Table of contents

1.0 Objectives / Purpose 4

2.0 Scope 4

3.0 Configuration Management Organization 4

3.1 Roles, Responsibilities and Assignment 4

3.2 CM Project Communication 6

3.3 Configuration Management Set-up 6

3.3.1 Project members and roles 7

4.0 Configuration Management 7

4.1 Project lifecycle activities mapping 7

4.2 Configuration Management Strategy 8

4.2.1 Configuration Management Tools 8

4.2.2 SharePoint 9

4.2.3 GIT Overview and access 9

4.2.4 Gitflow workflow 10

4.2.5 Git Development Tools 10

4.2.5.1 Git download 10

4.2.5.2 Git Bash 10

4.2.5.3 SourceTree 10

4.2.5.4 TortoiseGit 11

4.2.6 RFE Software GIT repository 11

4.2.7 Software Folder Structure 11

4.2.7.1 rfe 13

rsdk\_rfe\_abstract 15

4.2.8 NXP Radar Xplorer 15

4.2.8.1 Repo Naming Convention 15

4.2.8.2 Document Management 15

4.2.8.3 Software Folder structure 16

4.2.8.4 Release Baseline 17

4.2.9 Git Branching Strategy 17

4.2.9.1 develop Branch 17

4.2.9.2 master branch 17

4.2.9.3 custom,feature and bugfix 18

4.2.9.4 hotfix branches 18

4.2.9.5 release branches 18

4.2.10 Commit to develop 18

4.2.11 GIT commit messages 18

4.2.12 Git Hooks 19

4.2.13 Git Naming Conventions 19

4.2.13.1 Branch Naming 19

4.2.13.2 Tag Naming 19

4.2.14 Software Development Procedure 20

4.2.15 BitBucket Pull Request 21

4.2.16 Build Management 26

5.0 Software Release Management 27

5.1 Release Naming convention 28

5.2 Release location 28

5.1 Thirdparty deliverables and storage 28

6.0 Change Management 28

6.1 Change Management Strategy 28

6.1.1 JIRA Issue Logging Guideline 29

6.1.2 Jira workflow for RFE Software project 29

6.1.3 Reviewing the changes 31

6.1.4 Closing an issue 31

6.1.5 Responsible CCB 31

6.1.6 STRX RFE SW JIRA Board 32

6.1.7 STRX\_NXP RADAR XPLORERNXP RADAR XPLORER Jira Board 32

6.1.8 Non testable issue 32

6.2 CCB hierarchy, structure and members/roles 32

7.0 Configuration Identification 33

7.1 Identifying configuration items 33

7.2 Tools for SW development 33

8.0 Configuration Items Storage and Control 33

9.0 Configuration Status Accounting 34

9.1 Project configuration status accounting 34

10.0 Configuration Management Auditing 34

11.0 Back-up, Archiving and Disaster Recovery 34

12.0 Configuration Management Training 34

13.0 Annexes 35

14.0 Document Information 35

14.1 References 35

14.2 Terms/Acronyms and Definitions 36

14.3 Revision History 37

List of Figures

[Figure 1 GitFlow Workflow 10](#_Toc57725447)

[Figure 2 STRX RFE SW Folder Structure 12](#_Toc57725448)

[Figure 3 Document Folder Structure 14](#_Toc57725449)

[Figure 4 Test folder structure 15](#_Toc57725450)

[Figure 5 Document Folder Structure 16](#_Toc57725451)

[Figure 6 RFE GUI SW Folder Structure 16](#_Toc57725452)

[Figure 7 Branch Statergy 17](#_Toc57725453)

[Figure 8 Pull request workflow 21](#_Toc57725454)

[Figure 9 Clone URL 22](#_Toc57725455)

[Figure 10 Inline Comment 25](#_Toc57725456)

[Figure 11 Side by Side Diff 25](#_Toc57725457)

[Figure 12 Reviewer Approval 26](#_Toc57725458)

[Figure 13 Build Management 26](#_Toc57725459)

[Figure 14 Change Mangement Worflow 30](#_Toc57725460)

[Figure 15 Transition State NA for RFE 30](#_Toc57725461)

[Figure 16 Pull Request Worflow 31](#_Toc57725462)

List of Tables

[Table 1 Role and Responsibilities 4](#_Toc44338145)

[Table 2 Change Request and Owners 6](#_Toc44338146)

[Table 3 CM Tools 6](#_Toc44338147)

[Table 4 Configuration Items 21](#_Toc44338148)

[Table 5 References 23](#_Toc44338149)

[Table 6 Acronyms 24](#_Toc44338150)

[Table 7 Revision History 24](#_Toc44338151)

# Objectives / Purpose

The Software Configuration Management Plan (SCMP) provides an overview of the configuration, documentation and change management activities of STRX Radar Front End (RFE) software project.

The Automotive Configuration Management Process defined in BCAM7.0 is the main input for the creation of this document. This document is prepared in accordance with BU Auto CM policies and organizations CM Strategy and inputs from overall STRX SW CMP [[7]](#STRX_OverallSW_SCPM) .

The main purpose of this document is to define the Software CM rules which will be implemented in the project. The objectives of this plan are:

* Definition of Software CM organization, roles, main responsibilities and project assignments.
* Identification of all the content which needs to be under configuration management, organization of this content in various folders and how Configuration Management is achieved.
* Defining the access rules for configuration items, configurations, maintenance and reporting of Software CM data.
* Define deviations of change management rules for configuration items, configurations and CM data for software.
* Planning of auditing of configurations and CM data to assure their integrity.

Defined and described continous integration strategy and related tools used.

# Scope

The configuration management plan is applied for RFE software project.

For a description of the team roles and responsibilities please refer to RFE SW Project Management Plan located at [[1]](#PMP), for roles specifically defined for GIT usage, please see section [3.1](#_Roles,_Responsibilities_and)

# Configuration Management Organization

## Roles, Responsibilities and Assignment

The CM roles, CM related responsibilities and Responsible person that exist in STRX RFE SW and NXP Radar Xplorer are listed in table below

|  |  |  |
| --- | --- | --- |
| Role | Responsibilities | Role Assignment |
| Project Manager | The Project Leader/Manager is responsible for the proper functioning of the CM procedures, as defined in the SCMP in his project. For this PM:   * Appoints a Project CM * Assures CM process and tool training * Share project release plans with CM * Plan CM & Build Manager’s activities based on CMs other project priorities * Responsible for tracking CR/PRs. He is responsible for the process flow of the CR/PRs as described in the project. | Maulik Prabhudesai |
| Software Architect | Provides information to the project CM regarding the data structure during the initial project-setup in the CM database. Where his/her knowledge is needed S/he will be involved in the analysis of new problems and the verification of solved CR/PRs. | STRX One Chip RFE SW : Artur Burchard  STRX Remote : Lars Van Meurs  RFE GUI SW : Govin Ashrit |
| QAE | The Quality Assurance Engineer is responsible for ensuring that the projects are following defined procedures and practices; for taking any necessary action to improve the processes; and for safeguarding the quality of the end deliverables | STRX OneChip : Siddareddy Sadashivappa  STRX Remote : Bhavani H S |
| Configuration Manager | * Writes the Configuration Management Plan, in co-operation with the Project Manager * Supports the project member w.r.t. CM-tool/process. * Provides product workspaces for the development domain (‘private’/’working’), integration domain, testing domain, and release domain. * Carries out Configuration Audits to verify the integrity and completeness * Performs the promotion strategy and technical release handling, i.e. promoting configuration items (e.g. documents, code, etc.) from one status to another * Owns project Branching strategy, document and educate project teams * Owns project versioning strategy, document and educate project teams * Perform SCM related Tags whenever required * Perform SCM Branches whenever required * Manages Continuos integration system. * Perform formal project releases * Sets-up a directory structure and the environment that supports the way of working (WOW) * Communicates the availability of a new tag to stake holders * Responsible to setup Project Tracker for the project * Responsible for defining the WoW for issues/bugs. * Project CCB report creation and deliver it to CCB Board on time | Vinoth Ranee Kumar  Divya Vijayakumar |
| Document CM | * Administers Change Requests and Problem reports and distributes amongst CCB members * Sets up a CCB for the project * Overall document management and traceability.   SW Documents:   * Define, create and maintain the CM processes for documentation * Deploy the CM processes for documentation * Have overview of documentation conf. management | Burkhard Bräuer |
| Build Manager | * Setup automated project builds in Jenkins/Bamboo build environment * Schedule nightly builds and provide self-service model to project teams * Manage Jenkins/Bamboo server and project build environment (including software and build machine) * Configure plugins (Ex: Compiler warnings) based on project needs. * Monitor project builds and alert team if build fails * Provide release items (src / binaries) for release * Develop & maintain project build script and build scripts documentation, after build flow is defined by architects per software subsystem. * Responsible for resolving environment related build issues * Analyse build errors along with project team * Assists project team for solving build issues. * Develop & maintain project wrapper build scripts to drive top level builds. * Automate MISRA, Coverity project builds and made the results available to QA. | Vinoth Ranee Kumar |
| Project Team | * Responsible for applying CM tools and rules as defined in the SCMP. * Responsible for resolving source code related build issues * Assist build manager during build errors debugging | SW Dev Team |
| Tester/Integrator | The *Tester/Integrator* is responsible for integrating the sub-products produced by different developers and testing the integrated product or sub-product. This includes:   * Integration Test and in the Acceptance Test. * Checks content of releases * Checks the reproducibility * Writes release note for official releases | Shakti Prasad Shenoy |
| Tools/DB Administrator | Atlassian tools support is provided by IT via SNOW tickets. | IT |

Table 1 Role and Responsibilities

## CM Project Communication

Project communication structure is defined in Project Management Plan [[1]](#PMP). Please refer to this document for the configuration management communication rules and structure.

