



IEC 62304 Training Templates

www.softwarecpr.com

781-721-2921



www.softwarecpr.com

20 Berkshire Drive
Winchester, MA 01890
781-721-2921

Copyright

© Copyright 2010 Crisis Prevention and Recovery, LLC. (CPRLLC), all rights reserved. SoftwareCPR® is a DBA of Crisis Prevention and Recovery, LLC and the SoftwareCPR® logo is a registered trademark.

SoftwareCPR® authorizes its clients and SoftwareCPR®.com subscribers use of this document for internal review and training. **Any other use or dissemination of this document is expressly prohibited** without the written authorization of SoftwareCPR®. Individual original FDA documents in their original form without SoftwareCPR® annotations are public information and may be shared without restriction.

Legal Disclaimer

The training document that follows **should only be applied in the appropriate context with oversight by regulatory and software professionals with direct knowledge and experience with the topics presented.** The document should not be used as a cookbook since it is not adequate for some situations and may be excessive for others.

While SoftwareCPR® attempts to ensure the accuracy of information presented, no guarantees are made since regulatory interpretations and enforcement practices are constantly changing, and are not entirely uniform in their application.

Disclaimer of Warranties: The information is provided AS IS, without warranties of any kind. CPRLLC does not represent or warrant that any information or data provided herein is suitable for a particular purpose. CPRLLC hereby disclaims and negates any and all warranties, whether express or implied, relating to such information and data, including the warranties of merchantability and fitness for a particular purpose.

List of Training Templates Included

PROCEDURES

**SOFTWARE DEVELOPMENT PROCEDURE
SOFTWARE RISK MANAGEMENT PROCEDURE
SOFTWARE RELEASE PROCEDURE
SOFTWARE MAINTENANCE PROCEDURE**

PLANS

**SOFTWARE DEVELOPMENT PLAN
SOFTWARE CONFIGURATION MANAGEMENT PLAN
SOFTWARE TEST PLAN**

SPECIFICATIONS

**SOFTWARE REQUIREMENTS SPECIFICATION
USER INTERFACE DESIGN SPECIFICATION
SOFTWARE DESIGN SPECIFICATION
Template Training Example Mapping to 62304**

* If you are viewing this in Adobe format you can click on the tab labeled bookmarks to navigate easily to each template.

SOFTWARE DEVELOPMENT PROCEDURE

TRAINING EXAMPLE

DRAFT

Revision History

Date	Revision	Author	Description

Note: This example is conceived as the body of a procedure.

Explanatory comments are included in << comment >>.

Other text is example definition that outlines an example of Software Development Process.

This is not a complete SOP, just a training example to guide in the development of a Software Development SOP.

TABLE OF CONTENTS

1	Introduction.....	3
1.1	Purpose and Scope	3
1.2	Definitions.....	5
1.3	References.....	5
2	Roles and Responsibilities	5
3	Process Overview	5
4	Process Description.....	6
4.1	Software Safety Classification.....	6
4.2	Software Lifecycle.....	6
4.3	Activities.....	10
4.3.1.1	Planning	10
4.3.1.2	Requirement Analysis.....	10
4.3.1.3	SW Risk Analysis	11
4.3.1.4	Prototyping.....	12
4.3.1.5	SW Architectural Design	12
4.3.1.6	UI Design.....	13
4.3.1.7	SW Detailed Design.....	13
4.3.1.8	Coding.....	14
4.3.1.9	Integration and SW Build	15
4.3.1.10	Review	15
4.3.1.11	Test Design and Testing.....	20
4.3.1.12	SW Tools Validation.....	21
4.3.1.13	Configuration Management	21
4.3.1.14	Change Control	22
4.3.1.15	Problem Reporting and Resolution	23
4.3.1.16	Traceability	24
4.3.1.17	Release Activities.....	25
4.4	Phase transition requirements.	26
4.5	Metrics.	27
5	Record Keeping	28
5.1	Design History File.....	28
5.2	Device Master Record.....	28
5.3	Software Quality Manual.....	28
5.4	Record Retention	28
6	Training.....	28
7	External Supplier Management	29
8	SOUP Strategy.....	29

1 Introduction

1.1 Purpose and Scope

This procedure defines the Software Development process and activities within the <TBD> <<state the company, department, etc. to which this document applies. >>

Software is a component of the products which the <TBD> division designs and manufactures. The division is committed to the development of safe and effective software. To assure software quality this procedure specifies requirements for software development and quality assurance which includes design control, verification, and validation of software.

This procedure is intended to integrate with, and support, general design control requirements for product development overall.

This procedure applies to all software embedded in products designed by the <TBD> division. It does not apply to software used to support manufacturing or the quality system.

Conformance to this procedure is required only from the point in time that the project overall becomes subject to the general corporate design control procedures, although aspects may be useful on specific projects prior to this point.

Software developed to support research activities, early technical feasibility studies, or prototypes not intended for eventual release to customers is not subject to this procedure.

<<Detail applicability and extension of this procedure>>

Intended audience for this document will be all the personnel involved in software development activities for this project, in particular the SW development team and the SW Test team.

This procedure will also be used by the Project Manager, the QA Manager and the Product Manager as a tool to agree and control software development activities throughout the project life span.

<< Describe the expected users of this procedure. >>

There are five software-specific SOPs. The diagram below (Figure 1) provides an overview of their purpose and interrelationships.

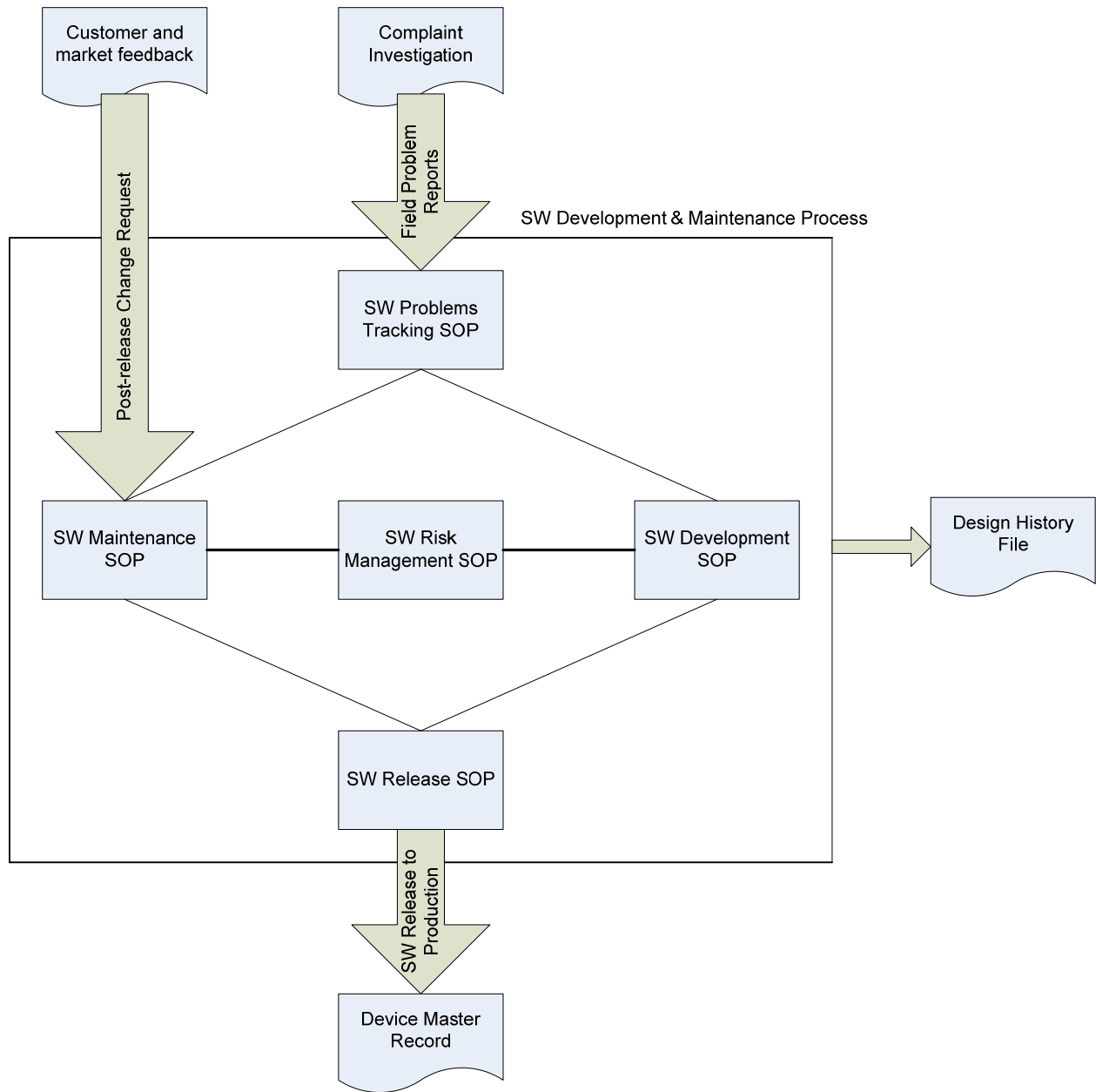


Figure 1

All sections of this procedure apply to development of software as part of new product development or significant enhancements to existing products.

Maintenance of released software is also subject to the Software Maintenance Procedure.

Release of product software to production is subject to the Software Release Procedure.

Tracking and handling of SW problems (both internally identified problems and field problems) is subject to the SW Problem Tracking SOP.

SW Risk Management (both Development and Maintenance) is subject to the SW Risk Management SOP.

Development of minor maintenance releases is subject to all sections of this procedure with the following exceptions:

- No Software Development Plan (Section 4.3.1.1) is required.

- Software Design Verification and Validation (Section 4.3.1.11) will only include testing relevant to the changes made and adequate regression testing as described in the Software Maintenance Procedure.
- Software Design Reviews (Section 4.3.1.10) will only be performed for the changes made – full reviews of the entire software system and associated documentation are not required.

1.2 Definitions

For department common definitions, please refer to <TBD> Common Definitions.
Additional specific terms are defined in Table 1 below.

SDP Software Development Plan

Table 1

<< Define any term, acronym or abbreviation used in the document that may be unfamiliar to readers. >>

1.3 References

Standards:

IEC 62304:2006, Medical device software—Software life cycle processes.

IEC 60601-1:2005 Medical Electrical Equipment – Part 1: General requirements for basic safety and essential performance.

AAMI TIR32:2004, Medical device software risk management. - << replace with IEC 80002 when released >>

.....
<< Add any relevant standard >>

Procedures:

SW Maintenance SOP

SW Release SOP

SW Risk Management SOP

SW Problems Tracking SOP

.....
<< Add any relevant SOP, including the Product Development and Design Control SOPs >>

2 Roles and Responsibilities

The mandatory steps identified in this procedure are responsibility of the SW Engineering and SW Quality Staff on a particular project.

The Manager of Software Development is responsible for verifying the application of this procedure and for control and maintenance of the software portions of the Design History File and the Device Master Record.

<< Define specific responsibilities for application of the procedure and approvals >>

3 Process Overview

The Software Development process defined in this procedure is the portion specific to Software of the entire Design Control Process for product development of <TBD> division.

<< Describe here how the software process is integrated into the design control process for product development. >>

The project software lifecycle shall consist of a series of phases that are performed incrementally to develop the software product and associated documentation. Each phase shall consist of activities that result in the creation, definition, or update of specified components and deliverable documentation.

Each software project, except minor maintenance releases, shall be governed by a specific Software Development Plan. The development plan shall define the application of this procedure to each specific project and indicate the specific documentation that needs to be produced. Software lifecycle phases, activities and documentation may be tailored to the unique requirements of each project, based on safety risk. Any tailoring shall be documented in the Software Development Plan of each project and needs to be approved by the Manager of Software Development..... << define specific responsibilities and eventual limitations to the tailoring allowed. >>

In cases where this procedure becomes applicable during or after development (e.g. big maintenance projects modifying older products) , the preexisting project documentation shall be evaluated to identify necessary actions. The project Software Development Plan shall identify what previous deliverables can be reused and what retrospective work shall need to be applied in terms of documentation update, applicable regression test, etc. with particular attention to risk and maintenance factors.

<< Define specific actions and limitations to address retrospective work. >>

4 Process Description

4.1 Software Safety Classification

The project team shall determine the safety class of the project software based on severity and according to the guidance of IEC 62304:2006, clause 4.3. The safety class shall be documented in the risk management report and can be used as base for tailoring the requirements of this procedure to the specific needs of the project.

Further classification of different software subsystems or software items, once the architectural design decomposition has been achieved, is allowed but not required and should be pursued only if necessary to justify further tailoring of the process for some software subsystems or items (e.g. limitations to design documentation or to technical reviews).

4.2 Software Lifecycle

The software lifecycle is broken up into three main phases: Requirement Definition, Construction and Validation/Release. Some activities are specific to each phase, while some other activities are spread up through all of the phases.

In the Requirement Definition phase, main planning activities are started, to be continued into the following phases. Requirement Analysis, architectural design and initial Risk Management activities are also characteristic of this phase. They shall be based on the System Level Requirements defined in the Product Specification or other design input document.

During Construction phase the Software System is designed, built and verified.

The Construction phase is divided in three sub-phases: Design, Code and Test.

These three sub-phases can be applied repeatedly throughout Construction phase for three reasons:

1. Once the high level software architecture is defined in Requirement Definition phase, the software system can be broken up into main software items or subsystems. Interfaces between subsystems are defined, then each subsystem could proceed into construction as much independently as possible one from the other through its own Design-Code-Test path. In this way several parallel Design-Code-Test path could be in progress.
2. The software construction is designed to be incremental, with an integration plan that defines several subsequent builds incorporating incremental sets of features. Each build is integrated and

- tested applying both integration tests and some level of system test appropriate for the integrated features. Regression test is also applied depending on features subsequently integrated.
3. Anomalies identified during Test sub-phase could require to recycle back to design and repeat the Design-Code-Test sequence as necessary until satisfied of the quality of the integrated subsystem or system.

During Validation and Release phase, a final set of validation testing is applied and the final release of the full integrated software system is built for release to manufacturing. Release activities take place and the final project documentation is prepared for the release.

Table 2 below shows the main activities that should take place in each phase and the main deliverables. Note that items referenced as deliverables are not intended as specific documents, but as types of deliverables. The documentation structure for each project shall be defined in the project SW Development Plan .

Activities not listed in a given phase can be performed in that phase for some identified portions of the project, provided that they are defined and justified in the SW Development Plan.

Figure 2 shows in graphical way the sequence of the main activities related to the lifecycle phases and shows how planned integrations could proceed in parallel, each one of them with its own design-code-test sequence.

<< This depicts an example of possible lifecycle. Adapt it to your needs, based on project size and project characteristics of your organization and tying it to the overall design control/device development phases of your organization.

It is also possible to define tailoring allowed based on software safety classification of each project. >>

Project Phase Activities	Requirement Definition	Design	Construction		
			Code	Test	Validation and Release
Management Activities	<ul style="list-style-type: none">•Planning•Requirement Analysis•Risk Analysis	<ul style="list-style-type: none">•Planning•Requirement Analysis•Risk Analysis	<ul style="list-style-type: none">•Planning•Risk Analysis	<ul style="list-style-type: none">•Planning•Risk Analysis	<ul style="list-style-type: none">•Planning•Risk Analysis•Release Activities
Development Activities	<ul style="list-style-type: none">•Prototyping•SW Architectural Design•UI Design (HL)	<ul style="list-style-type: none">•SW Design (HL)•SW Design (Detailed)•SW Design (Interface)•UI Design (Detailed)	<ul style="list-style-type: none">•Coding	<ul style="list-style-type: none">•Integration•SW Build	<ul style="list-style-type: none">•SW Build
Verification Activities	<ul style="list-style-type: none">•Planning Review•Requirement Review•System Architecture Review	<ul style="list-style-type: none">•Test Design•Planning Review•Design Review•RA Review	<ul style="list-style-type: none">•Test Design•Unit Test•Code Review•Test Review	<ul style="list-style-type: none">•Test Design•Integration Test•System Test•Regression Test•Test Review•Defects Review	<ul style="list-style-type: none">•Test Design•System Test•Regression Test•Test Review•Defects Review•Release Doc Review
Supporting Activities	<ul style="list-style-type: none">•SW Tools validation•Configuration Management	<ul style="list-style-type: none">•SW Tools validation•Configuration Management•Change control	<ul style="list-style-type: none">•Configuration Management•Problem Resolution•Change control	<ul style="list-style-type: none">•Configuration Management•Problem Resolution•Change control	<ul style="list-style-type: none">•Configuration Management•Problem Resolution•Change control
Deliverable Types	<ul style="list-style-type: none">•SW Development Plan•Configuration Management Plan•Test Plan (initial)•SW Requirement Spec•SW Risk Analysis (initial)•SW Design Spec (architecture only)•UI Design Spec	<ul style="list-style-type: none">•(Updated SDP)•(Updated SCMP)•Updated Test Plan•SW Design Spec•Updated UI Spec•Updated SRS•SW Risk Analysis with RCMs•Traceability Matrix (Initial)	<ul style="list-style-type: none">•Updated plan documents•Verified Code•Unit Test Cases•Unit Test Result•(Updated SW RA)	<ul style="list-style-type: none">•Updated plan documents•Integrated Code•Integration Test Cases•System Test Cases•Integration Test Result•System Test Result•(Updated SW RA)•(Updated Traceability Matrix)	<ul style="list-style-type: none">•Release Master Code•Test Plan (final)•System Test Cases•System Test Result•Test Summary Report•SW RA (final)•Traceability Matrix (final)•Release Documents•SW Maintenance Plan

Table 2

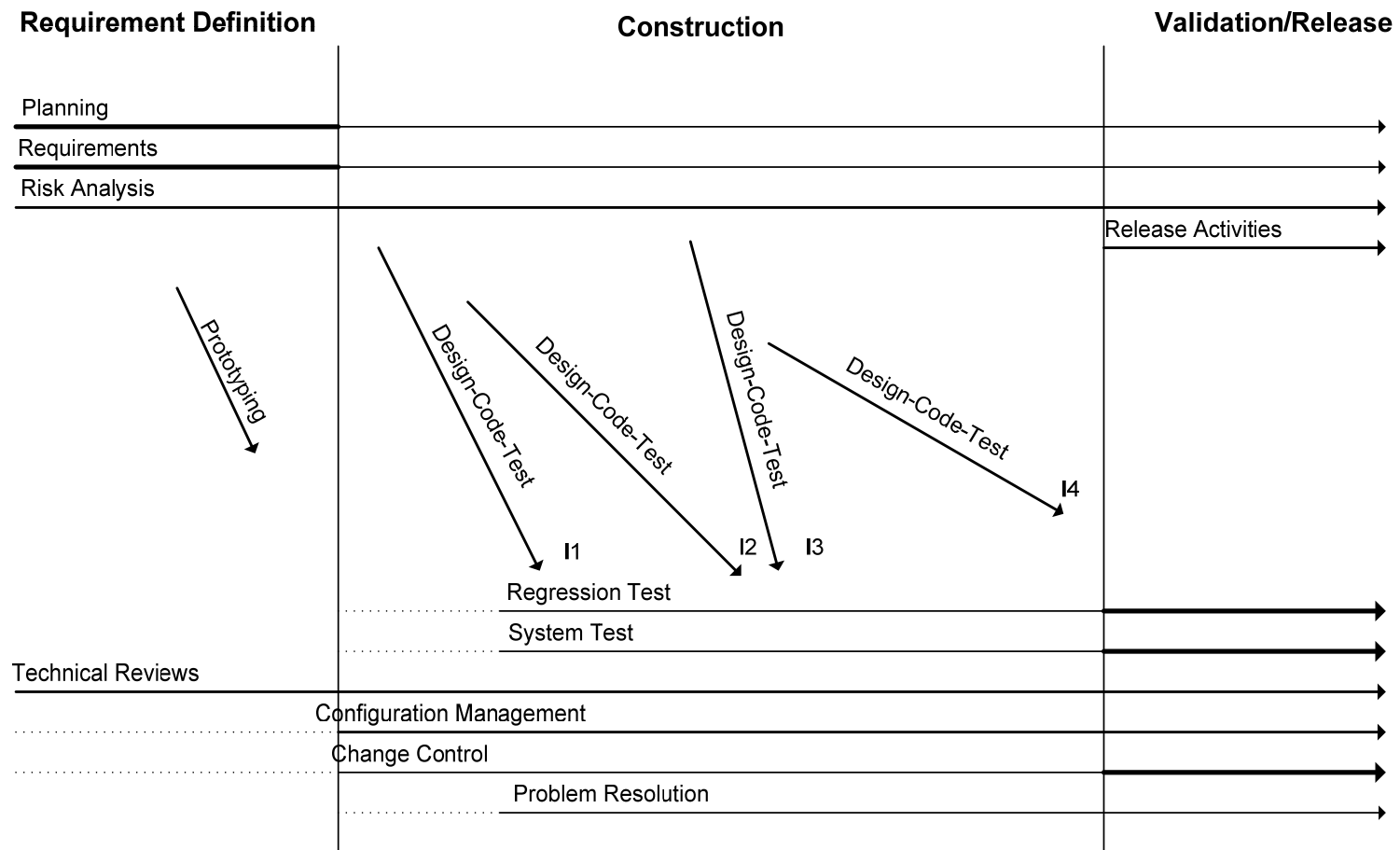


Figure 2

4.3 Activities

<< The following list of activities, with some example text, is modeled on the lifecycle described in Table 2Error! Reference source not found.. Add/Remove/Modify according to your own lifecycle description. >>

4.3.1.1 Planning

Input:

Main input for this activity will be:

- Project level planning
- System Requirements
- System Risk Analysis

<<Describe all the fonts on which this activity is based. >>

Output:

Main deliverables from this activity will be:

- Software Development Plan
- Software Configuration Management Plan
- Software Integration Plan <<Note: this could be included in the Software Development Plan>>
- Software Test Plan
- Software Risk Management Plan <<Note: this could be included in the Software Development Plan>>
- SW Maintenance Plan <<Note: if SW Maintenance Procedure is enough to define maintenance phase requirements, a specific maintenance plan could not be required. >>

<< Define the deliverables in terms of document types, not specific documents >>

Activity:

Planning activity shall start during Requirement Definition phase and continue for the entire project span.

The SW development plan must be well defined before construction phase is entered. However it will be an evolving document and will be updated as needed while the project progresses through the subsequent phases to reflect any change in the process or strategy or deliverables. The plan shall be formally reviewed at least at main phase transitions as described in Section 4.4. Additional reviews could take place in case of significant changes.

The Software Configuration Management Plan must be developed before configuration management is applied. This means that it needs to be defined early in the project in order to be able to control the deliverables from the development process.

Test planning will take place in all lifecycle phases. The Software Test Plan is the document that ties together all the test planning and test execution and needs to be started in the Requirement Definition phase, to be substantially updated during subsequent phases, as development and test progress.

The Software Maintenance Plan must be ready at software release to address the subsequent maintenance phase of the project. It could be omitted if no project specific deviations from the SW Maintenance procedure are planned.

<<Describe planning activities in your own environment. >>

4.3.1.2 Requirement Analysis

Input:

Main input for this activity will be:

- System Requirements
- System and Software Risk Analysis

- Knowledge of system design
- Experience with previous similar projects

<<Describe all the fonts on which this activity is based. >>

Output:

Main deliverables from this activity will be:

- Software Requirement Specification

<< Define the deliverables in terms of document types, not specific documents >>

Activity:

Requirement Analysis shall be with planning the main activity of the Requirement Definition phase.

Software Requirement Specifications, as complete as possible, shall be available, reviewed and approved before entering the Construction phase.

Requirement Analysis shall continue through the following Construction phase. Requirements will become more clear and detailed, new requirements will be added, existing requirements will be modified, based on better knowledge of the system, added risk control measures, identification of defects in the spec.

Updated versions of the SRS will be released and approved. Specific requirements need to be approved before their implementation is started.

<< Describe Requirement Analysis tasks, inputs, outputs, verification and its relationship with other activities. >>

4.3.1.3 SW Risk Analysis

Input:

Main input for this activity will be:

- System Requirements
- System Risk Analysis
- Software Requirements
- Knowledge of system and software design
- Experience with previous similar projects

<<Describe all the fonts on which this activity is based. >>

Output:

Main deliverables from this activity will be:

- Software Risk Analysis
- Software Risk Control Measures

<< Define the deliverables in terms of document types, not specific documents >>

Activity:

SW Risk Analysis shall start in the Requirement Definition phase and will go together with Requirement Analysis. The SW Risk Management procedure defines the process to be applied for Software Risk Analysis and Control within the general project risk management process..

SW Risk Analysis and Risk Management will span over the entire SW lifecycle.

RCMs identified by Risk Analysis and Risk Control as to be implemented in software shall be reflected into software requirements. Related risk control requirements shall be further analyzed to identify possible generation of new sequences of events resulting into hazardous situations.

SW Risk analysis shall also be applied to:

- Anomalies evaluation to rate the severity of the anomaly and identify possible impacts of the fix.
- Change control, to evaluate the risks connected with the implementation of changes.

The QA Manager and the Project Manager will be responsible for application of Software Risk Management activities.

<< Describe your approach, inputs, outputs, verification. >>

4.3.1.4 Prototyping

Input:

Main input for this activity will be:

- High level description of prototyping intent

<< Describe all the fonts on which this activity is based. >>

Output:

Main deliverables from this activity will be:

- Working prototype fulfilling the intent

<< Define the deliverables in terms of document types, not specific documents >>

Activity:

Prototyping can be used early in the project to support feasibility of technical concepts and solutions or to provide early presentation of some features, for example user interface structure, to support discussion and clarify requirements.

<< Describe how prototype construction is supported and controlled, what would be its use and limitation. >>

Code development during prototyping shall not follow the normal software process:

- Prototyping does not need to be based on formally approved requirements. It will be enough an informal concept description
- Prototyping design shall be developed as needed, but will not be formally reviewed
- Prototyping code shall not be formally inspected
- Testing shall be limited to verification that the prototype is fulfilling its intended use
-

<< Define your own specific or additional deviations from the normal code production and verification process. >>

If any of the code developed during prototyping will be re-used afterwards in the final device code, the following process shall be followed:

- The SRS shall include specific requirements related to the functionalities provided by the prototype code to be included.
- The design specification shall be updated to reflect the added code.
- The code shall be formally inspected according to the technical review process.
-

<< Define your own requirement for the process to convert prototype code into real code. >>

4.3.1.5 SW Architectural Design

Input:

Main input for this activity will be:

- System Requirements
- Software Requirements
- System and Software Risk Analysis
- System Architectural design

<< Describe all the fonts on which this activity is based. >>

Output:

Main deliverables from this activity will be:

06 Feb 2009

SW DEVELOPMENT SOP Training Example

Page 12 of 29

Copyright 2009 SoftwareCPR®

- Software Architectural Design document

<< Define the deliverables in terms of document types, not specific documents >>

Activity:

SW System partitioning into main SW items or subsystems will take place during Requirement Definition phase. Interfaces between the subsystems will be defined in a way to achieve segregation and to allow as much independency as possible so that parallel construction paths can be started on different subsystems.

An initial SW Design Specification will be available before construction phase is entered, describing high level architecture and subsystems interfaces.

SW architectural design will continue during Construction phase, adding increased detail and defining the architecture of each software item.....

<< Describe architectural design tasks, inputs, outputs, verification and its relationship with other activities. >>

4.3.1.6 UI Design

Input:

Main input for this activity will be:

- System Requirements
- Software Requirements
- System and Software Risk Analysis
- Usability Requirements

<< Describe all the inputs on which this activity is based. >>

Output:

Main deliverables from this activity will be:

- UI design description
- UI prototype (if planned at project level)

<< Define the deliverables in terms of document types, not specific documents >>

Activity:

User interface design should start as early as possible, at least with the definition of the UI structure and model. Prototyping could be used to support UI design. UI design should continue throughout the project and goes together with Requirement Analysis.

Usability testing should support the verification of the UI design.

<< Describe here your approach to UI design, its inputs, outputs and verification. >>

4.3.1.7 SW Detailed Design

Input:

Main input for this activity will be:

- System and Software Requirements
- Software Architecture
- System and Software Risk Analysis

<< Describe all the inputs on which this activity is based. >>

Output:

Main deliverables from this activity will be:

- Software Design Specification(s)
- (Headers and comments in code) << Sometimes some of the design is maintained within the code itself >>

<< Define the deliverables in terms of document types, not specific documents >>

Activity:

SW detailed design will be an iterative activity performed mainly during design sub-phase of the construction phase. Any single unit shall be fully designed and its design shall be reviewed before the unit code is developed. Interfaces between the units and to the hardware shall be defined.

Detailed design shall be updated for any change or fix affecting the design before the change is implemented into the code.

The structure of the design documentation for each project shall be defined in the project SW Development Plan.

<< Describe here your detailed design policies, tasks and their interrelation with other activities. There could be several options:

- You could require design for each item to be defined and approved before coding is started on that item, or you could allow design and code to proceed in parallel provided that design is reviewed and approved before the code is accepted. You could even have design and code reviewed together.
- You could completely document your detailed design in the SW Design Specification, or you could have different design documents for different items, or you could have some of the detail design as headers and comments in the code and use some documentation tools to extract design information.
- You could use any design techniques or design tools.

Whatever your approach, describe it here.

You could also keep this section of the procedure pretty vague and leave flexibility to each project to define their specific design policy in the SW Development Plan.

The main requirements in the procedure should be to have design documents and to keep them up to date throughout the project. >>

4.3.1.8 Coding

Input:

Main input for this activity will be:

- Software Requirements
- System and Software Risk Analysis
- Software Architecture
- Software Design

<<Describe all the fonts on which this activity is based. >>

Output:

Main deliverables from this activity will be:

- Code
- Updated software design

<< Define the deliverables in terms of document types, not specific documents >>

Activity:

Coding will be the main activity performed during Code sub-phase of the Construction phase. However coding will not be limited to that phase. Some coding will start already in the Requirement Definition phase as part of prototyping activity (see description of prototyping for additional details). Coding related to software fixes shall take place also during Validation and Release phase.

Code shall be verified though code reviews. SW Units will be subjected to formal code inspection or to code walkthrough based on risk considerations.

Code shall be subject to unit test before it is delivered for integration.

<< Describe coding tasks, inputs, outputs, verification and its relationship with other activities.>>

4.3.1.9 Integration and SW Build

Input:

Main input for this activity will be:

- Verified Code under configuration control
- Build scripts

<<Describe all the fonts on which this activity is based. >>

Output:

Main deliverables from this activity will be:

- Executable integrated Code
- Build release notes

<< Define the deliverables in terms of document types, not specific documents >>

Activity:

SW Integration and SW build are performed during Test sub-phase of the Construction phase.

Integrations and builds shall take place for different purposes according to an integration plan that shall be defined within the project Software Development Plan or as a separate document.

Final integration and formal build will be performed during Validation and Release phase.

Software Code to use in integration and build must be under configuration management. Any build script needs also to be under configuration management.

The build procedure for each project shall be specified in the project Software Configuration Management Plan.

Each integration shall be verified to include all the planned features and related software items and to correctly interface with hardware.

<< Describe integration and build task. Explain differences, if any, between tasks taking place during Construction and tasks taking place during Validation and Release (e.g. different level of formality and control). Define inputs, outputs, verification activities, etc. >>

4.3.1.10 Review

Input:

Main input for this activity will be:

- Documentation to review (see below)

<<Describe all the fonts on which this activity is based. >>

Output:

Main deliverables from this activity will be:

- Review Report

<< Define the deliverables in terms of document types, not specific documents >>

Activity:

Reviews will be the main verification technique used to verify requirements, design, code, risk analysis and test. Review will be extensively applied to all formal documentation produced in the project, therefore reviews will be planned in all lifecycle phases.

All technical reviews imply participation by persons with appropriate technical expertise other than the responsible programmer.

