**Title of Document : Software Requirement Specification**

**(SRS) for Emulator**

Table of Contents

**Type chapter title (level 1)1**

Type chapter title (level 2)2

Type chapter title (level 3)3

**Type chapter title (level 1)4**

Type chapter title (level 2)5

Type chapter title (level 3)6

**1 SCOPE**

**1.1 Identification**

This software Requirements Specifications (SRS) describes the requirements for SWInfra, CX, Display and CXXX Interface Computer Software Configuration Items (CSCIs) for GXXXXX on the MXX/HXX platform.

**1.2 System Overview**

An overview of the relevant system configuration is shown in Figure 1.

**Figure 1.1 : System Configuration**

The GXXXXX software comprises the following CSCIs:

* SWInfra CSCI in the Box (BOX) with the following capabilities:

1. Monitor Terminal (MT) on existing Avionics Bus (X-Mux)
2. I/O Management
3. System Services (SYS)
4. Error and Fault Management

The CSCI shall be executed under a real-time multi-tasking environment, which consists of three (3) different types of tasks, namely the synchronous tasks, asynchronous tasks and background tasks.

* CX CSCI in the Box (BOX) with the following capabilities:

1. Network and Radio Management (NET)
2. Entity Management System (EMS)
3. Message Management System (MSG)

* Display CSCI residing in the Display in the Tablet Toughbook with the following capabilities:

1. Process and display all HSD graphical symbols
2. Man-Machine Interface (MMI)
3. Digital map display
4. Track and Target Management
5. Image handling and processing

* CXXX Interface CSCI residing in the Display in the Tablet Toughbook:

1. Interface with CXXX

**1.3 Document Overview**

This document describes the functional requirements for the SWInfra, CX, Display and CXXX Interface CSCIs. It will be used as a basis for detailed CSCI design and testing.

Section 2 lists the documents referenced by this document. Section 3 defines the requirements of XXX software and requirements traceability. Section 4 describes preparation for delivery. Section 5 defines acronyms and abbreviations used in this document.

**2 REFERENCES**

Documents furnished by the Authority:

N/A

ST Documents:

Software Process Engineering Manual (SPEM/23) UML Standardization Rev -

Software Project Plan Rev - (866/ESW/SPP)

Software Configuration Management Plan Rev - (866/ESW/SCMP)

XX Interface Control Document for LRUA-2 Rev B (866/XXXX/XXX)

XX Operational Requirements Document for LRUA-2 Rev A (866/XXX/XX\_ORS)

Software Requirement Specification (SRS) for Project XX MC Rev - (872/XXX/XXX dated 16 August 2004)

**3 ENGINEERING REQUIREMENTS**

**CSCI External Interface Requirements**

The XXX software interfaces are shown in Figure 2. SWInfra CSCI use MIL-STD-1553B databus and discretes to exchange data with XXX CSCI and XXXXX.

Detailed data content for the external interfaces are contained in the Electrical Interface Control Document (EICD) and Interface Control Document (ICD).

**Figure 3.1: XXX Software Interface Diagram**

**3.2 CSCI Capability** **Requirements**

**3.2.1 Actors**



**Figure 3.2: Actor List**

**3.2.1.1 System User Actor List**



**Figure 3.3: System User Actor List**

**3.2.1.2 Subsystem User Actor List**



**Figure 3.4: Subsystem Actor List**

### **3.2.2** **Emulator Use Case - Operate Emulator**

**3.2.2.1 Sending Ownship Position**

**3.2.2.1.1 Use Case Name: Sending Ownship position through Multicast**

This use case describes the process by which the emulator sends ownship position read from a CSV file.

**Actors**

Emulator

**Pre-conditions**

Emulator is in operation

**Post-conditions**

Refer to Event Flow

**“Used” Use Cases**

**Traceability**

**Event Flow**

This use case begins when the user starts the emulator application.

Basic Path

1. User clicks on LoadXML() button. Emulator read the configurations from the predefined XML file.
2. User clicks on ConnectUDP() button. Emulator uses the configurations and set the parameter for Multicast UDP socket. Emulator reads the CSV file to store the ownship positions in a Data Array for sending.
   1. Emulator loads CSV file containing Flight Data
   2. Emulator reads each row of the exported CSV Data Table
   3. Converts each selected value into their relevant data types
      1. Integer
      2. Float
      3. Boolean
   4. Adds row’s columns into a Flight-Data Data List
   5. The row will be converted into a byte array with the bit converter
   6. The bytes convert into an object of type Data
   7. Data structure is converted into an array of bytes
   8. Period of sending is determined through the time difference between the first two timestamps
3. User clicks on SendPosition() button. Emulator check if it is time to send the Ownship Position on the Ethernet through a Multicast message (ICD-X). If yes, the emulator sends the Multicast message. If no, the emulator wait for the cycle to check the time.

The use case ends when Ownship position message is successfully send on the Ethernet.