## Configuration Management Set-up

This chapter contains the project specific configuration management set-up dedicated to software development.

### Project members and roles

Additionally, to the roles defined in section 3.1, the following roles shall be defined in the Atlassian solution:

* Source Code Administrator

This role contains all administrative activities on tickets and source code, including:

* Deletion of branches in Git
* Deletion and renaming of tags in Git
* Creation of new Git repositories
* Pushing commits without a ticket association
* Rebasing a Git branch
* Source Code Developer

This is a role with typical permissions for software developers. It includes:

* Pushing commits to central repositories with ticket association
* Push merge commits
* Pushing tags to central repositories
* Pushing new branches to central repositories
* Source Code Reviewer

This is an optional role that allows people to accept or reject patch sets, without the capability to upload their own changesets. The role includes:

* Access to the source code review system
* Permission to merge or reject a changeset

Alternatively, this role can be merged with the “Source Code Developer” role.

* Source Code Reader

This is a read-only role. Members of this role cannot contribute to source code development. Their only permission is to clone Git repositories.

All permissions from lower hierarchies are automatically granted to upper hierarchies.

# Configuration Management

## Project lifecycle activities mapping

* Project initiation activities

At the time of initiating a project, Product Owner, shall perform the following setup tasks related to Configuration and Change Management processes for the project.

* Define Configuration and Change Management strategy
* Create and approve the Software Configuration Management Plan document for the project or make all required project specific updates in this document
* Initiate the Change Control Board (CCB) for this project. This shall include as mandatory members the Project Manager and the Developers involved in the project.
* Set-up project CCB meetings
* Create JIRA project and set-up permissions
* Create bitbucket repo for the project.
* Define the version naming rule applicable for this project
* Identify all Internal and External Change Request (New features, bugs and Inquiries) sources and assign one or more team members to monitor these sources on a regular basis in the table below.

|  |  |  |
| --- | --- | --- |
| **Change Request Sources** | **Owner – person who submits the issues** | **Description** |
| salesforce | Customer |  |
| Jira | Customer Engineer/Tech Leads/Project Manager/Software Developers/Testers | Internally raised tickets |

Table 2 Change Request and Owners

* Applicable Policies, Procedures, Process, Templates:

Configuration and Change Management documentation package is composed of:

1. JIRA Guideline [[2]](#JIRA_Guidelines)
2. GIT/ JIRA/ Atlassian Integration/ Review Training [[3]](#BL_AP_Training)
3. SCMP Template [[4]](#SCMP_Template)
4. CM Audit checklist [[5]](#CM_AuditCL)

## Configuration Management Strategy

### Configuration Management Tools

| Tool | Vendor Information | Version | Function |
| --- | --- | --- | --- |
| JIRA | Atlassian | 8.20.8 | Defect tracking  New features tracking  Change request tracking  Task tracking |
| Bitbucket | Atlassian | 7.17.2 | Configuration Management  Release Management  Review record management with Pull Request |
| Git | Version | 2.27.x | Version control tool supported in bitbucket. |
| Source Tree | Atlassian | 3.1.2 | Git GUI |
| Tortoise GIT | Open Source | 2.10.0.0 | Git GUI tool from NXP IT |
| Crucible | Atlassian | 4.8.8 | Review record management |
| Share Point | Microsoft | NA | Document baselining |

Table 3 CM Tools

### SharePoint

Sharepoint is used for SW deliverable document baselining for all gates and milestones. This document will be baselined part of STRX SW document deliverables. The baseline strategy for the document baselining is described in STRX Overall SCMP [[7]](#STRX_OverallSW_SCPM) under Chapter 4.2.

### GIT Overview and access

The revision control tool for the day to day work and collaboration is GIT. The tool is available here:

<https://git-scm.com>

In order to access the STRX software project git repository, user access on the git server must be granted. The correct way to get access is to log on to

<https://bitbucket.sw.nxp.com>

So the Bitbucket account is created, then submit a request to [vinoth.raneekumar@nxp.com](mailto:vinoth.raneekumar@nxp.com) for access to <https://bitbucket.sw.nxp.com/projects/STRX>

Once access to the repository has been granted, the repositories can be cloned.

For example, [the RFESW repo](http://sw-stash.freescale.net/projects/ALB/repos/linux/browse) can be cloned with the command:

*git clone* ssh://git@bitbucket.sw.nxp.com/strx/rfe.git

or

*git clone* https://nxf60632@bitbucket.sw.nxp.com/scm/strx/rfe.git

For the exact git URL for each of the STRX software subsystem, log into Bitbucket, enter a repo, then see the information in the “Clone” section in the web interface for each repository.

This will create a complete clone (copy) of the upstream repository on the local development machine.

4,2,3 Adding SSH

To enable SSH key access please follow the instruction at

<https://confluence.atlassian.com/bitbucketserver068/using-ssh-keys-to-secure-git-operations-981145241.html?utm_campaign=in-app-help&utm_medium=in-app-help&utm_source=stash>

### Gitflow workflow

Standard workflow Gitflow workflow [[16]](#GitWorkflow) is adapted to RFE software project development. Various branching strategies are explained in section 8.0

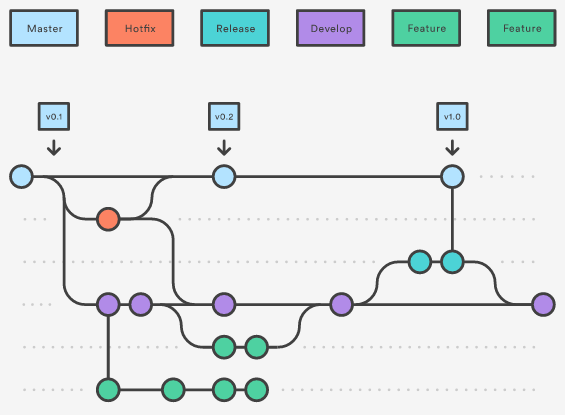


Figure 1 GitFlow Workflow

### Git Development Tools

#### Git download

Git can be dowloaded from git-scm.com/downloads

<https://git-scm.com/downloads>

Choose the GIT installer depending on the operating system of your choice.

#### Git Bash

Download and install GIT bash tool from <https://www.atlassian.com/git/tutorials/git-bash>

Git Bash is command line tool for windows to perform GIT operations.

#### SourceTree

Download and install SourceTree from Atlassian: <https://www.sourcetreeapp.com/>

During the installation, you will be prompted to introduce an account. You must create an account to start using sourcetree.

Refer to BL AP SW Generic\_JIRA\_Git\_Review\_Training [4] for steps to install and use source tree tool.

#### TortoiseGit

TortoiseGit is a Windows Shell Interface to Git and based on TortoiseSVN. It's open source tool and same is supported by NXP IT

Open a ticket via <https://nxp2.service-now.com/> to install TortoiseGit on your PC.

### RFE Software GIT repository

All software will be version controlled in Git. The central repositories will all be hosted in Atlassian Bitbucket.

<https://bitbucket.sw.nxp.com/projects/STRX>

RFE development code will be under STRX bitbucket project which host the ‘rfe’ repo

