**ITW Corporation**

NguyenTC8Project S-Record Application

**Module C**

**Version – 1.0**

**24th Dec 2019**

NNN BishopsXXXX

London

EC2N 4AY

Tel: +44 (0)20 7890 1234

Fax: +44 (0)20 7890 1235

# **Table of Contents**

[Table of Contents 2](#_Toc149835457)

[1 Introduction 4](#_Toc149835458)

[1.1 Scope 5](#_Toc149835459)

[1.2 Definitions, Acronyms and Abbreviations 5](#_Toc149835460)

[1.3 Overview 5](#_Toc149835461)

[1.4 References 6](#_Toc149835462)

[2 Software Requirements 7](#_Toc149835463)

[3 Design 8](#_Toc149835464)

[3.1 System Overview 8](#_Toc149835465)

[3.2 Sequence diagram 9](#_Toc149835466)

[3.2.1 Feature 1 9](#_Toc149835467)

[3.3 Class diagram 11](#_Toc149835468)

[3.3.1 Feature 1 11](#_Toc149835469)

[3.3.2 Feature 2 11](#_Toc149835470)

[4 Implementation 12](#_Toc149835471)

[4.1 VendorExt-Project XXXX 12](#_Toc149835472)

[4.2 Source code structure 12](#_Toc149835473)

[4.2.1 Source code file name of components: 12](#_Toc149835474)

[4.2.2 Directory structure 12](#_Toc149835475)

[4.3 Common 13](#_Toc149835476)

[4.3.1 Constants 13](#_Toc149835477)

[4.3.2 Structure list 13](#_Toc149835478)

[4.3.3 Function list 14](#_Toc149835479)

[4.4 Features Detail 16](#_Toc149835480)

[4.4.1 Feature 1 16](#_Toc149835481)

[4.4.2 Feature 2 17](#_Toc149835482)

[5 Unit Test 20](#_Toc149835483)

[5.1 Unit Test plan 20](#_Toc149835484)

[5.1.1 Prerequisite 20](#_Toc149835485)

[5.1.2 Testing environment 20](#_Toc149835486)

[5.1.3 Convention 20](#_Toc149835487)

[5.1.4 Assumptions 20](#_Toc149835488)

[5.2 Unit Test cases 21](#_Toc149835489)

[5.2.1 Common Functions Test cases 21](#_Toc149835490)

[5.2.2 Features Test cases 21](#_Toc149835491)

[5.3 Result of the Unit Test. 21](#_Toc149835492)

[6 Open Issues 21](#_Toc149835493)

[7 Appendix 21](#_Toc149835494)

RECORD OF CHANGES

|  |  |  |  |
| --- | --- | --- | --- |
| **Document Version History** | | | |
| **Version** | **Date** | **Author** | **Details** |
| 0.1 | 30/10/2023 | Trần Minh trí | Created |
| 0.2 | 31/10/2023 |  | Updated sau khi Trần Cao Nguyên |
| 0.3 |  |  |  |
|  |  |  |  |

**APPROVERS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Distribution List** | | | | |
| **Name** | **Role** | **Signature** | **Date** | **Version** |
| Trần Cao Nguyên |  |  |  |  |

**CONFIDENTIALITY**

This document is distributed on a restricted basis, is commercial in confidence to the recipient, and may not be used for any purpose other than that associated with an FPT Software Academy project. The contents of this document may not be disclosed to any third parties without the expressed advance written authorization of FPT Software Academy.

# Introduction

The document is to describe how Project S-Record Application

## Scope

N/A

Definitions, Acronyms and Abbreviations

Table : Abbreviations in this document

|  |  |
| --- | --- |
| **Acronym** | **Reference** |
| VSCode | Visual Studio Code |

Overview

This document includes following sections:

1. **Introduction:**

This section introduces some overview information about Project S-Record Application.

1. **Software Requirement:**

This section contains excerpted parts from the requirement (SRS document) which relate to Project S-Record Application to focus the functionality.

