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.

Copyright 2009 SoftwareCPR®

TABLE OF CONTENTS

Copyright 2009 SoftwareCPR®

Copyright 2009 SoftwareCPR®

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

Copyright 2009 SoftwareCPR®

	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

Copyright 2009 SoftwareCPR®

- 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

Copyright 2009 SoftwareCPR®

- 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. >>

Copyright 2009 SoftwareCPR®

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.

Copyright 2009 SoftwareCPR®

<< 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

Copyright 2009 SoftwareCPR®

- 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

Copyright 2009 SoftwareCPR®

		implementation toward Configuration Items defined in this plan.
<TBD>	<TBD>	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. >>

Copyright 2009 SoftwareCPR®

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. >>

Copyright 2009 SoftwareCPR®

<< 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.

Copyright 2009 SoftwareCPR®

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.

Copyright 2009 SoftwareCPR®

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 >>

Copyright 2009 SoftwareCPR®

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. ……

Copyright 2009 SoftwareCPR®

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). >>

Copyright 2009 SoftwareCPR®

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

Copyright 2009 SoftwareCPR®

<< 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>	…….

Copyright 2009 SoftwareCPR®

<< 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 un 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. >>

Copyright 2009 SoftwareCPR®