https://bitbucket.sw.nxp.com/projects/STRX/repos/rfe/browse

### Software Folder Structure

|  |
| --- |
| Rfe  ├── api.defs  ├── BITFILE.README  ├── build  ├── Build\_Script  ├── docs  │   ├── Change\_and\_Configuration\_Management  │   ├── Coding\_Standard  │   ├── Coverage  │   ├── Doxygen  │   │   ├── CMD\_RSP\_Format.pdf  │   │   ├── Doxyfile  │   │   ├── htmlSources  │   │   │   ├── dynamic\_table.html  │   │   │   ├── SAF85xx\_RFE\_FW\_Reference\_Manual.html  │   │   │   ├── SAF85xx\_RFE\_SW\_Reference\_Manual.html  │   │   │   └── static\_table.html  │   │   ├── images  │   │   ├── input  │   │   └── rfeApi\_DoxygenDocumentation.txt  │   ├── Functional\_Safety  │   ├── Project\_Management  │   ├── Quality\_Assurance  │   ├── Releases  │   ├── Requirements\_Management  │   ├── Software\_Design  │   └── Software\_Testing  │   ├── Docs  ├── GNUmakefile  ├── jenkins  │   └── Jenkinsfile  ├── pcHost  │   ├── apps  │   │   ├── proxycmd  │   │   ├── rfeProxy  │   │   ├── uartdownload  │   │   └── validationApi  │   ├── bin  │   │   ├── clients  │   │   │   ├── Labview  │   │   │   ├── Matlab  │   │   │   └── Python  │   │   ├── CMM  │   │   ├── doc  │   │   ├── examples  │   │   ├── M3SAStrx2\_G7.exe  │   │   ├── proxycmd.exe  │   │   ├── releaseinfo  │   │   ├── rfeDriver\_includes  │   │   ├── rfem7\_images  │   │   ├── testscripts  │   │   ├── uartdownload.exe  │   │   └── uartdownload\_files  │   ├── input  │   ├── output  │   ├── tools  │   │   └── apigen  │   └── units  │   ├── val  │   │   └── inc  │   ├── valHw  │   │   └── inc  │   ├── valHwReg  │   │   ├── inc  │   │   └── src  │   ├── valHwRx  │   │   ├── inc  │   │   └── src  │   └── valHwSpi  │   ├── inc  │   └── src  ├── README.md  ├── README.src.md  ├── release.sh  ├── rfeDriver  │   ├── apps  │   │   ├── rfeAppBasicExample  │   │   │   ├── inc  │   │   │   ├── Makefile  │   │   │   └── src  │   │   ├── rfeAppBasicExample32  │   │   │   ├── include  │   │   │   ├── Lauterbach\_scripts\_run\_A53\_and\_M7  │   │   │   ├── Project\_Settings  │   │   │   └── src  │   │   ├── rfeAppExtendedExample  │   │   │   ├── include  │   │   │   ├── Lauterbach\_scripts\_run\_A53\_and\_M7  │   │   │   ├── Project\_Settings  │   │   │   └── src  │   │   └── rfeAppLeadCustomer  │   │   ├── Debug\_RAM  │   │   ├── include  │   │   ├── Lauterbach\_scripts\_run\_A53\_and\_M7  │   │   ├── Project\_Settings  │   │   └── src  │   └── units  │   ├── rfe  │   │   └── inc  │   ├── rfeAbstractApi  │   │   └── inc  │   ├── rfeCfg  │   │   ├── inc  │   │   ├── src  │   │   └── unit.cmake  │   ├── rfeCmdClient  │   │   ├── inc  │   │   └── src  │   ├── rfeCmdClientStub  │   │   └── src  │   ├── rfeDynTbl  │   │   ├── inc  │   │   ├── src  │   │   └── unit.cmake  │   └── rfeSwPcMri  │   ├── inc  │   └── src  ├── rfem7  │   ├── apps  │   │   ├── readMe.md  │   │   ├── readMe.src.md  │   │   ├── rfeAnalogTest  │   │   │   ├── CMakeLists.txt  │   │   │   └── main.c  │   │   ├── rfeBringupExample  │   │   │   ├── CMakeLists.txt  │   │   │   └── main.c  │   │   ├── rfeDft  │   │   │   ├── CMakeLists.txt  │   │   │   ├── dftMatlab  │   │   │   ├── main.c  │   │   │   └── readMe.md  │   │   ├── rfeDsp  │   │   │   ├── CMakeLists.txt  │   │   │   ├── inc  │   │   │   ├── main.c  │   │   │   └── src  │   │   ├── rfeFw  │   │   │   ├── CMakeLists.txt  │   │   │   └── main.c  │   │   ├── rfeGPIOTest  │   │   │   ├── CMakeLists.txt  │   │   │   └── main.c  │   │   ├── rfeHwCrc  │   │   │   ├── CMakeLists.txt  │   │   │   ├── inc  │   │   │   ├── main.c  │   │   │   └── src  │   │   ├── rfeInit  │   │   │   ├── CMakeLists.txt  │   │   │   └── src  │   │   ├── rfeIpValFw  │   │   │   ├── CMakeLists.txt  │   │   ├── rfeM7Internal  │   │   │   ├── CMakeLists.txt  │   │   │   └── src  │   │   ├── rfeM7IntrinsicTest  │   │   │   ├── CMakeLists.txt  │   │   │   ├── inc  │   │   │   └── src  │   │   ├── rfePPOETest  │   │   │   ├── CMakeLists.txt  │   │   │   └── main.c  │   │   ├── rfeSwCalib  │   │   │   ├── CMakeLists.txt  │   │   │   ├── inc  │   │   │   ├── main.c  │   │   │   └── src  │   │   ├── rfeSysValFw  │   │   │   ├── CMakeLists.txt  │   │   │   └── main.c  │   │   ├── rfeUnitTest  │   │   │   ├── CMakeLists.txt  │   │   │   ├── readMe.md  │   │   │   ├── readMe.src.md  │   │   │   └── src  │   │   └── rfeValMiniFw  │   │   ├── CMakeLists.txt  │   │   └── main.c  │   ├── CMakeLists.txt  │   ├── code  │   │   ├── tests  │   │   │   ├── CMakeLists.txt  │   │   │   ├── testCmdServerDisp  │   │   │   ├── testDbgPrintf  │   │   │   ├── testDspMath  │   │   │   ├── testHwAcc  │   │   │   ├── testHwAtb  │   │   │   ├── testHwChirppll  │   │   │   ├── testHwCrc  │   │   │   ├── testHwCsi2  │   │   │   ├── testHwFccu  │   │   │   ├── testHwFlexIO  │   │   │   ├── testHwGbias  │   │   │   ├── testHwGldo  │   │   │   ├── testHwLldo  │   │   │   ├── testHwLoif  │   │   │   ├── testHwLpUart  │   │   │   ├── testHwMcgen  │   │   │   ├── testHwNvic  │   │   │   ├── testHwPacker  │   │   │   ├── testHwPdc  │   │   │   ├── testHwPit  │   │   │   ├── testHwPpd  │   │   │   ├── testHwRcOsc  │   │   │   ├── testHwReg  │   │   │   ├── testHwRx  │   │   │   ├── testHwRxAdc  │   │   │   ├── testHwRxbist  │   │   │   ├── testHwTe  │   │   │   ├── testHwTsense  │   │   │   ├── testHwTx  │   │   │   ├── testSwCalibration  │   │   │   ├── testSwCfgMngr  │   │   │   ├── testSwClocking  │   │   │   ├── testSwInit  │   │   │   ├── testSwMainFsm  │   │   │   ├── testSwOcOtp  │   │   │   ├── testSwPwrMngr  │   │   │   ├── testSwUtils  │   │   │   ├── testSysTick  │   │   │   ├── testWait  │   │   │   └── unity  │   │   └── units  │   │   ├── CMakeLists.txt  │   │   ├── Doxyfile  │   │   ├── rfe  │   │   ├── rfeCmdServer  │   │   ├── rfeCmdServerDisp  │   │   ├── rfeDbgPrintf  │   │   ├── rfeDft  │   │   ├── rfeDsp  │   │   ├── rfeFit  │   │   ├── rfeHw  │   │   ├── rfeHwAcc  │   │   ├── rfeHwAtb  │   │   ├── rfeHwChirppll  │   │   ├── rfeHwCrc  │   │   ├── rfeHwCsi2  │   │   ├── rfeHwCsr  │   │   ├── rfeHwFccu  │   │   ├── rfeHwFlexIO  │   │   ├── rfeHwGbias  │   │   ├── rfeHwGldo  │   │   ├── rfeHwLldo  │   │   ├── rfeHwLoif  │   │   ├── rfeHwLpUart  │   │   ├── rfeHwMcgen  │   │   ├── rfeHwNvic  │   │   ├── rfeHwPacker  │   │   ├── rfeHwPdc  │   │   ├── rfeHwPit  │   │   ├── rfeHwPpd  │   │   ├── rfeHwPpoe  │   │   ├── rfeHwRcOsc  │   │   ├── rfeHwReg  │   │   ├── rfeHwRx  │   │   ├── rfeHwRxAdc  │   │   ├── rfeHwRxbist  │   │   ├── rfeHwSpi  │   │   ├── rfeHwTe  │   │   ├── rfeHwTsense  │   │   ├── rfeHwTx  │   │   ├── rfeSwBist  │   │   ├── rfeSwCalibration  │   │   ├── rfeSwCfgMngr  │   │   ├── rfeSwCfgMngrIntegrated  │   │   ├── rfeSwClocking  │   │   ├── rfeSwInit  │   │   ├── rfeSwMainFsm  │   │   ├── rfeSwMainFsmIntegrated  │   │   ├── rfeSwOcOtp  │   │   ├── rfeSwPwrMngr  │   │   ├── rfeSwStartup  │   │   ├── rfeSwUtils  │   │   ├── rfeSysTick  │   │   └── rfeWait  │   ├── Global-after.cmake  │   ├── Global-before.cmake  │   ├── GNUmakefile  │   ├── README.md  │   ├── README.src.md  │   ├── tools  │   │   ├── arm-none-eabi-gcc-hard  │   │   ├── bfToC\_generator  │   │   ├── capture.py  │   │   ├── cmake  │   │   │   ├── arm-clang-toolchain.cmake  │   │   │   ├── arm-gcc-toolchain.cmake  │   │   │   ├── CompilerOpt.cmake  │   │   │   └── ghs-arm-toolchain.cmake  │   │   ├── coverity  │   │   │   └── config  │   │   ├── embed.sh  │   │   ├── examplesDftEclipse  │   │   ├── hex\_to\_bin.py  │   │   ├── image-sizes.jl  │   │   ├── lauterbach  │   │   ├── linker  │   │   ├── m3saSimulator  │   │   ├── make  │   │   ├── software\_gen\_header  │   │   ├── t32.sh  │   │   └── test.sh  │   └── units.required  ├── Test\_rfe  │   ├── integration  │   │   ├── generic  │   │   │   └── src  │   │   └── specific  │   │   └── STRX  │   ├── Makefile  │   ├── qualification  │   │   ├── generic  │   │   │   └── src  │   │   └── specific  │   │   └── STRX  │   ├── ratf  │   │   └── traceability  │   │   ├── Reference\_Artefacts  │   │   ├── traceability\_strx.rqtf  │   │   ├── traceability\_strx.rqtfimage  │   │   └── traceability\_strx.types  │   ├── STRX\_config.yaml  │   ├── test\_reports  │   │   └── STRX\_RFE\_SummaryTestReport.html  │   ├── test\_spec\_xml  │   │   ├── STRx\_RFE\_TS.xml  │   │   ├── topics  │   ├── tools  │   │   ├── bb\_api.py  │   │   ├── config\_tests  │   │   │   └── default.yaml  │   │   ├── templates  │   │   │   └── layout.html  │   │   └── test\_report\_to\_xml.py  │   ├── vnv\_config  │   │   └── specific  │   │   └── STRX  │   └── vnv\_config\_integration  │   └── specific  │   └── STRX  └── tools  ├── radarCycleScheduleGenerator  │   ├── generateRadarCycleSchedule.c  │   ├── Makefile  │   └── README.md  ├── rfeConfigGenerator  │   ├── docs  │   │   └── rfeConfigGeneratorUM  │   ├── netbeans  │   │   └── rfeConfigGenerator  │   └── release  └── rfeLTBcmmScript  ├── docs  ├── rsdk\_org  ├── scripts\_a53  ├── scripts\_appm7  ├── scripts\_bringup  │   └── STRX\_CM7\_RFE\_enable\_FW\_load.cmm  └── scripts\_rfem7 |