1. **Design:**

This section includes diagrams which describe functionality, from outline to detail, in Project S-Record Application. The diagrams are block diagram, sequence diagram, and others.

1. **Implementation:**

This section describes followings:

* List of parameters which are corresponding to requirement document.
* Source code’s directory structure of Project S-Record Application.
* Description about template files’ source code.
* List of constants, enum types, structures, other user-defined data types which are used.
* List of functions with detailed descriptions for using in source code.
* Some notice when coding Project S-Record Application.

1. **Unit test:**

This section includes information relates to Unit Test for Project S-Record Application as following:

* Description of the Unit Test plan.
* List of Unit Test cases and their description.
* Result of the Unit Test.

References

|  |  |
| --- | --- |
| **Documentation Name** | **Description** |
| HCM23\_FRF\_EMB\_09 - C Module - Mock Project.pdf | System Requirement Specification document which contains requirement of all features. |
| https://en.wikipedia.org/wiki/SREC\_(file\_format) | Technical specification about S-Record format. |
|  |  |

# Software Requirements

**Table 2: Software requirement table for Project XXXX**

|  |  |  |
| --- | --- | --- |
| **Module YYYY** | | |
| **Reference** | **Requirement** | **Note** |
| **1** | **Form a sequence of integer values to S-Records and add them to an S-Record file.** |  |
| **1.1** | Allowing the user to input a sequence of integer values |  |
| **1.2** | Converting these integer values into the S-Record format |  |
| **1.3** | Adding the generated S-Records to an S-Record file. |  |
| **2** | **Convert binary data from a specific file to S-Records and add them to an S-Record file.** |  |
| **2.1** | Allowing the user to input file binary path |  |
| **2.2** | Convert the binary data into S-Record format. |  |
| **2.3** | Add them to the S-Record file. |  |
| **3** | **Convert a normal binary file to an S-Record file. (1-1 conversion)** |  |
| **3.1** | Prompt the user to specify the specific binary file from which data will be converted. |  |
| **3.2** | Read the binary data from the specified file. |  |
| **3.3** | Convert the binary data into S-Record format. |  |
| **3.4** | Add the generated S-Records to new S-Record file. |  |
| **4** | **Check for corruption in an S-Record file.** |  |

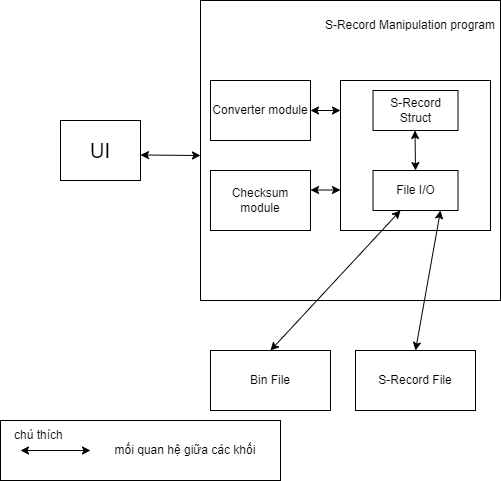
# Design

This section includes diagrams which describe behavior of components in Project S-Record Application.

The diagrams shown following using tool Drawer.io

System Overview

The following diagram shows system architecture of PROJECT S-Record Application



**Figure 1: PROJECT S-Record Application system block diagram**

* **UI:** Đây là thành phần giao diện người dùng chịu trách nhiệm trình bày hệ thống menu và hướng dẫn sử dụng phần mềm thân thiện với người dùng. Người dùng có thể chọn các hoạt động khác nhau và thành phần này tương tác với các mô-đun khác theo lựa chọn của người dùng**.**
* **Converter module** : Khối này thể hiện việc chuyển đổi từ các data như : Integer(số liệu người dùng nhập) hay các file binary thành định dạng của 1 struct trong Format Record.
* **Checksum Module:** Khối này kiểm tra xem tập tin S-Reacords có bị sai sot gì khi truyền tải hay không
* **S-Record struct – File I/0:** Khối này thể hiện việc tương tác giữa data đã được định dạng với file (bao gồm Read – Write – Append )
* **Bin File-Srecord File :** Khối này thể hiện nơi lưu trữ data ở định dạng Bin hay Srec

Sequence diagram

### Form a sequence of integer values or Binary File Path input by user convert to a S-Record format and add them to an S-Record file.