Formal reviews will require that the documentation to be reviewed is put under configuration control and distributed to participants in the review at least <TBD> in advance. A review leader shall be designated and shall be responsible for the orderly execution of the review.

Table 3 below list the required formal reviews to be performed during the software project lifecycle, including the indication of the review objectives, review timeframe, required participants and a list of checks to perform during the review. Note that the description only indicate the types of deliverables to submit for review, not the actual document names: actual project document structure will be defined in each project SW Development Plan. The list of checks can be integrated by additional checks.

Each formal review shall be documented with a review report, indicating eventual action items. The review leader shall designate a person responsible for the verification that the action items get implemented in an appropriate timeframe.

Follow up from the review, depending on the amount of action items and on the review leader decision, could imply an additional review to verify implementation of the action items. In case a second review is not considered necessary, the person designated to verify the action items will be responsible for checkout of the completed action items on the review report.

In addition to the reviews indicated in the table, other spontaneous formal or informal reviews are encouraged.

Review Type	Description	Checklist	Participation	When
Initial Requirements Review	One or more reviews that cover the first official versions of the Software Requirements Specification, SW Risk Analysis, and User Interface Design Spec.	<ul style="list-style-type: none"> • SW requirements implement System requirements and are traceable to them • SW requirements implement RCMs and are traceable to them • SW requirements uniquely identifiable, unambiguous, not contradictory and testable • Risk Analysis according to SW Risk Management SOP, as complete as possible for the related project phase. • Usability requirements correctly implemented in UI design 	Involve at least Project Manager, SW Development Leader, 1 Test representative, 1 MKTG representative.	End of Requirement Definition phase
Final Requirements review	One or more reviews that cover the final versions of the Software Requirements Specification, Risk Analysis, and User Interface Design Spec.	<ul style="list-style-type: none"> • SW requirements implement System requirements and are traceable to them • SW requirements implement RCMs and are traceable to them • SW requirements uniquely identifiable, unambiguous, not contradictory and testable • Risk Analysis according to SW Risk Management SOP, as complete as possible for the related project phase. • Usability requirements correctly implemented in UI design 	Involve at least Project Manager, SW Development Leader, 1 Test representative, 1 MKTG representative.	End of Construction phase
Architectural Design Review	Formal walkthrough of the overall architectural decomposition with particular focus on mitigation relevant to hazards.	<ul style="list-style-type: none"> • SW Architecture implements System and SW requirements, including RCMs • Interfaces between architectural elements defined. • Hardware interfaces defined • SOUP items identified and supported 	Involve at least Project Manager, SW Development Leader, 1 Test representative, 1 designer from outside the project team.	End of Requirement Definition phase
SW Planning Review	One or more reviews of SW Development Plan, SW Test Plan, SW Configuration Management Plan, and other planning support documents.	<ul style="list-style-type: none"> • SW Planning compatible and coordinate with System planning • Planning as complete as possible and up to date for the related project phase. • Resources and responsibilities defined • Deliverables defined 	Involve at least Project Manager, SW Development Leader, SW Test Leader	End of Requirement Definition phase. End of Construction phase. When needed, for significant changes.

Review Type	Description	Checklist	Participation	When
SW Design Review	Formal design reviews to be planned when design of software items is defined, with particular focus on mitigations relevant to hazards. Could be held separately or together with code inspections on the same SW items.	<ul style="list-style-type: none"> • Detailed design implements the SW architecture and is not contradictory to it. • Interfaces are defined and are compatible with the overall architecture. • RCMs are addressed in the design • Design is implementable. 	Besides the author, involve at least another designer with significant experience on the application.	As needed during construction phase. All reviews must be completed and documented before transition to Validation phase
Code inspections	Formal code inspections to be held at least on the SW items identified as potentially critical by SW Risk Analysis. Could be held separately or together with design review on the same SW items.	<ul style="list-style-type: none"> • Coding performed according to Coding Standard • Interfaces implemented according to design • Code implements requirements, including RCMs • Additional checks on proper event sequence, data and control flow, resources allocation, error handling, variables initialization, self diagnostics, memory and stack management, boundary conditions. 	Besides the author, involve at least another programmer with significant experience on the application.	As needed during construction phase. All inspections must be completed and documented before transition to Validation phase
Test Design Review	Formal inspection of the test designs and test cases for coverage and adequacy.	<ul style="list-style-type: none"> • Verification strategies and test procedures are appropriate. • Unit and Integration test trace to design • System test traces to requirements • Test cases include verifiable pass/fail criteria. • All planned types of test are covered • All SW Requirements, including RCMs are covered by at least one type of test 	Besides the author, involve at least the SW Test Leader and 1 representative from the SW Development group.	As needed during construction and validation phase. Review must happen before the test cases are used for a formal launch. All reviews completed and documented before final release launch.
Test Result Review	Review of the test results and of the proper handling of identified anomalies.	<ul style="list-style-type: none"> • Test results meet the required pass/fail criteria • Test launch completed and documented as planned • Identified anomalies handled through the Problem Resolution process. 	Involve at least Project Manager, SW Development Leader and SW Test Leader.	As needed during construction and validation phase. Reviews can be planned at the end of each test session.
Phase Transition	At end of each phase a review will be held to	<ul style="list-style-type: none"> • Phase transition criteria are met • Project documents are correctly handled 	Involve at least the Project Manager, the QA Manager,	End of Requirement Definition phase.

Review Type	Description	Checklist	Participation	When
Reviews	determine if the objectives of the phase have been met and if its deliverables are adequate to warrant progression to the next phase. These reviews will focus on risk management and assurance that the project is proceeding in conformance with all relevant procedures and plans.	<ul style="list-style-type: none"> • Planning for next phase is available and up to date. 	the SW Development Leader and the SW Test Leader	End of Construction.phase.
SW Release Review	To be held before final release to determine if the objectives of the project have been met and if its deliverables are adequate to warrant release to Operations.	<ul style="list-style-type: none"> • Phase transition criteria are met • Release documentation complete • SW residual anomalies correctly handled and compatible with release. • Maintenance procedures and post-release planning available and complete 	Involve at least, the VP R&D, the Project Manager, the QA Manager, the SW Development Leader and the SW Test Leader.	End of Validation phase

Table 3

<< Add/Modify/Replace according to your review policy >>

4.3.1.11 Test Design and Testing

Input:

Main input for this activity will be:

- Software Test Plan
- Software Requirement Specifications
- System and Software Risk Analysis
- Software Architecture
- Software Design
- Code

<<Describe all the fonts on which this activity is based. >>

Output:

Main deliverables from this activity will be:

- Test Cases
- Test Reports
- Identified Anomalies
- Test summary report

<< Define the deliverables in terms of document types, not specific documents >>

Activity:

Software test for each project shall be governed by a project specific Software Test Plan, that shall define and coordinate all the testing activities.

The Software Test Leader shall be responsible for the definition and the application of the test plan.

Software Testing shall include:

- Unit test
- Integration test
- System test
- Validation (user) test
- Regression test

In addition to specific software testing, some software design validation shall be performed as part of system design validation, including clinical or user trials when required.

Unit test design shall start when detailed design for the unit is available and could proceed in parallel with unit coding. Unit test design will be responsibility of <TBD>.

Unit testing shall be performed during Code sub-phase of Construction phase.

<< Describe your own unit test policy, when applied, by whom, what inputs, what outputs, etc. >>

Integration test design can be started on any integration item as soon as integration plan is available and the interfaces between the items to be included in the integration are defined, therefore integration test design is not limited to the Test sub-phase of the Construction phase, but could be started before.

Integration testing shall be performed during Test sub-phase of the Construction phase. Some integration testing could also be applied as part of regression testing during Validation and Release phase.

<< Describe your own integration test policy, when applied, by whom, what inputs, what outputs, etc. >>

SW System test design can be started as soon as formally approved Software Requirements Specification and an approved SW System Architecture are available, therefore it can start already towards the end of the Requirement Definition phase, to continue through the Construction and the Validation and Release phase. SW System Test on integrated portions of the System will be applied during Test sub-phases of the Construction phase.

A final formal SW System Tests shall be applied during Validation and Release phase.

<< Describe your own system test policy, when applied, by whom, what inputs, what outputs, etc. >>

Regression Test will be extensively applied on changes, fixes, incremental releases.....

<< Describe your own regression test policy, when applied, by whom, what inputs, what outputs, etc. >>

4.3.1.12 SW Tools Validation

Input:

Main input for this activity will be:

- Tool specification
- Tool intended use

<< Describe all the fonts on which this activity is based. >>

Output:

Main deliverables from this activity will be:

- Tool validation protocol
- Tool validation report

<< Define the deliverables in terms of document types, not specific documents >>

Activity:

Tools used for SW development, SW testing and any other SW related activities in the project need to be validated.

Tool validation is responsibility of <TBD>

Whenever new tools are introduced to be used in a project, a validation plan shall be written and executed.

Validation needs to be performed before formal use of the tool for producing, documenting or verifying/validating product software.

Validation reports shall be available for each tool and shall be kept in <TBD>.

<< Describe here tasks related to this activity and how, when and from whom will be carried out. >>

4.3.1.13 Configuration Management

Input:

Main input for this activity will be:

- Software Configuration Management Plan
- Software Configuration Items

<< Describe all the fonts on which this activity is based. >>

Output:

Main deliverables from this activity will be:

- Controlled Software Configuration Items
- Configuration Status Reports

<< Define the deliverables in terms of document types, not specific documents >>

Activity:

All SW project documents shall need to be under configuration management before they are formally reviewed, therefore document control will be applied starting from Requirement Definition phase.

Each project document structure shall be defined in the project SW Development Plan.

All Software Items, including SOUP, shall be put under configuration management before they are formally tested (unit test included) or are submitted for integration, whichever comes first.

All Software Test Cases and Test Scripts will be put under configuration management before they are reviewed and used for formal testing.

All Software Tools used for Development, Test and Process support will also be put under configuration management, before they are used for formal activities within the project.

A specific Software Configuration Management Plan shall be developed for each project to detail project SW configuration process, use of specific configuration tools and any project specific requirements such as paths and directories of data, items to be put under SW configuration, login and access requirements.

Any changes to code under SW configuration control and/or to approved software documents will be handled through formal change control.

An audit trail shall be maintained to demonstrate correct application of configuration control and allow configuration status accounting.

The Project Manager will be responsible to implement the configuration management process for the project.

<< Add/Modify/Replace according to your configuration management policy >>

4.3.1.14 Change Control

Input:

Main input for this activity will be:

- Software Change Requests

<<Describe all the fonts on which this activity is based. >>

Output:

Main deliverables from this activity will be:

- Change Request Disposition
- Implemented changes documentation

<< Define the deliverables in terms of document types, not specific documents >>

Activity:

Once the Software Requirements are established and the Software Requirement Specification has been formally reviewed and approved, requests for changes to requirements will be collected through the Change Control mechanism.

Requirement change requests can come from different sources:

- Marketing requests
- On-going risk analysis and risk management activities
- Enhancements identified during development
- V&V activities
-

Change requests affecting requirements, code and/or any other approved software documents shall be stored in the Change Control database.

A Change Control Board will be established and shall meet with appropriate frequency to evaluate pending requirement change requests and define their disposition. Evaluation will take into account the impacts of changes on product scope and quality, schedule, budget, resources and risk factors.

The Change Control Board composition shall be defined in each project SW Development Plan. The composition itself will vary during the different project phases, depending on how much implementation of a change can affect schedule and delivered quality. During Validation phase a QA representative shall be required to approve changes.

Approved changes will be assigned a responsible for change implementation and an integration milestone for incorporation into the system. The responsible for change implementation shall be responsible to verify that all the documentation affected by the change gets appropriately updated before the change is considered complete.

As part of the change evaluation and implementation, risk analysis of the change shall be performed to verify that changes do not interfere with existing RCMs and to identify possible new sequences of events that could lead to hazardous situations. If necessary, additional RCMs shall be identified and implemented to address the new sequences of events.

As part of the change evaluation and implementation, the test procedures and the test plan shall be updated to reflect additional testing for the change itself and any required regression testing.

Verification of the change shall include verification that it has been implemented as specified and that all required activities have been performed.

Changes that are not approved shall be rejected or postponed for subsequent evaluation. A reason for rejecting or postponing shall be added to the change request record in the database.

<< Add/Modify/Replace according to your change control policy >>

4.3.1.15 Problem Reporting and Resolution

Input:

Main input for this activity will be:

- Software Anomaly Report

<<Describe all the fonts on which this activity is based. >>

Output:

Main deliverables from this activity will be:

- Anomaly Disposition
- Implemented fixes documentation

<< Define the deliverables in terms of document types, not specific documents >>

Activity:

Formal Problem reporting and resolution process shall be applied starting with Code sub-phase of Construction phase as soon as any formal test is applied to software units.

It shall be used to track any anomaly found through formal testing as well as any anomaly reported from outside the software organization. It shall not be used to track problems found through initial debug or informal testing.

The SW Problems Tracking SOP describes the overall issue tracking and problem resolution process. It addresses issue identification and assignment, and problem identification, reporting, investigation, correction and verification processes. This procedure address the specific use of any issue management tool and implements the Issue Tracking and Problem Resolution processes.

Specific responsibilities for SW problems handling in each project shall be defined in the project SW Development Plan. Responsibilities can vary during the different project phases, depending on how much implementation of a fix can affect schedule and delivered quality. During Validation phase a QA representative shall be required to review anomalies and approve fixes.

<< Describe your own problem reporting and resolution policy >>

4.3.1.16 Traceability

Input:

Main input for this activity will be:

- System and Software Requirements
- System and Software Risk Analysis
- Software Architecture and Detailed Design
- Software Test Documents
- Labeling documentation

<<Describe all the fonts on which this activity is based. >>

Output:

Main deliverables from this activity will be:

- Traceability Matrix(ces)

<< Define the deliverables in terms of document types, not specific documents >>

Activity:

Traceability is used to demonstrate coverage at different levels:

- SRS coverage of system requirements and risk analysis
- Design coverage of requirements
- Test coverage of requirements and risk control measures

Traceability is also an important tool to allow an easier impact analysis of changes and anomalies.

Traceability shall be started as soon as approved input documents are available and shall be kept up to date throughout the software lifecycle. Use of specific traceability tools, if any, shall be specified for each project in the project Software Development Plan.

The diagram in Figure 3 shows an example of traceability between Software project documentation.

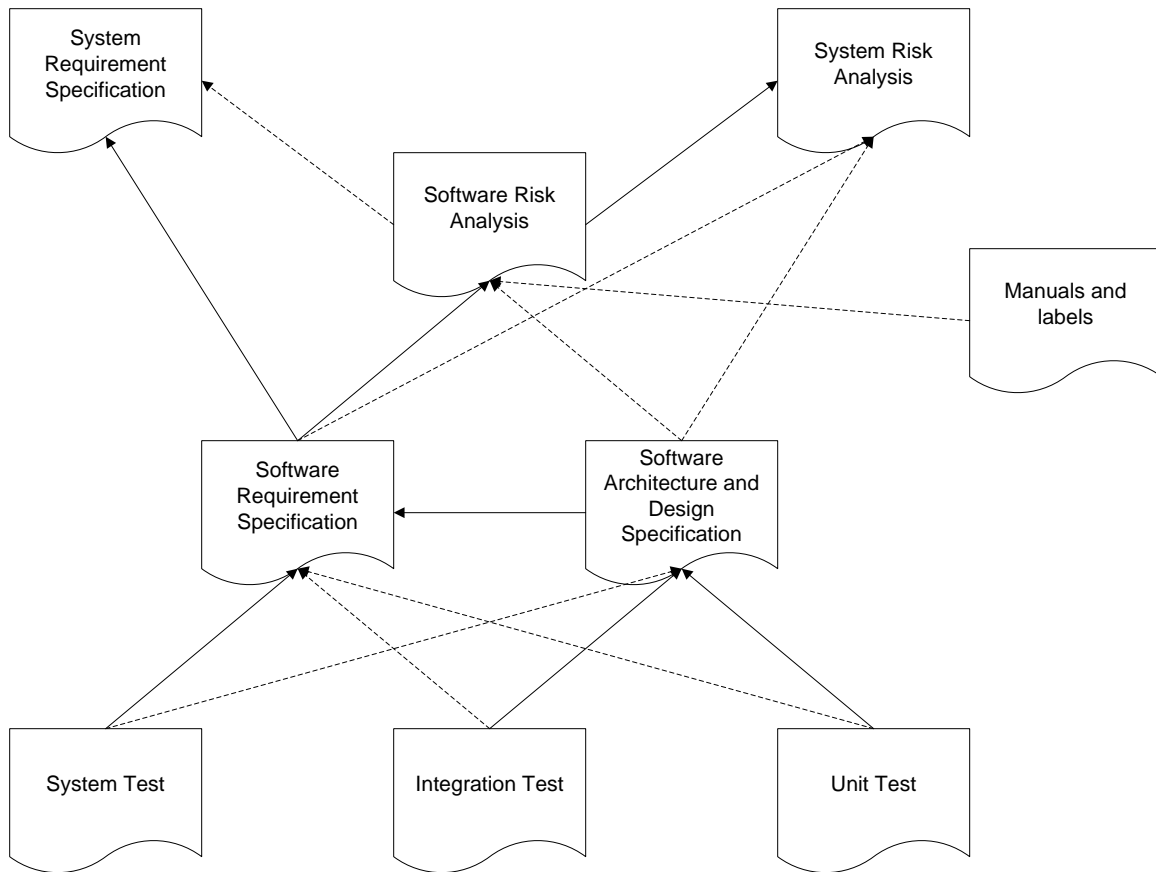


Figure 3

Solid lines indicate main traceability paths, while dotted lines indicate secondary traceability paths. For example:

- System Test will mainly verify requirements, but there could be some requirements that can only be verified in Integration or Unit test
- Risk Control Measures will normally reflect into requirements, but there could be some risk control measures that are only implemented into design.

Traceability will be implemented by using specific and unique traceability tags within the documents. Specific tags to use in each project shall be defined in the project Software Development Plan.

Traceability matrix will be created to link the different documents.

Test cases traceability will demonstrate that all functional requirements and types of testing have been adequately addressed. Details of project specific test traceability shall be defined in the project Software Test Plan.

<< Add/Modify/Replace according to your traceability policy >>

4.3.1.17 Release Activities

Input:

Main input for this activity will be:

- Software Development Plan
- Software Configuration Management Plan

- Integrated, verified and validated build.

<<Describe all the fonts on which this activity is based. >>

Output:

Main deliverables from this activity will be:

- Master Executable(s)
- Release documentation

<< Define the deliverables in terms of document types, not specific documents >>

Activity:

Release must be explicitly approved after verification that all activities are complete and appropriately documented and all planned verification and validation has been performed and documented. Approved final release software and associated documentation shall be transferred (directly or by specific reference) to the Device Master Record (see Section 5.2) as part the product release process with some specific requirements as defined in the Software Release Procedure.

In addition:

- Manufacturing staff will be trained in aspects of the use of the software and its diagnostic functions that are relevant to the device manufacturing process.
- A procedure defining the software replication (and associated QC checks) and installation process for use by manufacturing or its vendors will be provided.

<< Describe release tasks, inputs, outputs, verification and their relationship with other activities.>>

4.4 Phase transition requirements.

<< The purpose of defining project phases is to define quality control checkpoints to ensure adequate in-process reviews, verification and validation. Performing the different activities in a proper sequence and under control prevents costly design mistakes that are usually difficult and expensive to correct in later phases of the project. It is necessary to define some phase transition criteria. It is also necessary to have phase transition reviews to verify that the criteria are met and approve the transition to next phase. Below an example is given. Change and/or complete it with your own phase transition criteria. >>

Phase transition reviews will take place at the end of each phase to verify that the activities have been performed as planned and the software project is ready to proceed to next phase. Table 4 below shows the phase transition criteria for the 3 main phases of the lifecycle.

Phase	Phase exit criteria
Requirement Definition	<ul style="list-style-type: none"> • Software Development Plan Approved • Software Requirement Specification Approved • Software System Architecture Approved • Software Configuration Management Plan Approved • Software development infrastructure in place and validated • Integration Plan is defined • Initial SW Risk Analysis exists • Initial SW Test Plan exists
Construction	<ul style="list-style-type: none"> • SW design spec Approved • Code and SW documents under configuration control • Code integrated and verified and build reproducible. • SW Test Plan Approved • SW Risk Analysis Approved • Software Development Plan up-to-date and Approved

Phase	Phase exit criteria
Validation and Release	<ul style="list-style-type: none"> • Software Configuration Management Plan up-to-date and Approved • Software Requirement Specification up-to-date and Approved • Unit and Integration test completed, documented and Approved • SW Anomalies documented and handled through Problem Resolution Process • System Test Cases exists according to Test Plan • Traceability Matrix exists • All planning documents up-to-date and Approved • System Test and Regression Test Completed, documented and Approved • Test Summary Report Approved • Software Requirement Specification up-to-date and Approved • Software Risk Analysis up-to-date and Approved • Traceability Matrix Approved • Release Master Code available, with reproducible build • Release documentation Approved • SW Maintenance Plan Approved

Table 4

Within Construction phase, for each Design-Code-Test sequence, some sub-phase transition criteria are in place to guarantee that a proper sequence is followed. The following Table 5 shows these criteria:

Sub-Phase	Sub-Phase exit criteria
Design	For the software items to be implemented: <ul style="list-style-type: none"> • Design is complete and Approved • Software Requirements complete and Approved • Risk analysis exists
Code	For the software items to be implemented: <ul style="list-style-type: none"> • Code complete and reviewed according to plan • Unit test completed and documented • Code under SW Configuration Control • Anomalies documented and handled
Test	For the software items to be implemented: <ul style="list-style-type: none"> • Code Integrated and under SW Configuration Control • Integration testing completed and documented • Planned system test completed and documented • Planned regression test completed and documented • Anomalies documented and handled through Problem Resolution Process

Table 5

Deviations from the criteria indicated must be defined and justified for each project in the project Software Development Plan.

4.5 Metrics.

At a minimum trending of problems will be performed in each phase of development based on severity related to safety and effectiveness and number of unresolved problems at the end of each phase.

Any project specific metrics collection shall be defined in project Software Development Plan.

<< Add/Remove/Modify according to your process >>

5 Record Keeping

5.1 Design History File

During development, records will be maintained to demonstrate conformance with procedures and plans and to serve as the software design history. These records will be retained as part of (or by reference in) the overall product Design History File for the project. Such records will include:

- Approved revisions of the Software planning documents (SDP, SCMP, STP, etc)
- Reviewed versions of Software Requirements Specifications
- Reviewed versions of the Software risk/hazard analysis
- Reviewed versions of Software Design Specifications
- Revisions of software source code and build scripts for internal releases that were subjected to formal documented testing
- Records of formal technical reviews including documentation of issue resolution
- Formal test plans & test cases
- Test results from formal test execution
- Change control records starting at the point formal control is established for each item.

<< [Add/Remove/Modify according to your process](#)>>

5.2 Device Master Record

As part of release of software to production (i.e., design transfer) final release documents will be added (directly or by reference) to the Device Master Record for the product. Final versions of software documents to be part of the Device Master Record will include:

- Software Requirements Specifications
- Software Risk/hazard Analysis
- Software Design Specifications
- Software source code and build scripts (including versions of compilers & other required tools)
- Software manufacturing (replication, verification, & installation) procedure(s)
- Software Maintenance Plan

<< [Add/Remove/Modify according to your process](#)>>

5.3 Software Quality Manual

The Manager of Software Development will collect and organize all software related division and departmental procedures, guidance, relevant training materials, and sample documents into a software quality manual (SQM). This manual will include a table of contents and will be maintained up-to-date as procedures are revised.

<< [Add/Remove/Modify according to your process](#)>>

5.4 Record Retention

Record retention requirements for the DHF and DMR components are governed by the corporate record retention policy/procedure(s).

6 Training

All SW staff will be trained (or have prior experience) in relevant aspects of this procedure and associated procedures. The Software Quality Manual shall be available to them for reference as needed.

Everybody shall be notified when relevant procedures change.

Based on individual experience and role, specific training will be provided on the main tools used for SW development and testing.

Training may consist of self study and on the job by mentoring from experienced personnel based on staff qualifications and experience. Project specific training needs and approach will be defined in the SDP.
<< Add/Remove/Modify according to your process>>

7 External Supplier Management

If an outside vendor is used for software development and/or quality assurance the vendor's procedures will be reviewed to assure that they satisfy the requirements of this and related procedures. If the vendor's procedures are inadequate the project-specific Software Development plan will address these inadequacies.

In addition, a project and contract-specific procedure for change control and software transfer between the vendor and the company will be defined to explicitly defined the process and responsibilities of each party.
<< Add/Remove/Modify according to your process>>

8 SOUP Strategy.

If one or more SOUP items (either Commercial Off the Shelf or internally developed) are included in a project development:

- the Software Development Plan shall indicate all the SOUP items with manufacturing and revision information.
- the Software Configuration Management Plan shall indicate how SOUP items configuration shall be managed and shall state how maintenance of the SOUP items (e.g. revisions, patches) will be handled
- Software Integration planning shall include integration of SOUP items
- Software Risk Analysis and Risk Management shall include Analysis and management related to SOUP items, including hazardous situations caused by malfunction or unexpected behavior of SOUP items.
- Software Requirement Analysis shall take into consideration hardware and software requirements imposed by the use of SOUP items and shall specify functional and performance requirements of the SOUP items.
- Static (review) and dynamic (test) verification shall explicitly verify the SOUP items for their intended use.
- Known anomalies of SOUP items shall be evaluated for potential contribution to hazardous situations.
- The architectural and detailed design shall take into account SOUP items requirements in terms of hardware and software and shall specify functional and performance requirements of the SOUP items.

<< Add/Remove/Modify according to your process>>

SOFTWARE RISK MANAGEMENT PROCEDURE

TRAINING EXAMPLE

DRAFT

Revision History

Date	Revision	Author	Description

Note: This example is conceived as the body of a procedure.

Explanatory comments are included in << comment >>.

Other text is example definition that outlines an example of Software Risk Management Process.

In the appendix, an example template for Software Risk Analysis report is provided.

This is not a complete RA, just a training example to guide in the development of a RA for a certain type of device.

TABLE OF CONTENTS

1	Introduction.....	3
1.1	Purpose and Scope	3
1.2	Definitions.....	3
1.3	References.....	3
2	Roles and Responsibilities	3
3	Process Overview	4
4	Process Description.....	4
4.1	Risk Analysis	4
4.1.1	Identify Intended Use.....	4
4.1.2	Identify SW Related Hazards.....	4
4.1.3	Identify SW Causes.....	4
4.2	Risk Evaluation.....	4
4.3	Risk Control.....	5
4.4	Overall Residual Risk Evaluation.....	6
5	Report.....	7
6	APPENDIX: report template example.....	8
1	Introduction.....	10
1.1	Purpose.....	10
1.2	Scope.....	10
1.3	Definitions, acronyms, and abbreviations.....	10
1.4	References.....	10
1.5	Overview.....	10
2	Hazards and Risk Summary.....	11
3	Hazards Analysis	11
3.1	Specific hazards	12
3.1.1	Wrong parameter result presented to the user.....	12
3.1.2	Therapy dosage delivery speed too high.....	13
3.1.3	13
3.1.4	13
3.1.5	13
3.2	Indirect/Common Causes.....	14

1 Introduction

1.1 Purpose and Scope

This procedure defines the Software Risk Management process and activities within the <TBD> <<state the company, department, etc. to which this document applies. >>

This procedure describes the way software related risks will be identified, evaluated, classified, documented and controlled.

Software Risk Management is part of System Risk Management..... << describe how these two processes interrelate to each other. >>

This procedure will be used with risk analysis tools such as Hazard Analysis, Fault Tree Analysis and Failure Mode Effects and Criticality Analysis.

This procedure applies to:

- All new product software developments
- All software changes through product maintenance
-
-

<<Detail applicability>>

This procedure does not apply to: << If necessary, detail where the procedure is not applicable >>

Intended audience for this document are.....<<Define intended audience>>

1.2 Definitions

For department common definitions, please refer to <TBD> Common Definitions.

Additional specific terms are defined in **Table 1** **Error! Reference source not found.** below.

ALARP	Acronym for “As Low As Reasonably Practicable”
Failure Path	The causes or combinations of causes that can lead to the top level event (in the case of an FTA) or the event of interest, i.e. results in hazard.
Foreseeable Misuse	Use of the medical device in a manner the manufacturer did not intend but could have reasonably predicted as a consequence of human behavior.
Residual Risk	Risk remaining after protective measures have been taken
Risk Control Effectiveness (RCE)	Capability of Risk Control Measure to reduce or eliminate the risk of the hazard it is meant to mitigate.

Table 1

<< Define any term used in the document that may be unfamiliar to readers. >>

1.3 References

ISO 14971:2007, Medical devices— Application of risk management to medical devices.

IEC 62304:2006, Medical device software—Software life cycle processes.

AAMI TIR32:2004, Medical device software risk management. - << replace with IEC 80002 when released >>

2 Roles and Responsibilities

The QA Manager is responsible for supervising the application of this procedure to every project.....

The Project Manager of each project is responsible to ensure that Risk Management activities are properly performed in every phase of the software development and maintenance lifecycle.....

<< Define specific responsibilities for application of the procedure and approvals >>

3 Process Overview

The risk management process is composed of 4 main phases:

- Risk analysis to identify the potential risks and the causes that could lead to hazardous situations.
- Risk evaluation to evaluate the possible consequences of risks
- Risk control to define and apply mitigations that allow to reduce the risks
- Residual risk evaluation to estimate the levels of risk after applying the mitigations and decide on its acceptability.

The process is repeated cyclically during all the software lifecycle phases and is reapplied after changes or anomaly fixes.....

[<< Modify/Expand to describe the entire process. >>](#)

4 Process Description

4.1 Risk Analysis

4.1.1 Identify Intended Use

Describe <device> software intended use and software foreseeable misuse.

4.1.2 Identify SW Related Hazards

Compile a list of all known and foreseeable hazards associated with <device> software in both normal and abnormal (faulty) conditions.

These could be derived from the hazards identified in the <device> Risk Analysis, applying a filter to isolate those that could be software related.

Additional specific hazards can be investigated and added to the list, by using a variety of sources, including:

- Historical field information for similar products such as review of complaints.
- Software related Medical Device Reports (MDRs) and product recalls reported to the FDA
- Standards and guidance documents, such as TIR32
-

[<< Detail the process and specify all the sources to consider >>](#)

4.1.3 Identify SW Causes

Once the hazards for the intended use have been identified, evaluate the possible software related causes for each of them. These could be software causes, or hardware causes and use causes handled by software. In this sense it is important to identify the chain of events that can lead to the harm to occur.

Fault Tree Analysis tool could be used to help identifying the possible failure paths.

An adequate number of software engineers and system engineers should be involved in this analysis so that multiple levels of analysis occur, ranging from the hardware interface layer to the user interface layer.

TIR32 Annex A could be used to help in this analysis.

In addition to function-specific failure causes, software indirect/common causes for the specific application must be taken into consideration. These are software initiating causes that could lead to hazards through an unpredictable chain of events. TIR32 Annex B could be used to help in this analysis.

SOUP items unexpected failures shall be included in this analysis. If SOUP unexpected behavior could be a contributor to the risk, published anomaly lists shall also be examined.

[<< Detail the methods used for identifying SW causes. >>](#)

4.2 Risk Evaluation

Risk evaluation will be an iterative process. Risk will be evaluated prior to any risk control measures being applied and then re-evaluated any time a risk control measure is identified and applied.

For software related causes initial risk evaluation shall only be based on severity of the risk, while probability of occurrence is not considered, because software errors are considered systematic.

In subsequent evaluations after application of Risk Control Measures, reductions in likelihood of harm based on the effectiveness of the risk control measures taken are utilized based on risk control effectiveness rationales as explained below.