Figure 2 STRX RFE SW Folder Structure

This document covers the folders applicable for RFE software development from the STRX project repo

RFE SW is standalone monolith repo which doesn’t have any other BL AP driver/subsystem which will be coming in as submodule.

#### rfe

This folder contains software developed as part of STRX RFE SW project and related documents. The content for this is generated by STRX Radar Front end development team.

Root of this has this folder structure:

rfem7 #rfe code for m7 core

pcHost #<desc>

build #Build output dir

rfem7/apps #sample apps folder

docs # Designated for project documentation

Test\_rfe # Rfe Test docs

jenkins # CI Pipeline scripts

rfeDriver # Driver code

rfem7/tools # All tools intended towards build

rfem7/code/units # Unit code work area. Refer folder

structure for different units

rfem7/code/tests # Unit test code work area. Refer folder

structure for different unit test

tools # Tools area config tool generator

rfem7/tools/m3saSimulator # m3sasimulator code

rfem7/tools/cmake # cmake configuration files

rfem7/tools/make # make configuration

docs/Doxygen # Doxygen files and html file

A component folder may include multiple sub-dir.

rfem7

This folder contains all the RFE SW that is internal to RFE-M7 processor. This includes

* RFE API functions
* Main FSM
* RFE Specific drivers
  + Analog RFE IPs
  + Digital RFE IPs
* Infra IP Drivers

Modified from ZEBRA or recreated, this may be nedded due to code size restrictions.

rfe-host

This folder the RFE SW that is dedicated to communication between RFE and Host, including

* RFE Command Server

To be run on RFE-M7

* RFE Command Client

To be run on Application core

* RFE Driver

To be run on Application core integrates RFE Command Cliend and IPCF to implement remote procedure call for exposing RFE API to radar Application.

pc-host

This folder the RFE SW that is dedicated to PC Host, including

* Dedicated PC drivers for accessing RFE (e.g. RFE-SPI)
* Example Applications showing use of RFE API (to be defined)
* RFE GUI

apps

This folder contains the reference and demo application for the RFE software project, as we all the overall demo applications that use RFE API

At least two RFE demo applications are anticipated

* RFE Demo App demonstrating direct use of RFE API at 3 different API layers:
  + Radar System Cycle API
  + Radar Frame API
  + Low Level Drivers API
* RFE Demo App demonstrating use of RFE API via RFE Abstract

docs

This folder is dedicated for storing project planning documentation, project presentations, quality assurance reports, or engineering documentation.

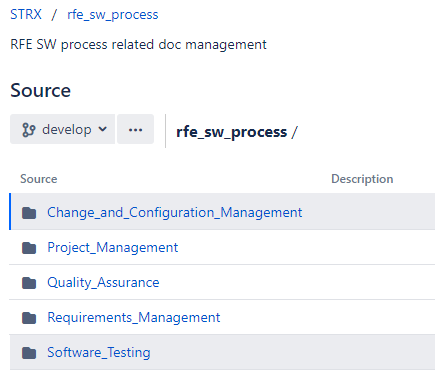


Figure 3 Document Folder Structure

**Contents:**

* **Configuration and Change Management** folder contains the RFE SW Configuration Management plan – this document
* **Project Management** folder contains the RFE SW Project Management Plan and other documents related to project management.
* **Quality Assurance** folder contains the RFE Quality Assurance Plan, RRR Reports, Metric Reports and other documents related to quality assurance
* **Release Management** folder contains software releases , release criteria and other documents related to release process.
* **Requirements Management** folder contains software requirements, customer use-cases and other documents related to requirements management. The document extract from doorsNG will be placed in this folder.
* **Software Testing** folder contains test plans, test reports and other documents related to testing

Test\_rfe

This folder contains test I&V development code as part of STRX RFE SW project and related documents. The content for this is generated by overall STRX SW I&V test team.

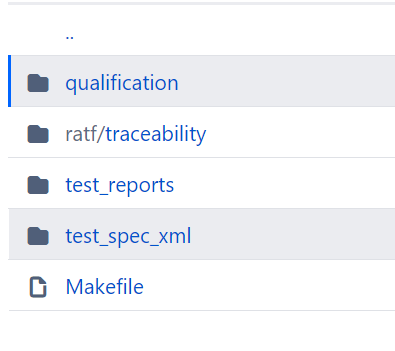


Figure 4 Test folder structure

build

All build related files will reside this folder.

#### rsdk\_rfe\_abstract

This folder is anticipated to contain

* RFE Abstract

will use RFE driver for integrating with RFE Application. This is developed as part of RSDK project but will be defined and implemented in a close cooperation with RFE SW Team.

### NXP Radar Xplorer

The NXP Radar Xplorer GUI is developed by GUI team will be version controlled in below bitbucket project. The deliverables to RFE SW will be binary, no source code will be released for GUI SW.

https://bitbucket.sw.nxp.com/projects/BLRFPGT/repos/nxpradarxplorer/browse

Project communication structure for NXP Radar Xporer GUI Developmentis defined in NXP Radar Xplorer Project Management Plan [[18]](#GUI_PMP).

The configuration management tools, workflow and startegy of RFE SW project is also applicable to Radar Xplorer as well

NXP Radar Xplorer is not safety compliance product and GUI tool will be qualified in RFE SW development .

#### Repo Naming Convention

All development repo will following the naming convention with <IC\_LABEL>rfegui

#### Document Management

All Radar Xplorer NXP Radar Xplorer related documentation will be managed in bitbucket git repo.

<https://bitbucket.sw.nxp.com/projects/BLRFPGT/repos/nxpradarxplorer/browse/docs>

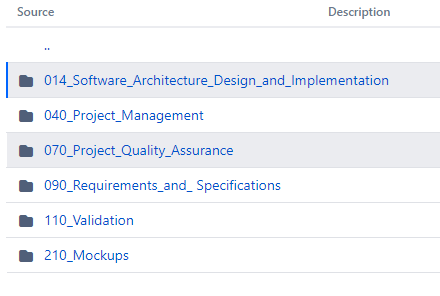
**

Figure 5 Document Folder Structure

**Contents:**

* **Project Management** folder contains the Project Management Plan and other documents related to project management.
* **Project Quality Assurance** folder contains the Quality Assurance Plan, RRR Reports, Metric Reports and other documents related to quality assurance
* **Release Management** folder contains software releases , release criteria and other documents related to release process.
* **Requirements and specification** folder contains software requirements, customer use-cases and other documents related to requirements management. The document extract from doorsNG will be placed in this folder.
* **Software Testing** folder contains test plans, test reports and other documents related to testing
* **Validation** All validation related document placeholder

#### Software Folder structure

Below folder primarily contributes the development code base for NXP Radar Xplorer NXP Radar Xplorer. All configuration management strategy applicable for RFE SW will be applicable to NXP Radar Xplorer

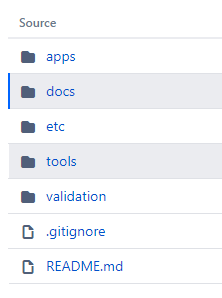


Figure 6 RFE GUI SW Folder Structure

apps : All reference application released to customer

docs : document management area

etc : externals tool license file maintenance and customer release.

tools : tools work area

validation : Test and validation code development area

#### Release Baseline

Any deliverables for STRX RFE from GUI will be built. Released and baselined and anticipated to be GIT repo per naming convention defined in later chapter. This will be intake for RFE GUI SW and also for STRX radar.

### Git Branching Strategy

The diagram below depicts the various branching strategy for STRX RFE software and STRX RFE STRX GUI SW project development workflow.