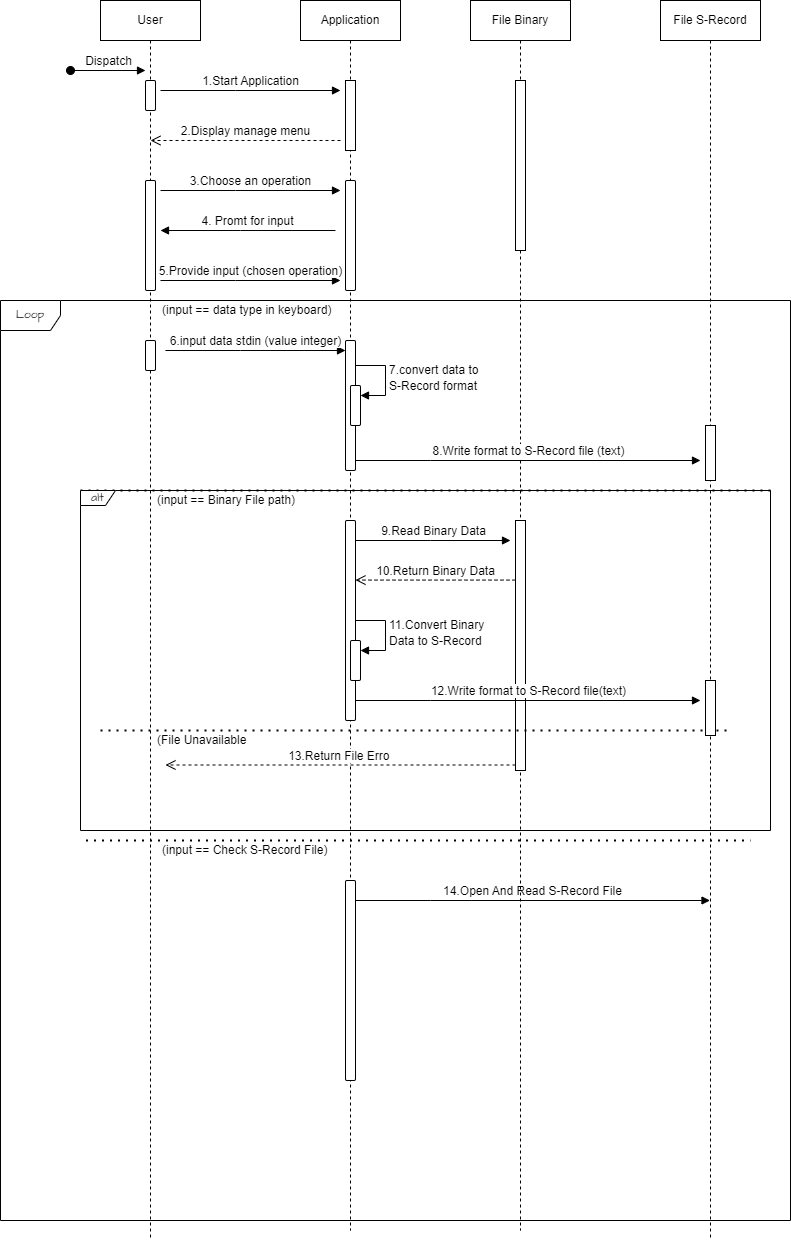


Figure 1: Allowing the user to input a sequence of integer values or binary file path