<< Further detail according to your process. >>

The severity is evaluated according to the following table: << this is just an example, replace with your own levels. Severity levels could also be unique to the intended use of the device either by clarification of types of harm in the definitions or in the severity levels themselves (e.g., radiation overdose, infection, minor burn, temporary cosmetic ...) >>

Severity Rank	Severity Level	Severity Description
1	Negligible	No potential for direct or indirect harm.
2	Marginal	Potential of minor direct or indirect harm not necessitating medical intervention. May cause reversible damage to the system, reagents or consumable materials, other property, or the environment.
3	Serious	Potential for significant direct or indirect harm necessitating medical intervention. May cause major, irreversible damage to the system, reagents or consumable materials, other property, or the environment.
4	Critical	Potential death, serious injury, or serious deterioration in state of health.

Table 2

4.3 Risk Control

For each cause that requires risk reduction, appropriate risk control measures will be identified and applied. Risk control measures can be applied at different stages of the casual chain, therefore could be different for different causes and sub-causes, or sometimes, if applied to the last point of control, could be common to several causes and sub-causes.

Again, Fault Tree Analysis could be used as a powerful tool to identify risk control measure and apply them in the most effective point in the failure path.

Depending on the type of risk control measures applied their effectiveness in reducing the risk will be different.

The most effective risk control measures are those that are “inherently safe,” that is, the possibility of the hazard is eliminated altogether. Early in the product concept phase of the project, the designers should consider the potential hazards and how they might be eliminated through inherent design. An example could be only use of static memory allocation to eliminate the possibility of memory fragmentation or memory leaks.

When inherently safe design is not possible, design or process mitigations will be applied. General mitigation will be applied to indirect/common causes, where the casual chain of events is not defined, while specific mitigations will be applied to specific causes and failure paths. Effectiveness of the mitigations will depend on the type of mitigation and on its reliability. Diverse RCMs are more effective than simple RCM. Diversity depends on the specific design: in some case could be use of a different processor to implement the mitigation, in other cases could be execution of the main code and of the controlling code as totally independent processes on the same processor so that one cannot affect the other.

Another level of mitigation is labeling and training. By providing the user with instructions for use, warning, cautions, training, and other information, the user can become a key element in reducing the risk,

but the goal should always be to reduce risk prior to the device's last point of control. Effectiveness of this type of mitigations should always be evaluated with the assistance of medical/clinical experts.

Multiple risk control measures can sometimes combine to increase risk control effectiveness (RCE).

Software failures being analyzed will be assigned a RCE rating in order to determine residual risk and risk acceptability.

The following table can be used to evaluate the RCE. << this is just an example, replace with your own levels >>

RCE Rank	RCE Level	RCE Description
1	Safe Design	Inherently Safe Design – failure can not happen
2	Diverse RCM	Diverse RCM(s) considered highly reliable (cannot fail in a latent manner), independent, redundant, highly effective.
3	Simple RCM	Effective but non-diverse RCM.
4	Labeling	Only labeling, training and/or clinical practice to mitigate the concern.

Table 3

Each risk control measure will be verified to be correctly designed and implemented and will be traceable to one or more requirements in the requirement specification and to one or more test procedures or test cases.

Risk Control Measures shall be further analyzed to verify they do not introduce new failure paths. In case they do, further analysis shall be applied to document the additional causes and to verify if additional controls need to be introduced

<<Add/Replace as needed with your process >>

4.4 Overall Residual Risk Evaluation

The residual risk evaluation is an iterative process. For each failure path (cause and sub-causes), residual risk will be evaluated after every risk control measure is applied.

The RCE residual risk acceptability can be evaluated according to the following table: << this is just an example, replace with your own levels >>

Severity>	Negligible	Marginal	Serious	Critical
RCE				
Safe Design	Acceptable	Acceptable	Acceptable	Acceptable
Diverse RCM	Acceptable	Acceptable	Acceptable	Further Evaluation
Simple RCM	Acceptable	Acceptable	Further Evaluation	Further Evaluation
Labeling	Acceptable	Further Evaluation	Further Evaluation	Further Evaluation

Table 4

For items that fall in the cautionary “Further Evaluation” Category attempts will be made to identify alternative risk control measures to result in the Acceptable range. If this is not practicable then the acceptability of the residual risk will be evaluated based on the overall device risk analysis which includes the likelihood that if such failures occurred the hazardous situation would occur and if so then the likelihood of actual harm occurring as part of the resulting residual risk rating.

A final overall residual risk acceptability will be determined and documented. This will be based on all the different risk control measures applied (although each one may not be fully effective in itself, they can combine to result in a very effective risk mitigation) and will also take into account external factors, such as intended use and common clinical practice.

Whenever residual risk acceptability does not result immediately Acceptable according to the above table, a detailed justification for final acceptability of the risk is required.

<< This last portion of the process could be part of the overall device risk analysis and could be omitted from the Software Risk Analysis, or could be used to support the overall device risk analysis. >>

5 Report

A Software Risk Analysis report will be compiled as a result of this process.

This will be a document growing through the phases of the process.

An example template for this document is provided as an attachment to this procedure.

The main features in the report will be the list of the identified hazards, and, for each hazard, a table illustrating causes, sub-causes, risk control measures, and risk evaluation.

Each table could be structured with the following columns, as illustrated in the template:

- **Cause** Various ways the hazard can be caused; these are high-level causes.
- **Sub-Cause** This column lists various ways the hazard can be caused at a more detailed level: indicate here the initiating cause, or the chain of events that could lead to the high level cause and the hazard. If fault tree has been used to document failure paths, reference here the branch illustrating the chain of events.
- **Severity** Risk Severity level according to [Table 2](#) above
- **Internal RCM** This column lists various controls built into the device or its software that control or mitigate the hazard. These could include also labeling (manuals, labels, user screens) and specific user training.
- **Traceability** This column has linkage information either to the requirement implementing the RCM or to the verification test verifying its implementation (or both). << alternatively a separate trace table could be maintained >>
- **RCE (risk control effectiveness)** For each risk control an effectiveness rate attributed according to [Table 3](#) above.
- **Initial Residual Risk rating** Acceptability level of risk mitigation for any single RCM, according to [Table 4](#)
- **External RCM** These are related to the actual way the instrument is expected to be used. Could be accepted clinical practice or intended use limitation.
- **Final Residual Risk Rating** This is the acceptability level of risk mitigation assigned to each sub-cause. It takes into account all the different internal RCMs applied and the external RCMs to evaluate the overall effectiveness of all mitigations applied. Even if none of the single mitigations taken by itself could be sufficient to reduce the risk to an acceptable level, they could when considered all together to increase effectiveness.
- **Justification for acceptance** The rationale to accept the residual risk without further mitigation. Required only if none of the internal RCMs leads to Initial Residual risk rank = ACCEPTABLE.

6 APPENDIX: report template example.

SOFTWARE RISK ANALYSIS REPORT

TEMPLATE

DRAFT

Revision History

Date	Revision	Author	Description

Note: This template is conceived as a partial example template for a generic small device with embedded real time control. Explanatory comments are included in << comment >>.
Other text is example definition that you should replace with your own text.

TABLE OF CONTENTS

<< Insert Report TOC >>

1 Introduction

1.1 Purpose

Purpose of this Software RISK ANALYSIS REPORT is to describe the risk analysis and risk control activities performed related to the software of <device> . <<state the version of the software to which this document applies. >>

1.2 Scope

The <device> is meant to<< Describe intended use of the device, >>

The software in the <device> executes the following tasks..... <<Describe shortly the main software tasks and software intended use >>

This report documents the following activities related to the software of <device> :

- Risk analysis.
- Risk evaluation.
- Risk control.
- Residual risk evaluation

This report is limited to software; it does not document any of the hardware or overall device risk analysis activities.

<<The information in this report could be incorporated into the overall device risk analysis report if desired.>>

1.3 Definitions, acronyms, and abbreviations

RA	Risk Analysis
RC	Risk Control
RCE	Risk Control Effectiveness
RCM	Risk Control Measure

<< Define any acronyms, abbreviations, or terms used in the document that may be unfamiliar to readers. If a project level definitions document exists, it can be referenced here and this section limited to specific terms used only in this document. >>

1.4 References

Standards:

ISO 14971:2007, Medical devices— Application of risk management to medical devices.

IEC 62304:2006, Medical device software—Software life cycle processes.

AAMI TIR32:2004, Medical device software risk management. - << replace with IEC 80002 when released >>

.....

<< Include any **relevant** standard >>

Project Documents:

<device> Software Requirement Specifications, Rev. x.y,<<you could add any information useful to locate the document>>

.....

<< Include **relevant** project documents such as device risk analysis, device and software risk management procedures, software architecture/design specs >>

1.5 Overview

This document is organized in 3 main sections.

Section 1 (this section) puts the document into its context and gives an overview of it.

Section 2 gives a high level summary of the identified software risks.

Section 3 details the risk analysis for each identified risks and for the software indirect/common causes considered.

<< Add anything apt to describe content and organization of this document >>

2 Hazards and Risk Summary

- Wrong parameter result presented to the user.....
 - Therapy dosage delivery speed too high
- << State the main risks of the device and as the project progresses adding how these are handled. These are explained in a general way, and generally how they are mitigated (within the device or externally once these are defined). >>

3 Hazards Analysis

The tables below describe in detail the causes and methods of control for each software hazard.

Each cause, sub-cause and RCM are numbered as Cn , $SCn.n$, RCn , where n is a consecutively assigned number.

<< The following are just examples of possible hazards in a medical device. Replace with your own. Consider also possible failures of SOUP components >>

3.1 Specific hazards

3.1.1 Wrong parameter result presented to the user.

Cause	Sub-Cause	Severity	Internal RCM	Traceability	RCE	Initial Residual Risk rating	External RCM	Final Residual Risk Rating	Justification for acceptance.
C1 – The value stored in Patient DB is wrong	SC1.1 The value was correctly stored, but got corrupted after storing.	Critical	RC1 – Multiple (3) copies handled by different process for critical info.	SRS111 TC063	Diverse RCM	ALARP	The clinical practice is never to use single parameter data...	Acceptable	Highly effective risk control measures including multiple methods of error detection together with clinical practice/labeling warning
			RC2 – CRC on data stored in Patient DB.....	SRS1234 TC159	Simple RCM	Not Acceptable			
			RC3 – Check on defined limits when retrieving data from DB.....	SRS2468	Simple RCM	Not Acceptable			
			RC4 -			
	SC1.2 Wrong A/D conversion...	Critical	RC5 – Periodic check on A/D using known reference values....	SRS3721	Simple RCM	Not Acceptable	The clinical practice is never to use single parameter data...
			RC6 -			
C2 – Malfunction in data formatting algorithm during presentation									

<< Complete with all identified causes and all identified paths and RCMs. >>

3.1.2 Therapy dosage delivery speed too high.

Cause	Sub-Cause	Severity	Internal RCM	Traceability	RCE	Initial Residual Risk rating	External RCM	Final Residual Risk Rating	Justification for acceptance.

3.1.3

Cause	Sub-Cause	Severity	Internal RCM	Traceability	RCE	Initial Residual Risk rating	External RCM	Final Residual Risk Rating	Justification for acceptance.

3.1.4

3.1.5

3.2 Indirect/Common Causes

The following table lists the software failure causes that are not tied to specific functionality of the device and individual hazards. They can cause unpredictable effects:

Cause	Sub-Cause	Severity	Internal RCM	Traceability	RCE	Initial Residual Risk rating	External RCM	Final Residual Risk Rating	Justification for acceptance.
C33 - Arithmetic error	SC33.1 Division by zero	Critical	RC33 - Error reported by arithmetic coprocessor		Safe Design << Provided error is correctly handled >>	Acceptable			
	SC33.2 Numeric Overflow	Critical	RC34 – Range checks are used		Simple RCM				
			RC35 - User manual indicate to verify all results outside limits.....		Labeling				

<< Complete this table with all the common causes that have been considered in software risk analysis. Use Annex B of AAMI TIR32 for reference. Explicitly consider also possible failures of SOUP components (e.g. operating system failures) >>

SOFTWARE RELEASE PROCEDURE

TRAINING EXAMPLE

DRAFT

Revision History

Date	Revision	Author	Description

Note: This example is conceived as the body of a procedure.

Explanatory comments are included in << comment >>.

Other text is example definition that outlines an example of Software Maintenance Process.

This is not a complete SOP, just a training example to guide in the development of a Software Release SOP.

TABLE OF CONTENTS

1	Introduction.....	3
1.1	Purpose and Scope	3
1.2	Definitions.....	3
1.3	References.....	3
2	Roles and Responsibilities	4
3	Process Overview	4
4	Process Description.....	4
4.1	Creating Master Executable Packages	4
4.2	Release Notes.....	5
4.3	Manufacturing Replication and Installation Procedure	5
4.4	Training.....	5
4.5	Special Handling for non-customer versions.....	5
5	Record Keeping	5

1 Introduction

1.1 Purpose and Scope

This procedure defines the specific requirements for the design transfer/release of software to production, integrating the requirements defined in the Software Development SOP and the Software Maintenance SOP. This applies to initial product release and subsequent maintenance releases. The purpose is to ensure that adequate controls are in place to:

- prevent inadvertent release of software versions not intended for production
- prevent inadvertent release of software versions that have not been subject to adequate verification and validation
- ensure adequate quality control in the creation of master copies provided to manufacturing
- ensure software aspects of the Device Master Record are up-to-date
- ensure software media is appropriately controlled, delivered, and labeled to prevent mix-ups.

This procedure is intended to integrate with, and support, general design control requirements for product release overall.

This procedure applies to all software embedded in products designed by the <TBD> division to use in units to be shipped to the customer. It does not apply to software used to support manufacturing or the quality system.

<<Detail applicability and extension of this procedure>>

Intended audience for this document will be the R&D and QA personnel involved in handling a software release and the Production personnel involved in receiving the SW release and in handling SW duplication and installation on the product.

This procedure will also be used by the divisional managers as a tool to agree and control software release activities coordinating them with overall system release activities..

<< Describe the expected users of this procedure. >>

If necessary, details on single product release or specific tailoring of the process described in this procedure shall be defined in the project specific Software Configuration Management Plan.

1.2 Definitions

For department common definitions, please refer to <TBD> Common Definitions. Additional specific terms are defined in Table 1 below.

SCMP Software Configuration Management Plan

Table 1

<< Define any term, acronym or abbreviation used in the document that may be unfamiliar to readers. >>

1.3 References

Standards:

IEC 62304:2006, Medical device software—Software life cycle processes.

.....

<< Add any relevant standard >>

Procedures:

SW Development SOP

SW Maintenance SOP

SW Risk Management SOP

SW Problems Tracking SOP

.....
<< Add any relevant SOP, including the Product Development and Design Control SOPs >>

2 Roles and Responsibilities

The mandatory steps identified in this procedure are responsibility of the SW Engineering and SW Quality Staff assigned to release and of the production staff assigned to software installation and quality control. The Manager of Software Development is responsible for verifying the application of this procedure.
<< Define specific responsibilities for application of the procedure and approvals >>

3 Process Overview

The division ECN and <TBD new product release> procedures define general requirements for release to manufacturing and apply to software as well as hardware and labeling. No product-embedded software is to be released to manufacturing without adhering to the requirements of these procedures.

The software release process shall imply creating master copies of the software to release on the appropriate media, transferring the master copies to manufacturing together with the release documentation and updating the Device Master Record.

In case of first release of a new product, the release process shall include also defining the software replication and installation procedure and providing adequate training to operation personnel.
<< Add/Remove/Modify according to your own process >>

4 Process Description

4.1 Creating Master Executable Packages

The delivery media for the executable package will be defined in the project specific Software Configuration Management Plan.

R&D shall produce in 3 copies of the executable package on the delivery media: first copy to deliver to Operations, second copy to deliver to document control, third copy to archive in R&D.

<< Define your own policy of master software distribution, detail if necessary how the copies are to be produced. >>

Labels to be applied to the delivery media shall include SW Package name, SW version and build number. It will also include distribution identifier, which will be different on each of the 3 copies: OP on the copy sent to operation, DC on the copy sent to Document Control, RD on the copy stored in R&D.

<< Define your own labeling requirements. >>

Any equipment used to create master copies needs to be properly identified, calibrated if the case (e.g. prom programmers) or validated for proper operation.

Each one of the three copies will be verified by simulating software installation on a unit according to a specific test protocol.....

<< Define your own verification procedure, that could include verification of the checksum, verification that the copies are identical, verification of installation on a new CPU, verification of installation on a CPU that had already a different version installed, etc. depending on the type of hardware and software and what makes sense to test. >>

4.2 Release Notes

In addition to any documents defined in the division ECN and <TBD new product release> procedures, Software Release Notes shall be included in the release package as requested by the specific Software Configuration Management Plan for any software release. The release notes shall provide details on the content of the release and its verification and validation. It shall also provide a list of known residual anomalies present in the release.

<<Add/Remove/Modify according to your own policy >>

4.3 Manufacturing Replication and Installation Procedure

For any new product release, a procedure will be provided for use by manufacturing (and/or vendors) to replicate, verify, and install copies of the software in production units. For maintenance releases this procedure will be modified if necessary.

<< There are cases where just replication is required (e.g. if software is loaded from a CD at any <device> switch on), cases where only installation is required (e.g. if firmware is installed into flash memory in manufacturing through a specific procedure) or both are required (e.g. firmware is installed on-board, but also provided as backup on disk).>>

The procedure shall address how copies need to be labeled and verified, how the software shall be installed and correctness of installation shall be verified, how to keep track of copies distributed and/or installations performed.

<< This is important to keep track of software versions in the field >>

If royalties need to be paid for any OTS software item included in the release package, the procedure shall also indicate how this needs to be done and traced.

4.4 Training

For any new product release, training to manufacturing personnel shall be planned to cover the replication and installation procedure and the verification and validation of the installation.

<<Add/Remove/Modify according to your own policy>>

4.5 Special Handling for non-customer versions

If it is necessary for development to provide test versions for use on the manufacturing floor as part of verification and validation plans, special care shall be used to avoid that any such version is erroneously shipped to the customer:

- The Manager of Software Development must approve delivery of such test versions.
- The media used and the devices on which the test release is installed must be clearly labeled as “NOT FOR RELEASE TO CUSTOMERS” or “FOR INVESTIGATIONAL USE ONLY”
- After testing is completed the media must be removed from manufacturing and all units on which it was installed must be decommissioned or returned to the appropriate released version.

<< Add/Remove/Modify according to your own policy. Other more rigorous controls could be added. >>

5 Record Keeping

For new product release the Device Master Record must include (physically or by reference to) approved versions of all required software documentation (as specified in the Software Development SOP and the project-specific Software Development Plan) prior to, or concurrent with, release of the executables to manufacturing. If the Device Master Record references, but does not physically include, any of the software items, those items must be under appropriate configuration management and document controls to ensure the integrity and accuracy of the total Device Master Record and the reference must be specific enough to permit unambiguous determination of the location of the exact items (e.g., specific code versions) referenced.

For maintenance releases all software specifications that are part of (physically or by reference to) the Device Master Record that are affected by software changes are to be updated and approved prior to, or concurrent with, release of executables to manufacturing.

SOFTWARE MAINTENANCE PROCEDURE

TRAINING EXAMPLE

DRAFT

Revision History

Date	Revision	Author	Description

Note: This example is conceived as the body of a procedure.

Explanatory comments are included in << comment >>.

Other text is example definition that outlines an example of Software Maintenance Process.

This is not a complete SOP, just a training example to guide in the development of a Software Maintenance SOP.

TABLE OF CONTENTS

1	Introduction.....	3
1.1	Purpose and Scope	3
1.2	Definitions.....	3
1.3	References.....	3
2	Roles and Responsibilities	4
3	Process Overview	4
4	Process Description.....	4
4.1	Handling of external problem reports	4
4.2	Handling of internal problem reports.....	5
4.3	Handling of SOUP	5
4.4	Change Control	5
4.5	Change Verification and Validation	6
4.6	Change Release	6
5	Record Keeping	6

1 Introduction

1.1 Purpose and Scope

This procedure defines the specific requirements for the maintenance of released software, integrating the requirements defined in the Software Development SOP and the Software Release SOP.

This procedure is intended to integrate with, and support, general design control requirements for product maintenance overall.

This procedure applies to all software embedded in products designed by the <TBD> division, after it has been released to production to use in units to be shipped to the customer. It does not apply to software used to support manufacturing or the quality system.

<<Detail applicability and extension of this procedure>>

Intended audience for this document will be the personnel involved in handling of complaints and requests coming from the field and R&D and QA personnel involved in the support for released products.

This procedure will also be used by the divisional managers as a tool to agree and control software maintenance activities coordinating them with overall released products support.

<< Describe the expected users of this procedure. >>

If necessary, details on single product maintenance or specific tailoring of the process described in this procedure shall be defined in the project specific Software Development Plan and/or Software Maintenance Plan.

1.2 Definitions

For department common definitions, please refer to <TBD> Common Definitions.
Additional specific terms are defined in Table 1 below.

SMP Software Maintenance Plan

Table 1

<< Define any term, acronym or abbreviation used in the document that may be unfamiliar to readers. >>

1.3 References

Standards:

IEC 62304:2006, Medical device software—Software life cycle processes.

IEC 60601-1:2005 Medical Electrical Equipment – Part 1: General requirements for basic safety and essential performance.

AAMI TIR32:2004, Medical device software risk management. - << replace with IEC 80002 when released >>

.....

<< Add any relevant standard >>

Procedures:

SW Development SOP

SW Release SOP

SW Risk Management SOP

SW Problems Tracking SOP

.....

<< Add any relevant SOP, including the Product Development and Design Control SOPs >>

2 Roles and Responsibilities

The mandatory steps identified in this procedure are responsibility of the SW Engineering and SW Quality Staff assigned to product support.

The Manager of Software Development is responsible for verifying the application of this procedure and for control and maintenance of the software portions of the Design History File and the Device Master Record.

<< Define specific responsibilities for application of the procedure and approvals >>

3 Process Overview

The Software Maintenance process defined in this procedure is the portion specific to Software of the entire Design Control Process for product maintenance of <TBD> division.

<< Describe here how the software process is integrated into the design control process for product maintenance. >>

Software maintenance will mainly consist of evaluating problem reports and change requests related to released software products, decide on their acceptance or rejection, planning their implementation and release, implementing changes according to the plans and keeping project documentation up-to-date.

Small maintenance software releases shall be managed through a simplified process not requiring a specific Software Development Plan, while major software releases, including several changes and fixes, shall be handled as a full software project and shall be subject to the entire lifecycle requirements defined in the Software Development Procedure.

4 Process Description

4.1 Handling of external problem reports

All field problems reported via the complaint handling procedures shall be investigated and the ones that could potentially be attributed to software will be reported to the Manager of Software Development (or person designated to handle product support).

All received complaints shall be tracked in a specific complaints database, logging at least date, source of the complaint, product attribution, software revision and description of the problem.

After an initial investigation, the complaints that are identified as real software problems shall be entered in the Problem Tracking System. In the problem tracking system a reference shall be maintained to the original complaint(s) so that there is traceability between the software problem report log and the corporate and division complaint files and it is possible to follow through and track the complaint to closure.

Complaint that, after initial investigation, are not attributed to software problems or are considered totally irrelevant, shall not be entered into the problem tracking system, but shall be rejected. The reason for rejection shall be added to the complaint database and corporate complaint handling shall be informed.

<< Add/Remove/Modify according to your own policy. In some cases, it could be avoided to have a field complaints database separated from the Problem Tracking System. One reason to keep this initial separation is that often complaints attributed to software after initial investigation turn out not being software related or being irrelevant. Another reason is that often several complaints are similar and can be investigated altogether as a single anomaly, so it is not worthwhile to fill the Problem Tracking System with many identical records without applying a preventive filter.

In any case, the requirement here is to have a mechanism to feed the externally received complaints to the Problem Tracking System and a mechanism to ensure that each complaint is tracked to its disposition and closure. >>

4.2 Handling of internal problem reports

Internally submitted problem reports for released software are handled through the Problem Tracking System.

They will be evaluated in terms of potential for safety and efficacy impact in the field. If the potential malfunction could affect safety or efficacy this will be escalated to determine if any field action (e.g., a recall) is required or if Medical Device Reports need to be submitted to the FDA.

<< Add/Remove/Modify according to your own policy.>>

4.3 Handling of SOUP

The life of SOUP items included in a product shall be monitored. For Off –the-Shelf commercial product this shall include frequent checks on published anomaly lists, availability of new releases, patches and service packs.

If, as result of the above checks, it is required un upgrade of the SOUP item or it is necessary to implement additional risk control measures to address possible anomalies of the SOUP item, a change request or an anomaly report shall be filed and shall be addressed by Change Control in the same way as all other change requests and problem reports.

<< Add/Remove/Modify according to your own policy.>>

4.4 Change Control

A Change Control Board (CCB) shall be in charge of evaluating Change Requests and Anomaly Reports for released products. Components of the board shall be defined for each product and shall include representatives of R&D and QA management, as well as the Manager of Software development. Technical staff can be invited to the board as needed. The board shall decide on changes and defects investigation and propose what changes and fixes will grant a new software release. An Engineering Change Request (ECR) shall be used as the standard mechanism to plan for change release: a single ECR can include several changes and fixes, provided that they are individually described.

Software design control activities apply to changes made after release, as well as to initial development, as described in the Software Development Procedure. This includes risk analysis, updating requirements and design specifications and use of Software Configuration Management process for the implementation of the changes.

Risk analysis for changes will include a review of the baseline analysis to determine if:

- New intended use hazards are being introduced
- New casual chains for existing potential hazards are being introduced
- Hazard mitigations could be affected
- Any new hazard mitigations are needed due to the change.

The results of this analysis will be used as input to the requirements, design and V&V planning for the change and associated regression testing as well as to determination of any regulatory action (e.g. the need for a new 510(k)).

If the software product is part of a bigger system or is or can be interfaced to other products or systems, the analysis of the change shall include analysis of possible impacts of the change on other parts of the system or on interfaces. This will be taken into account in the decision to proceed with the change and will eventually indicate the need to extend the information on the change itself to other organizations.

If major enhancements are required or changes and fixes are extensive then the associated software development work should be performed as a major project and a Software Development Plan is required.

<< Add/Remove/Modify according to your own handling of change requests for released products.>>

4.5 Change Verification and Validation

Reviews and testing applied will depend on the specific changes and fixes implemented, on their complexity and potential impact on system functionalities, safety and efficacy.

Change documentation, including design and code, shall be reviewed singularly or several changes can be reviewed all together in a single review.

Every change shall be tested using specifically developed test cases or original test cases modified to include the testing of the change. If several changes are implemented in the same integration, integration testing shall be applied to ensure that they do not interact in problematic ways.

All releases shall be subject to appropriate regression testing, based on evaluation of the impact of the changes. Main focus of regression testing shall be on safety-critical aspects of the software and to confirm that failure modes are still handled properly.

All formal testing is to be performed according to documented protocols and results of testing are to be recorded and these records are to be retained according to record keeping policies.

<< Add/Remove/Modify according to your own V&V policies.>>

4.6 Change Release

Divisional and Corporate management shall decide on the plans to support the maintenance of the product with software releases and what changes and fixes will be incorporated in each release. Frequency and consistency of new software releases shall be based on amount and urgency of changes and fixes, also related to the severity of the anomalies to be handles in a release.

Release process shall be in accordance with Software Release SOP.

Release information to the customers shall be according to the corporate procedures.

<< Add/Remove/Modify according to your own release policies.>>

5 Record Keeping

The DHF shall be updated as needed during the implementation of the maintenance changes and fixes with the most recent approved copies of all the documents that have been updated as part of the change process.

As part of release of software to production final release documents will be added (directly or by reference) to the Device Master Record for the product. The list of documents to add to the DMR is the same as for the original release. See the Software Development SOP for details.

<< Add/Remove/Modify according to your own record keeping policies.>>

SOFTWARE DEVELOPMENT PLAN

TEMPLATE

DRAFT

Revision History

Date	Revision	Author	Description

Adaptation of IEEE Std. 1058-1998 IEEE Standard for Software Project Management Plans and IEEE Std.730-2002 IEEE Standard for Software Quality Assurance Plans.

Note: This template is conceived as a partial example template for a generic small device with embedded real time control. Explanatory comments are included in << comment >>.

Other text is example definition that you should replace with your own text.

This is not a complete Software Project Management Plan, just a training example to guide in the development of a Software Project Management Plan for a certain type of device.

TABLE OF CONTENTS

1	Introduction.....	4
1.1	Project Summary.....	4
1.1.1	Purpose, Scope and Objectives	4
1.1.2	Project deliverables	4
1.2	Audience	4
1.3	Definitions, acronyms, and abbreviations.....	4
1.4	References.....	4
1.5	Document Overview	5
2	Project Organization	6
2.1	Organizational Structure	6
2.2	Interfaces.....	6
2.3	Roles and Responsibilities	7
3	Managerial Process	8
3.1	Objectives and Priorities	8
3.2	Assumptions, dependencies and constraints	8
3.3	Resources Management - OPTIONAL	8
3.3.1	Estimation	8
3.3.2	Staff.....	8
3.3.3	Other resources	8
3.3.4	External Suppliers Management	8
3.3.5	Training.....	9
3.3.6	Budget.....	9
3.4	Work Plan	9
3.4.1	Schedule.....	9
3.4.2	Integration Plan.....	9
3.5	Monitoring and controlling mechanisms – OPTIONAL	10
3.5.1	Requirements Control	10
3.5.2	Schedule Control.....	10
3.5.3	Budget Control.....	10
3.5.4	Quality Control	10
3.5.5	Metrics	10
3.5.6	Reporting.....	11
3.6	Project Risk Management – OPTIONAL	11
3.7	Regulatory Plan.....	11
3.8	Project Closeout Plan – OPTIONAL	12
4	Technical Process	13
4.1	Process Model.....	13
4.1.1	Software Lifecycle	13
4.1.2	Activities	13
4.1.3	Legacy Software - OPTIONAL	13
4.1.4	Phase transition requirements.	13
4.1.5	Tasks and assignments.....	13
4.2	Methods, Tools and Techniques	14

4.3	Infrastructure.....	14
5	Quality Plan	15
5.1	Standards, Practices and Conventions	15
5.2	Documentation Plan.....	15
5.2.1	Input to Software Development	15
5.2.2	Output from Software Development.....	16
5.3	Reviews and Audits	20
5.4	SOUP and OTS management.....	20
5.5	Verification and Validation.....	20
5.6	Code and Media Control	21
5.7	Record Collection, Maintenance and Retention	21
6	Supporting Process Plans.....	22
6.1	Configuration Management	22
6.2	Problem Resolution.....	22
6.3	Software Risk Management.....	22
6.4	Traceability	22
6.5	Process Improvement.....	22
7	Maintenance Plan.....	23

1 Introduction

1.1 Project Summary

1.1.1 Purpose, Scope and Objectives

The purpose of this project is to develop a new <TBD> analyzer, based on <TBD> technology and delivering <TBD> results that are clinically used for.....

The <TBD> analyzer will be part of a set of analyzers used for and as such will be integrated into.....

Execution of <TBD> types of analysis is outside the scope of this analyzer.....

<< Detail what is the main purpose of the project, what is within its scope and what is outside of its scope. The statement of scope needs to be consistent with similar statements in the project agreement and other relevant system-level documents, such as the product specification. >>

1.1.2 Project deliverables

The project will deliver the following features:

-
-
-

<< Describe what are the main deliverables of the project. If a phased release is planned, list features for each planned release. >>

1.2 Audience

Intended audience for this document will be all the personnel involved in software development activities for this project, in particular the SW development team and the SW Test team.
This plan will also be used by the Project Manager, the QA Manager and the Product Manager as a tool to agree and control software development activities throughout the project life span.