Figure 7 Branch Statergy

Branches are to be created from corresponding JIRA ticket. This ensures cross tracking among Atlassian tools and consistency in branch names.

The initial proposal is to break up branches in following category,

#### develop Branch

As the name suggests this branch is used to commit newly developed features or bug fixes to the repository. Developers shall not directly commit to this branch but create feature branches on which to develop new features or fix bugs and merge the changes back to the develop branch once the feature or bug fix has been reviewed and tested.

#### master branch

The *master* branch is used to track versions that have passed builds with all deliverables for RFE SW. These versions are considered candidates for Feature Complete, and release branching (though release branches can also be started from develop if needed).

The *master* branch will follow the develop branch and is updated only when builds on pass on *develop*.

#### custom,feature and bugfix

This branch should be pulled with associated JIRA ticket for the feature. Work happens in isolation, on feature branches. Collaboration among team members working on the same feature is done in the feature branch. Intermediate milestones (could be daily, e.g.) are pushed (after a merge/rebase operation, via a pull request) into the *develop* branch. A continuous integration testing mechanism can be put in place when pushing to *develop.*

#### hotfix branches

This branch is for implementation of hotfixes. These branches are created from the release branch and are never merged back to any other branch, it remains unended.

#### release branches

This is branched off from d*evelop* and it accommodates all activities for a release. Once release baseline is established by CCB, relevant content is sent (via pull request) to *develop* by the release manager only and release branch can be frozen (not allow any commits).

**Naming conventions**

Proposed branching naming conventions is as follows,

Master will not be taking any commits/merge, so it will be intact.

All branch names use lower cases only.

Any merge or commit will be associated with JIRA ticket and the branch would be of,

<type>/<JIRA number>\_<description>

For e.g.,

Jira request, which is a feature would be like,

feature/STRX-12\_RFE\_Clock\_Timing\_Issue

develop, release/10.0, hotfix/10.0.2,release/release\_<version>

### Commit to develop

Direct commits to *develop* branch are not allowed, one must use pull requests.

### GIT commit messages

Commit messages in GIT obey this convention:

>>>

[STRX-xx] [RFE] Short Description that fits on a single line (50 chars max). <xx> represents the ID of the JIRA issue associated with the working branch.

<empty line>  
Optional extended description

<<<

Branch naming convention will be applicable as RFE is bundled to STRX software project. The information on branch naming convention can be found in overall STRX software SCMP[[7]](#STRX_OverallSW_SCPM).

### Git Hooks

Git hooks shall be implemented before or after events such as: commit, push etc., Git hooks is built-in feature of git which runs locally.

### Git Naming Conventions

To assure consistent branch and tag naming across the repository following conventions shall be followed.

#### Branch Naming

* Names shall be all lower case, except Jira issue number
* **Release branches** shall start with the branch type release, followed by Jira ticket number for the release branch, SDK name and the release number.

Naming convention: **release/<JIRAID>-** SAF85XX\_<Product>\_<ReleaseType>\_<Version>\_D<YYMMDD>Where:

<JIRAID> - SW Release request ticket ID

**<ReleaseType> -** release type (CD,EAR,PRC,RFP)

**<Version>**- release version using lowercase and without special chars (e.g. 1.0.0 as 100)

**<YYMMDD>** - YYMMDD

Example:

release/STRX-4737-SAF85XX\_RFE\_SW\_CD\_0.8.4\_D220426

* **Topic branches** shall start with the branch type (bugfix/ or feature/), followed by Jira ticket number and a short description of the issue/feature.   
  Example: origin/feature/STRX-11

**Note:** the ‘origin/’ prefix is a git convention of prepending a ‘remote’ name to branches that are server-side.

#### Tag Naming

Tags can be added automatically by the build system upon a successful build.

The naming convention for tags can be as follows:

SAF85xx\_RFE\_SW\_BN\_<date>\_<time>\_BuildNumber

Where:

* **<date>**: date in yymmdd format (e.g. 0501420)
* **<time>**: time is format HHMMSS
* **BN - Baseline**
* **[#<release>]**: optional string, if the build is done on a release branch, contains the release branch name
* **<BuildNumer>**: Jenkins CI build Number (e.g. 1327)

Eg.,

SAF85xx\_RFE\_SW\_BN\_150622\_101235\_1327

### Software Development Procedure

Standard workflow Gitflow workflow is adapted to RFE SW and RFE GUI SW project development.

* Source code commits must be related to STRX JIRA tickets to respective repo to fulfil the traceability requirements. The annotation method of the Atlassian platform must be used.

Any code change shall be reviewed via Atlassian BitBucket pull request. Feedback procedure to external parties must be defined separately for each party.

* Feature branch should always be created from JIRA ticket.
* STRX bitbucket project and BLRFPGT project is configured such that default branch is develop.
* Any direct commits to master or develop is disabled. Master will not be allowed for any commit or merge from any developers, its manged for releases by configuration manager.
* Any commit for development activity should be merged to develop branch of the respective repo.

Commit Configuration:

1. Any git commit needs a valid JIRA id with STRX-<xxxx> and the comment should have it, detailed in git commit message section.
2. Email ID and user ID has to be configured with exact name in the bitbucket which can be found in the bitbucket settings, eg.,

|  |
| --- |
| git config --global user.name "Vinoth Ranee Kumar"  git config --global user.email [vinoth.raneekumar@nxp.com](mailto:vinoth.raneekumar@nxp.com) |

3. No merge is allowed without approved pull request.

a. Any pull request for merge to develop will need minimum of 2 approvers to be signed off.

b. Configured CI build to pass the build.

Build tools (like make/cmake), compiler configuration and other supporting tool needs to be defined and installed by respective component owners. CM automated system (CI builds bamboo plan/Jenkins) like build VM will be configured with inputs from respective team.

If the software does not require a custom build code generation system, it shall use the “Cmake” system to generate Makefiles for the platform to build. This is applicable to any new software that will be developed by project team.

The default operating system anticipated to b Linux and the default working machines shall be used from SWIS cloud environment.

### BitBucket Pull Request

Pull request is a feature that makes it easier for dev community to collaborate using BitBucket. Both code and doc can be reviewed.

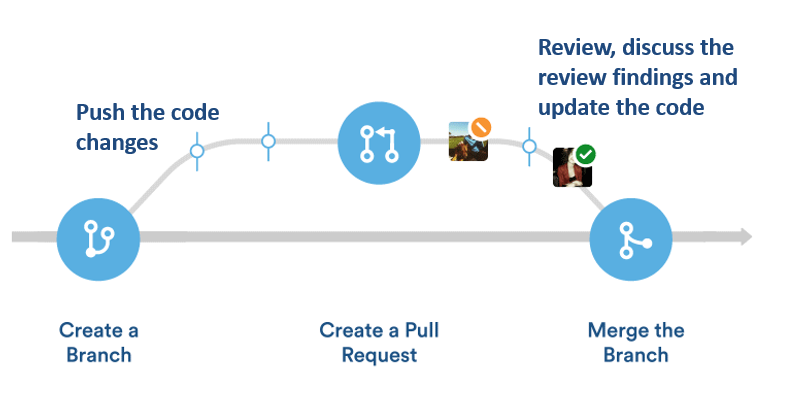


Figure 8 Pull request workflow

Following are the steps to submit Pull request

**Clone the Repo:**

Git clone the repo using your convenient tool (gitbash/ sourcetree/Tortoise Git)

If you have tortoise git right click and should see clone. If sourcetree, open the app and clone the repo in your required directory

Eg., with gitbash,

|  |
| --- |
| cd <ws>  git clone <git\_url> .    If you have SSH key configured. Then you can use ssh based clone, else https would work.  Eg.,  git clone ssh://git@bitbucket.sw.nxp.com/strx/ipc.git |

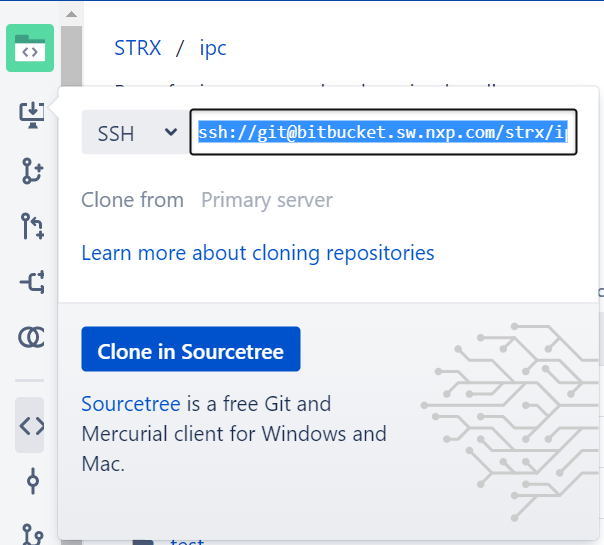


Figure 9 Clone URL

**Create Feature branch**

Create branch from JIRA ticket that is filed to track your issue via create branch link in the JIRA ticket.