Class diagram

### Form any values to S-Records and add them to an S-Record file.

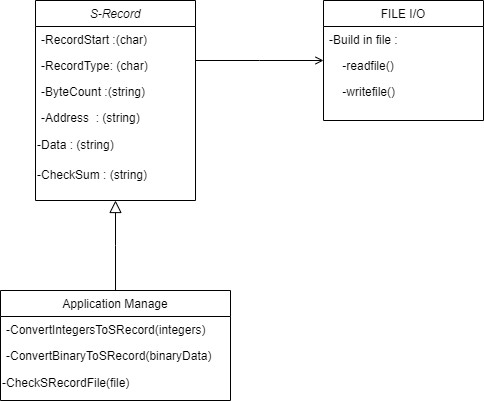


Figure 1: Class Diagram for application organiztion

Flowchart diagram

#### Flowchart demonstrates data analysis when reading out fields

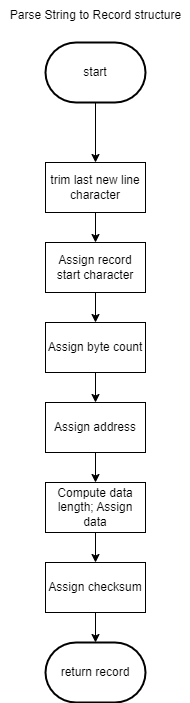


Figure 1: Parse String into each component

#### Flowchart demonstrates high address value in Srec file

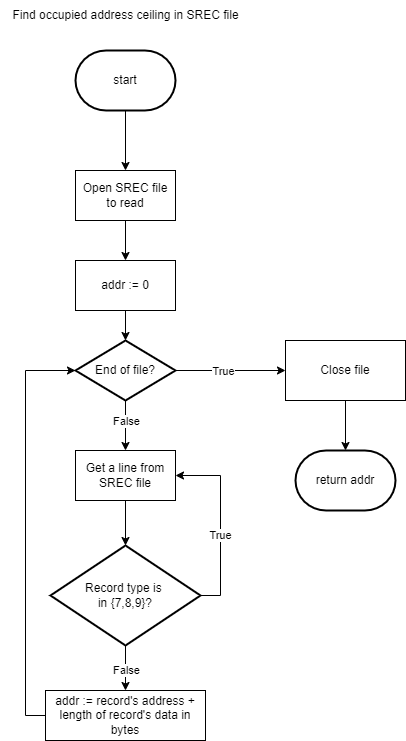


Figure 2: Determine high address in file SRec

#### Flowchart for feature 1

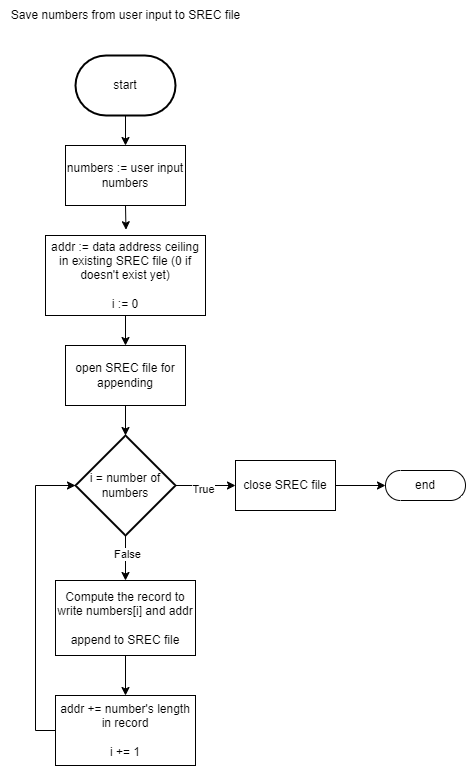


Figure 3: Feature 1 in Programing

# Implementation

Git Repository : Project S-Record.git

VendorExt-Project XXXX

We enclose here the VendorExt-Project XXXX for what we collect from the SoR. The Data Model template and the dimclient data model will get this file as input its own data model.