<< Add/Modify/Replace as fitting to your own purpose and organization. >>

1.3 Definitions, acronyms, and abbreviations

RA	Risk Analysis
RCM	Risk Control Measure
SDP	Software Development Plan
SRS	Software Requirement Specification

<< Define any acronyms, abbreviations, or terms used in the document that may be unfamiliar to readers. If a project level definitions document exists, it can be referenced here and this section limited to specific terms used only in this document. >>

1.4 References

IEC 62304:2006, Medical device software—Software life cycle processes.
IEC 60601-1:2005 Medical Electrical Equipment – Part 1: General requirements for basic safety and essential performance.
IEEE Std. 1058-1998 IEEE Standard for Software Project Management Plans.
IEEE Std.730-2002 IEEE Standard for Software Quality Assurance Plans.

.....
<< Include any **relevant** standard >>

Software Development SOP
Software Risk Management SOP
Software Problem Management SOP
Software Maintenance SOP
Software Release SOP

.....
<< Include any relevant procedure >>

1.5 Document Overview

This document is organized in 7 main sections.

- Section 1 (this section) puts the document into its context and gives an overview of it.
- Section 2 describes the project organization, including main interfaces and roles and responsibilities.
- Section 3 describes project management, including resources management, main milestones, monitoring and controlling tasks, project risks management and regulatory activities.
- Section 4 defines the SW development process, including SW lifecycle, activities, tasks and their assignments, methods and tools used.
- Section 5 describes the Quality activities, including V&V activities, standards used, documentation control, code and media control.
- Section 6 describes the supporting processes: Configuration Management, Problem Resolution, Software Risk Management, Traceability, Process Improvement
- Section 7 describes maintenance after release

<< Add anything apt to describe content and organization of this document >>

2 Project Organization

2.1 Organizational Structure

The following organizational chart defines the structure of the teams involved with <device> software development.

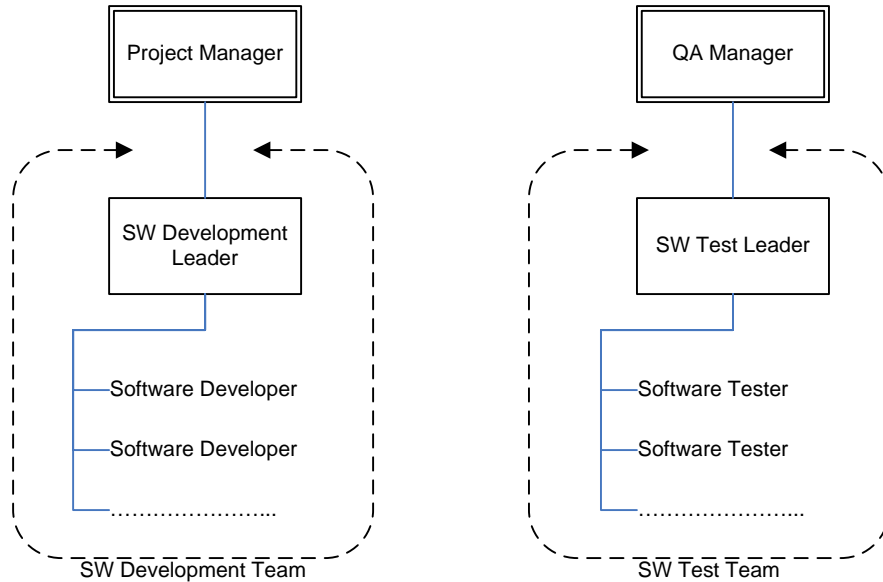


Figure 1

<< Describe software team organization and its internal interfaces. Use organizational charts, as in the simple example above or other graphical forms to better depict the organization. >>

2.2 Interfaces

The portion of the project organization relevant to software development is shown in the following organizational chart.

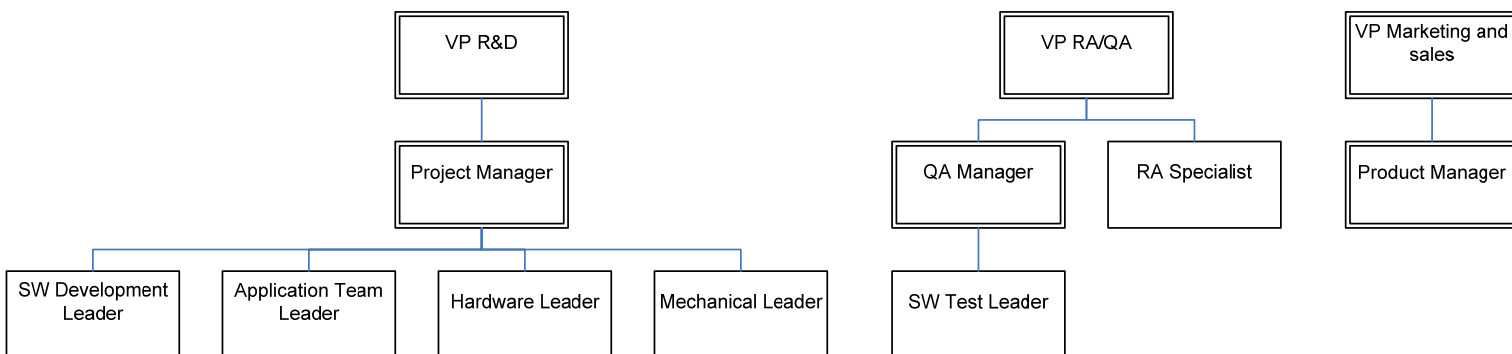


Figure 2

<< Describe the portion of project organization relevant to Software and define the interfaces between the SW team and the other organizations. Use organizational charts as in the example above or other relevant graphics or tabular forms to depict the interfaces. >>

2.3 Roles and Responsibilities

Table 1 below shows the main actors in software development and testing and their roles and responsibilities

Actor	Roles and responsibility
Project Manager	<ul style="list-style-type: none">• Coordinates the work of the project teams to reach project objectives• Assigns the SW project leader•
SW Development Leader	
SW Development Team	
QA Manager	
SW Test Leader	
SW Test Team	
.....	
VP R&D	
IS Organization	

Table 1

<< Describe the roles and responsibilities of the people and organizations involved in the SW development, including also support organizations, such as IS for network support, etc. Preferably do not use people names, but titles and organization names, so that if people change or are added/removed from the project organization the plan is still valid. Use a tabular form as in the example above or any other suitable form.>>

3 Managerial Process

3.1 Objectives and Priorities

<< Describe specific objectives of software development within the project, the development strategy and the main priorities. >>

3.2 Assumptions, dependencies and constraints

The <device> software development is meant to use resources that are currently used by project XYZ, that is closing out in

It is assumed that the necessary resources will be freed up by

In case of delay.....

<< Describe assumptions and dependencies that would affect project management and require contingency plans in case of change. Identify also possible constraints related to the software project management. >>

3.3 Resources Management - **OPTIONAL**

<< This section and its subsections are optional. Information could be managed elsewhere at program level or for very simple projects could not be relevant. >>

3.3.1 Estimation

<<Drop if not relevant.>>

<< Detail the resource estimations and estimation method or refer to other project documentation where the estimation is maintained. >>

3.3.2 Staff

<<Drop if not relevant.>>

<< Specify the number of staff required by skill level, map staff level over the project forecasted life span or by project phase, define how the staff is acquired. Use a tabular form or a graphical form. >>

3.3.3 Other resources

<<Drop if not relevant.>>

<< Specify the requirements related to office space, workstations and other hardware resources, software licenses, etc. Indicate when the resources are needed and how they are acquired. >>

3.3.4 External Suppliers Management

<<Drop if not relevant.>>

Table 2 below lists the products and services that are externally acquired and indicates how they are controlled.

Product or Service	Supplier ID / location	Supplier Management	Product Evaluation
RTKern25 real time multitasking OS.	RTStar, inc. 111 RTDrive, Burlington, MA 01803	Supplier Qualification Visit on	Validation test will be performed internally by

Table 2

<< List products and services that are externally acquired and indicate the plans to manage the related external supplier and the plan to evaluate the quality of the product or service. Use a tabular form as in the example above or other suitable form. >>

3.3.5 Training

<<Drop if not relevant.>>

Table 3 below shows a tentative plan for specific training classes. The plan will be adjusted as needed.

Training session	When and where	# of participants	Type of participants
Requirement Analysis Workshop	April 1, 2009 Auditorium	20	SW development and SW testing staff

Table 3

<< Describe training needs, how they will be fulfilled and provide a training plan. A tabular form, as in the above example or other suitable form may be used. >>

3.3.6 Budget

<<Drop if not relevant.>>

<< Define the budget for the SW portion of the project. If budget is maintained elsewhere, just state that. >>

3.4 Work Plan

3.4.1 Schedule

The detailed project schedule for software is maintained separately in document

Current targets for phase completions are:

- Requirement Definition: June 2009
- Construction: June 2010
- Validation and Release: October 2010

<< Normally the detailed project schedule is not maintained within the development plan, but in a separate document, eventually using tools like MS-project or similar, to help in work breakdown, identify constraints and possible parallel paths. In this section, it is enough to indicate where the schedule can be found and define same major project target dates. >>

3.4.2 Integration Plan

The strategy of development includes construction of a series of vertically integrated slices of the system. These are versions that are executable as part of the full system framework at various stages of development. Integration testing and evaluation of various aspects of functionality from a user and performance standpoint are performed on each version to minimize the risk and degree of modifications to system requirements and design late in the project.

Table 4 below lists the main integration milestones

Milestone	Purpose	Includes	Target date
Real Time Framework	To provide the platform of the system (e.g., OS) running on actual device hardware and stubs for applications within which application tasks can be inserted.	<ul style="list-style-type: none">• Integrated OS• Integrated DB manager• Stubs for HW drivers• Stubs for system functionalities• Stubs for UI	
User Interface Prototype	To demonstrate key aspects of interface to obtain user feedback	<ul style="list-style-type: none">• Main screens• Analytical screens• Patient screens	

Milestone	Purpose	Includes	Target date
Service and Diagnostics	early To verify hardware software interaction and HW control performances	<ul style="list-style-type: none"> • Basic navigation In addition to previous: <ul style="list-style-type: none"> • Real HW drivers • User Interface for Service and Diagnostics 	
Basic Functionality			
Full Analytical Version			
Alpha Test Version			
Clinical Trials Version			

Table 4

<< Define major project integration milestones, with their purpose and scope, expected content and target dates. Use a tabular form as in the example above, or any other suitable form. >>

3.5 Monitoring and controlling mechanisms – OPTIONAL

<< This section and its subsections are optional. Information could be managed elsewhere at program level or for very simple projects could not be relevant. >>

3.5.1 Requirements Control

<<Drop if not relevant.>>

<<Indicate any specific deviation from change control process defined in the Software Development SOP. >>

3.5.2 Schedule Control

<<Drop if not relevant.>>

<< Describe your own schedule control mechanism, including responsibilities for control, frequency and tools used to control. >>

3.5.3 Budget Control

<<Drop if not relevant.>>

<< Describe your budget control mechanism, including responsibilities for control, frequency and tools used to control. If budget control is performed elsewhere in the project, just state that >>

3.5.4 Quality Control

<<Drop if not relevant.>>

Quality control mechanisms will include:

- in-process audits to verify that the development process is conforming to the quality standards.
- Design reviews and code inspections
- V&V of the work products and the final software
- Continuous improvement program
-

<< Expand as needed >>

Refer to Section 5 of this plan for details on quality control tasks and objectives.

3.5.5 Metrics

<<Drop if not relevant.>>

<< Describe specific project metrics collection, if any >>

3.5.6 Reporting

<<Drop if not relevant.>>

<<Describe project specific status reporting mechanism. >>

3.6 Project Risk Management – OPTIONAL

<< This section is optional. Information could be managed elsewhere at program level or for very simple projects could not be relevant. >>

This section identifies risks to the project that could result in non meeting project scope, project quality, customer acceptance, time of delivery, project costs.

Table 5 below lists the risks considered, a qualitative analysis performed, and the risk control actions taken.

Risk	Consequence	Risk Control
Cannot Meet Performance	If the specified level of analytical performance is not reached, intended use of the instrument needs to be limited	Hardware and software performance will be assessed early on one of the first integrated version, so that in case performance is not attained, adequate changes can be planned on time.....
High defect rate	Many severe defects are reported, requiring strong effort for analyzing and/or fixing, thus resulting in schedule delay.	Testing will be applied early in the project, to all the different SW integrations to discover and fix problem early and assess project quality. More resources will be applied if needed to reduce schedule problems.
Major natural disaster (earthquake, hurricane,...)	This risk could strongly affect the project in different ways.....	Daily backup of all project documentation will be executed both on local server and via internet to a remote server installed in a safer location.....

Table 5

<< Define risks to the project or the plan. Identify mitigations and contingency plans to cope with the risks. Include in this analysis risks in the acquirer-supplier relationship, contractual risks, technological risks, risks caused by the size and complexity of the product, risks in the development and target environments, risks in personnel acquisition, skill levels and retention, risks to schedule and budget, and risks in achieving acquirer acceptance of the product. Use a tabular presentation as in the example above or any other suitable form. >>

3.7 Regulatory Plan

For release in the U.S. this product requires 510(k) clearance by FDA. The submission materials will include a section on software as described in FDA's software submission guidances. This section will be written as a summary of the software and its development and test process according to current FDA expectations and no separate documents will be written solely for regulatory purposes. It is assumed that the actual project documentation defined in this plan will be adequate to support any additional information requests by an FDA reviewer or inspector.

The intent of FDA software related guidances listed below are addressed by the company's procedures and this and related plans.:

- Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices
- General Principles of Software Validation (GPSV)
- Guidance for Industry, FDA Reviewers and Compliance on Off-the-Shelf Software Use in Medical Devices

-

A CE mark is required for release in Europe and the software documentation will need to comply with TUV Munich requirements.....

The RA Specialist is responsible to monitor the project development planning and documentation to ensure that all regulatory requirements are met at time of release.

<< Define your own regulatory requirements. Consider all countries where the instrument is supposed to be marketed. Specify how compliance will be reached, who is responsible to assure compliance. If necessary define also timelines to reach documentation compliance (e.g. release in different countries could happen at different times). >>

3.8 Project Closeout Plan – OPTIONAL

Relevant project documents are maintained in the Design History File.

Project post-mortem review will be planned.....

<< Define your own closeout plan This section of the plan does not need to be included at the beginning of the project, but should be added towards the end of it to allow for an orderly project shutdown. >>

4 Technical Process

4.1 Process Model

4.1.1 Software Lifecycle

<< Deviations from the lifecycle defined in the Software Development SOP or any additional requirements, sub-phases or details specific to the project should be stated here >>

4.1.2 Activities

<< Deviations from the activities defined in the Software Development SOP or any additional requirements or tailoring specific to the project should be stated here. >>

4.1.3 Legacy Software - **OPTIONAL**

<< In case of projects making strong reuse of legacy code derived from previous projects (e.g. big maintenance projects to update older instruments), indicate here specific activities required to bring the project deliverables to the required quality level. Indicate what deliverables can be reused as they are, even if maybe do not meet current standards (e.g. code developed not according to current coding standard), what retrospective work is planned, what regression testing is necessary, etc.

Drop this section is not applicable. >>

4.1.4 Phase transition requirements.

<< Deviations from the phase transition requirements defined in the Software Development SOP or any additional requirements or tailoring specific to the project should be stated here >>

4.1.5 Tasks and assignments

<< Within the lifecycle activities, tasks specific to the project should be defined and responsibilities for them assigned to the project team. A table as the example provided below could be used to define these assignments. Change and/or complete as is fitting to your project >>

Table 6 below lists the tasks defined within the activities of the Software Lifecycle and assign responsibilities to the different actors as follows:

- **Attend** for a meeting task, participate in the meeting
- **Lead** for a meeting task, manage the meeting
- **Create** for a document, author the document
- **Review**
- **Approve**.....

<< Change\complete with your own list >>

Task	Programmers	Chief Engineer	Testers	Project Manager	Quality Engineer
Project Meetings	Attend	Attend	Attend	Lead	Attend
Development Planning	Review	Review	Review	Create	Review
Milestone Coordination					
Team building & personnel management					
Time Tracking & Analysis					
Standards & Procedures					
Phase Transition Reviews					
Risk Analysis					
Requirements Definition					
Requirements Review					

Internal SW Design Code & Debug Technical Reviews Test Planning Test Plan Review Test Case Review Test Execution Change Control Release Control Problem Tracking & Resolution In-process audits	
--	--

Table 6

4.2 Methods, Tools and Techniques

Planning and schedule documents shall be produced and maintained using Microsoft Office tools.....

Requirement and Design Specification shall be produced and maintained using

Code will be written in C and C++ with the use of the << define editor, compiler, linker, build tools or whatever integrated suite is used >>

Risk analysis will make use of

Unit Test will be supported through.....

.....

.....

<< Detail for every activities methods, tools, specific techniques you plan to use. For tools make sure to indicate also version of build or reference any document having this information. >>

4.3 Infrastructure

<< Describe the policy to establish and maintain the development and testing environment, including office space and facilities, hardware and software tools, network capability, etc.

Define also responsibilities for infrastructure support. >>

5 Quality Plan

5.1 Standards, Practices and Conventions

Project-specific procedures in addition to this plan are listed below. Software development will be audited for compliance with these procedures during project phases within which they are required.

- <SW Department> C/C++ Coding Standard
- <SW Department> Software Development procedure.
- <SW Department> Software Maintenance procedure
- <SW Department> Problem tracking procedure.
- <device> SW Configuration Management Plan
- <device> SW Test plan
-
-

<< List all the company procedures and specific project documents that apply. Project personnel should be trained on these procedures and plans and the project will be audited to verify compliance. >>

The project will also be compliant with the following external standards:

- IEC62304:2006 Medical device software – Software Lifecycle Processes
- IEC 60601-1:2005 Medical Electrical Equipment – Part 1: General requirements for basic safety and essential performance.
-
-

<< List standards to which you claim compliance. Make sure to avoid listing standard you have used but to which you cannot claim full compliance. >>

The following external standard have been used as reference:

- IEEE Std. 1058-1998 IEEE Standard for Software Project Management Plans
- IEEE Std.730-2002 IEEE Standard for Software Quality Assurance Plans
-
-

<< List standards that the project uses as reference for its activities and/or its documentation, but to which you do not claim full compliance. >>

5.2 Documentation Plan

<< This section identifies main project documents that are input to the Software development process and documents that are produced as deliverables from the software development process. The example shows a tabular form, but you can use a graphical form to show document sequence and/or interrelationship, or whatever other form you find suitable >>

5.2.1 Input to Software Development

Table 7 below lists the documents that serve as the primary inputs to software development.

Document	Description	Responsible	Target availability
System Hazard Analysis	This document identifies potential device hazards.	QA Manager	Draft available prior to completion of Requirement Definition phase
Product Specification	This document defines the device from a system and user need perspective. It is the primary Design Input for the system overall.	Product Manager	Draft available at start of Requirement Definition phase

Document	Description	Responsible	Target availability
System Requirement Specification	This document details the system requirements. It can be part of the Product Spec above	Product Manager	Draft available at start of Requirement Definition phase
Hardware Specifications	This includes schematics, block diagrams, interface descriptions, and I/O and interrupt maps.	Hardware Leader	Drafts available during Requirement Definition phase, most documents need to be finalized and approved before entering Construction phase
User Manual	To be developed outside the software group but reviewed by the software group and tested as part of final software test.	Product Manager	Available at start of Validation and Release phase
Service Manual			

Table 7

<< List all relevant documents that will be used by software development, indicating also the person or group responsible for it and a target availability in terms of date or relationship to lifecycle phases. >>

5.2.2 Output from Software Development

Table 8 below lists the key software documentation to be produced and maintained for this project. All documents listed in this table will be maintained under revision control. Relevant final release documents will be maintained as part of the Device Master Record (DMR) for the project. In-process records of design control will be maintained as part of the Design History File (DHF).

Document Title	Filename	Description	Responsible	Intended Audience	Reviews	Target availability
<device> Software Risk Analysis Report	XYZ-SRA_Report.doc	This document identifies software functions or defects in software function that could result in patient or user hazards, evaluates the risks, and defines mitigation for these hazards..	SW Development Leader	Project Management, SW development and SW test team	At least at phase transition.	Draft available at start of Construction phase, final approved at the end of Validation phase
<device> Software Development Plan		This document. It defines the process for software development, quality assurance, verification, and validation.	Project Manager	Project Management, SW development and SW test team	At least at phase transition	Approval before start of Construction phase
<device> Software Requirements Specification		This document will include all software functionality. Detail will be the greatest for requirements that are related to key safety hazards. This document is critical for effective test planning as well as for software development.	Project Manager	Project Management, MKTG, SW development and SW test team	At least at phase transition	Approval before start of Construction phase
<device> User Interface Design Specification		This document fully specifies screen design and keystroke command operation.	SW Development Leader	SW development and SW test team	At start of Construction phase	Approval before start of Construction phase
<device> Software Design Specification		This document will contain the architectural breakdown, a description of key software design decisions and algorithms, and detailed software design structure of software items .	SW Development Leader	SW development team	At least at phase transition.	Architectural breakdown approved before start of Construction phase, final complete design document approved at start of Validation phase
Code	Filename conventions	The source code, build files, test code, and a listing of revisions of	SW Development	SW development	Code inspection for critical units,	Complete code available and

Document Title	Filename	Description	Responsible	Intended Audience	Reviews	Target availability
<device> Software Test Plan	defined in <device> SW Configuration Management Plan	tools used to generate executables . Headers and comments within the source code provide key detailed documentation of design and functionality.	Team	and SW test team	based on risk analysis, walkthrough for other unit, upon developer request.	reviewed at start of Validation phase
		This document is intended as a high-level plan for software testing and to tie together aspects of software testing that may be accomplished in concert with hardware, system, and clinical tests.	SW Test Leader	Project management, SW Test team	At least at phase transition.	Draft available at start of Construction phase, final approved at start of Validation phase.
Test Cases	Filename convention defined in <device> Software Test Plan	Specific test cases to be executed.	SW Test Team	SW test team	Before formal execution	Available as needed during Construction phase, complete set available at start of Validation phase
Test Records	Filename convention defined in <device> Software Test Plan	Results of test execution in the form and containing the information prescribed in the test plan.	SW Test Team	Project management, SW Test team	According to availability	Available during Construction and Validation phase
<device> Test Summary Report		Summary of test execution, including what and when tests ran, list of anomalies and their disposition, evaluation of test coverage and completeness.	SW Test Leader	Project management	At release	Partial summaries available at each test session, final approved summary report at the end of Validation phase
Defect Tracking Records		Defects encountered after first system integration will be recorded according to the Problem Tracking and Resolution process. Defect analysis and disposition will also be recorded.	Project Manager	Project Management, SW development and SW test team	Ongoing, by CCB	During Construction and Validation phases
<device> SW Configuration Management Plan		Identifies items to be managed under Configuration control and defined the project Configuration	Project Manager	SW development and SW test	Before application.	Approval at start of Construction phase

Document Title	Filename	Description	Responsible	Intended Audience	Reviews	Target availability
Change Requests		Control process. Modification requests will be maintained according to Change Control process and retained until incorporated in an updated requirements and/or design specification.	Project Manager	team Project Management, SW development and SW test team	Ongoing, by CCB	During Construction and Validation phases
Release Control Records		Records for each release to formal software system testing will include a fixes list, changes implemented list, and a source code diff list. These records will be retained until the release is no longer in use.	SW Development Leader	Project management, SW test team	At release	During Construction and Validation phases
Audit reports		Each in-process audit by the QA or external auditor will be documented and the audit reports will be retained with project records	QA Manager	Project management		According to audit plan
Review records		Records of key reviews will be created and retained, using the template in Section Error! Reference source not found.. Resolution of action items will be documented.	Project Manager	Project Management, SW development and SW test team		According to Review plan

Table 8

<< List all relevant documents that will be produced by the SW development process, indicating responsibilities and due dates or relationship to the lifecycle phases. >>

5.3 Reviews and Audits

<< The minimum requirements include records required for formal review in the Software Development SOP. Provide in this section additional details and/or additional planned formal reviews. >>

5.4 SOUP and OTS management

Management of SOUP Software shall be based on the September 1999 Off-The-Shelf Software Used in Medical Devices FDA Guidance.

Each SOUP SW items shall be identified by manufacture and version. Its functionality shall be described to ensure it is appropriate for its intended use.

SOUP software items shall be considered during the Risk Assessment Process. Software shall be tested, verified and validated as necessary. Configuration of the SOUP software items shall be controlled and the software specification shall include the system configuration for which the software will be validated.

Table 9 below lists the SOUP Software Items that will be used on the project:

SOUP SW Item	Description	Provider
RTKern25. Version 123 build 33	Real time multitasking kernel. The following features will be used: <ul style="list-style-type: none">• Task management• Message queues• Semaphores••	RTStar, inc. 111 RT Drive, Burlington, MA 01803
CrazyDB22. Version 999 build 99	Database manager used for patient DB management. The following features will be used: <ul style="list-style-type: none">• <TBD>•	Crazy Software, inc. 999 Crazy Lane Burlington, MA 01803
Format Conversion Utility	SW allowing to convert any type of internal data to the required format for display and print.	Internally developed for project ABCD.

Table 9

<< List of the software items that you plan to use, either acquired from outside or internally reused from other projects. In the last case, list only the ones you plan to use as is, without specific adaptation to your project. The ones that get adapted will need to undergo all the process defined for the project and will be accordingly documented. >>

5.5 Verification and Validation

SW V&V activities will include:

- Static Testing (reviews and inspections). See Section 5.3 for details.
- Unit Testing
- Integration Testing
- System Testing
- Regression Testing

Refer to the <device> Software Test Plan for details on planned V&V activities and how they are conducted.

<< Describe V&V activities as needed or reference external documents detailing them.>>

5.6 Code and Media Control

Source code and build scripts will be controlled throughout development using SourceSafe and according to the <device> Software Configuration Management Plan.

All C code destined for inclusion in the product will be subject to the Software Department C Coding Standard.

Software for installation on the system during Construction phase will be on USB key. This will be inserted in an envelope marked with SW version and including simple release notes.

During Validation phase SW will be distributed on CD for installation via network connection. The CD will be labeled with version number and the statement FOR SW VALIDATION USE ONLY.

Final release software for Operation will be produced according the release procedure <TBD>

<< Detail your own code and media control process. >>

5.7 Record Collection, Maintenance and Retention

Each document or record mentioned in this plan will be considered for inclusion in the Device Master Record or Design History File as required in the Software Development SOP and will be retained as required by corporate Record Retention policies.

<< Add/Delete/Replace as needed according to your own policy. >>

6 Supporting Process Plans

6.1 Configuration Management

SW Configuration management shall be applied according the SW Configuration Management Plan starting from.....

<< Describe here any specific detail related to this project or any deviation from requirement of the Software Development SOP. >>

6.2 Problem Resolution

The Problem Reporting Procedure describes the overall issue tracking and problem resolution process.

The issues will need to be tracked in the problem resolution process as soon as any formal testing is applied already during Construction phase.

<TBD> will be responsible for analyzing problems and approving corrections.

<< Describe here only details related to this project or any deviations from what stated in the Software Development SOP and in the Problem Resolution SOP. Include indication of when problem resolution needs to be applied and the person or team responsible for it. >>

6.3 Software Risk Management

The Software Risk Management procedure defines the process to be applied for Software Risk Analysis and Control within the general project risk management process.

The QA Manager and the Project Manager will be responsible for application of Software Risk Management activities.

<< Describe here only details related to this project or any deviations from what stated in the Software Development SOP and in the Software Risk Management SOP. >>

6.4 Traceability

<< Detail any specific detail on traceability or any deviations from what required in the Software Development SOP. Detail any tool used in the project to implement traceability. >>

6.5 Process Improvement

Attention during reviews will be applied to identify possible improvements to the Software development process.

This will not limited to the in-process audits and the Phase Transition reviews.

During defect analysis, some effort will be applied to identify trends that could indicate the need or the opportunity to apply changes to the process. Process improvements will be evaluated to understand their expected impact and decide the best time to apply them without disrupting the process.

A project post-mortem review will be held after release to identify process improvements to apply to following projects.

<< Describe your own process improvement plan >>

7 Maintenance Plan

The focus during maintenance is on monitoring field reliability and usability issues, and controlling the changes that are made to the software.

Maintenance process will be implemented according to the <TBD> SW Maintenance procedure.

A specific Maintenance phase Change Control Board will be installed to evaluate and control the changes that are implemented during maintenance phase.

Only minor changes and fixes will be allowed during maintenance. Any major change, if approved, will require a new project to be started, following the entire project development process.

<< Use this section to indicate deviations from the maintenance procedures or specific details related to this project. Drop if a specific Maintenance Plan is developed. >>

SOFTWARE CONFIGURATION MANAGEMENT PLAN

TEMPLATE

DRAFT

Revision History

Date	Revision	Author	Description

Adaptation of IEEE Std. 828-1998 IEEE Standard for Software Configuration Management Plans

Note: This template is conceived as a partial example template for a generic small device with embedded real time control. Explanatory comments are included in << comment >>.

Other text is example definition that you should replace with your own text.

This is not a complete Software Configuration Management Plan, just a training example to guide in the development of a Software Configuration Management Plan for a certain type of device.

TABLE OF CONTENTS

1	Introduction.....	4
1.1	Purpose.....	4
1.2	Scope.....	4
1.3	Evolution of this plan.....	4
1.4	Definitions, acronyms, and abbreviations.....	4
1.5	References.....	5
1.6	Overview.....	5
2	SCM Management.....	5
2.1	Organization and Responsibilities.....	5
2.2	Applicable policies, directives and procedures.....	6
3	SCM Activities.....	6
3.1	Configuration Identification.....	6
3.1.1	Configuration Items.....	7
3.1.2	Baselines.....	8
3.1.3	Releases.....	8
3.2	Configuration Control.....	8
3.3	Configuration Status Accounting.....	9
3.4	Configuration Audits and Reviews.....	9
3.5	Subcontractor/Vendor Control.....	9
3.6	SOUP control.....	9
4	SCM Schedules.....	10
5	SCM Resources.....	10
5.1	Personnel.....	10
5.2	Infrastructure and Tools.....	10
6	Appendices.....	12
6.1	Appendix A: Detailed SCM process definition.....	12
6.1.1	Configuration Items Database.....	12
6.1.2	Project structure.....	12
6.1.3	Adding files to the configuration control database.....	12
6.1.4	Modifying files under configuration control.....	13
6.1.5	Configuration identification.....	13
6.1.6	Build and Promotion Process.....	14
6.1.6.1	Labeling.....	14
6.1.6.2	Build.....	14
6.1.6.3	Promotion.....	14
6.1.6.4	Handling of multiple baseline versions.....	15
6.1.7	Media Release Control.....	15
6.1.8	Records collection and retention.....	15
6.2	Appendix B: Detailed Software Release Process for Manufacturing Release.....	16
6.2.1	Overview.....	16
6.2.2	Creating Master Executable Packages.....	16

6.2.3	Release Notes.....	16
6.2.4	Manufacturing Replication and Installation.....	16
6.3	Appendix C: Release Form template	18
6.3.1	Identification	18
6.3.2	Purpose.....	18
6.3.3	Scope.....	18
6.3.4	References.....	18
6.3.5	Released material	18
6.3.5.1	Released Components	18
6.3.5.2	Compatibility	19
6.3.5.3	Distribution	19
6.3.5.4	Installation.....	19
6.3.6	Release content	19
6.3.6.1	New features	19
6.3.6.2	Fixed anomalies	19
6.3.7	Testing.....	20
6.3.8	Known anomalies.....	20

1 Introduction

1.1 Purpose

This plan identifies the procedures for managing the configurations of the <device> computer programs, support software and test software during their development and maintenance. This plan is intended for use:

- by the software development and test team during their software development activities
- by QA as a basis to audit the configuration activities on the project.