Choose the appropriate git repo .

|  |
| --- |
| Choose branch- feature  Branch from - develop    Choose create branch |

If you have already cloned the repo, you need to sync from remote to get the feature branch created from JIRA ticket.

|  |
| --- |
| # Sync the code to latest if needed  $ git pull  # Now checkout the branch created from JIRA ticket  $ git checkout <feature/<yourbranchname>  Eg.,  (You can copy the branch name from bitbucket server "…" drop down "Copy branch name" )  $ git checkout feature/STRX-104\_strx\_review\_test  Switched to branch 'STRX-103\_strx\_review\_test'  # Check if you are in your feature branch  $ git branch  master  **\* feature/STRX-104\_strx\_review\_test** |
|  |

Make changes to the document or required source files.

Perform required UT for commit preparation.

**Git add and commit**

**For new files added do not forget to git add.**

Execute git add . (Note if you want files individualy to add , do git add <filename>)

Followed by git commit -m "[STRX-407]:Review doc"

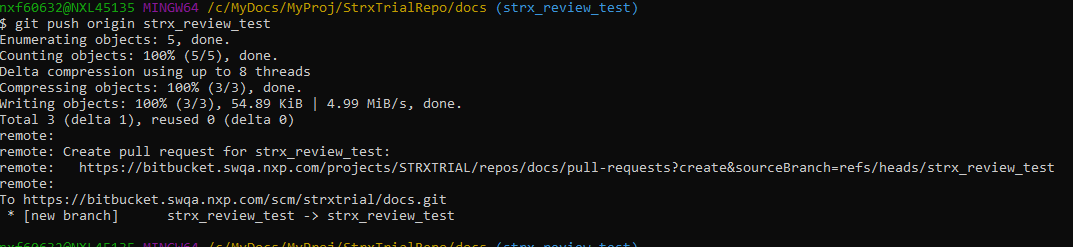
Machine generated alternative text:
nxf68632@NXL4S13S MINGV.'64 /c/myDocs/myProj/StrxTria1Repo/docs (strx_review_test) 
$ git status 
n branch strx review test 
Changes not staged for commit: 
(use "git add 
" to update what will be committed) 
(use "git restore . 
" to discard changes in working directory) 
modified: 
eae.æ smartTRX . dacx 
no changes added to commit (use "git add" and/or "git commit -a") 
nxf68632@NXL4S13S MINGV.'64 /c/myDocs/myProj/StrxTria1Repo/docs (strx_review_test) 
$ git commit -a -m "Review for doc" 
[strx_review_test 88f3f8a] Review for doc 
1 file changed, insertions(+), deletions(-) 

The same can be done viaTortoise GIT client options.

**Push changes to branch**

Use Tortoise Git and push the changes Or via using git bash,

git push origin </feature/targetbranch>



**Submit a Pull Request**

Open the URL from about git push command in your preferred internet browser.

This will open up the pull request page. Select the branch to merge (master/feature) for feeback.

Update the comments and click create pull request.

Your review is now submitted.

Reviewing Pull Request

Review comments can be added to general comment under activity.

Add comment/observation in comment "what to say"

Machine generated alternative text:
RFE SW SCMP for review for STRX PPA 
Overview Diff Commits 
Details 
Vinoth Ranee Kumar created a pull request 5 days ago 
STRX RFE SCMP Review with the alignment on repo structure and other updates. 
Activity 
What do you want to say? 
Maulik Prabhudesai marked the pull request as 
NEEDS WORK 
Yesterday 

Record all your review comments for review records.

Pull request will capture the review feedbacks.

Machine generated alternative text:
Activity 
What do you want to say? 

If it’s a document review a reviewed inline document can be attached in the same activity section.

Any inline comments to the code can be added from the diffs section via inline comment option.

Machine generated alternative text:
Overview 
Show diff of 
Diff 
Commits 
All changes in this pull request 

Machine generated alternative text:
qual tc øøøl.c 
Add a comment on this 'irk. 

Figure 10 Inline Comment

You can have a Side-by-Side comparison of the changes. Go to "…" on the right on same page,

And then select “Side-by-side diff”

Machine generated alternative text:
generic J' src J' Port.c 
Razvan Lucaci (re4471) 
Razvan Lucaci (re4471) 
Vo Tuan Anh (544846) 
16/e3/2e15 
08/04/2015 
24/e7/2e15 
ENGRøø34882e 
ENGRøø351e52 
ENGRøø363263 
Updated to be 
Fix misra err 
Remove DET er 
81 
83 
85 
Razvan 
Razvan 
Lucaci 
Lucaci 
(re 4471) 
(re 4471) 
85 
88 
95 + 
vo Tuan Anh (044846) 
16,'ø3/2e15 
e8,'ø4/2e15 
24/0,' 2015 
3e,'1e/2e15 
07,'a1/2e16 
27/06/2015 
17,'1e/2e16 
Ignore whitespace 
Hide comments 
Hide e-ditt 
Unified ditt 
Side-by-side ditt 
Duc Ta 
(053913) 
30/1ø/2e15 
e7/e1/2e16 
27/e6,'2e15 
ENGRøø368573 
ENGRøø372117 
ENGRøø378357 
in Port_lpw.c 
Inconsistent/ 
Kepnords and 
Add support f 
Verify all Mi 
88 
93 
Duc Ta 
(553913) 
Razvan Lucaci (nxa19458) 
Ngoc Le (055593) 
Razvan Lucaci (nxa19458) 
, )dnl - DO NOT modify this M4 line! 
Ngoc Le 
Ngoc Le 
(555593) 
(b55593) 
ENGRae372117 
ENGRae378357 
E%Røe384424 
Keywords and 
.20dd support fc 
Verify all Mis 
Check paramete 
DO NOT modify this '44 line! 

Figure 11 Side by Side Diff

Review rework

Author fixes the review comments and commits to the same feature branch to update the pull request.

Approving Pull Request

Once the reviewer feedback is addressed. Reviewer signs off the review via approve button in the pull request link,

Machine generated alternative text:
o e Merge 
— T"-is is 
to merge 

Figure 12 Reviewer Approval

Merging Pull Request

Software delopment procedure section [4.2.14](#_Software_Development_Procedure) details the merge critera.

The merge ikon in the pull request page gets enabled for developer to merge the changes.

### Build Management

All RFE SW builds are managed through Atlassian Bamboo.

<http://lsv05432.swis.nl-cdc01.nxp.com:8080/view/STRX_BB_Project/job/SmartTRX/job/rfe/>

The build script and other tools should be installed by developer for the automated build system to use.

Any change to toolchain/tool version has to be notified to the CM for updating in the automated system.



Figure 13 Build Management

**Integration Build**

The Integration build shall be run on each change (Pull Request) before it is merged. The build must pass before the change can be merged. The Integration build includes checking of MISRA e.g. no undocumented violations are introduced and that code compiles on all affected platforms. Other checks might be introduced with the intention of increasing code quality.

**Nightly Build**

The Nightly build generates a package output. The package output includes at least a standalone installer .exe file or a .zip file containing the same files to be installed by the installer. Other outputs of the build can be defined as need arises. (This is yet to be decided).

**Build Artifacts**

All the build artifacts will be stored in car buildshare under STRX folder.

<https://bitbucket.sw.nxp.com/projects/STRX/repos/bin_rfe/browse>

Subfolder will be created for daily and release builds.

All team members will have read access to the shared folder.

# Software Release Management

This chapter contains the release management activities performed for RFE software project, the release procedures and release criteria applicable for RFE project.

After succesfull testing of the configuration in the release branch, a release candidate will be created depending on the naming convention of the image defined in later sections.

The tools delivered along with the package information should be captured and documented in release notes for reference. This is needed for tools traceability. The list of tools delivered in package is captured in release plan document.

Automated system shall be used to generate final image for release.

A target in the top level Makfile should build all required binaries and then create the final package by packing parts of the whole tree into executable format.

By this approach, any release package can be recreated later from the corresponding snapshot.

This candidate is then handed to

* Developers for final release testing
* V&V Team
* Quality doc info, for release readiness review

The RFE software release plan is managed part of STRX overall software release which can be found at strx release [[8]](#STRX_ReleasePlan).

Each release is tagged in the git repo and maintained. The release artifacts are stored in the flexera for release.

STRX\_NXP Radar Xplorer team will be releasing zip which will be consumed by RFE SW team. There wouldn’t be source code release from RFE SW GUI team.

## Release Naming convention

Following release naming convention for release package can be followed,

SAF85xx\_<Product>\_<ReleaseType>\_<Version>\_D<YYMMDD>

<Product> => Product name like RFE\_SW/RADARFW

<ReleaseTtype> => EAR/PRC/RFP/CD

<Version> => product Release version number.

<DYYMMDD> => Date of the release year and month 2202

<extrainfo> => Any additional info (Ex: Customer specific release) (Optional)

<uniqueID> => Build ID / TimeStamp

ext - .bin /.exe

Example

SAF85xx\_RFE\_SW\_CD\_0.8.0\_D220212.zip

SAF85xx\_RadarXplorer\_CD\_v1.2.0\_D220212\_Setup.exe

Note: The recommended practice is to automate the BIN naming conventions via build / packaging scripts to avoid typo errors.

This package will be bundled with the overall STRX Software release product.

## Release location

Release packages are stored on Flexera.

NXP intranet persons can access following link to register in order to download STRX software packages: <https://are.nxp.com/FlexNetCatalog.aspx>

RFE GUI SW will be delivered to RFE SW project via git repo management.