Refer to Data Model file here:

…

Source code structure

### Source code file name of components:

**Common:**

Project\_XXXX\_common.h

Project\_XXXX\_common.c

**Feature 1:**

Project\_XXXX\_feature1.h

Project\_XXXX\_ feature1.c

**Feature 2:**

Project\_XXXX\_ feature2.h

Project\_XXXX\_ feature2.c

### Directory structure

$PROJECT\_HOME/modules/

├── Project\_XXXX

│   ├── inc

│      ├── project\_XXXX\_common.h

│      ├── project\_XXXX\_feature1.h

│      └── project\_XXXX\_feature2.h

│   └── src

│   ├── project\_XXXX\_common.c

│      ├── project\_XXXX\_feature1.c

│      └── project\_XXXX\_feature2.c

Common

### Constants

|  |  |  |
| --- | --- | --- |
| **Constant Name** | **Value** | **Description** |
| PROJECT\_XXXX\_IFACENAME\_LENGTH | 24 | Maximum length of an interface |
| PROJECT\_XXXX\_IFACE\_MAX\_NUM | 16 | Maximum number of interfaces |
| PROJECT\_XXXX\_TIMER\_STATUS\_OFF | 0 | 3 these constants belong to the enum PROJECT XXXX\_TIMER\_STATUS\_E with following meanings:  *STATUS\_OFF*: Timer does not exist  *STATUS\_RUNNING*: Timer exists and is running  *STATUS\_STOP*: Timer exists but is not running |
| PROJECT\_XXXX\_TIMER\_STATUS\_RUNNING | 1 |
| PROJECT\_XXXX\_TIMER\_STATUS\_STOP | 2 |
|  |  |  |

### Structure list

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Description** |
| PROJECT\_XXXX\_TIMER\_INFO: Structure for timer information which is used by any component of Project XXXX FM | | |
| lTimerID | XXXX\_LONG | Timer ID |
| uInterval | XXXX\_UINT32 | Timer interval |
| uStatus | PROJECT XXXX\_TIMER\_STATUS\_E | Timer status |
| bNeedRestart | XXXX\_BOOL | Indicate that timer needs to start |
| Related constants: No | | |
| *STRUCT\_NAME*: Description | | |
| fieldName1 | DATA\_TYPE[number of elements] | Array field |
| fieldName2 | DATA\_TYPE |  |
| Related constants: | | |

### Function list

|  |  |  |  |
| --- | --- | --- | --- |
| XXXX\_STATUS XXXXInit  (  PXXXX\_CONTEXT pContext  ) | | | |
| **Arguments** | pContext | PXXXX\_CONTEXT | Pointer to Context |
| **Description:**   * Outline:   This function is automatically generated from template file of Project XXXX.  In this function, the input Context (via pContext argument) will be stored to global var of Project XXXX.  In this function there are some processings is inserted for components’ purposes.   * Returned data type: XXXX\_STATUS | | | |
| **Flowchart:** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| XXXX\_VOID XXXXMeasureData  (  PXXXX\_DATA\_MEASUREMENT pDataMeasurement,  XXXX\_UINT32 uMeasuredValue  ) | | | |
| **Arguments** | pDataMeasurement | PXXXX\_DATA\_MEASUREMENT | Pointer to structure which stores data measurement. |
| uMeasuredValue | XXXX\_UINT32 | Value which was got from system and will be stored into the storing array. |
| **Description:**   * Outline:   Update current got value into the storing array, and update other related fields.   * Returned data type:   None. | | | |
| **Flowchart:** | | | |

## Features Detail

### Feature 1

#### Notes

Feature 1’ main functionality:

* Functionality 1:
* Functionality 2:

#### Structure List

Table : List of structures for Basic HG Info

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Description** |
| struct *XXXX\_INFO*: Structure is used by global variable to store information. | | |
| cDeviceType | XXXX\_CHAR[16] | Device Type |
| cHardwareVersion | XXXX\_CHAR[16] | Hardware version |