<< detail the purpose of the plan and its intended audience >>

1.2 Scope

This procedure is applicable to all personnel involved in the development of new and/or modified system software and associated documentation for the <device> software.

The <device> is a <TBD> analyzer used for.....

The Project Manager shall use this guide to establish common practices for software configuration management.

This plan is applicable to software development lifecycle starting from Construction phase, and shall apply also to Validation/Release and maintenance phases.

This plan applies to all software elements embedded in the <device> as well as to any tool used during development for software production and testing.

Configuration Management of hardware is not covered in this plan.

Documentation Control, as applied to project plans and project specification documents, is covered at system level, not in this plan.

<< Add/Remove/Replace as fit. Describe range of applicability and eventual limitations. Indicate when in the project the SW configuration management is applied. >>

1.3 Evolution of this plan

This plan will be maintained by the Project Manager or person designated by the Project Manager to be the Software Configuration Management Administrator. The plan will be updated as needed during lifecycle to add details or to correct defects in the plan. After first formal review of the plan, subsequent versions of the plan will be reviewed again, according to the document review process.

<< Describe planned maintenance for this plan. >>

1.4 Definitions, acronyms, and abbreviations

Build	A build is usually designed to phase in feature additions. With regard to this plan, a build is a set of files that can be loaded and executed. Builds are generated from Software Packages using a build process.
Clean Build	A clean build is a build in which all files are retrieved fresh from the SCM tool's database. This is best accomplished by wiping out the working project directory prior to retrieving the software package associated with the build label.
Configuration Control	An element of configuration management, consisting of the evaluation, coordination, approval or disapproval, and implementation of changes to configuration items after formal establishment of their configuration

	identification.
Configuration Identification	An element of configuration management, consisting of selecting the configuration items for a system and recording their functional and physical characteristics in technical documentation.
Executable Package	Files that may be executed directly by a computer.
Release	A release is a particular build that is being distributed for a specific purpose.
Software	Computer program instructions and data that are executed out of RAM, ROM, Flash PROM, EPROM or EEPROM. The instructions and data are generated using general engineering tools and disciplines. Packaging of the instructions use technology technically appropriate for the media.
Software Package	A set of source code files required for the creation of a build. The files are associated in such a manner that the appropriate revision of individual files is linked (e.g. by applying labels).

<< Define any acronyms, abbreviations, or terms used in the document that may be unfamiliar to readers. If a project level definitions document exists, it can be referenced here and this section limited to specific terms used only in this document. >>

1.5 References

Standards:

IEC 62304:2006, Medical device software—Software life cycle processes.

IEEE Std. 828-1998 IEEE Standard for Software Configuration Management Plans

.....

<< Include any **relevant** standard >>

Project Documents:

<device> Software Development Plan, Rev. x.y,<<you could add any information useful to locate the document>>

.....

<< Include **relevant** project documents such as device risk analysis, other project plans, etc. >>

1.6 Overview

This document is organized in 6 main sections.

Section 1 (this section) puts the document into its context and gives an overview of it.

Section 2 describes how the SCM will be managed.

Section 3 describes how the SCM will be implemented and all the related activities.

Section 4 gives a SCM implementation schedule within the software lifecycle.

Section 5 indicates the resources necessary to implement the SCM

Section 6 (appendices) details the SCM process and SW Release process, including a template for release notes.

<< Add anything apt to describe content and organization of this document >>

2 SCM Management

2.1 Organization and Responsibilities

The <device> Software project development is under the direct responsibility of the Project Manager.

Different units participate in the software development effort:

- A software development team, in charge for the software development

- A testing team, in charge of defining the test protocols and developing the test scripts.
- A tool support team, in charge to develop and/or install the software tools used for development and testing
-

Quality Assurance is responsible for ensuring that the software is developed and validated according to the company procedures. QA is responsible for configuration management audits and reviews.

An SCM Administrator will be designated by the Project Manager. This Administrator will be in charge of managing the configuration management tool and ensuring support to all groups using SCM. He/She is responsible for the implementation of the configuration tool and database, for the definition of naming and labeling conventions, assignment of release baselines labels, and performance of release builds.

All software engineers and test engineers involved in the <device> project are responsible for the implementation of the process defined in this plan.

Each author is responsible for the identification of the configuration items, for insertion and extraction from the SCM database and for appending history notes to the configuration items assigned to him/her.

The Project Manager is responsible for identifying the baseline releases as defined in this plan. The Project Manager is also responsible to ensure that configuration items developed by subcontractors/vendors are managed according to this plan.

Responsibilities for Change Management process and Defect Management process implementation and execution are.....

<< Add/Complete/Modify as needed. Briefly describe your specific organizational structure related to users and managers of the SCM, detailing SCM responsibilities within the group. The example text above assumes the existence and use of a configuration management database, managed through a tool. Of course text needs to be changed/adapted in case configuration management is performed manually, e.g. by naming folders in your system >>

2.2 Applicable policies, directives and procedures

<< List here applicable procedures, work instructions, etc. that are used for the implementation of this plan, e.g. any Configuration Management procedure, any work instruction for access to the Configuration Database,>>

3 SCM Activities

<< The example text used in this training example assumes the use of a configuration database managed through a tool. Of course the text should be modified/adapted according to your own SCM process. Use of a specific tool could require adapting the language (for example labeling could be called in a different way by your tool), in case no tool is used and configuration management is implemented manually the specificity of some action could be different or be lacking. >>

3.1 Configuration Identification

Configuration identification of software and documentation during the development effort consists of established baselines and releases that are synchronized with the development activities identified in the <device> Software Development Plan.

There are the following different categories of documents that fall under this Software Configuration Management Plan:

- Embedded software code and tools

- Test software and tools
-

Each of them shall have independent lifecycle and identification process, but shall be labeled consistently at each major software release.

3.1.1 Configuration Items

Configuration items (CI) for embedded software shall include:

- Source code files
- Include and header files
- Software build files
- Compiler Option files
- Executable files
- Tools used in the build process
-

Configuration items for Test software shall include:

- Test scripts.
- Test data files.
- Scripting engine source files.
- Tools used in the build process
-

Each configuration item shall have a descriptive name and a standard extension, identifying the file type.

Table 1 below lists the supported file types:

EXTENSION	FILE TYPE	ASCII/BINARY
.asm	Assembler source code files	ASCII
.bld	Build Files	Binary
.c, .cpp	C, C++ source code files	ASCII
.dll	Dynamic link library	Binary
.doc	MS Word file	Binary
.exe, .hex	Executable files	Binary
.h	C, C++ header files	ASCII
.inc	Assembler header files	ASCII
.ini	Initialization/Configuration File	ASCII
.lib	Library files	Binary
.mak	Make file	ASCII
.o, .obj	Object files	Binary
.txt	Text	ASCII

Table 1

A version number and a descriptive note shall be associated within the configuration management database. The version number shall be updated any time a configuration item is updated. A descriptive note shall be appended, describing the reasons and the contents of the update.

Note: Object files .o and .obj for this project are not maintained in the SCM database. The consideration here is that the object files can always be recreated based off of the other source files that are maintained in the database.

<< Add/Delete/Modify based on your own configuration items identification and handling. >>

3.1.2 Baselines

Baselines are defined as points during the software development life cycle at which configuration control of design, product, and engineering changes are synchronized with the software development phases.

Each of the different categories of documents defined above could have different baselines, but shall come together at major software releases.

At every baseline, a descriptive label shall be associated with all the configuration items included in the baseline: the label shall include Baseline Type and Revision number. In addition, it could include additional descriptive information.

<< Add/Delete/Modify based on your baseline definition and baseline handling >>

3.1.3 Releases

Software baseline releases include the following:

- a) **Verification:** the software is released only for internal use (development and testing).
- b) **Validation:** the software is released for use in controlled laboratory validation setting.
- c) **Production:** the software is released for use in full-scale production devices.

Within each of the major baseline releases, recycles can cause the implementation of minor releases.

At each Release, all the configuration items are placed under configuration control and shall be labeled with the release information.

After a baseline release, changes are performed using the Change Control process.

<< Add/Delete/Modify based on your release definition and handling >>

3.2 Configuration Control

All software configuration items are released to, and maintained by, software configuration management.

The Change Control process is the mechanism by which changes are presented and authorized. Change Requests, from whatever source, are entered in a Change Request database. The Change Control Board meets periodically to approve or reject proposed changes. Approved changes can impact several project documents. Some of them affect Configuration Items managed through SCM. Changes to CIs will be managed through Configuration Control. For additional details on Change Control process, please refer to

The Issue Tracking and Problem Resolution process is the mechanism by which anomalies are submitted for implementation into configuration controlled items. Software anomalies are evaluated by the Defect Review Board. The anomalies approved for implementation and affecting Configuration Items will be managed through Configuration Control. For additional details on Problem Resolution process, please refer to

Each approved change or approved defect implementation to a Software Configuration Item shall require extraction of the item from the database, implementation of the change, followed by insertion of the modified item into the database. A descriptive note shall be mandatory when updating a configuration item.

<< Add/Delete/Modify based on your change control, problem resolution and configuration control processes >>

3.3 Configuration Status Accounting

A detailed change history is maintained by the configuration management tool for each Configuration Item through the notes appended at each update into the database.

The tool shall clearly identify the latest version of each configuration item, as well as those items that are undergoing change.

Appropriate labeling shall identify the latest released revision of each configuration item, as well as back tracking to any previous released revision.

It is the responsibility of the current author of a configuration item to review the current configuration status of the item when it is released. This includes verifying that the correct version is in the version control system.

<< Add/Delete/Modify based on your Configuration Status accounting processes >>

3.4 Configuration Audits and Reviews

Audits are used to verify the effectiveness of the SCM system. They shall be conducted by QA at each planned major release.

A review of the process shall be conducted after implementation and after each audit.

<< Add/Delete/Modify based on your audit and review plan >>

3.5 Subcontractor/Vendor Control

If software items to be incorporated into the project or software tools used for SW development or testing are developed externally by a subcontractor, all the configuration items developed shall be put under configuration control, according to the process defined in this plan.

<< Add/Delete/Modify based on your subcontractor management process. For example:

- you could specify if the subcontractor will have direct access to the configuration database via a remote client, or internal personnel are responsible for receiving the configuration items from the subcontractor and putting them under configuration control
- you could specify how the subcontractor will be audited for compliance
- you could specify the need of some level of acceptance testing to be performed before putting subcontracted items under configuration control
- you could specify how proprietary items are to be handled for security of information and traceability of ownership
- you could specify how changes and anomalies to subcontracted configuration items are to be handled and what the involvement of the subcontractor in the process. >>

3.6 SOUP control

Any SOUP item, either OTS or internally developed, is incorporated into the project, as part of the embedded software or of a development/testing tool, it shall be identified as a configuration item and shall be put under configuration control, according to the process defined in this plan.

<< Add/Delete/Modify based on your policy to handle SOUP items. For example:

- you could specify how the SOUP configuration items are identified as such

- you could specify what is the process for accepting a SOUP configuration item into the configuration control database, e.g. the need of some level of acceptance testing
- you could specify how proprietary items are to be handled for security of information and traceability of ownership, e.g. copyrights and royalties
- you could specify how anomalies in SOUP configuration items are to be handled
- you could specify how revisions and patches are to be handled >>

4 SCM Schedules

The Software Configuration Management Process shall be implemented during the construction phase and prior to the establishment of any formal baselines.

Configuration items that are meant to be included in a baseline release, need to be put under configuration control before the baseline is labeled and the release software is built.

Any unresolved process issues found once the SCM process is implemented must be promptly resolved prior to the establishment of next baseline.

<< Add/Delete/Modify based on your SCM process schedule. >>

5 SCM Resources

5.1 Personnel

No human resource will be exclusively dedicated to the implementation and day to day handling of the SCM process.

The SCM Administrator will be designated by the Project Manager among the project personnel involved in SW development or testing activities, based on his/her skills.

All the staff that need to be involved in SCM activities will be trained on this plan, the process and the related tools before SCM process is established.

IS department will provide support for the establishment of the infrastructure and tools necessary for the implementation of the SCM process.

<< Add/Delete/Modify based on your specific process needs. >>

5.2 Infrastructure and Tools

A space of <TBD> GigaBytes will be reserved on the <TBD> R&D Server for the implementation of the SCM database.

Table 2 below lists the tools used to implement the SCM process

Tool Name	Version	Use
<TBD>	<TBD>	Control of all the Configuration Items defined in this plan.
<TBD>	<TBD>	Implement the SCM database
<TBD>	<TBD>	Front end client application to connect to SCM database
<TBD>	<TBD>	Defect management system used to manage defect submission, approval and

<TBD>	<TBD>	implementation toward Configuration Items defined in this plan. Change management system used to manage change submission, approval and implementation toward Configuration Items defined in this plan.
-------	-------	--

Table 2

<< Add/Delete/Modify based on your specific process needs. >>

6 Appendices

6.1 Appendix A: Detailed SCM process definition

<< This is structured as a work instruction on use of the specific SCM work environment and tools. It could be kept as an appendix of the plan or as a separate work instruction document. Add/Delete/Modify, based on your own SCM environment and tools you use.

In this example we try to use generic terminology. You should replace it with the proper terminology used by your process and your tools. For example, if your tool uses the term “check out” to indicate extraction of a configuration item from the SCM database for modification, you should use that term in place of “extract from SCM database” >>

6.1.1 Configuration Items Database

A SCM database has been created in a specific location on the network to store the CIs.

<TBD> is the tool used to manage the CIs and <TBD> tool is used as the front end client to access the SCM database.

Each user of <TBD tool> needs to have login capability to the above database. Contact the network administrator to setup access to the database.

6.1.2 Project structure

The SCM project structure matches the categories of CIs indicated in section 3.1, as shown in the figures below:

<< Insert screen captures showing the folder structure of the project in the SCM database>>

Under each main folder, a subproject structure can be created by using the command <TBD>.....

Each user shall have a matching directory structure on his/her local workstation.

If it doesn't already exist, the directory structure is automatically created by the tool when files are retrieved from the SCM database.

This local directory structure is the default location where all the interaction with <TBD> SCM tool (files retrieval, files insertion, etc.) takes place.

A similar directory structure needs to be created on any build workstation used to build release software packages.

<< You could extend the description by including all the subproject tree structure of the SCM database and describing commands to create and maintain the structure. >>

6.1.3 Adding files to the configuration control database

During development, as source files (and related documentation) are created, the files shall be added to the SCM database directly through the <TBD> client interface as indicated in the displays below.

To add files to any project <<describe the command(s) to be used to add files to the database. Some tools allow for different ways to perform this task (drop down menus, icons, drag and drop). You could explain here all the possibilities, or just indicate a preferred way to perform this task. Most tools prompt the user to enter a comment when a file is first added to the database, or allow anyway to add a comment, although often this is not mandatory. You should indicate here in the work instruction that entering a comment is strongly recommended (or required by your process): such comment will appear as first item in the file history documentation. >>

<< Insert screen captures showing the command sequence to add files. >>

6.1.4 Modifying files under configuration control

Anytime one or more documents under configuration control need to be modified, the following operations must be done:

- Extract the file from SCM database to your local workspace. When a file is checked out, no other user can check it out for changing. Other users can only get read only copies of the file for consultation.
- Alter the file(s) as necessary.
- Verify the changes (for example, if it is source code, verify that the modified file compiles correctly, debug the changes, etc.).
- Insert the modified file into the database.

<< you could add other mandatory or recommended actions to this list, for example you could suggest to make a diff between all files in your local directory and the files in file folder in the database, to verify if other files that could be related to the file you have updated did also change in the meanwhile. In such case maybe you need to verify that the file you have updated is still compatible with the other files. >>

Extraction from SCM database is done with << indicate the proper command or command sequence. Insert screen captures showing the command sequence to extract files. >>

By default, the latest version of the file is copied from the database to the matching directory structure in the user's workspace. It is possible to select a previous version instead of the latest one, but pay attention that this would create a branch on the database structure for that file when you return the file to the database after modification. Branching is not recommended, unless you need to create and keep alive in parallel several versions of the same file.

<< Expand your explanation, based on your own policy or recommended practice and the possibilities offered by the tool. Some tool allow extracting the file for modification on different places, some tool prompt or allow comments to be added to the file history when you extract a file from the database for modification. Eventually add screen captures to illustrate the different options. >>

While the file is extracted from the database for modification, no other user can extract it (the file is locked), therefore the tool shall << explain the mechanism the tool uses to lock the file and to show that a file is unavailable for modification. Eventually use screen captures.

Note that some tools allow modifications in parallel by different users on the same file. If your tool allows it, you should be very cautious in the use of such feature and maybe warn the user against using it or impose some rules for its use and indicate them here. >>

Once the changes have been applied and verified, the file can be returned back from the user's workspace to the SCM database, by using the <TBD> command

A comment is mandatory for this operation and it should explain what has been changed and the reasons for the change. This will appear in the file history. If the change is due to a formal Change Request or to a Defect Fix (as should be the case after any baseline build), the comment must make explicit reference to the Change and/or the Defect Tracking Number, so that they will show up in the file history.

<< Add screen captures to show how this operation is performed >>

Returning the file to the database will make it available for extraction for further modification (unlock the file), unless you explicitly choose to keep it locked for some reasons. << Many tools allow to keep a file locked, explain how this is accomplished and eventually give directions on the cases where it is preferable to do so. Also here screen captures could be helpful. >>

6.1.5 Configuration identification

<< Give here some rules on your labeling mechanism for source files, executable files, builds etc. This is very specific to your model of versioning.>>

6.1.6 Build and Promotion Process

6.1.6.1 Labeling

Before any baseline build takes place, all the source files and related documentation files must be returned back to the database.

The SCM Administrator must ensure that all team members are informed and have returned their files to the database.

The files are then labeled with a label indicating baseline identification, according the guidelines in section 6.1.5.

To apply labels, execute the <TBD> command A comment should be entered to explain the reasons for the baseline build.

<< Further explain your baseline labeling process. Screen captures could be helpful. >>

6.1.6.2 Build

Get all files that are necessary for the build in a work directory of the build workstation, by using.....

<< Describe the command or sequence of commands to perform this operation, eventually adding screen captures. >>

To perform a clean build, delete all the files in the working folder before getting the ones needed for the build.

Execute the build according to the relevant procedure to generate an integrated executable package.

<< Describe the build procedure relevant to your system and your tools. >>

6.1.6.3 Promotion

A log file shall be maintained for each project to list all baseline builds. The structure shall be an excel type table, including the following information:

Build Label	Date/Time	CRC	Identity of builder	Comments

Promotion to Verification baseline.

<< Describe the steps to perform for Verification baseline build. These will be specific to your SCM environment and tools. They will typically involve:

- Assigning a development software version and a baseline identifier.
- Ensuring that all files necessary for the build are updated in the SCM database
- Labeling files as described in section 6.1.6.1
- Applying same labeling to all tools used for the build to ensure to be able in the future to repeat the build.
- Performing the build as described in section 6.1.6.2.
- Updating the build log table with the build information.
- Making the Verification baseline executable package available to its users. >>

Promotion to Validation baseline.

<< Similarly describe the steps to perform for Validation baseline build. Normally the Validation baseline will be obtained from a Verification baseline that has been verified to be suitable for Validation testing. Typically the steps will be similar to the ones for Verification baseline, except that only the baseline identifier, and consequently the label, should change.

After build, it is worthwhile to verify through a diff with the Verification baseline that only appropriate memory locations have been altered. >>

Promotion to Production baseline.

<< Similarly describe the steps to perform for Production baseline build. Normally the Production baseline will be obtained from a Validation baseline that has been validated and considered suitable for release to Operations.

Typically the steps will be similar to the ones for Validation baseline, except that only the baseline identifier, and consequently the label, should change.

After build, it is worthwhile to verify through a diff with the Validation baseline that only appropriate memory locations have been altered. >>

6.1.6.4 Handling of multiple baseline versions

<< During project development, multiple baseline versions of the software can be alive at the same time. For example, a validation baseline is under test, and requires update, while the developers are working at the following verification baseline. It is helpful to give directions on how to proceed under different scenarios.

Consider at least the following scenarios:

- A different existing version of a file needs to be in a baseline already released.
- You have released a baseline, proceed development on the following one and realize that a fix is needed on one or more files of the released baseline, but these files have already changed and you don't want the changes to be included in the released baseline.

The handling of the different scenarios will depend on your policy and on the features provided by your tool. >>

6.1.7 Media Release Control

The following distribution method shall be followed for the associated baseline:

Verification <TBD>

Validation <TBD>

Production <TBD>. Refer to Appendix B, Section 6.2 and to Software Release SOP.

<< Define the way the different baseline software packages will be made available to the users (e.g. stored on the network with limited accessibility, distributed on media, etc.) and the accompanying release documentation>>

6.1.8 Records collection and retention

At the time of production release, the SCM database and the revision of all tools required for that build shall be copied onto non-volatile media for off-site storage.

<< Add/Modify according to your own collection and retention policy >>

6.2 Appendix B: Detailed Software Release Process for Manufacturing Release

6.2.1 Overview

<< Describe at high level the purpose and the content of this release procedure. Note that this should only describe the project specific details, since the general requirements for software release are defined in the Software Release SOP.

This includes the body of a manufacturing installation and replication procedure that will need to be developed when releasing new software for production installation.

>>

6.2.2 Creating Master Executable Packages

The delivery media for the executable package will be <TBD>.

The equipment used to produce the master copy will be <TBD>.

<< Define your own policy of master software distribution, detail if necessary how the copies are to be produced. >>

Labels to be applied to the Master copies will be <TBD>.

<< Define your own specific labeling requirements. >>

The master copies will be verified by <TBD>.

<< Define your own specific verification requirements >>

6.2.3 Release Notes

Release Notes in the form detailed in Appendix C, section 6.3, will accompany the release package to fully describe its content from a functional point of view.

<< Define what documentation will accompany the release. >>

6.2.4 Manufacturing Replication and Installation

The replication process in manufacturing must follow the following steps:

1.
2.
3.

Each copy created through the replication process will be labeled as follows:

.....

Verification of the replicated copies must follow the following steps:

1.
2.
3.

A master list will be kept in the form <TBD> to register copies created and where used.....

<< If replication in manufacturing is required, define the replication process, the applied verification and the documentation produced. >>

The installation of the software on manufactured <device> units must follow the following steps:

1.
2.
3.

Installations will be verified as follows:

1.
2.

3.

A master list will be kept in the form <TBD> to register the software installation on unit.....

<< If firmware installation in manufacturing is required, define the installation process, the applied verification and the documentation produced. >>

The <OTS software TBD> requires the payment of royalties to <TBD>.....

<< If royalties need to be paid, describe the procedure to do so and the process to demonstrate royalties have been paid (e.g. labels to apply or whatever). >>

6.3 Appendix C: Release Form template

<< The following example is a generic template, that can be tailored for use as release notes both for R&D internal releases for verification testing and for release to operation. >>

6.3.1 Identification

Instrument Name: <device>

Software ID: <software>

Version ID: <major version>.<minor version>

SW Build: <build>

Change Order: <#>

Release date: <date>

Authorizations: << signatures and titles of people authorizing the release>>

<<Add/Change/Remove according to your process and to the scope of the release. A tabular for can also be used. >>

6.3.2 Purpose

Intended use of this release is analytical verification of tests <TBD> to be performed in house by the analytical team.....

In addition the release can be used to verify the fix of.....

<< Describe the purpose and intended use of the release >>

6.3.3 Scope

This release applies only to <TBD>.....

It is not meant to be used for

<< Define limitations of the release >>

6.3.4 References

Ref.	Document Title	Files
Ref. 1		
Ref. 2		

<< List all relevant documents referenced in the release >>

6.3.5 Released material

6.3.5.1 Released Components

Item	Type	Version	Description
Main	CD	<version>	Installation SW for main CPU
DeviceXXX	PROM	<version>	Firmware for TBD

<< Depending on the structure of the hardware and software of the device, a release could be constituted by several different components that could be on different media and have different versions. >>

6.3.5.2 Compatibility

The following compatibility table shows the compatibility of the released components with other components of the system.

System comp. Rel. comp.	Printer
Main rev. <revision> DeviceXXX rev. <revision>	Rev. <revision>

<< Depending on the complexity of the system and on how much independency you want to leave between the different components, the compatibility section could be as simple as a single statement saying that the new release requires upgrade of all components in the system or as elaborated as one or more cross-reference table showing compatibility of the different versions of several hardware and software items. >>

6.3.5.3 Distribution

The release is distributed as

<<Describe how the released material is made available to the intended users >>

6.3.5.4 Installation

<< This section could describe how the released material is meant to be installed on the device, or just make reference to the installation procedure that could be in another document. >>

6.3.6 Release content

6.3.6.1 New features

With reference to previous release <TBD>, this release adds the following new features.

Change #	Feature	Limitations
<TBD>	New test <TBD>	The test can be performed only as batch test, not implemented yet as urgency test

<< Describe all the new implementations, eventually referencing a change request number if relevant. Indicate eventual limitations in the implementation of the new features. A tabular form can be used as in the example above or any other suitable form. >>

6.3.6.2 Fixed anomalies

With reference to previous release <TBD>, in this release the following anomalies have been addressed and fixed.

Defect #	Description
<TBD>

--

<< List all the anomalies that were fixed, referencing the defect numbers in the defect tracking system and adding a title or a short description. A tabular form can be used as in the example above or any other suitable form. >>

6.3.7 Testing

<< Describe what testing has been applied to the released material, and/or reference testing documents. >>

6.3.8 Known anomalies

At the date of release, this software is released with the following known defects. Defects are tracked through <TBD>. Please check frequently the defects database to get an update on defects identified on this release.

Defect #	Description	Severity	Workaround
<TBD>

<< List all known anomalies, making reference to their numbers in the defect tracking system and adding any suitable information. A tabular form can be used as in the example above or any other suitable form. >>

SOFTWARE TEST PLAN

TEMPLATE

DRAFT

Revision History

Date	Revision	Author	Description

Note: This template is conceived as a partial example template for a generic small device with embedded real time control. Explanatory comments are included in << comment >>.
Other text is example definition that you should replace with your own text.
This is not a complete Test Plan, just a training example to guide in the development of a Test Plan for a certain type of device.

TABLE OF CONTENTS

1	Introduction.....	4
1.1	Purpose.....	4
1.2	Scope.....	4
1.3	Definitions, acronyms, and abbreviations.....	4
1.4	References.....	4
1.5	Overview.....	5
2	Test items.....	5
3	Features to be tested.....	5
4	Features not to be tested.....	6
5	Overall Approach.....	6
5.1	Test Types.....	6
5.1.1	Hazard.....	7
5.1.2	Negative	7
5.1.3	Error Message Handling	7
5.1.4	Fault	7
5.1.5	Stress	7
5.1.6	Installation and Setup.....	7
5.1.7	Startup and Shutdown	7
5.1.8	Regression.....	8
5.1.9	Platform.....	8
5.1.10	Workflow	8
5.1.11	Functional	8
5.1.12	Documentation.....	8
5.1.13	Algorithm.....	8
5.1.14	Static	8
5.1.15	Multiuser/external access.....	8
5.1.16	Security	8
5.1.17	Data Integrity	9
5.1.18	Ad hoc.....	9
5.2	Unit Testing	9
5.2.1	Strategy and approach.....	9
5.2.2	Documentation.....	9
5.3	Integration testing	10
5.3.1	Strategy and approach.....	10
5.3.2	Documentation.....	10
5.4	System Testing.....	10
5.4.1	Strategy and Approach.....	11
5.4.2	Documentation.....	11
5.5	Validation (user) Testing	11
5.6	Test Summary	11
6	Test Coverage	11
6.1	Physical Coverage.....	11
6.2	Requirements to test coverage	11

6.3	Risk Control Measure to test coverage	12
6.4	Test types to test coverage	12
7	Item pass/fail criteria	12
8	Start, Suspension and Resumption requirements.....	12
9	Test deliverables	13
10	Testing tasks	13
11	Environmental needs.....	13
12	Supporting Software and processes	13
12.1	Software	13
12.2	Processes	14
12.2.1	Documentation Configuration Control	14
12.2.2	Software Configuration Management.....	14
12.2.3	Issue Tracking and Problem Reporting problem resolution process	14
12.2.4	Change control	14
13	Responsibilities.....	14
14	Staffing and training needs	14
15	Schedule.....	14
16	Risks and contingencies.....	15
17	Appendices.....	16
17.1	Appendix A - Test Case Format	16
17.2	Appendix B – Test Summary Outline.....	20
17.3	Appendix C – Test List	23
17.4	Appendix D - Trace Matrices	23

1 Introduction

1.1 Purpose

Purpose of this Software Test Plan is to define the testing approach, testing tasks, test resources, test sequences and test documentation required to test the software of <device>.

This plan has been developed in accordance with the overall quality assurance effort as defined in the <device> Software Quality Assurance and Development Plan.

This document is meant to cover all the software test phases in the software lifecycle, including unit, integration and system testing.

<<this is an example of a single test plan covering everything. If the project is large or if different organizations are responsible for different stages or types of testing then details relevant to specific organizations could be separated into additional plans. Another approach would be to create specific test plans for unit test, integration test and system test or to create specific test plans for specific portions of the project, or stages of the lifecycle, and refine this into a strategy level plan intended to tie it all together to ensure coverage and minimize unnecessary redundancy. >>

1.2 Scope

The <device> is meant to<< Describe intended use of the device, >>

The software in the <device> executes the following tasks..... <<Describe shortly the main software tasks >>

This plan is defining only the testing activities related to software testing. Hardware testing, overall system validation and usability testing are outside the scope of this plan.

<<detail all the exclusions.>>

Main users of this document are the software development team and the software test team to plan their testing activities.

The intended audience of this plan includes also..... << detail all the functions or groups that should have access to this document.>>

1.3 Definitions, acronyms, and abbreviations

Black box (Functional testing)	Testing that ignores the internal mechanism or structure of a system or component and focuses on the outputs generated in response to selected inputs and execution of the system.
STP	Software Test Plan
TC	Test Case
White box (Structural Testing)	Testing that takes into account the internal mechanism or structure of a system or component.

<< Define any acronyms, abbreviations, or terms used in the document that may be unfamiliar to readers. If a project level definitions document exists, it can be referenced here and this section limited to specific terms used only in this document. >>

1.4 References

Standards:

IEC 62304:2006, Medical device software—Software life cycle processes.

829-1998 IEEE Standard for Software Test Documentation.

.....

<< Include any **relevant** standard >>

Project Documents:

<device> Software Development and Quality Plan, Rev. a.b,<<you could add any information useful to locate the document>>
<device> Software Requirement Specifications, Rev. x.y,<<you could add any information useful to locate the document>>
.....
<< Include **relevant** project documents >>

1.5 Overview

This document is organized in 17 sections.
Section 1 (this section) puts the document into its context and gives an overview of it.
Section 2 describes the items that will undergo testing under this plan.
Section 3 lists the features that will be tested.
Section 4 lists the features that will not be tested in each release.
Section 5 details the test approach, including applied types of testing and testing strategy
Section 6 describes how test coverage is documented and verified.
Section 7 indicates the item pass/fail criteria.
Section 8 outlines how decision is taken to suspend or resume testing
Section 9 lists the documents delivered during and at the end of testing activities
Section 10 define how testing is accomplished.
Section 11 defines the environment where test is executed
Section 12 lists the processes that are needed to support testing activities
Section 13 outlines testing responsibilities.
Section 14 defines test staff and their experience
Section 15 gives an outline of the schedule
Section 16 describes possible risks to the plan and how they are addressed
Section 17 (appendices) provides some forms to use to support test activities
<< Add anything apt to describe content and organization of this document >>

2 Test items

The following items are subject to test under this plan:
<device> firmware.....<<describe>>
<device> software.....<<describe>>
<device> user documentation.....<<describe>>
.....
<< list all the items that will undergo testing according to this plan >>

The following documents are used for development of specific test cases:
<device> Software Requirement Specification.
<device> Software Risk Analysis.
<device> Software Design Specification.
.....
<< list all the documents that will be used to develop the tests >>

3 Features to be tested

All software features that will be tested prior to final release, including:

- User Interface
- Device Control
- Data Acquisition and reduction
- Calibration
- Error handling and recovery
- Startup and Shutdown
-

<< list all main features that will undergo testing according to this plan. Note that, even if all software features are to be tested, writing down the list is useful to verify feature coverage. >>

Extra focus shall be on hazards and risk control measures, faults and recovery, extreme conditions.