## Thirdparty deliverables and storage

Third party deliverables like Cadence Xtensa tool chain and libraries are managed and stored in STRX project area or separate location like carbuild share.

Safety library should be checked in to the git repo. The tools delivered along with the package information should be captured and documented in release notes for reference and the same needs to be checked into the git repo.

All RFE GUI SW related tools will be managed and stored in carbuildshare Build\_SWs directory,

\\wiv6016.wbi.nxp.com\CarBuildShare\Build\_SWs

# Change Management

## Change Management Strategy

RFE Software project is following BL AP JIRA Guideline [[2]](#JIRA_Guidelines) for all types of JIRA issues, issue workflow and fields.

This chapter contains RFE specific Change Management information, not covered by generic procedure.

JIRA Project used for tracking JIRA issues of this project is called STRX. JIRA issue key is STRX (e.g., STRX-1).

<https://jira.sw.nxp.com/projects/STRX>

### JIRA Issue Logging Guideline

When an issue is identified or a new feature is implemented, first a Jira issue must be opened.

Issue summary description:

* Length should be maximum **10 words / 60 characters (besides template)**.
* Should have a consistent format for all the bugs entered in a campaign:

**[STRX:<driver>] may be line [STRX:RFE]**

* **The component for RFE SW development issues should be ‘RFESW’, subcomponents can be later defined on the requirement basis.**

Jira labels:

Labels can be used in Jira to classify the ticket. It is mainly intended for tickets that are created for a release. Its mandatory to apply filter to track issues.

See the BL AP SW JIRA Guideline [2] for details on how to create an issue for other required fields.

### Jira workflow for RFE Software project

Refer STRX Overall SW SCMP for the JIRA workflow and other information under section section 6.1

<https://www.collabnet.nxp.com/sf/docman/do/viewDocument/projects.smarttrx/docman.root.es0.030_configuration_management/doc405541?nav=1>

### Reviewing the changes

All changes in source code and documentation which are done as result of implementing a ticket are reviewed per the process defined in the AMP SW Review Guideline [[6]](#BL_AP_SWGuidelines). The review records are linked / referenced to each ticket.

Crucible will be used for following ty

Bitbucket pull request will be used for code review; PR will also be used for documents reviews.

The type of review (inspection, walkthrough, mini walkthrough) depends on the type, complexity and maturity of the work product under review.

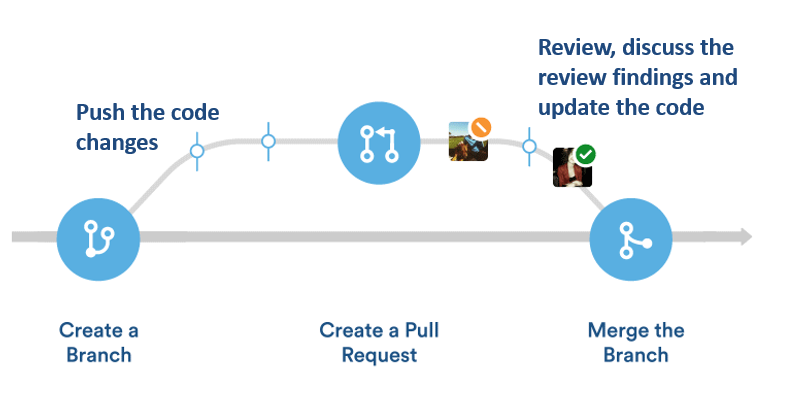


Figure 16 Pull Request Worflow

Crucible review will be used for following types of review:

 DOORS baselines, since the baseline exists in the tool it cannot be 'attached' to a PR, so we open a crucible review and mention the baseline ID

 Documents that are not yet part of git (e.g. drafts/first revisions)

### Closing an issue

Any JIRA ticket that must be closed by one of the following CCB Members defined in JIRA project [3].

Artur Burchard

Grzegorz Kulik

Maulik Prabhudesai

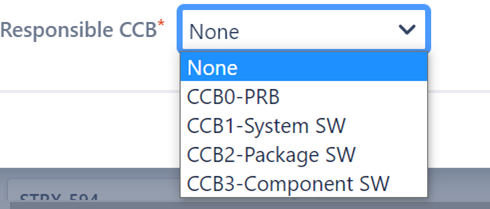
Marthijn de Man

Sushil Kumar Gupta

### Responsible CCB

‘**Responsible CCB**’ mandatory field enabled for STRX JIRA for **Bug and NewFeature** ticket type for CCB levels transitions.

The input are following, (None is not valid input)



From STRX WoW, CCB0-PRB will be in collabnet. Under SW , we will apply CCB3-Component SW which will be component level CCB which is RFESW component.

If a ticket has to be escalated and tracked at SYSTEM SW level CCB, then the responsible CCB can be changed to CCB-System SW so that it will discussed part of System CCB.

Note any new feature that is coming from CCB1 System SW CCB will be transitioned to CCB3 part of the overall STRX SW CCB.

### STRX RFE SW JIRA Board

All tickets associated to RFESW component are managed and viewed by STRX\_RFE\_SW\_BOARD.

This way the SPRINT can be associated to respective boards,

### STRX\_NXP Radar Xplorer Jira Board

All tickets associated to RFE GUI SW component are managed and viewed by STRX\_NXP Radar Xplorer\_BOARD under

https://jira.sw.nxp.com/projects/BLRFPGT/summary

NXP Radar Xplorer is anticipated to have JIRA subcomponent like GUIdocuments, GUI UI etc., depending on the requirement.

### Non testable issue

Any JIRA issue which doesn’t need run test needs to be marked with NOT\_TESTABLE JIRA label to enable the transition flow to CLOSED from RESOLVED with appropriate comment for CCB to take a decision to close.

## CCB hierarchy, structure and members/roles

STRX Software project is using Agile daily scrum meetings to track and plan JIRA tickets.

* Scrum meeting frequency is daily.
* Meeting duration is 30 minutes.
* Participants are all team members.

Maulik Prabhudesai is Scrum master and he is leading the scrum meeting.

# Configuration Identification

This chapter contains the following information:

* The list of Configuration Items and Configuration Items identification
* The Configuration Items naming conventions, labelling instructions and archiving rules

## Identifying configuration items

All identified components as per the RFE project SW ADS [[9]](#RFE_SWADS) shall be in configuration management.

All component defined in SW AS [[9]](#RFE_SWADS) and dependencies will be available in git repo.

## Tools for SW development

All list of tools applicable for software development process are captured in STRX Tools set [[12]](#STRX_ToolList).

STRX RFE SW project will be using tools applicable for development from the same.

# Configuration Items Storage and Control

The storage locations will be managed to ensure proper identification, control, status accounting, maintenance, retrieval, and backup protection.

Configuration Items are stored under GIT repository, Compass, SharePoint, DOORs and JIRA.

The same is applicable for RFE GUI SW as well.

| **Configuration Item** | **Storage** |
| --- | --- |
| Software requirements | DOORS |
| Source code | Git (Bitbucket) |
| Architectural Design [9] | DoorsNG |
| SW Design | Enterprise Architect |
| Software User Manual | Git (Bitbucket) |
| Programmers Manual/Data Sheet | Git (Bitbucket) |
| SW release note document | Git (Bitbucket) |
| SW errata | Git (Bitbucket) |
| V&V test specification | GIT |
| Test Results | Git (Bitbucket) |
| Unit Integration Test reports | Git (Bitbucket) replicate to collbabnet(if need be) |
| Qualification Test | Git (Bitbucket) |
| Release packages | Flexera |
| Regression Test Reports | Git |
| Build Makefiles | GIT |
| Test plan | GIT |

Table 4 Configuration Items

# Configuration Status Accounting

## 9.1 Project configuration status accounting

Configuration status accounting (CSA) is the process of creating and organizing the knowledge base necessary for the performance of configuration management. CSA is accomplished using various tools used by the project (JIRA, GIT/Bitbucket). The CSA process will report on various configuration management data such as:

* Metrics (Atlassian metrics, defect metrics, etc.)
* Engineer Documentation status (Configuration Items Status).

This will be done for releases preparation. This will be partly done by Doc Manger defined in role and responsibilities. Project Metrics KPIs

Configuration and Change Management Metrics (KPIs) planned for STRX is maintained in BL AP SW Metrics Plan [[17]](#BL_AP_SWMetrics).

Below are some of the KPI’s pertaining to RFE GUI SW releases (refernced from earlier released),

The list below are the one which are anticipated and definite list will be planned during first code drop or release.

1. Code quality report

2. Junit Unit test report

3. Code Coverage report

4. Requirement coverage report

5. Project metrics via PMI or JIRA charts

6. Black duck report and TA approvals

7. Code reviews report from Bitbucket

8. Release Notes

9. Updated SW\_UM

10. Java compiler warning

11. Git Baseline

12. Functional test reports

# Configuration Management Auditing

Project SQE shall conduct periodic Configuration Management audit and Change Management audits These audits are performed based on the Atlassian process audit checklists [10].

Before each release, a final audit is performed, as part of the readiness review, according to RRR checklist [11].

# Back-up, Archiving and Disaster Recovery

System and database back-up, archiving, disaster and recovery management is defined and performed by the IT department.