#### Function list

|  |  |  |  |
| --- | --- | --- | --- |
| XXXX\_BOOL XXXXInfoInit(); | | | |
| **Arguments** | None |  |  |
| **Description:**   * Outline:   Initialize default value for Info structure.  The function is called when the feature wants to initiate this component.  The function gets information cDeviceType, cHardwareVersion and store to another disk.   * Returned data type: bool.   The function returns whether the component initiation is successful or not. | | | |
| **Flowchart:** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| XXXX\_BOOL XXXXGetInfo**();** | | | |
| **Arguments** | sample\_intervalue | uint16 |  |
| **Description:**   * Outline:   The function gets device information by calling the API ifx\_get\_device\_info of ZZZZ SDK. The information includes Device Type, hardware version, firmware version.  The got information is stored to global variable of this component.   * Returned data type: XXXX\_BOOL.   The function returns whether getting is successful or not. | | | |
| **Flowchart:** | | | |

### Feature 2

#### Notes

ZZZZ SDK supports to perform get/set data from XYZ store to do following funtionality:

* Functionality 1.
* Functionality 2.

#### Structure List

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Description** |
| *XXXX\_RESULT*: Structure for storing checking result of all factors which are CPU, memory. | | |
| nCpuResult | XXXX\_RESULT\_E | CPU check result |
| nMemoryResult | XXXX\_RESULT\_E | Memory check result |
| Related constants:  N/A | | |

|  |  |  |
| --- | --- | --- |
| *XXXX\_RESULT*: Structure for storing result of synchronization. | | |
| cIfaceName | XXXX\_CHAR[XXXX\_DIAG\_IFACENAME\_LENGTH] | Interface name |
| nIfaceResult | XXXX\_RESULT\_E | Interface sync result |
| Related constants:  N/A | | |

|  |  |  |
| --- | --- | --- |
| **Constant List** | | |
| **Constant Name** | **Value** | **Description** |
| XXXX\_SYNC\_INTERVAL | 900 | Default interval for Sync timer. It is 15 minutes. |

#### Function list

|  |  |  |  |
| --- | --- | --- | --- |
| XXXX\_BOOL XXXXResultInit(); | | | |
| **Arguments** |  |  |  |
| **Description:**   * Outline:   Initialize default value for structure. Moreover, default interval for the timer is assigned.   * Returned data type: XXXX\_BOOL.   The function returns whether initiation is success or not. | | | |
| **Flowchart:** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| XXXX\_BOOL XXXXSync ()**;** | | | |
| **Arguments** | None |  |  |
| **Description:**   * Outline:   This function performs DSL Synchronization then stores result into the field of component’s global variable.  This function is called in 2 cases:   * + Called when feature performs sync first time.   + Called periodically by timer hanlding function. * Returned data type: XXXX\_BOOL.   The function returns whether performing is successfully or not. | | | |
| **Flowchart:** | | | |

# Unit Test

Unit Test plan

### Prerequisite

CPPUtest <http://cpputest.github.io/>

Unit test coverage percentage SHOULD be confirmed by Trainer.

### Testing environment

* All code added to target feature module source code for UT MUST be placed under the unit test flag: ProjectXXXX\_UT

### Convention

#### Test file name

* Each test file (.c) will include all test cases for all functions in a source file.
* Name of the test file will be the combination of the source file name with suffix “\_test”
* Example :

Source file : ProjectXXXX\_common.c

UT test file : ProjectXXXX\_commonTest.c

#### Function to test

TEST(UTSuite,TestXXXX\_ID\_xx)

* xx is the ID of the test case starting from ‘01’
* For testing static function, a wrapper function need to be placed in where can called to static function.
* Wrapper function name : STUB\_{name of static function to be tested}

### Assumptions

* XXXX generated source code from template will not be defined unit test
* SUCCESS = 0; FAILURE = 1

Unit Test cases

### Common Functions Test cases

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Test Name** | **Function Name** | **Description** |
|  |  |  |  |
|  |  |  |  |

### Features Test cases

#### Feature 1

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Test Name** | **Function Name** | **Description** |
|  |  |  |  |
|  |  |  |  |

#### Feature 2

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Test Name** | **Function Name** | **Description** |
|  |  |  |  |
|  |  |  |  |

Result of the Unit Test.

The result of Unit Test is reported in the document named as “UT Coverage Report”. The document is introduced separately.

# Open Issues

N/A

# Appendix