4 Features not to be tested

All features will be tested prior to final release, however the software development plan defines several partial releases, to be used for internal and field verification, before the final one. For this reason, some features will be excluded from integration and/or system testing of the partial releases as follows.

Release Alpha1:
Release Alpha2:
Release Beta1:
.....

<< List here the planned releases with the exclusions. Note that exclusions could be related to entire features or to some types of testing. The list of features could also be kept in an other document (e.g. a integration plan document) and you could refer to it instead of listing the features here. >>

5 Overall Approach

The approach to testing the software includes: Unit, Integration and System Test.

The Unit and Integration testing may overlap with System Test or be executed concurrently. Depending on the software development and integration plan, some software items could proceed through system test, while others are still under development.

Test cases will be developed for Unit, Integration, and System test that specify the steps to perform and the expected results. Generally, test case steps will also define the actions to perform to obtain objective evidence that the test passed, such as screen shots or printouts of pertinent files, to include in the test report. These test cases will be reviewed and approved by..... before they are formally executed.

<< Give any useful detail on the approach and define the approval line >>

As extensive coverage as possible will be applied, while trying to avoid redundancy, except for critical software items, as identified through the risk analysis, where some level of redundancy is desirable.

Verification of test coverage will be accomplished by verifying the following in trace matrices:

- Coverage of requirements
- Coverage of test types
- Coverage of change requests or problem reports that turn into software modifications
- Coverage of Risk Control Measures

Device testing may be used to cover certain software functionality if explicitly traced to it.

<< Detail how coverage will be achieved and documented. >>

5.1 Test Types

The following subsections describe different types of testing to be performed. These subsections are in lieu of separate test designs. Test cases will be written to detail the steps and methods to be employed in accomplishing each type of testing. There may be multiple test cases for a given test type or one test may apply to multiple test types. A series of cross references and trace matrices are provided in Appendix D.

<< Note: if any of the following types of testing are not applicable, it is better to leave the related subsection indicating “not applicable” instead of removing it. This way you show that some thought to it has been given. >>

5.1.1 Hazard

All software hazards and risk control measures identified in the risk analysis will be used as a reference to ensure that testing covers and focuses extra testing on significant hazard conditions and associated risk control measures.

Test cases will be generated to attempt to induce hazards both by detecting improper normal operation and by generating fault conditions that the software is expected to detect.

Testing of hazards shall be applied at all levels, by using both black box and white box testing, since some of the hazardous situations could not be easily created at system level.

<< describe how testing of hazards is accomplished >>

5.1.2 Negative

The software will be tested to ensure that it responds to unexpected or erroneous user inputs or to unexpected missing functionality in an acceptable way.

<< describe with some level of detail in your application what the negative testing would be, e.g. erroneous keystrokes, loss of communication during operation, etc. >>

5.1.3 Error Message Handling

Test cases will be developed to induce error conditions and present all the error messages defined in the specs (alarms, warnings, etc.).

<< describe how this testing will be accomplished. This may overlap some of the other test types (hazard, negative, fault, etc.), but it is important to verify that all error messages are properly presented. >>

5.1.4 Fault

Test cases will be developed to verify that the software properly handles the error conditions.

<< Describe at high level the different types of errors that will be tested, e.g. loss of communication, improper calibration, error on motor control, etc. >>

5.1.5 Stress

Test cases will be developed that are likely to “stress” software operation. Some of these tests will be based on functional knowledge of the instrument, some will be based on knowledge of the internal software design, and some will be based on conditions that tend to stress software in devices in general. These stress conditions will include but not be limited to:

1. Fast & extended key presses during critical operations
2. Power failure
3. Running (potentially real) cycles with the maximum number of concurrent software tasks or ISRs executing
4. Variations in system voltage and power surges
5. Extended running without shutdown to accomplish a software life test to uncover counter rollovers, stack overflow, memory leaks, and other anomalies that exhibit symptoms only after repeated execution without reset or power-up.
6.

<< Add/Remove stress conditions as applicable according to specificity of the device under test >>

5.1.6 Installation and Setup

Test cases will be developed to verify the installation and configuration sequence both on hardware where software has never been installed and on hardware where a previous installation has been performed with a different setup. Different configurations will be verified.

<< Describe any specific installation condition to test on your device >>

5.1.7 Startup and Shutdown

Test cases will be developed to verify proper operation at startup and shutdown and that a series of startups and shutdowns results in consistent normal operation.

<< Define startup/shutdown conditions to verify according to specific knowledge of the behavior of your device, e.g. testing startup interruption at various stages, testing power failure recovery under different condition, testing battery replacement, etc. >>

5.1.8 Regression

For any new release of the software implementing changes or additions during the test period, an impact analysis of the changes and new implemented features will be performed to determine the tests that need to be repeated unless a full regression of rerunning all test cases will be performed. If a full regression is not performed, a subset of the tests will be selected and re-run. The test summary report will contain the rationale for re-running only selected tests.

<< Add/Replace as applicable with your specific regression policy >>

5.1.9 Platform

Testing will be performed on instruments produced using a variety of lots of components wherever possible. After Beta release new releases will be tested on a variety of instruments including new manufactured units and previously manufactured units.

<< Add/Replace as applicable with your specific platform testing policy, depending on your device hardware and software. In some cases, also use of different lots of disposables could be important to uncover problems >>

5.1.10 Workflow

Test cases will be developed to verify that the expected user workflow can be followed without error.

<< Add/Replace as applicable with your specific workflow test policy. In some cases a few different workflows can be defined and tested accordingly >>

5.1.11 Functional

The <device> Software Requirements Specification and the <device> System Specification will be used to generate test cases (or portions of the test cases) to verify that all required software operations are functional and behave as intended. This will include all documented aspects of system behavior including the user interface and calibration.

<< Add/Replace as applicable with your specific description of functional test >>

5.1.12 Documentation

The portions of the user and service manuals related to software functions will be tested to ensure that the documentation and the software are consistent and that the documentation is clear and easily followed.

<< Add/Replace as applicable with your specific description of documentation test >>

5.1.13 Algorithm

Test cases will be developed to verify that the algorithms used for are correct.

<< Describe specific testing related to the device algorithms. Describe testing techniques used: these could include tracing partial and final results, the use of simulation, the construction of defined data files, etc. Some of this types of testing could only be performed white box >>

5.1.14 Static

Code inspection techniques will be applied to 100% of the developed code. Any subsequent change will be re-inspected. See the <device> Software Development and Quality Plan for additional details on application of static testing.

<< Add/Replace as applicable with your specific description of static test. Walkthroughs could be used in some cases in place of formal code inspection meetings. In some cases application of code inspection could be limited to specific software items or units, with a choice based on risk analysis results. >>

5.1.15 Multiuser/external access

Test cases will be developed to verify the possibility of concurrent access to software functionalities.....

<< If applicable to <device>, describe how the test of concurrent access will be conducted >>

5.1.16 Security

Test cases will be developed to verify that password protection of patient information is working as expected and cannot be broken from any of the possible access ways.

<< Add/Replace as applicable with your specific description of security test, based on security options of <device>. >>

5.1.17 Data Integrity

Test cases will be developed to check the protections against corruption of critical data, such as

<< Describe as appropriate the testing of data protection techniques used in <device>, such as CRC on critical areas of the database, redundant data storage, etc., as applicable. >>

5.1.18 Ad hoc

Ad hoc testing can be considered exploratory or creative challenge testing, It is the least formal of test methods.

<< Ad hoc testing has been criticized because it isn't structured, but this can also be a strength: important things can be found quickly. It is performed with improvisation, the tester seeks to find bugs with any means that seem appropriate. Completely informal ad hoc testing could be used early in the development lifecycle and then formalized or eliminated in later stages . There are also degrees of ad hoc testing. For instance usability testing often requires test cases that are very general and not very detailed so that users are free to do things their own way and is best done at least partially in ad hoc manner. Since all testing is a sampling use of some amount of ad hoc testing can prevent too narrow a perspective in test design.>>.

Trained and un-trained staff will be asked to try and make the software fail. Although ad hoc and not following a specified protocol the results of ad hoc testing could be formally documented – and must be if used for test coverage. Some of the ad hoc tests could also be formalized for future use as structured test cases.

<< Add/Replace as applicable with your specific description of ad hoc test. >>

5.2 Unit Testing

Unit testing focuses on discrete parts of the software and uses knowledge of the inner workings of the software to determine correct functionality. Unit tests may be specific structural test cases, or may employ code inspections to perform static testing.

5.2.1 Strategy and approach

Unit test shall be executed white box.....

Unit test shall mainly be based on unit detailed design as described in <device> Software Design Specification.....

Unit test shall be applied to 100% of the software items.....

Any implemented changes shall be unit tested.....

Unit test shall be developed and executed under the responsibility of the Software Development team.....

The TestItAll test tool shall be used to develop and run unit tests in a simulated environment

Any uncovered anomaly shall be handled through the Problem Resolution Process as appropriate for this phase of the project.....

<< Add/Replace as appropriate with your own strategy and approach to unit test. Just to give some examples:

- In some cases, unit test could apply to a mix of specific structural test cases and application of code inspection techniques.
- In some case unit test could be applied only to critical units identified through the results of software risk analysis
- In some cases unit test could be applied by groups different from the implementers of the code.
- In some cases instead of using a simulated environment, the unit code could be tested in the target environment by using dedicated stimulus code and such tools as in-circuit emulators.

>>

5.2.2 Documentation

Formal structural test cases shall be developed with the use of TestItAll test tool, which shall also provide the ability to define specific stimulus data files. << If the same test requirements apply to all or a subset of

software units/items a single test case or procedure could be defined to be run on all such units rather than separate unique test cases for each. >>

The tool shall also record the test results for each test run.

Test cases, test data files and test result files shall be archived.....

All the test documentation shall be reviewed and approved by

<< Add/Replace as appropriate with your own unit test documentation production and handling. It is important that some formal documentation, proving the application of unit testing, is kept in the project file. It is also important that the documentation shows that the appropriate sequence in the testing tasks has been applied: unit test of one unit should be applied prior to integration and integration testing of the same unit, although it is not required that all software units are unit tested, before integration and system testing are started. Parallelism of development and testing tasks is allowed, where some subsystems are integrated and tested, while other subsystems are still being designed or are under development. >>

5.3 Integration testing

Integration testing consists of tests that determine that the various parts of the system are working together.

5.3.1 Strategy and approach

Integration of the different units into subsystems and the final complete system shall be performed and verified according to the software integration plan.....(see)

Integration of each subsystem shall be tested.....

Integration testing shall focus especially on verification of the interfaces between the units and between subsystems and on timing/concurrency requirements.....

Integration testing shall be developed and executed under the responsibility of the Software Test team.....

The following test tools shall be used to develop and execute integration testing.....

The integration testing shall be executed in the target environment.....<< or if not, explain how and why that is adequate >>

Any uncovered anomaly shall be handled through the Problem Resolution Process as appropriate for this phase of the project.....

Integration testing shall include some regression testing to be repeated after changes are implemented.....<< Add/Replace as appropriate with your own strategy and approach to integration test. Just to give some examples:

- In some cases the integration could be incremental and the test would be designed accordingly.
- In some cases the integration testing could be combined with the system testing in a single testing strategy.
- In some cases integration testing at subsystem level could be performed in simulated environment.

>>

5.3.2 Documentation

Integration test cases shall be developed with the use of

Stimuli shall be produced through.....

Test cases, test data files and test result files shall be archived.....

All the test documentation shall be reviewed and approved by

<< Add/Replace as appropriate with your own integration test documentation production and handling. It is important that all formal documentation, proving the application of integration testing, is kept in the project file. It is also important that the documentation shows that the appropriate sequence in the testing tasks has been applied. Test result documentation should also identify the tester, the equipment and test environment used for testing. >>

5.4 System Testing

System Test consists of functional, user-oriented testing to verify that the functions of the software meet the requirements.

5.4.1 Strategy and Approach

Software System testing shall be mainly based on <device> Software Requirement Specification and on <device> Software Risk Analysis.....

All software requirements shall be covered. Requirements shall be covered as much as possible through system testing, but some requirements, especially related to fault conditions, could only be covered at unit level.....

System testing shall be developed and executed under the responsibility of the Software Test team.....

The following test tools shall be used to develop and execute system testing.....

The system testing shall be executed in the target environment.....

<< In case of incremental integration>> System test on the integrated subsystems/features shall start as soon as integration is positively accomplished and verified, according to the integration plan, without waiting for the entire system to be fully integrated.....

Any uncovered anomaly shall be handled through the Problem Resolution Process as appropriate for this phase of the project.....

System testing shall include some regression testing to be repeated after changes are implemented.....

<< Add/Replace as appropriate with your own strategy and approach to system test. >>

5.4.2 Documentation

System test cases shall be developed with the use of

Stimuli shall be produced through.....

Test cases, test data files and test result files shall be archived.....

All the test documentation shall be reviewed and approved by

<< Add/Replace as appropriate with your own system test documentation production and handling. It is important that all formal documentation, proving the application of system testing and its coverage, is kept in the project file. It is also important that the documentation shows that the appropriate sequence in the testing tasks has been applied. Test result documentation should also identify the tester, the equipment and test environment used for testing. >>

5.5 Validation (user) Testing

<<Detail the purpose, scope and type of user and usability testing and what aspects are covered under this plan versus overall device validation testing.>>

5.6 Test Summary

A test summary report, describing the test activities performed in the different sessions, their results and the uncovered anomalies and their disposition, will be created using the outline provided in Appendix B.

6 Test Coverage

Test coverage will be demonstrated through different types of traceability tables, as detailed below.

The format presented in Appendix D will be used.

<< Replace with your own traceability scheme. Traceability could be defined also in other project documents, e.g. the Software Development and QA Plan. In this case, just reference the appropriate document. >>

6.1 Physical Coverage

Each software item will be covered at least by one test case, either unit, integration or system test, although at this level traceability will mainly be at unit level.

<< Describe your own physical coverage policy >>

6.2 Requirements to test coverage

Each numbered requirement from the following documents will be covered at least by one test case, either unit, integration or system test:

- <device> Software Requirement Specification

-
[<< Describe your own requirements coverage policy >>](#)

6.3 Risk Control Measure to test coverage

Each numbered RCM from the following documents will be covered at least by one test case, either unit, integration or system test:

- <device> Software Risk Analysis Report
-
[<< Describe your own RCM coverage policy >>](#)

6.4 Test types to test coverage

A specific traceability table will demonstrate coverage of the test types listed in section 5.1.

7 Item pass/fail criteria

Each test case will determine its own pass/fail criteria which will be defined as the “expected results” for each step. Generally, if a step fails the entire test fails and if all steps pass, the test passes.

Test cases will be reviewed and approved by the <TBD> or designee before execution.

Each executed test case will be reviewed by the <TBD> or designee and a final test pass/fail disposition will be determined.

Identified anomalies will be handled through the Problem Resolution Process.

[<< Add/Replace as appropriate with your own pass/fail criteria. >>](#)

Test case variances – if the tester needs to deviate, if the variance is minor and does not affect the purpose of the test, or if the tester decides to add steps, then the tester can proceed and document the variance for subsequent review and possible approval for passing the test despite the variance. Rationale shall be documented and this should only occur on an occasional basis at least in later stages of the life cycle when working with mature test cases.

[<< Add/Replace with your own test case variances handling >>](#)

8 Start, Suspension and Resumption requirements

Unit test will start on a developed units after the code has been reviewed (code inspection or walkthrough) and the developer has updated it according to the review results.

[<< Add/Replace with your own unit test start policy >>](#)

Integration and system test on an integrated software item will start when the responsible for the integration formally releases the item and the software build has been completed and properly documented according to the software release policy.....

All the software items that are part of the integrated software item and the integrated software item itself must be put under configuration control and properly labeled.....

[<< Add/Replace with your own integration and system test start policy. >>](#)

Execution of each test case will continue to completion unless anomalies detected do not allow to proceed or have side affects that would invalidate further testing or the expected fix is broad enough that it would invalidate further testing anyway.

[<< Add/Replace with your own test suspension criteria >>](#)

At test resumption after a fix, at a minimum tests related to the specific items found defective will be repeated and the regression test will be repeated. Other tests may be added based on internal knowledge of the fix itself.

[<< Add/Replace with your own test resumption criteria >>](#)

9 Test deliverables

The records of testing will include:

1. The final Software Test Plan for the release
2. Test Cases
3. Test Records – include executed test case reports, samples or sample references, measurements from independent test records, output from the instrument display, etc. as indicated in the test case,.....
4. Test Summary - a summary of testing and release decisions and the rationale for each release.
5. Problem reports
6. Code Review Report

<< Add/Replace with your own deliverables >>

10 Testing tasks

Testing tasks include:

- Test planning and management
- Test case design
- Test execution
- Test results recording and review

Test planning is documented in this plan.

Test management includes:

- administrative work procuring and coordinating staff and equipment for test execution
- training of staff involved in testing in test procedures and test documentation requirements
- coordination of testing to ensure detailed test cases are developed to support each type of testing indicated in this plan and to avoid redundancies and waste of efforts.
- review of testing to ensure procedures are followed and documentation conforms with standards
- review of testing and anomaly rates to assess status and adequacy and to make ongoing decisions regarding the need for retesting or creation of additional tests

Test case design is the detailed planning and documentation of test steps including documentation of expected results (including how they are independently determined) where relevant. The level of detail of test protocols will be appropriate for the experience and knowledge level of the staff to be involved in execution of the test.

Test case execution may be performed by the test case designer himself or by another tester. If the staff member that executes a test is not sufficiently trained to determine whether the test was successful, the test results (e.g., a sample) will be reviewed by an appropriately trained staff member.

The test will be documented to allow its repetition and the results will be recorded, including all identified anomalies. The need to subsequently repeat the test will be determined based on the results of the first test and the degree to which software changes are made that could affect the outcome of the test.

11 Environmental needs

<< Describe test environment, including office space, special test equipment, any simulation tools, number and types of instruments, hardware revision control during testing. >>

12 Supporting Software and processes

12.1 Software

The Software Data Acquisition simulation tool SimulaAll rev. x.y will be used for.....

.....

<< List and describe all software used to support the testing activities, including if necessary spreadsheet and document editing tools. Indicate also revisions as appropriate. >>

12.2 Processes

The following procedures will be followed during testing:

12.2.1 Documentation Configuration Control

<< Describe the Documentation Configuration Control process or reference the project document or SOP where it is defined. >>

12.2.2 Software Configuration Management

<< Describe the Software Configuration Management process or reference the project document or SOP where it is defined. >>

12.2.3 Issue Tracking and Problem Reporting problem resolution process

The Software Quality Assurance plan requires that formal issue tracking begin in the xx phase. From that point forward all issues found during testing will be entered into the issue tracking system. This includes issues related to specific test cases as well as any others found serendipitously. For each new release submitted for formal test the list of resolved issues will be reviewed to determine appropriate regression testing.

<< Describe the Problem Resolution Process used to track anomalies uncovered in the different phases of testing or reference the project document where it is defined. >>

12.2.4 Change control

<< Describe the Change Control process or reference the project document or SOP where it is defined. >>

13 Responsibilities

The Software Project Leader is responsible for assigning test staff to the test activities, preparing plans and coordinating test and test reviews, anomaly tracking and resolution....

The developer of the unit is responsible for unit test development and execution.....

The <TBD> are responsible for formally defining, executing, and recording integration and system tests.....

<< Add/Replace as appropriate with your own defined responsibilities for the different test tasks. >>

14 Staffing and training needs

<TBD> Test engineers are required to..... << Define your own staff requirements and map them to the phases of the project. If staff requirements are defined elsewhere, e.g. the <device> Software Development and QA Plan, just reference the appropriate document >>

All staff used as testers must be trained in proper test procedures including documentation and problem reporting. << Add/Replace with your own training requirements >>

15 Schedule

The exact dates of testing will vary as needed and be tracked and controlled separately from this document. Test cases will be approved prior to execution, but the order of running specific unit, integration, or system tests is not pre-determined, but will be based on efficiency decisions with the objective to start testing as early as possible. As soon as a software item is ready for test, testing on that software item will start accordingly to resource availability and priority assignments.

If all unit, integration and system tests are not run on the final version of the software, then appropriate regression analysis will be done. In that case, the regression analysis rationale and the regression testing that is performed will be discussed in the Test Summary document.

Expected testing dates are as follows:

.....

<< Add/Replace as appropriate. Insert here schedule details or reference external schedule documents. >>

16 Risks and contingencies

<< Describe any risk to the test activities that could affect performance, accuracy, schedule, quality, etc. and define how to minimize or address them. >>

17 Appendices

17.1 Appendix A - Test Case Format

<< The following is an example template that can be used to develop test cases. Some example information is included to show how to fill in the template. You can modify it as fit your application or use one of your own. It is important to indicate the following information:

- Type of tests
- Test Steps, including actions, expected results, information on what objective evidence should go in the report
- Reference to supporting documentation (e.g. Requirement Number, RCM number)
- Signature and date of approval.

Note: A test case could be of multiple types (unit, integration, and system) >>

<device> Test Case Protocol– <TC name>

VERSION:	<TC version number>	DATE LAST MODIFIED:	<date>
S/W TEST TYPE:	<input type="checkbox"/> Unit Test <input type="checkbox"/> Integration Test <input checked="" type="checkbox"/> System Test	MODIFIED BY:	<name of staff member that developed or modified the TC>
TEST TITLE:	Calibration Drift check << this is just an example >>		
TEST PURPOSE :	This test verifies calibration drift handling under the following conditions:		
CHECK ALL BLOCKS THAT APPLY	Test verifies: <input checked="" type="checkbox"/> SW interface/Workflow <input type="checkbox"/> Local data structures <input type="checkbox"/> Events and states <input checked="" type="checkbox"/> Boundary/limit conditions <input type="checkbox"/> Installation/Platform <input checked="" type="checkbox"/> Algorithms	<input type="checkbox"/> Stress <input type="checkbox"/> Startup/Shutdown <input type="checkbox"/> Documentation <input checked="" type="checkbox"/> Errors/Faults/Diag <input type="checkbox"/> Regression <input type="checkbox"/> Security <input type="checkbox"/> _____	Functionality Under Test: <input checked="" type="checkbox"/> SRS Functional <input type="checkbox"/> SDD <input checked="" type="checkbox"/> RCM <input type="checkbox"/> Special Test Types

DEPENDENCIES <<all actions required to place the system in the proper state before executing the test >>	
Hardware Preparation:	Standard Instrument. No need of calibrants (simulation is used)
Software Preparation:	Activate the acquisition simulation tool
Other / Set-up:	The instrument must be on, warm up completed, uncalibrated.
SUPPORTING LINKS	
Document References:	<< identify here the documents on which this TC is based (SRS, RA,) >>
Automated Scripts:	<< list here any test script used to automatically run this TC >>
SRS IDs /RCM IDs /Change IDs	<< list here any referenced requirement or RCM ID numbers, if the TC is testing changes reference the change ID >>
Test Data Sets:	CalSim1 (calibration simulation) << list any set used for this TC >>

Signatures	
Approval:	<< Print Name of person reviewing and authorizing use of this TC, date and sign >>

TEST Inputs and Expected Results				
Step	Actions	Expected Results	Objective evidence	Reference
Execute calibration << this is just an header to document the purpose of the following steps >>				
1.	Preparation steps: Start acquisition simulation using file CalSim1	<< for preparation steps, output does not need to be monitored >>		<< pointers to related SRS/RCM >>
2.	Press the “CALIBRATE” soft key. At the same time start the stopwatch.	Calibration is started. “CALIBRATION IN PROGRESS” is presented with a “STOP” soft key.		SRS-Calxx
3.	Look at screen and stop the stopwatch when the “CALIBRATION IN PROGRESS” disappears.	After 20±1 sec from start, calibration is completed. “READY” is presented.	Write down the stopwatch count. Execute “Print Screen” and attach the printout to test result form.	SRS-Calyy
Cause calibration drift and verify handling.				
4.	Preparation steps: Start acquisition simulation using file CalDrift1. At the same time start the stopwatch.			
5.	Look at screen and stop the stopwatch when the “READY” disappears	After 15±1 sec, “READY” is replaced by error message “CAL DRIFT xxxxx”. An acoustic alarm of type yyyyyy is activated.	Write down the stopwatch count. Execute “Print Screen” and keep the printout. Register the alarm sound as MP3 file and refer it in test result form.	SRS-Calzz RCMabc
6.				
7.				
8.				
9.				
10.				
11.				
12.				
13.				
End of Test Case.				

<< Note that the test step definitions must be more or less detailed, based on the purpose and tester knowledge and training . If it is expected to be used by testers very experienced with the application, less detail may be required, if it is expected to be used by inexperienced people, it is necessary to add more details (e.g. it would not be enough to write “Start acquisition simulation using file CalSim1”, but it would be necessary to specify how to accomplish this action or reference some document specifying it). If the exact test step is critical to the purpose of the test (versus just getting to the right place) it would need to be more detailed than if not. >>

<< The following is an example of test result form. Test run information could be entered and objective evidence could be added either by attaching printouts or by referencing archived files. Use/modify it or use your own. >>

Test Case ID _____ Version _____ Tester Name/Initials _____
Signature _____ Date _____

Instructions: This form should be stapled to a printed copy of the Test Case that was executed. Other hard copy evidence should also be attached, if any. Draw a bold horizontal line after the entries for each test input/step.		TEST CASE PASSED?		YES	NO
		Reviewed By: _____ Date Reviewed: ____/____/____			
Software Revs: Equipment ID: Include calibrated test equipment identifier as well if relevant.					
Comments/ Deviations:					
TC Step #	<u>Actual Results</u> ("Set-up" steps (if any) can be marked as <u>Done</u> with no results if there are no expected results relevant to passing/failing test. All actual values for variable responses and any specific responses not listed in procedure shall be recorded as expected results. When the procedure completely characterizes a non-variable response and the actual response matches, the "as expected" box may be used.) <u>As</u> Enter actual response here or check "As Expected or <u>Expected</u> <u>reference additional test evidence.</u>			<u>Status</u> (Check one column per step.) <u>Done</u> <u>Pass</u> <u>Fail</u>	

Test Case ID _____ Version _____ Tester Name/Initials _____
 Signature _____ Date _____

PAGE _____

TC Step #	<u>Actual Results</u>		<u>Status</u>		
	<u>As</u> <u>Expected</u>	<u>Enter actual response here or check "As Expected" or</u> <u>reference additional test evidence.</u>	<u>Done</u>	<u>Pass</u>	<u>Fail</u>

17.2 Appendix B – Test Summary Outline

<< The following is an example of a test summary outline that can be used to report the result of a test session >>

TEST SUMMARY ID:

1 Testing Summary:

<< Provides overview of testing performed including specific items tested (e.g., sequence compiler, embedded code, user manual) and a summary of actual test activities.
Provide a list of specific test cases executed and references to associated test documentation.>>

1.1 SW Release xxx.yy

1.1.1 Overview

<< High level description of testing activities performed on the release, including period of test, staff involved in testing, etc. >>

1.1.2 Unit test

<< The following is just an example of organization of this information >>

The following table lists the unit test activities performed on the release:

TC executed	Result	Reference Documentation	Items tested	Anomalies
<< name of TC >>	<< pass/fail >>	<< pointers to doc supporting test run>>	<< list of items tested>>	<<IDs of anomalies detected>>

1.1.3 Integration test

The following table lists the integration test activities performed on the release:

TC executed	Result	Reference Documentation	Items tested	Anomalies
<< name of TC >>	<< pass/fail >>	<< pointers to doc supporting test run>>	<< list of items tested>>	<<IDs of anomalies detected>>

1.1.4 System test

The following table lists the system test activities performed on the release:

TC executed	Result	Reference Documentation	Items tested	Anomalies
<< name of TC >>	<< pass/fail >>	<< pointers to doc supporting test run>>	<< list of items tested>>	<<IDs of anomalies detected>>

1.1.5 Regression test

<< Describe regression test executed and rationale for it or reference any impact analysis documents supporting the decision on regression testing. >>

1.2 SW Release xxx.zz

1.2.1 Overview

.....

1.2.2 Unit test

.....

1.2.3 Integration test

.....

1.2.4 System test

.....

1.2.5 Regression test

.....

2 Anomalies, Variances, & Incidents:

<< Lists anomalies found & fixed, open, or unexplained incidents and describe or reference their resolutions or further course of action.

List any variance from test steps and justification. Variances unrelated to the purpose of the test might on review be considered acceptable on an exceptional basis. >>

2.1 SW Release xxx.yy

2.1.1 Anomalies

<< The following is just an example of organization of this information >>

The following table lists the anomalies identified by testing on the release:

Anomaly ID	Description	Severity	Resolution
<< anomaly ID>>	<< short description or just title. The full description will be in the issue tracking system >>	<< according to risk analysis ratings >>	<< e.g. fixed on release xxx.zz, not to be fixed, fix postponed to, >>

2.1.2 Variances

<< The following is just an example of organization of this information >>

The following table lists the variances on test steps applied during testing on the release:

Test Case	Variance Description	Reason for variance and its acceptability

2.1.3 Incidents

<< The following is just an example of organization of this information >>

The following table lists the incidents during testing on the release:

Incident Description	Subsequent Investigation	Further proposed action

2.1.4 Anomaly analysis

<< Provide a general anomaly analysis and assessment of anomaly rates from a reliability perspective.>>

2.2 SW Release xxx.zz

2.2.1 Anomalies

.....

2.2.2 Variances

.....

2.2.3 Incidents

.....

2.2.4 Anomaly analysis

.....

3 Comprehensiveness Assessment

<< Identify aspects of the system not fully addressed during this test period or where the experience during this test period indicates further testing is needed.

On final release, if the entire set of testing has not been repeated, justify why the results of test sessions on previous intermediate releases have been considered acceptable to proceed with the release. >>

4 Evaluation

<< Provide an overall evaluation of the results of testing and whether the software has attained the reliability required to move to the next test period or release. List all known remaining anomalies and provide individual rationales or an overall rationale for whether or not their existence is acceptable for release or progression to the next phase. >>

4.1 Open anomalies

<< The following is just an example of organization of this information >>

The following table lists the anomalies still open:

Anomaly ID	Description	Severity	Rationale for acceptability
<< anomaly ID>>	<< The description here should provide enough detail to understand the impact of the anomaly >>	<< according to risk analysis ratings >>	<< give a detailed rationale, based on risk evaluation, of why it is acceptable to proceed with this anomaly still present >>

4.2 Overall evaluation

.....

Approvals:

<< Names, signatures and dates of people reviewing and approving the test summary >>

17.3 Appendix C – Test List

<< The following is an example of a table listing the test cases developed with their purpose >>

Test Case	Type of Test – Unit, Integration, System
<< Fill in with Test Case name and short description >>	<< Indicate test type >>

17.4 Appendix D - Trace Matrices

<< The following is an example of a table tracing software items to related testing (physical coverage). Granularity of coverage table can be selected according to your policy. A software item could be a software module, or a software function, or whatever makes sense. Higher granularity has the advantage of making easier the impact analysis of changes, but has the drawback to making more cumbersome the maintenance of the traces >>

Item	Description	Test type	Test Case(s)
<name of the item>	<short description>	<static, unit, integration, system>	<TC name>

<< The following is an example of table tracing requirements to related testing (functional coverage). >>

Software requirement	Test Cases
Calibration Verification	
SRS-Calxxx	CAL1
SRS-Calyyy	CAL1
SRS-Calzzz	CAL2
Data Acquisition	
SRS-DAxxx	ACQ1
SRS-DAyyy	ACQ2
SRS-DAzzz	ACQ1

<< The following is an example of table tracing RCM to related testing (risk control coverage). >>

Software Risk Analysis Item	Test Cases
System calibration drift alarm	CAL3

<< The following is an example of table tracing test types to related test cases (test types coverage) >>

Test Type	Test Cases
1. - Hazard	[see RCM table]
2. - Negative	<< list TC names >>
3. - Fault	<< list TC names >>
4. - Stress	<< list TC names >>
5. - Startup & Shutdown	<< list TC names >>
6. - Functional	<< list TC names >>
7. - Documentation	<< list TC names >>
8. - Ad hoc	<< list TC names >>
9.	
10.	
11.	
12.	
13.	