# Configuration Management Training

Atlassian Trainings:

* GIT [[3]](#BL_AP_Training)
* JIRA (Auto Schema) [[3]](#BL_AP_Training)
* Atlassian Integration [[3]](#BL_AP_Training)
* Review with Crucible and PullRequest [6]
* Agile methodology overview [[3]](#BL_AP_Training)

# Annexes

# Document Information

## References

| **Item** | **Description** |
| --- | --- |
| [1] | RFE software PMP  https://bitbucket.sw.nxp.com/projects/STRX/repos/rfe/browse/docs/Project\_Management/RFE\_SW\_Project\_Management\_Plan\_(Safety\_Plan).docxhttps://bitbucket.sw.nxp.com/projects/STRX/repos/rfe/browse/docs/Project\_Management/RFE\_SW\_Project\_Management\_Plan\_(Safety\_Plan).docx |
| [2] | BL AP SW JIRA Guideline  <https://nxp1.sharepoint.com/:f:/s/ampsoftware/EmSAxm3NpStOvmiD7Xn8tpcBMkJwrIPP-oxaLTmBk12IAg?e=yR2lmb> |
| [3] | [BL AP SW Generic\_JIRA\_Git\_Review\_Training](https://nxp1.sharepoint.com/:p:/r/sites/ampsoftware/Shared%20Documents/AMP%20SW%20Process%20-%20BU%20BCaM7%20Process%20Adoption%20in%20AMP%20BL/Trainings%20for%20Engineering%20(Quality%20and%20Tools)/AMP%20SW%20Generic_JIRA_Git_Review_Training.pptx?d=wee7e5437062a48c991714aa698e1fdc3&csf=1&web=1&e=IZE7jJ) |
| [4] | [SCMP Template](https://nxp1.sharepoint.com/:u:/r/sites/ampsoftware/Shared%20Documents/AMP%20SW%20Process%20-%20BU%20BCaM7%20Process%20Adoption%20in%20AMP%20BL/AMP%20Software%20-%20Change%20Configuration%20Management%20and%20Reviews/BU%20Auto%20-%20Configuration%20Management%20Plan%20Template.url?csf=1&web=1&e=pMr05v) |
| [5] | [Audit checklist](https://nxp1.sharepoint.com/:x:/r/sites/ampsoftware/Shared%20Documents/AMP%20SW%20Process%20-%20BU%20BCaM7%20Process%20Adoption%20in%20AMP%20BL/AMP%20Software%20-%20Change%20Configuration%20Management%20and%20Reviews/AMP%20SW%20Configuration%20Management%20Audit%20Checklist.xlsx?d=w055afc87ceba4270ab6133e2a32c5b2a&csf=1&web=1&e=Xb0rI0) |
| [6] | [BL AP SW review Guidelines](https://nxp1.sharepoint.com/sites/ampsoftware/Shared%20Documents/Forms/AllItems.aspx?originalPath=aHR0cHM6Ly9ueHAxLnNoYXJlcG9pbnQuY29tLzpmOi9zL2FtcHNvZnR3YXJlL0VvMGZGUktWYl9CR2tBT0t0Q29jRzRNQjc3UXh3Nm4wblBOTVNSdmtaMksyUXc%5FcnRpbWU9LUdUUnRjZjQxMGc&id=%2Fsites%2Fampsoftware%2FShared%20Documents%2FAMP%20SW%20Process%20%2D%20BU%20BCaM7%20Process%20Adoption%20in%20AMP%20BL%2FAMP%20Software%20%2D%20Change%20Configuration%20Management%20and%20Reviews%2FAMP%20SW%20Review%20Guideline%2Epdf&parent=%2Fsites%2Fampsoftware%2FShared%20Documents%2FAMP%20SW%20Process%20%2D%20BU%20BCaM7%20Process%20Adoption%20in%20AMP%20BL%2FAMP%20Software%20%2D%20Change%20Configuration%20Management%20and%20Reviews) |
| [7] | SmartTRX overall SCMP <https://www.collabnet.nxp.com/sf/go/doc312075> |
| [8] | STRX Release Plan  <https://nxp1.sharepoint.com/:p:/t/206_16/EXGRNWYI_6tPsZmL2BA_kzQBUtEzVGuUls1EG23iAsygVA?e=MSSDAz> |
| [9] | RFE SW ADS  [https://doorsng.nxp.com/rm/web#action=com.ibm.rdm.web.pages.showProjectDashboard&componentURI=https%3A%2F%2Fdoorsng.nxp.com%2Frm%2Frm-projects%2F\_r8ZsYRVKEeiLn-6Dce4MoA%2Fcomponents%2F\_entLQI3hEeqaeff\_JhQ61g&oslc.configuration=https%3A%2F%2Fdoorsng.nxp.com%2Fgc%2Fconfiguration%2F425](https://doorsng.nxp.com/rm/web" \l "action=com.ibm.rdm.web.pages.showProjectDashboard&componentURI=https%3A%2F%2Fdoorsng.nxp.com%2Frm%2Frm-projects%2F_r8ZsYRVKEeiLn-6Dce4MoA%2Fcomponents%2F_entLQI3hEeqaeff_JhQ61g&oslc.configuration=https%3A%2F%2Fdoorsng.nxp.com%2Fgc%2Fconfiguration%2F425) |
| [10] | Atlassian process audit checklists  <https://nxp1.sharepoint.com/:f:/s/ampsoftware/EmSAxm3NpStOvmiD7Xn8tpcBMkJwrIPP-oxaLTmBk12IAg?e=TjntSf> |
| [11] | RRR Checklist  <https://nxp1.sharepoint.com/sites/ampsoftware/Shared%20Documents/Forms/AllItems.aspx?id=%2Fsites%2Fampsoftware%2FShared%20Documents%2FAMP%20SW%20Process%20-%20BU%20BCaM7%20Process%20Adoption%20in%20AMP%20BL%2FAMP%20Software%20-%20Quality%20Assurance> |
| [12] | STRX Tools List  <https://www.collabnet.nxp.com/sf/go/doc428127?nav=1&pagenum=1&pagesize=15> |
| [13] | Confluence page <https://confluence.sw.nxp.com/display/BLRFP/BL+Radio+Frequency+Processing+%28RFP%29+Home> |
| [14] | GIT REPO  <https://bitbucket.sw.nxp.com/projects/STRX> |
| [15] | Share Point  <https://nxp1.sharepoint.com/teams/206_16> |
| [16] | GIT workflow  <https://www.atlassian.com/git/tutorials/comparing-workflows/centralized-workflow> |
| [17] | [BL AP software metrics](https://nxp1.sharepoint.com/sites/ampsoftware/Shared%20Documents/Forms/AllItems.aspx?id=%2Fsites%2Fampsoftware%2FShared%20Documents%2FAMP%20SW%20Process%20%2D%20BU%20BCaM7%20Process%20Adoption%20in%20AMP%20BL%2FAMP%20Software%20%2D%20Metrics%20Program&p=true&originalPath=aHR0cHM6Ly9ueHAxLnNoYXJlcG9pbnQuY29tLzpmOi9zL2FtcHNvZnR3YXJlL0VrcGNXdXlPQkE5UHNRX19GeVRuRGFZQnB3MEhEYUxHZ3I0TDZmRTNiWXJoSEE_cnRpbWU9T2E3U29wWF8xMGc) |
| [18] | <https://bitbucket.sw.nxp.com/projects/BLRFPGT/repos/saf85xxrfegui/browse/docs/Project_Management> |

Table 5 References

## Terms/Acronyms and Definitions

|  |  |
| --- | --- |
| **Acronyms** | |
| **Term/Acronym** | **Definition** |
| CCB | Change Control Board |
| CM | Configuration Management |
| PMP | Project Management Plan |
| SCMP | Software Configuration Management Plan |
| CSA | Configuration Status Accounting |
| CI | Configuration Identification |
| RRR | Release Readiness Review |
| SQE | Software Quality Engineer |
| SRS | Software Requiremnet Specificiation |
| PI | Project Initiation |
| PCA | Project Concept Approval |
| PDA | Project Definition Approval |
| PPA | Project Planning Approval |
| CR | Change Request |
| BCAM7 | Business Creation and Management version 7 |
| QAP | Quality Assurance Plan |
| PI | Platform Integration |
| SW | Software |
| GUI | Graphical user interface |

Table 6 Acronyms

## Revision History

| **Document Author** | **Version** | **Status** | **Date** | **Description of Change** | **Document Owner** |
| --- | --- | --- | --- | --- | --- |
| Vinoth Ranee Kumar | 0.1 | Draft | 05.18.2020 | Intial draft |  |
| Vinoth Ranee Kumar | 0.2 | Draft | 06.24.2020 | Rework on review comments |  |
| Vinoth Ranee Kumar | 1.0 | Approved | 07.03.2020 | Review comments agreed and closed. [R-STRX-153](https://crucible1.sw.nxp.com/cru/R-STRX-153) |  |
| Vinoth Ranee Kumar | 1.1 | Draft | 11.25.2020 | Changes for PPA and include RFE GUI SW related CM items |  |
| Vinoth Ranee Kumar | 1.2 | Approved | 06.15.2022 | Changes for CES Release | Vinoth Ranee Kumar |
| Vinoth Ranee Kumar | 1.3 | Approved | 09.26.2022 | Update for remote | Vinoth Ranee Kumar |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |

Table 7 Revision History