SOFTWARE REQUIREMENTS SPECIFICATION

TEMPLATE

DRAFT

Revision History

Date	Revision	Author	Description

Adaptation of IEEE Std. 830-1998 Recommended Practice for Software Requirements Specifications.

Note: This template is conceived for the case where the project is small enough to grant for a single SRS document. There are cases where the project is large and it is impractical to keep all the software requirements in a single document. You could then choose to have a logical partitioning of the system and distribute the requirements in several documents, each of them covering for example a functional area.

Alternatively there could be one SRS at a high level and several detailed SRSs or functional specifications . In this case this template requires some adaptations.

Explanatory comments are included in << comment >>.

Other text is example definition that you should replace with your own text or consider for pros and cons of the approach and suitability. This is not intended for blindly filling in the blanks.

This is not a complete SRS, just a training example to guide in the development of a SRS for a certain type of device.

TABLE OF CONTENTS

1	Introduction.....	3
1.1	Purpose.....	3
1.2	Scope.....	3
1.3	Definitions, acronyms, and abbreviations.....	3
1.4	References.....	3
1.5	Overview.....	4
2	Overall Description.....	4
2.1	Product Perspective.....	4
2.2	Software Functions	4
2.3	User Characteristics	4
2.4	Constraints	4
2.5	Assumptions and Dependencies	5
2.6	Requirement Distribution.....	5
3	Use cases/user workflows.....	5
4	Specific Requirements	5
4.1	External Interface Requirements.....	5
4.1.1	User Interfaces	5
4.1.1.1	General requirements	6
4.1.1.2	Input devices	6
4.1.1.2.1	Purpose.....	6
4.1.1.2.2	Functional Requirements	6
4.1.1.1	Display	6
4.1.1.1.1	Purpose.....	6
4.1.1.1.2	Functional Requirements	6
4.1.2	Hardware Interfaces	6
4.1.3	Communications Interfaces	7
4.1.4	Hardware and Software Platform.....	7
4.2	System Features	7
4.2.1	Motor 1 Control	7
4.2.1.1	Purpose.....	7
4.2.1.2	Functional Requirements	8
4.2.1.3	Stimulus Response Sequence.....	8
4.2.2	Feature 2.....	8
4.2.2.1	Purpose.....	8
4.2.2.2	Functional Requirements	8
4.2.2.3	Stimulus Response Sequence.....	8
4.3	Performance Requirements.....	8

4.4	Safety Requirements	9
4.5	Software System Attributes	9
4.6	Other Non-Functional Requirements	9
5	Appendices.....	9
5.1	Appendix A: Data dictionary	9
5.2	Appendix B: Alarm/Messages dictionary	9
5.3	Appendix C: List of TBD	9

1 Introduction

1.1 Purpose

Purpose of this Software Requirements Specification is to detail the software requirements for the <TBD> product.

The software requirements are derived from the system requirements defined in <TBD> Product Specification.

The document is meant to drive the software design and implementation activity and is the main reference for the software system level test activity.

<<Identify the purpose of this document and its intended audience>>

1.2 Scope

This Software Requirement Specification specifies requirements for <TBD> device software rev. A1.

The <TBD> device is meant to

This spec applies to the software embedded into the device to control its operations based on user input and patient information acquired from specific sensors.....

This spec does not applies to any external system software interfaced to the <TBD> device.....

<<Identify with name and details the software product(s) that this SRS document is specifying.

Give an high level description of what the software product will do and, if necessary, will not do.>>

1.3 Definitions, acronyms, and abbreviations

SRS	Software Requirements Specification
TBD	To Be Defined

<< Define any acronyms, abbreviations, or terms used in the document that may be unfamiliar to readers. If a project level definitions document exists, it can be referenced here and this section limited to specific terms used only in this document. >>

1.4 References

Standards:

IEEE Std. 830-1998 Recommended Practice for Software Requirements Specifications

.....

<< Include also any **relevant** standard >>

Project Documents:

<TBD> Product Requirement Specifications, Rev. 1.1,<<you could add any information useful to locate the document>>

.....

<< Include **relevant** project documents >>

1.5 Overview

This document is organized in 4 main sections and some appendices.
Section 1 (this section) puts the document into its context and gives an overview of it.
Section 2 describes the project and the product from a high level user point of view.
Section 3 defines usage scenarios of the device under different conditions.....
Section 4 details the software requirements and has a hierarchical functional organization based on features.....
Section 5 (the appendices) add some detailed supporting information.....
<< Add anything apt to describe content and organization of this document >>

2 Overall Description

<< This section should describe the requirements at high level from a user perspective >>

2.1 Product Perspective

The <TBD> project is derived from <TBD TBD TBD> device project originally designed in 1995.
The main changes over its predecessor are.....
It is part of a family of products used for
It can be remotely controlled via a network connection
<< Describe the context and origin of the product being specified in this SRS. If the SRS defines a product that is a component of a larger system, then this subsection relates the requirements of the larger system to functionality of the software and identifies interfaces between that system and the software.
A block diagram showing the major components of the larger system, interconnections, and external interfaces can be helpful to provide context. Keep the diagram at a functional and interface level, not at design level. >>

2.2 Software Functions

The main functions performed by the software are:

- User Interface....
- Data Acquisition and reduction.....
- Report generation....
- Communication with the host....
-

<< List and describe the main functions performed by the software and their relationships. The textual explanation can be incremented with diagrams, such as a top level data flow diagram or a state transition diagram, to better understand functionality from a user perspective. >>

2.3 User Characteristics

The <TBD> device is aimed to be used in
Personnel using it is expected to have limited clinical practice, but to be trained in its use.....
Use of this device is indicated for
This device should never be used for
<< Define the intended users of the system, their educational level and technical expertise. State how the system is intended to be used and eventual limitations to its use. >>

2.4 Constraints

The <TBD> will require an FDA 510(k) clearance prior to marketing and is a medical device with a Moderate level of concern.
This device must conform to Standard << indicate any applicable regulatory policy >>
The device must be operated in temperature controlled environment.....
In case of loss of control due to any cause, the device must be put in safe state condition.....

<< Indicate here non-functional requirements that limit the developer's option, including, but not limited to, regulatory policies, hardware limitations, reliability requirements, safety and security requirements, operating environment. Note that these should be defined at high level from a user point of view. >>

2.5 Assumptions and Dependencies

It is assumed that the Ethernet driver xyz-eth, designed by Eth SW, Inc. is available by

<< List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, etc. The project could be affected if these assumptions are incorrect or change.

Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project. >>

2.6 Requirement Distribution

<< If the product is meant to have different releases, define here what requirements are included in each release. This section should be omitted if there is no defined release plan or if the information is to be retrieved in another project document. >>

3 Use cases/user workflows

<< If the system has significant user interactions, defining use cases or user workflows helps clarifying the actual requirements and could be used as a preliminary activity, together with some User Interface prototyping, preceding formal requirement definition. This activity can also help understanding the scenarios and therefore writing better design and better testing especially for usability testing. Although they are not requirements per se, it can be good to keep them to support the requirements. This section can be used to document this activity when it is small and informally performed, just recording sequences of user actions/system responses. If you use formal methodology for this activity, it is probably more valuable to document it elsewhere and just reference it in this spec. If the system has limited or no user interaction, just omit this section. >>

4 Specific Requirements

<< The following subsections define specific system requirements directly or via reference to other documents. These shall be detailed requirements stated from a developer perspective. The level of detail should be sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements.

To allow traceability, each formal requirement should be identified by a requirement tag. Do not use section header numbers as requirement tags, because section header numbers tend to change when you add/remove sections in the document. Requirement tags need to be specific and unambiguous and should never change. If requirements are deleted, their tags should never be reused. Tags could be numerical or alphanumerical, built by the union of an alphabetic prefix (e.g. an abbreviation for the specific functional area) and a sequential number. Examples are given below. Tag granularity depends on the level of granularity you want to use in your traceability to design and to testing, taking into account that higher granularity allow for better tracing of changes, but requires more effort and could be less efficient.>>

4.1 External Interface Requirements

<< Detail all inputs into and outputs from the software system, including their purpose, formats and ranges. Normally this would be a list of external interfaces with a summary of their purpose and reference(s) to detailed communication/interface specifications.>>

4.1.1 User Interfaces

<< Detail the requirements for each interface between the software product and the users.

This may include any GUI standards or style guides that are to be followed, screen layout constraints, user functionality requirements (e.g. help, keyboard shortcuts), error message display standards, etc. Also list here any usability requirements that should drive the design of the user interface. This is not intended to be a detailed description of the user interface in terms of screens or navigation and that is generally done in a user interface design specification. Indicate here only main requirements that must drive the actual screen and navigation design. Some examples are given below.>>

4.1.1.1 General requirements

[UI0001] The user interface shall be designed according to GUI standard xxxxxxxx.....

<< Add all additional general requirements associated with this feature >>

4.1.1.2 Input devices

4.1.1.2.1 Purpose

The user interface supports 2 types of input devices:

- A touch screen.....
- A full alphanumeric keyboard.....

In addition to that a remote control functionality is implemented through the Ethernet network connection.....

<< Add any descriptive information apt to explain this feature and put it into perspective. >>

4.1.1.2.2 Functional Requirements

[UI0001] A auto-repeat functionality shall be provided on touch screen when touching the following types of objects:

The repeat function shall start after xx seconds of continuous pressure. The repetition rate shall be every yy seconds.....

[UI0002].....

<< Add all additional functional requirements associated with this feature >>

4.1.1.1 Display

4.1.1.1.1 Purpose

A VGA LCD display is use to display all information to the user. The display presents status information on the first line, a series of icon functional keys below the status information, while the remaining portion of the screen is the main interface window.....

<< Add any descriptive information apt to explain this feature and put it into perspective. >>

4.1.1.1.2 Functional Requirements

[UI0013] The status bar on the first line shall present the following information:

- Date and Time
- A main status indication (BUSY or READY)
- The current operation in progress
-

[UI0014] The date and time indication shall have the following possible formats:

[UI0015] The date and time indication shall be updated every minute.....

<< Add all additional functional requirements associated with this feature >>

4.1.2 Hardware Interfaces

<< Detail the logical and physical requirements for each interface between the software product and the hardware components of the system (e.g. motor drivers, digital output signals, analog and digital inputs, etc). Normally, if extensive this would be a list of interfaces with a summary of their purpose and reference(s) to detailed hardware control specifications. A tabular form could be used as in the example below. >>

The following table lists the hardware interfaces with the indication of the documents where details on each interface can be found:

Req. Tag	Name	Type	Description/Usage	Reference Document
IF0001	Sens 1	Digital sensor	Motor 1 ax position sensor	<TBD> HW Design specification
IF0002	Enc 1	Gray Code	Motor 1 position/speed encoder	<TBD> HW Design specification
IF0003	Anal 1	A/D converted digital value	Output from acquisition sensor 1	<TBD> HW Design specification

4.1.3 Communications Interfaces

<< Detail the requirements for any communications functions required by this product (e.g. server link, network interface, e-mail server, LIS interface, etc.). Identify any communication standards that will be used. Specify any communication security or encryption issues. This does not need to be an extensive description of the communication layers, just reference external communication protocol documents for that. A tabular form could be used as in the example below. >>

The following table lists the hardware interfaces with the indication of the documents where details on each interface can be found:

Req. Tag	Name	Type	Description/Usage	Reference Document
IF0007	Ether 1	Ethernet interface	Communication with remote PC	<TBD> Communication Protocol
IF0009	USB 1	USB interface	External Printer interface	<TBD> Printer Postscript Driver

4.1.4 Hardware and Software Platform

[IF0017] This software shall run on the custom board ABCD from SILLYHW corporation, rev. 2.2 or higher.....

<< Detail any characteristics of the HW that you consider important and need to be verified. >>

[IF0021] The SureReal 123 real time kernel from UnrealSoft corp. shall be used to.....

<< Detail requirements related to processor, memory, system software and any other platform related requirements. >>

4.2 System Features

<< This template organizes the functional requirements by system feature. Other organization may be possible: mode of operation, user class, object class, functional hierarchy, etc.

Whatever the organization, it should reflect a logical partitioning of the functional requirements, and should not directly drive or impose the structure of the software design. >>

4.2.1 Motor 1 Control

4.2.1.1 Purpose

Motor 1 is a stepper motor used to position and move the sample <TBD> device << Describe use of the motor >>.

The motor is controlled through two sensors: a Home_Position digital sensor used to indicate when the <TBD> device reaches the position <TBD>, and a grey code encoder to verify the motor speed and the device position based on number of motor revolutions. This is necessary to guarantee the correct handling of the device also in case steps lost due to high resistance encountered in the movement.

<< Provide a short description of the feature, including its purpose and importance to the user. >>

4.2.1.2 Functional Requirements

<< List the functional requirements associated with this feature. The requirements that are Risk Control Measures should be specifically identified as with the tag **RCM** in the example below. Include error handling requirements: how the software shall respond to anticipated system errors or use errors. Try to be concise and unambiguous and to define the requirements in a way that they are testable. Use tables when necessary to help clarify conditions like in the example below. Tables can also be an efficient way to state requirements rather than all text and also can be used to create test cases and test records in a more efficient manner.>>

[PC0001] The software shall drive the motor using the following parameters:

Phase	Motor Control	Speed	Ramp
Phase 1: <TBD>	Half Step	100 steps/sec	Yes
Phase 2: <TBD>	Quarter Step	20 steps/sec	No

[PC0002]

[PC0003] **RCM:** During any motor movement, the software shall check the motor speed using the Grey Code Encoder. If the measured speed differs more than $\pm 10\%$ from the programmed speed, the software shall raise alarm ZZZZZ and stop the motor. << Add any additional action >>

[PC0004]

4.2.1.3 Stimulus Response Sequence

<< Describe input/output sequences or user action/system response. Tables can be used. If the requirement is simple and input/output sequences are limited, this information could be combined in the previous section and this section can be dropped.>>

4.2.2 Feature 2

<< Replace "Feature X", etc. in the headings with proper feature names. >>

4.2.2.1 Purpose

4.2.2.2 Functional Requirements

4.2.2.3 Stimulus Response Sequence

4.3 Performance Requirements

<< If there are performance requirements for the product under various circumstances, state them here and explain their rationale. Make such requirements as specific as possible: they must drive the design and implementation and must be testable. Consider such attributes as throughput, responsiveness, number of users/transactions, parallel operations, timing relationship for real time systems especially for control of hardware unless this is defined in other requirements or design specifications, etc. >>

[PR0001] The software shall respond to user commands within 0.005 seconds, measured from the moment the command is received by software to the moment the consequent action (e.g. motor control action) is activated. This excludes the hardware delays that are quantified in the hardware spec. This responsiveness is required to fulfill the system requirement of

[PR0002]

4.4 Safety Requirements

<< List here any Risk Control Measure that was not included in the feature by feature requirement description. These should be more system wide RCM. >>

[SR0001] RCM: Several programmed and configuration parameters are critical for the correct behavior of the device. To avoid the risk of corruption and consequent malfunctioning, the critical parameters shall be kept in 3 redundant copies organized in different data structures. Any time one such parameter is used, all 3 copies are read and compared to verify that are matching. In case of mismatch, the following actions shall be taken.....

[SR0002].....

4.5 Software System Attributes

<< Indicate here software attributes that can affect design and therefore become requirements to be considered in the design. Examples could be reliability (e.g. in terms of MTBF), availability (e.g. procedures to automatically recover after failure), security (e.g. protection against unauthorized access), maintainability (e.g. required modularity), portability (e.g. language or operating system independence), testability (e.g. test hooks or operation trace requirements), etc. Write these to be specific, quantitative, and verifiable when possible. >>

[SSA0001] After any failure, the software shall put the system in a safe condition within 0.01 sec << detail what this condition would be >>, then attempt to recover and make the system available for further operation within 0.1 sec.....

[SSA0002]

4.6 Other Non-Functional Requirements

<< Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, user documentation, installation and acceptance requirements, user maintenance requirements, etc.>>

5 Appendices

5.1 Appendix A: Data dictionary

<< There could be different interface and/or functional requirements that refer the same data. Instead of defining the data attributes in every one of them, it is suggested to define the data in a separate list or table. This table could be an appendix to the SRS or could be kept in a separate document (to be available to other spec documents) and referenced here.

The data dictionary should include at least Name, meaning or intended use, type, size, format, unit of measure, resolution, validity range, default value of all data and data structures. >>

5.2 Appendix B: Alarm/Messages dictionary

<< Although response to malfunction or misuse should be specified in the functional requirements, it is often useful to keep a list (e.g. in tabular form) of all software generated alarms, warnings, operator messages. These should include information on condition, triggering factor(s), recovery, etc. It would not necessarily include the actual text of the message, that could reside in the UI design specification. >>

5.3 Appendix C: List of TBD

<< When writing SRS there are often requirements that cannot be totally defined or issues that cannot be resolved from the beginning. These can be marked with “TBD” in the text of the requirement. It is suggested to keep a dynamic reference list of all TBDs in the spec to be able to trace them to resolution in a timely way. Omit this appendix if there are other ways to trace TBDs to resolution >>

USER INTERFACE DESIGN SPECIFICATION

TEMPLATE

DRAFT

Revision History

Date	Revision	Author	Description

NOTE: this template assumes a graphical user interface using some type of screen and standard operating devices such as keyboard and a pointing device. If different forms of physical or logical interface is used, the template must be modified and adapted accordingly.

TABLE OF CONTENTS

1	Introduction.....	4
1.1	Purpose.....	4
1.2	Scope.....	4
1.3	Definitions, acronyms, and abbreviations.....	4
1.4	References.....	4
1.5	Overview.....	4
2	General Characteristics	4
2.1	Approach.....	4
2.1.1	Human Factors.....	4
2.2	Common Properties for all screens	4
2.2.1	Colors.....	4
2.2.2	Fonts.....	5
2.2.3	Icons.....	5
2.3	Input Functions	5
2.3.1	Touch Screen	5
2.3.2	Keyboard.....	5
2.3.3	Pointing Device.....	5
2.3.4	Bar Code Reader	5
2.4	Screen Objects	5
2.4.1	Windows	5
2.4.2	Menus.....	5
2.4.3	Status Bar	5
2.4.4	Scroll Bar	5
2.4.5	Controls.....	6
2.4.5.1	Buttons	6
2.4.5.2	Check Boxes	6
2.4.5.3	Text Boxes	6
2.4.5.4	List Boxes	6
2.4.5.5	Numerical Fields.....	6
2.4.6	Objects Alignment	6
2.5	Error Handling	6
2.6	Help Handling.....	6
2.7	Screensaver	6
2.8	Internationalization	6
2.9	Other output devices	6
2.9.1	Loudspeaker.....	6
2.9.2	Other visual indications	6
2.10	Application Startup and Shutdown	6
3	UI Structure and Navigation Flow	7
3.1	Screen structure.....	7
3.2	Navigation	7
4	Screen Details	8
4.1	Screen 1	8
4.1.1	Purpose.....	8
4.1.2	Description and properties.	8

4.1.3	Controls.....	8
4.1.4	Other objects.	8
4.1.5	Navigation.....	8
4.2	Screen 2	8
4.2.1	Purpose.....	8
4.2.2	Description and properties.	8
4.2.3	Controls.....	8
4.2.4	Other objects.	8
4.2.5	Navigation.....	8
5	Appendices.....	8
5.1	Appendix A: Messages.	8
5.2	Appendix B: Memory Budget.....	9
5.3	Appendix C: List of TBD.	9

1 Introduction

1.1 Purpose

<<Identify the purpose of this document and its intended audience>>

1.2 Scope

<<Identify with name and details the software product(s) that this UI design document is related to. If this design applies only to a part of the product, identify what lays within the scope of this document and what is outside of it.>>

1.3 Definitions, acronyms, and abbreviations

<< Define any acronyms, abbreviations, or terms used in the document that may be unfamiliar to readers. If a project level definitions document exists, it can be referenced here and this section limited to specific terms used only in this document. >>

1.4 References

<< Include **relevant** documents that are part of the project's documentation set (e.g. Product Specification, Software Requirement Specification, etc.) and any other document referenced elsewhere in the UI Design document. Include also any **relevant** standard (e.g. Human Factors Policies, Screen Layout Design Standard, etc) >>

1.5 Overview

<< Describe content and organization of this document >>

2 General Characteristics

2.1 Approach

<< Describe the main philosophy driving the design of this user interface and the reasons for the choices made in the design. Give an high level view of the structure of the user interface. >>

2.1.1 Human Factors

<< Describe usability issues that need to be addressed in the design and how they have been addressed. Indicate if special provisions have been made for color-blind people, left-handed people, deaf people, etc. >>

2.2 Common Properties for all screens

<< The following are just a few examples of common properties that could be addressed in this section. Add/remove subsections according to structure and design of your UI. >>

2.2.1 Colors

<< Define the color palette to be used, detailing RGB and indicating where and when used. A tabular form could be used such as:

Screen Area	Object Type	Color (RGB)	Color Name
Status Bar	Status Bar background	50, 50, 178	Blue

>>

2.2.2 Fonts


<< Define type(s) of font, font dimensions, font attributes (bold, italics, underlined, etc.). Define where and when used. A tabular form could be used such as:

Screen Area	Object Type	Font and rules
Status Bar	Instrument status	Arial 12 Bold, First letter in upper case in each word.

>>

2.2.3 Icons

<< Define the icons to be used, their meaning, where and when used. A tabular form could be used such as:

Icon	Meaning	Use
	Warning	In message box, when reporting a warning message.

>>

2.3 Input Functions

<< The following are just a few examples of common input devices that could be addressed in this section. Add/remove subsections according to input devices present in your system. >>

2.3.1 Touch Screen

<< Define touch screen characteristics, touch area covered, minimum size for active objects, logical use of the device (e.g. auto-repeat function, how editing actions are started and closed, etc. >>

2.3.2 Keyboard

<< Define keyboard type(s) supported, physical and logical use of the device >>

2.3.3 Pointing Device

<< Define type(s) of supported pointing device, physical and logical use of the device >>

2.3.4 Bar Code Reader

<< Define BCR type, supported code types, usage >>

2.4 Screen Objects

<< The following are just a few examples of screen objects that could be addressed in this section. Add/remove subsections according to structure and design of your UI. >>

2.4.1 Windows

<< Define types of windows, common characteristics, ways to open/close windows, etc. >>

2.4.2 Menus

<< Describe Menu structure, common characteristics, ways to open/close windows, etc. >>

2.4.3 Status Bar

<< Define Status Bar structure and information present on status bar >>

2.4.4 Scroll Bar

<< Define Scroll Bar layout, behavior and handling >>

2.4.5 Controls

2.4.5.1 Buttons

<< Describe common Button characteristics and behavior (e.g. raised/pressed button appearance) >>

2.4.5.2 Check Boxes

<< Describe common Check Box characteristics and behavior (e.g. checked/not checked appearance) >>

2.4.5.3 Text Boxes

<< Define types of text boxes, appearance, editing rules >>

2.4.5.4 List Boxes

<< Define types of list boxes, appearance, scroll and selection rules >>

2.4.5.5 Numerical Fields

<< Define types of numerical fields, common presentation rules, editing rules, numbers and date/time formats, etc. >>

2.4.6 Objects Alignment

<< Describe any common rules to align objects on the screen. >>

2.5 Error Handling

<< Describe common rules for Error presentation and user acknowledge. >>

2.6 Help Handling

<< If any form of help is required, describe how to access help and its behavior. Omit this section otherwise. >>

2.7 Screensaver

<< If screensaver is required, describe its behavior, timings, etc. Omit this section otherwise>>

2.8 Internationalization

<< Indicate the languages provision (e.g. labels should provide +20% space respect to English to allow for translations) and eventual list of supported languages. >>

2.9 Other output devices

<< The following are just a few examples of possible output devices supported by UI. Add/remove subsections according to output devices present in your system. Omit this section otherwise. >>

2.9.1 Loudspeaker

<< Describe tones, sound lengths of signal cycles and how and when used. >>

2.9.2 Other visual indications

<< Describe eventual other visual indications (e.g. alarm lights) >>

2.10 Application Startup and Shutdown

<< Describe startup and shutdown sequences, eventual user interaction if any >>

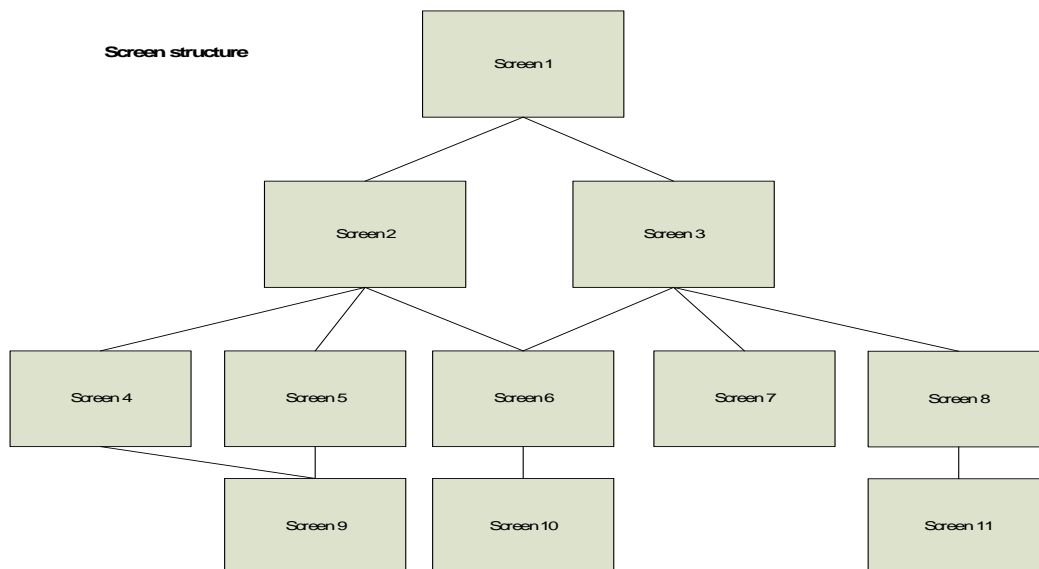
3 UI Structure and Navigation Flow

3.1 Screen structure.

<< Describe the structure of the user interface. A graphical presentation as in the example below can be used (use specific screen names, not Screen 1, 2, 3, etc. as in the example).

If the screen structure is more complicated, as sometimes is the case, you can partition the diagram in several hierarchical diagrams or use a different way to present it.

Sometimes is also valuable to build an actual prototype of the user interface to help in designing the structure and navigation rules. >>



3.2 Navigation .

<< Describe navigation rules and ways to move around to the different screens. Only common navigation behavior should be described in this section. If exceptions to common behavior are present on some screens, these should be addressed in the specific screen description.>>

4 Screen Details

<< This will be the biggest section of the design spec, detailing the design and behavior of each screen. >>

4.1 Screen 1 .

<< Use the actual screen names instead of “Screen 1, 2, 3, etc.”. The following subsections are just an example of how the screen description could be structured and of the type of information that could be provided. The actual structure should depend on the specific application and on your preferences. >>

4.1.1 Purpose

<< Describe the intended use of the screen in the system or specific application. Discuss reasons for any specific design choice related to this screen. >>

4.1.2 Description and properties.

<< Define the screen type (e.g. modal/non modal) and screen properties (e.g. scrollable or not, resizable or not). Describe screen layout or provide a diagram illustrating screen layout. This could be a printout from a screen prototyping environment.>>

4.1.3 Controls.

<< Describe all the controls of this screen with indication of the type of control, format of displayed information, type/origin of data, triggered action, etc. (whatever applicable). The description could be in one or more tables. Use of tables is often more efficient than text both for specification and for test case development. >>

4.1.4 Other objects.

<< Describe any other objects present on screen, besides controls (e.g. graphical objects, information messages, etc.) >>

4.1.5 Navigation.

<< Define navigation to/from the screen, unless the general navigation pattern described in section 3 applies. Any exception or specific navigation rules should be described here>>

4.2 Screen 2 .

4.2.1 Purpose

4.2.2 Description and properties.

4.2.3 Controls.

4.2.4 Other objects.

4.2.5 Navigation.

5 Appendices

5.1 Appendix A: Messages.

<< Optional section. Provide a list of all user messages (alarms, warnings, information, etc.). This could be an appendix to the UI Design spec or could be in another referenced document (e.g. an excel file could provide all the messages in all the supported languages.) >>

5.2 Appendix B: Memory Budget.

<< Optional section. Define the memory requirements for each graphical object of the UI.
Can be useful to calculate the memory size required for the application to run. >>

5.3 Appendix C: List of TBD.

<< It is suggested to keep a dynamic reference list of all TBDs in the spec to be able to trace them to resolution in a timely way. Omit this appendix if you use other ways to trace TBDs to resolution >>

SOFTWARE DESIGN SPECIFICATION

TEMPLATE

DRAFT

Revision History

Date	Revision	Author	Description

NOTE: This template can be used for describing the SW design of an entire project or of part of it, in which case an higher level architecture partitioning document is required.

It is conceived as a document to grow with the design, where different sections of the document are filled at different design stages, adding detail at each stage. Alternatively this template could be shortened to address just the overall architecture/high level design and separate detailed design specifications could be created for specific components or algorithms.

TABLE OF CONTENTS

1	Introduction.....	3
1.1	Purpose.....	3
1.2	Scope.....	3
1.3	Definitions, acronyms, and abbreviations.....	3
1.4	References.....	3
1.5	Overview.....	3
2	Design Considerations	3
2.1	Assumptions and dependencies	3
2.2	General constraints.....	3
2.3	Risk Management	3
2.4	Design Guidelines.....	4
2.5	Architectural Strategies.....	4
3	Hardware Architecture.....	4
4	System Environment.....	4
5	Software Architecture	4
5.1	Functional decomposition.....	4
5.2	Control/Data Flow.	4
5.3	Hazard Mitigation.....	4
5.4	Key Algorithms.....	4
6	Interfaces.....	5
6.1	Interface Mechanisms	5
6.2	Data Structures.....	5
6.3	Error Handling	5
7	Detailed Design.....	5
7.1	Component 1	5
7.1.1	Purpose and scope.....	5
7.1.2	Further decomposition	5
7.1.3	Interfaces.....	5
7.1.4	Data/file structures	6
7.1.5	Processing	6
7.2	Component 2.....	6
7.2.1	Purpose and scope.....	6
7.2.2	Further decomposition	6
7.2.3	Interfaces.....	6
7.2.4	Data/file structures	6
7.2.5	Processing	6
7.3	Component 3 (SOUP component)	6
7.3.1	Purpose and scope.....	6
7.3.2	Hardware and Software requirements.....	6
8	Appendices.....	6
8.1	Appendix A: List of TBD.	6

1 Introduction

1.1 Purpose

<<Identify the purpose of this document and its intended audience>>

1.2 Scope

<<Identify with name and details the software product(s) that this design document is related to. If this design applies only to a part of the product, identify what lays within the scope of this document and what is outside of it.>>

1.3 Definitions, acronyms, and abbreviations

<< Define any acronyms, abbreviations, or terms used in the document that may be unfamiliar to readers. If a project level definitions document exists, it can be referenced here and this section limited to specific terms used only in this document. >>

1.4 References

<< Include **relevant** documents that are part of the project's documentation set (e.g. Product Specification, Software Requirement Specification, etc.) and any other document referenced elsewhere in the Design document.
Include also any **relevant** standard >>

1.5 Overview

<< Describe content and organization of this document >>

2 Design Considerations

<< This section helps to identify the issues and constraints that need to be addressed by the software design. The following subsections give some examples of classification, but feel free to add and/or remove sections as needed by the structure of your project/product. >>

2.1 Assumptions and dependencies

<< List any assumed factors (as opposed to known facts) that could affect the design stated in this document. This could include related software or hardware, availability of specific tools, etc. >>

2.2 General constraints

<< Indicate here any limitation that could have impact on the architecture and design and specify what the impact could be. Note that these should be specific design constraints. It is not worth to repeat here whatever is already specified in the Software Requirement Specification, that is the main reference for the architectural and detailed design. >>

2.3 Risk Management

<< Indicate any risk management issues that affect the architecture and design. This section should not refer to specific risk control measures, that should be addressed in the SW architecture and Detailed design sections. It should discuss more high level consideration that could have impact on the overall design approach, such as for example the need to implement redundant checks on some critical system behavior. >>

2.4 Design Guidelines

<< Describe goals, guidelines and priorities that should drive the design task. For example “focus on fast system response instead of memory usage”, or “build as much as possible on reuse of a previous application to save design and implementation effort”, or “emphasize testability by providing an high number of test hooks”, etc. >>

2.5 Architectural Strategies

<< Explain how the goals, guidelines, risk considerations, etc. defined in previous subsections shall drive the architectural design decisions affecting the system organization and its design structures: approaches to control, data storage and flow, synchronization mechanisms, concurrency, memory management, resources management, interface paradigms, etc. >>

3 Hardware Architecture

<< Give a short description the structure of the hardware that the SW Application described in this design document is controlling, or use a hardware context diagram. If hardware architecture is defined in another document, just reference it and drop this section. >>

4 System Environment

<< Describe in this section the environment in which the software of this design document will execute (e.g. Operating System(s), file system, memory management, directory structure, startup/shutdown, etc.) >>

5 Software Architecture

5.1 Functional decomposition

<< Identify the software items that fulfill the requirements defined in the SRS and implement the architectural strategies of section 2.5. One or more context diagrams or other graphical presentation could be used for this purpose in addition to a short description of each main component or subsystem. Note that this should be a first level of decomposition still allowing to see the entire system. Further decomposition can be achieved in the Detailed Design. In the decomposition, identify any component that is classified as SOUP (Software of Unknown Provenance/Off the shelf software) >>

5.2 Control/Data Flow.

<< Specify how the different components of the SW architecture described in section 5.1 interact with each other. Depending on the structure of the application, you can use data flow diagrams or other types of graphical and/or descriptive presentation. >>

5.3 Hazard Mitigation

<< Define any necessary component segregation (i.e., isolation to prevent certain components from having indirect side effects on others either due to defects or common resource utilization) required to reduce the risks and how segregation is achieved. Describe also any other measure taken at architectural level to reduce or control the risks>>

5.4 Key Algorithms

<< List the main algorithms implemented in the system. A tabular form can be used. The detailed description of the algorithms could reside in the detailed design section or in another design document. >>

6 Interfaces

<< In this section detail the main hardware and software interfaces that are implemented for the architectural components defined in previous section. It is not necessary to define here low level interfaces eventually used within the single component and that could be described in the detailed design of that component. >>

6.1 Interface Mechanisms

<< Describe the interface mechanisms used in the system (e.g. message queues, shared memory, etc.) and where, when, for what purpose, by what components, etc. they are used. >>

6.2 Data Structures

<< List the data structures used for interfacing, where, when, by whom they are used. Define the data formats or refer to a data dictionary. >>

6.3 Error Handling

<< Describe any specific error handling used with the interface mechanisms and data. If error handling is fully described elsewhere, omit this section. >>

7 Detailed Design

<< In this section each of the components identified in the architectural decomposition is addressed in detail. Use specific component name in place of “Component 1”, “Component 2”, etc.

This section could have a hierarchical structure, since each main architectural component could be decomposed in subcomponents and each subcomponent could again be decomposed and so on.

You could decompose the system in subsystems or layers, based on hardware (e.g. different CPUs) or based on a functional decomposition (e.g. main control, data acquisition, database handling, reports, and so on), and then decompose each subsystem in physical components, each of them further decomposed in smaller components.

In case of object oriented design you could decompose in classes and sub-classes.

Each component implementing a risk control measure should be specifically identified.

We are giving here just a simplified example for the structure of this section. You could augment it or organize it in different way.

In case the system is complex, you could use a detailed design document for each subsystem or main component. >>

7.1 Component 1

7.1.1 Purpose and scope

<< Describe the function of the component within the system, what it is expected to fulfill, what are the limitations in its usage, etc. >>

7.1.2 Further decomposition

<< If an higher level of decomposition is necessary, describe the items that fulfill this decomposition. Use context diagrams, data flow, control flow diagrams or any other tools achieve this description. >>

7.1.3 Interfaces

<< Describe the specific interfaces of this component. Note that if the interfaces are fully and clearly detailed in the code, it is not necessary to repeat the information here. To avoid redundancy and documentation maintenance issues, it is better to keep the detailed information in one place only. >>

7.1.4 Data/file structures

<< For data intensive components, for example the ones implementing database management, this section should describe, also in tabular form, the data or file structures. Again, avoid redundancy: if data structures are documented in the code, do not repeat the detail here.

If the component is not data intensive, you could describe the data in the following section and omit this section.

If a project level data dictionary exists, refer to it. >>

7.1.5 Processing

<< Describe the component activity to fulfill its purpose and scope. This section should help understanding the component behavior without the need to look into the details of the code, so it is not good practice to use the code or portions of it to achieve this description, because it would not give added value and would be difficult to keep up-to-date. Better to use a descriptive paragraph and/or some graphical tools. The tools will depend on the type of component and design methods used. Could be state transition diagrams, data flow diagrams, class diagrams, UML diagrams, entity relationship diagrams, or whatever other form suggested by your specific design method and tools. >>

7.2 Component 2

7.2.1 Purpose and scope

7.2.2 Further decomposition

7.2.3 Interfaces

7.2.4 Data/file structures

7.2.5 Processing

7.3 Component 3 (SOUP component)

<< The structure for SOUP component could be different, this is just an example. >>

7.3.1 Purpose and scope

<< Describe the function of the component within the system, what it is expected to fulfill, what are the limitations in its usage, what are the functional and performance requirements for its intended use. >>

7.3.2 Hardware and Software requirements

<< Identify system hardware and software necessary to the correct operation of this component. >>

8 Appendices

8.1 Appendix A: List of TBD.

<< It is suggested to keep a dynamic reference list of all TBDs in the spec to be able to trace them to resolution in a timely way. Omit this appendix if there are other ways to trace TBDs to resolution >>

SoftwareCPR® 62304 Template Training Example Mapping to 62304

Training Template Documents Referenced

DSPEC → Software Design Specification Template (DSPEC-template-cpr020609.doc)
RA → Software Risk Analysis Procedure Training Example(RA-template-cpr020609.doc)
SCMP → Software Configuration Management Plan Template (SCMP-template-cpr020609.doc)
SDP → Software Development Plan Template (SDP-template-cpr020609.doc)
SDSOP → Software Development Procedure Training Example (SDSOP-template-cpr020609.doc)
SMSOP → Software Maintenance Procedure Training Example (SMSOP-template-cpr020609.doc)
SRSOP → Software Release Procedure Training Example (SRSOP-template-cpr020609.doc)
SRS → Software Requirements Specification Template (SRS-template-cpr020609.doc)
STP → Software Test Plan Template (STP-template-cpr020609.doc)
UID → User Interface Design Specification Template (UID-template-cpr020609.doc)

5 Software Development Process

5.1 Software development planning

ANSI/AAMI/IEC 62304 Conformity Requirements	Y/ N /NE/ NA	Procedure, Plan, or Document references	Comments
5.1.1 a – e The software development/quality plan(s) addresses:			
a. the processes to be used	Y	SDP Section 4.1, SDSOP Section 3 and 4	Process defined in the procedure, tailoring in the plan.
b. the deliverables of the activities and tasks	Y	SDP Section 5.2.2, SDSOP Section 4.2 and 4.3	Detailed document definition in the plan.
c. traceability between system requirements, software requirements, software system test and risk control measures.	Y	SDP Section 6.4 STP Sections 6.2 and 6.3 STP Section 17.4 SDSOP Section 4.3.1.16	Traceability is addressed within the Software Development Procedure and detailed in the Development Plan. Traceability to test detailed in the Software Test Plan to address test coverage.
d. configuration and change management including SOUP configuration items and software used for development	Y	SDP Section 6.1 SCMP Section 3.1, 3.2, 3.5 and 3.6 SDSOP Section 4.3.1.13, 4.3.1.14 and 8	
e. software problem resolution procedure	Y	SDP Section 6.2 SCMP Section 3.2 SDSOP Section 4.3.1.15	Details to be covered in the SW Problem Report Procedure to be developed.
5.1.2 Software development/quality plan(s) get updated	Y	SDSOP Section 4.3.1.1	
5.1.3 a. Software development plan(s) references system requirements as inputs	Y	SDP Section 5.2.1 SDSOP Section 4.2	
5.1.3 b. The plan includes or references procedures for coordinating the software development and the design and development validation necessary to meet quality management system requirements.	Y	SDP Section 5	

5.1.4 Standards, methods and tools defined in plan(s). Class C.	Y	SDP Section 4.2 and 5.1	
5.1.5 software integration and software integration test are included in the plan. Including SOUP. Class B, C	Y	SDP Section 3.4.2 and 5.4 SDSOP Section 4.3.1.11 and 8 STP Section 5.3	
5.1.6 software verification plan(s) include a) deliverables requiring verification, b) the verification tasks required for each life cycle activity, c) the milestones at which deliverables are verified d) acceptance criteria for verification	Y	SDP Section 5.2 and 5.3 SDSOP Section 4.3.1.10, 4.3.1.11 and 4.4	
5.1.7 Risk management planning is included in the plan(s) and includes risk management related to SOUP.	Y	SDP Section 5.4 and 6.3 SDSOP Section 4.3.1.3 and 8 RA Section 4.1.3 and 6	
5.1.8 Documentation planning is included in the plan(s) and includes the following for documents to be produced during the software development life cycle: a) Title, name or naming convention b) Purpose c) Intended audience d) Procedures and responsibilities for development, review, approval and modification.	Y	SDP Section 5.2.2, 5.3 and 4.1.4 SDSOP Section 4.3.1.10 and 4.4	
5.1.9 Plans include CM information including: a. items to be controlled. b. SCM activities and tasks c,d.. organizational responsibilities for CM e. points when the items are to be placed under formal CM f. when the problem resolution process is to be used.	Y	SDP Section 6.1 and 6.2 SCMP Section 2.1 and 3 SDSOP Section 4.3.1.13	

5.1.10 Supporting development tools, items or settings are included in CM Class B, C	Y	SCMP Section 3.1.1 SDSOP Section 4.3.1.13	
5.1.11 Plans require software items are placed under formal CM before they are verified. Class B, C	Y	SDP Section 6.1 SDSOP Section 4.3.1.10 and 4.3.1.13	

5.2 Software Requirements Analysis

Section Conformity Requirements	Y/N/ NE/ NA	Procedure, Plan, or Document references	Comments
5.2.1 software requirements are defined and documented from System Requirements.	Y	SDP Section 5.2.1 SDSOP Section 4.3.1.2 and 4.3.1.16 SRS Section 1.1	
5.2. As appropriate check for the following types of requirements 5.2.2 a. include Functional and capability requirements	Y	SRS Section 4.1.4, 4.2, 4.3 and 4.5	
5.2.2 b. Software system inputs and outputs	Y	SRS Section 4.2.1.3 and 5.1	
5.2.2 c. Interfaces between the software system and other systems.	Y	SRS Section 4.1	
5.2.2 d. Alarms, warnings, operator messages	Y	SRS Section 4.2.1.2 and 5.2 UID Section 2.5 and 5.1	
5.2.2 e. Security	Y	SRS Section 4.5	
5.2.2 f. Usability requirements that are sensitive to human error and training.	Y	SRS Section 2.3 and 4.1.1 UID Section 2.1.1	
5.2.2 g. Data Definition and database requirements	Y	SRS Section 4.6 and 5.1	
5.2.2 h. Installation and acceptance reqs at the operation and maintenance site.	Y	SRS Section 4.6	
5.2.2 i. reqs for operation and Maintenance	Y	SRS Section 4.5	
5.2.2 j. user documentation required	Y	SRS Section 4.6	
5.2.2 k. user maintenance reqs	Y	SRS Section 4.6	
5.2.2 l. regulatory reqs such as from performance standards for the device type, regulatory guidance documents for functionality for the device type, ...	Y	SRS Section 2.4	
5.2.3 risk control measures included as reqs. Class B, C	Y	SRS Section 4.2.1.2 and 4.4	
5.2.4 device risk analysis re-evaluated and updated based on software reqs.	Y	SDSOP Section 4.3.1.3	Not much detail, could be expanded

5.2.5 System requirements updated based on software reqs	N		Not covered, this is more a system level process than a SW level process but could be covered here if software is the device
5.2.6 Verify the software requirements including that: a) system and risk control reqs implemented.	Y	SDSOP Section 4.3.1.10	
b) Do not contradict one another	Y	SDSOP Section 4.3.1.10	
c) in terms minimizing ambiguity	Y	SDSOP Section 4.3.1.10	
d) testable	Y	SDSOP Section 4.3.1.10	
e) uniquely identified	Y	SDSOP Section 4.3.1.10	
f) are traceable to System requirements	Y	SDSOP Section 4.3.1.10 and 4.3.1.16	

5.3 Software Architectural Design (No Class A requirements)

Section Conformity Requirements	Y/N/ NE/ NA	Procedure, Plan, or Document references	Comments
5.3.1 Documented software architecture including structure and software items. Class B, C	Y	SDSOP Section 4.3.1.5 DSPEC Section 5.1	
5.3.2 Documented architecture includes the interfaces between the software items and between software items and external components (HW and SW). Class B, C	Y	SDSOP Section 4.3.1.5 DSPEC Section 6	
5.3.3 Functional and performance requirements are specified for SOUP items. Class B, C	Y	DSPEC Section 7.3.1 SDSOP Section 8	
5.3.4 System hardware and software necessary for SOUP items are specified. Class B, C	Y	DSPEC Section 7.3.2 SDSOP Section 8	
5.3.5 segregation essential to risk control is specified. Class C	Y	DSPEC Section 5.3 SDSOP Section 4.3.1.5	
5.3.6 Verify and document the architecture including that it: a) implements system and software and risk control reqs b) supports internal and external interfaces c) supports proper operation of SOUP items Class B, C	Y	SDP Section 5.2.2 SDSOP Section 4.3.1.10	

5.4 Software Detailed Design (No Class A requirements)

Section Conformity Requirements	Y/N/ NE/ NA	Procedure, Plan, or Document references	Comments
5.4.1 refine the architecture to the software unit level. Class B, C	Y	SDSOP Section 4.3.1.7 DSPEC Section 7	

5.4.2 detailed design exists for each software unit. Class C	Y	SDSOP Section 4.3.1.7 DSPEC Section 7	
5.4.3 Detailed design exists for the interfaces between the software units and between software units and external components (hw and software). Class C	Y	SDSOP Section 4.3.1.7 DSPEC Section 7.1.3	
5.4.4. Verification that the detailed design a) implements the software architecture b) is free from contradiction with the architecture. Class C	Y	SDP Section 5.2.2 SDSOP Section 4.3.1.10	

5.5 Software unit implementation and verification

Section Conformity Requirements	Y/N/ NE/ NA	Procedure, Plan, or Document references	Comments
5.5.1, Implement units (Class A,B,C)	Y	SDSOP Section 4.3.1.8	
5.5.2 -Procedures, methods and strategies exist for verifying each software unit. -Test procedures evaluated for correctness. Class B, C	Y	SDSOP Section 4.3.1.8, 4.3.1.10 and 4.3.1.11 SDP Section 5.2.2 STP Section 5.1.14 and 5.2	
5.5.3 Acceptance criteria - established for software units prior to integration - Units met acceptance criteria Class B, C	Y	SDSOP Section 4.3.1.10 SDP 5.2.2 STP Section 5.1.14 and 5.2.2	
5.5.4 unit acceptance criteria shall included proper event sequence, data and control flow, planned resource allocation, fault handling, initialization of variables, self diagnosis, memory management, memory overflows and boundary conditions. Class C	Y	SDSOP Section 4.3.1.10 SDP Section 5.2.2 STP Section 5.1.14 and 5.2.2	

5.5.5 Unit test verification has been performed and results documented. Class B, C	Y	SDSOP Section 4.3.1.10 and 4.4 SDP 5.2.2 STP Section 5.2	
---	---	---	--

5.6 Software integration and integration testing (No Class A requirements)

Section Conformity Requirements	Y/N/ NE/ NA	Procedure, Plan, or Document references	Comments
5.6.1 Software units integrated in accordance with the integration plan. Class B, C	Y	SDP Section 3.4.2 SDSOP Section 4.3.1.9	
5.6.2 Verify and record (not testing usually by review) a) units have been integrated into items and the system b) hardware and software items have been integrated Class B, C	Y	SDSOP Section 4.3.1.9	Could be made more explicit.
5.6.3 software items have been tested in accordance with the integration plan and the results are documented. Class B, C	Y	SDSOP Section 4.3.1.11 and 4.4 STP Section 5.3	
5.6.4 integration testing (NOTE: may be combined with system testing) verifies that the software item performs as intended Class B, C	Y	STP Section 5.3 SDSOP Section 4.3.1.11	
5.6.5 integration test procedures shall be evaluated for correctness. Class B, C	Y	SDP Section 5.2.2 STP Section 5.3.2 SDSOP Section 4.3.1.10	
5.6.6 regression testing to identify defects in other units that show up after integration of new units Class B, C	Y	STP Section 5.1.8 and 5.3.1 SDSOP Section 4.3.1.11	
5.6.7 Integration test records contain: a) the test result including pass/fail determinations and a list of anomalies b) records to permit repeating the test and c) tester identification Class B, C	Y	STP Section 5.3.2, 7, 9 and 17.1	

5.6.8 formal process exists and anomalies found during integration and integration testing are recorded. Class B, C	Y	SDP Section 5.2.2 STP Section 5.3.1, 7 and 9 SDSOP Section 4.3.1.15 and 4.4	
--	---	--	--

5.7 Software System Testing (No Class A requirements)

Section Conformity Requirements	Y/N/ NE/ NA	Procedure, Plan, or Document references	Comments
5.7.1 Testing covers all requirements and Tests include input stimuli, expected results, pass/fail criteria and cover all requirements. Note: it is acceptable to combine integration and system testing in earlier phases. Class B, C	Y	STP Section 5.4.1 and 17.1	
5.7.2 Anomalies handled using the formal problem resolution process. Class B, C	Y	SDP Section 5.2.2 and 6.2 STP Section 5.3.1, 7 and 9 SDSOP Section 4.3.1.15 and 4.4	
5.7.3 Regression testing after changes and perform any relevant risk management activities Class B, C	Y	STP Section 5.1.8 and 8 SDSOP Section 4.3.1.11 and 4.3.1.14	
5.7.4 Verified that a) verification strategies and test procedures are appropriate, b) that test procedures trace to software requirements, c) that all requirements have been tested or otherwise verified and d) test results meet required pass/fail criteria. Class B, C	Y	SDP Section 5.2.2 STP Section 5.4.2, 6.2, 6.3, 7, 17.1 and 17.4 SDSOP Section 4.3.1.10 and 4.3.1.16	
5.7.5 Software test records contain a. document the test result and anomalies b. sufficient records to permit the test to be repeated and c. identify of the tester. Class B, C	Y	STP Section 5.4.2, 7, 9 and 17.1	

5.8 Software Release (For Class A, 5.8.4 is the only required section)

Section Conformity Requirements	Y/N/ NE/ NA	Procedure, Plan, or Document references	Comments
5.8.1 verification is completed and results evaluated before release. Class B, C	Y	SDP Section 5.2.2 SCMP Section 6.3 SDSOP Section 4.3.1.17 and 4.4	
5.8.2 Known residual anomalies are documented. Class B, C	Y	STP Section 17.2 SCMP Section 6.3.8 SRSOP Section 4.2	
5.8.3 Known residual anomalies have been evaluated to ensure they do not pose an unacceptable risk. Class B, C	Y	STP Section 17.2 SCMP Section 6.3.8	
5.8.4 Versions of the software that are released are documented. Class A, B, C	Y	SCMP Section 6.3 SRSOP Section 4.1 and 5	
5.8.5 The procedure and environment used to build the release version is documented. Class B, C	Y	SCMP Section 6.1.8 SRSOP Section 4.1	
5.8.6 All required lifecycle tasks, activities and documentation are complete. Class B, C	Y	SDP Section 5.2.2 SDSOP Section 4.3.1.17 and 4.4	
5.8.7 The software, product and configuration items, documentation are archived for a period longer than the life of the device or as specified by relevant regulatory requirements. Class B, C	Y	SDP Section 5.2.2 and 5.7 SDSOP Section 5 SMSOP Section 5 SRSOP Section 5	
5.8.8 Procedures ensure that released software can be reliably delivered without change or corruption covering: - replication - media labeling - packaging -protection - storage - delivery Class B, C	Y	SCMP Section 6.2 SRSOP Section 4	

6 Maintenance Process

6.1 Establish Software Maintenance Plan (all are for all classes)

Section Conformity Requirements	Y/N/ NE/ NA	Procedure, Plan, or Document references	Comments
6.1 A software maintenance Plan is established. It includes:	Y	SDP Section 7 SDSOP Section 1.1, 4.3.1.1, 4.4 SMSOP Section 1.1	
a) procedures for receiving, documenting, evaluating and tracking feedback after release.	Y	SMSOP Section 4	
b) criteria for determining whether feedback is considered to a problem.	Y	SMSOP Section 4	
c) use of the software risk management process.	Y	SMSOP Section 4.4	
d) use of the formal problem resolution process. (also in 6.2.2)	Y	SMSOP Section 4.1 and 4.2	
e) use of configuration management process	Y	SMSOP Section 4.4	
f) procedures to evaluate and implement upgrades, bug fixes, patches and obsolescence of SOUP.	Y	SMSOP Section 4.3 SMSOP Section 8 SCMP Section 3.6	

6.2 Problem and Modification Analysis

Section Conformity Requirements	Y/N/ NE/ NA	Procedure, Plan, or Document references (If level of detail in section 4 is not considered a sufficient mapping)	Comments
6.2.1.1 feedback on released software products are monitored - within the organization - and from users.	Y	SMSOP Section 4.1 and 4.2	
6.2.1.2 Feedback is documented (as problem reports) and evaluated to determine whether a problem exists. Problem reports include actual or potential adverse events or deviations from specifications.	Y	SMSOP Section 4.1 and 4.2	

6.2.1.3 problem reports are evaluated for safety of released products and whether a change to the released product is needed.	Y	SMSOP Section 4.4	
6.2.2 Problem report process is used to address problems	Y	SMSOP Section 4.1 and 4.2	
6.2.3 Each change request is analyzed for its effect on the organization, released software products and systems with which it interfaces. Class B, C	Y	SMSOP Section 4.4	
6.2.4 Modifications to released software products are evaluated and approved.	Y	SMSOP Section 4.4	
6.2.5 Changes are communicated to users and regulators as required, including: a) Any problem in released software and the consequences of continued unchanged use. b) The nature of any available changes to released software and how to obtain and install the changes.	Y	SMSOP Section 4	

6.3 Modification Implementation

Section Conformity Requirements	Y/N/ NE/ NA	Procedure, Plan, or Document references (If level of detail in section 4 is not considered a sufficient mapping)	Comments
6.3.1 Uses the formal software development process or an established maintenance process to implement modifications.	Y	SMSOP Section 4.4 and 4.5	
6.3.2 Changed software shall be released according to a 5.8 software release process. <i>Note: 6.3.2 is for all Safety Classes but 5.8 and Table A.1 are explicit that 5.8 is not required for Class A.</i>	Y	SMSOP Section 4.6	

7 Software Risk Management Process (only 7.4.1 applies to Class A software)

7.1 Analysis of software contributing to hazardous situations

Section Conformity Requirements	Y/N/	Procedure, Plan, or Document references	Comments
---------------------------------	------	---	----------

	NE/ NA		
7.1.1 Software items that could contribute to a hazardous situation are identified. Class B, C	Y	RA Section 4.1.3	
7.1.2 Potential causes of hazardous situations have been identified including: a) Incorrect or incomplete specification of functionality b) Software defects c) Failure or unexpected results from SOUP d) Hardware failures or other software defects that could result in unpredictable software operation (indirect/common causes) e) Reasonably foreseeable misuse. Class B, C	Y	RA Section 4.1.3 and 6	
7.1.3 If SOUP failure is a potential cause supplier published anomaly lists were evaluated for relevance. Class B, C	Y	RA Section 4.1.3 and 8	
7.1.4 Potential causes of software items contributing to hazards have been documented. Class B, C	Y	RA Section 5 and 6	
7.1.5 The sequence of events that could result in a hazardous situation are documented. Class B, C	Y	RA Section 5 and 6	

7.2 Risk Control measures

Section Conformity Requirements	Y/N/ NE/ NA	Procedure, Plan, or Document references	Comments
7.2.1 Risk control measures have been identified for each potential cause. Class B, C	Y	RA Section 4.3	
7.2.2 Risk control measures implemented in software a) are included in software requirements b) the items have safety classes consistent with the risk being controlled	Y?	RA Section 4.3 SRS Section 4.2.1.2 and 4.4	Safety classes not addressed. Should be added.

7.3 Verification of Risk Control Measures

Section Conformity Requirements	Y/N/ NE/ NA	Procedure, Plan, or Document references	Comments
7.3.1 documented verification for all risk control measures . Class B, C	Y	RA Section 4.3 STP Section 5.1.1, 6.3 and 17.4	
7.3.2 Risk control measures in software were evaluated to identify any new sequences they could cause that could lead to hazards. Class B, C	Y	SDSOP Section 4.3.1.3 RA Section 4.3	
7.3.3 Documented traceability from a) hazardous situation to the software item b) software item to specific software cause c) software cause to RCM d) RCM to verification of RCM Class B, C	Y	RA Section 4.1, 4.3, 5 STP Section 5.1.1, 6.3 and 17.4	

7.4 Risk Management of Software Changes

Section Conformity Requirements	Y/N/ NE/ NA	Procedure, Plan, or Document references	Comments
7.4.1 Changes to the software are analyzed to determine whether: a) additional software risk control measures are required. b) additional potential causes are introduced contributing to a hazardous situation Class B, C	Y	SDSOP Section 4.3.1.3 and 4.3.1.14 SMSOP Section 4.4	
7.4.2 software changes, including changes to SOUP are analyzed to determine if the modification could interfere with existing RCMs. Class B, C	Y	SDSOP Section 4.3.1.14 SMSOP Section 4.4	
7.4.3 Risk mgmt activities have been performed based on the analysis of the changes. Class B, C	Y	SDSOP Section 4.3.1.14 SMSOP Section 4.4	

8 Configuration Management Process

8.1 Configuration Identification (all are for all classes)

Section Conformity Requirements	Y/N/ NE/ NA	Procedure, Plan, or Document references	Comments
8.1.1 Unique identification for configuration items and their versions and includes software documentation.	Y	SCMP Section 3.1 SDP 6.1 SDSOP Section 4.3.1.13	
8.1.2 Each SOUP item is identified by title, manufacturer, and unique SOUP designator/version/patch # etc.	Y	SDP Section 5.4 SCMP Section 3.5 and 3.6 SDSOP Section 8	
8.1.3 System configuration documentation includes versions for all items	Y	SCMP Section 3.1 and 3.3	

8.2 Change Control (all are for all classes)

Section Conformity Requirements	Y/N/ NE/ NA	Procedure, Plan, or Document references	Comments
8.2.1 Configuration items are changed only in response to an approved change request. NOTE: Different acceptance processes can be defined for different lifecycle phases. Note if there are.	Y	SCMP Section 3.2 SDSOP Section 4.3.1.13 and 4.3.1.14 SMSOP Section 4.4	
8.2.2 Changes are implemented as specified in the change request. Activities that need to be repeated as a result of the change have been performed.	Y	SDSOP Section 4.3.1.14 SMSOP Section 4.4 and 4.5	
8.2.3 Changes are verified including repeating any verification that has been invalidated by the change.	Y	SDSOP Section 4.3.1.14 SMSOP Section 4.5	
8.2.4 Each change request, relevant problem report and approval of the change can be traced.	Y	SDSOP Section 4.3.1.14 SMSOP Section 4	

8.3 Configuration Status Accounting Tasks (all are for all classes)

Section Conformity Requirements	Y/N/ NE/ NA	Procedure, Plan, or Document references	Comments
8.3 Retrievable records are retained that show the history of the controlled configuration items including system configuration.	Y	SCMP Section 3.3	

9 Software Problem Resolution Process (all are for all classes) – NOT COVERED

Section Conformity Requirements	Y/N/ NE/ NA	Procedure, Plan, or Document references	Comments
9.1 Problem reports exist and are classified by Type, Scope and Criticality			
9.2 Problem are investigated a) to determine the cause, b) evaluate the problem's relevance to safety c) investigation results are documented d) change requests are created for actions needing correct or and rationales for taking no action are documented			
9.3 Relevant parties are advised of the existence of the problem, as appropriate.			
9.4 Change requests are approved observing the requirements of the change control process. NOTE: <i>a special process may exist for emergencies and their appropriateness and overuse checked. If none exists consider if the company is prepared to handle an emergency related to the risk of the device.</i>			
9.5 Records of problem reports and their resolution and verification are kept. The Risk Management file is updated as appropriate.			
9.6 Problem reports are analyzed for trends not just individually			

<p>9.7 Resolutions of problems are verified to determine whether:</p> <ul style="list-style-type: none"> a) problems are resolved and the problem report closed b) adverse trends have been reversed c) change requests have been implemented in all relevant software items and associated documents d) additional problems have been introduced by the changes. 			
<p>9.8 Testing and regression testing documentation following a fix, includes:</p> <ul style="list-style-type: none"> a. Test results b. Anomalies found c. Software version tested d. Relevant hardware and software test configurations e. Relevant test tools f. Date tested g. Identification of the tester. 			



www.softwarecpr.com

20 Berkshire Drive
Winchester, MA 01890
781-721-2921

Copyright

© Copyright 2010 Crisis Prevention and Recovery, LLC. (CPRLLC), all rights reserved. SoftwareCPR® is a DBA of Crisis Prevention and Recovery, LLC and the SoftwareCPR® logo is a registered trademark.

SoftwareCPR® authorizes its clients and SoftwareCPR®.com subscribers use of this document for internal review and training. **Any other use or dissemination of this document is expressly prohibited** without the written authorization of SoftwareCPR®. Individual original FDA documents in their original form without SoftwareCPR® annotations are public information and may be shared without restriction.

Legal Disclaimer

The training document that follows **should only be applied in the appropriate context with oversight by regulatory and software professionals with direct knowledge and experience with the topics presented.** The document should not be used as a cookbook since it is not adequate for some situations and may be excessive for others.

While SoftwareCPR® attempts to ensure the accuracy of information presented, no guarantees are made since regulatory interpretations and enforcement practices are constantly changing, and are not entirely uniform in their application.

Disclaimer of Warranties: The information is provided AS IS, without warranties of any kind. CPRLLC does not represent or warrant that any information or data provided herein is suitable for a particular purpose. CPRLLC hereby disclaims and negates any and all warranties, whether express or implied, relating to such information and data, including the warranties of merchantability and fitness for a particular purpose.