**3.2.2.1.2 Use Case Name: Sending 2525 Symbols through Multicast**

This use case describes the process by which the emulator sends Symbols read from a CSV file.

**Actors**

Emulator

**Pre-conditions**

Emulator is in operation

**Post-conditions**

Refer to Event Flow

**“Used” Use Cases**

**Traceability**

**Event Flow**

This use case begins when the user starts the emulator application.

Basic Path

1. User clicks on LoadXML() button. Emulator read the configurations from the predefined XML file.
2. User clicks on ConnectUDP() button. Emulator uses the configurations and set the parameter for Multicast UDP socket. Emulator reads the CSV file to store the symbols data in a Data Array for sending.
   1. Emulator loads CSV file containing Symbol Data
   2. Emulator reads each row of the exported CSV Data Table
   3. Converts each selected value into their relevant data types
      1. Integer
      2. Float
      3. Boolean
   4. Adds row’s columns into a Symbol-Data Data List
   5. The row will be converted into a byte array with the bit converter
   6. The bytes convert into an object of type Data
   7. Data structure is converted into an array of bytes
   8. Period of sending is determined through the time difference between the first two timestamps
3. User clicks on SendSymbols() button. Emulator check if it is time to send the Symbol Data on the Ethernet through a Multicast message (ICD-X). If yes, the emulator sends the Multicast message. If no, the emulator wait for the cycle to check the time.

The use case ends when 2525 Symbol data message is successfully send on the Ethernet.

**3.2.2.1.3 Use Case Name: Writing Send Message Logs**

This use case describes the process by which the emulator write into a CSV file of what was sent.

**Actors**

Emulator

**Pre-conditions**

Emulator is in operation

**Post-conditions**

Refer to Event Flow

**“Used” Use Cases**

**Traceability**

**Event Flow**

This use case begins when the user starts the emulator application.

Basic Path

1. User clicks on LoadXML() button. Emulator read the configurations from the predefined XML file.
2. User clicks on ConnectUDP() button. Emulator uses the configurations and set the parameter for Multicast UDP socket. Emulator reads the CSV file to store the symbols data in a Data Array for sending.
3. User clicks on SendPosition() or SendSymbols() button.
4. Every message that was sent on the Ethernet, Emulator record timestamp and message sent into CSV file
5. If row information cannot be sent successfully, record timestamp and description of error into CSV file, and print onto message console, but process for the next row continues without breaking flow

The use case ends when sent messages are successfully written in the CSV file.

**3.2.2.1.4 Use Case Name: Starting the TCP Server**

This use case describes the process by which the emulator starts an instance of a TCP server.

**Actors**

Emulator

**Pre-conditions**

Emulator is in operation

**Post-conditions**

Refer to Event Flow

**“Used” Use Cases**

**Traceability**

**Event Flow**

This use case begins when the user starts the emulator application.

Basic Path

1. User clicks on LoadXML() button. Emulator read the configurations from the predefined XML file.
2. User clicks on ConnectTCP() button. Emulator uses the configurations and set the parameter for TCP Server.
   1. Emulator creates a socket and listens to port
   2. For every successful client connected, the server creates a separate connection for communication.
3. TCP Clients will be able to connect to the Server, to start transacting messages between the applications.

The use case ends when Emulator application is exited/closed.

**3.2.2.1.5 Use Case Name: Viewing messages sent**

This use case describes the process by which the emulator displays what was sent periodically in a text window.

**Actors**

Emulator

**Pre-conditions**

Emulator is in operation

**Post-conditions**

Refer to Event Flow

**“Used” Use Cases**

**Traceability**

**Event Flow**

This use case begins when the user starts the emulator application.

Basic Path

1. User clicks on LoadXML() button. Emulator read the configurations from the predefined XML file.
2. User clicks on ConnectUDP() button. Emulator uses the configurations and set the parameter for Multicast UDP socket. Emulator reads the CSV file to store the symbols data in a Data Array for sending.
3. User clicks on SendPosition() or SendSymbols() button.
4. User select which message type to view from the message select combo box.
5. Every message that was sent on the Ethernet, Emulator display the message data on the text window for viewing.

The use case ends when Emulator application is exited/closed.

**3.2.2.1.6 Use Case Name: Sending System Status Messages**

This use case describes the process by which the emulator displays what was sent periodically in a text window.

**Actors**

Emulator

**Pre-conditions**

Emulator is in operation

**Post-conditions**

Refer to Event Flow

**“Used” Use Cases**

**Traceability**

**Event Flow**

This use case begins when the user starts the emulator application.

Basic Path

1. User clicks on LoadXML() button. Emulator read the configurations from the predefined XML file.
2. User clicks on ConnectUDP() button. Emulator uses the configurations and set the parameter for Multicast UDP socket.
3. User selects the status of each subsystem to send through a drop-down menu, which includes the following:
   1. PASSED (Default selected)
   2. FAIL
   3. DEGRADED
   4. IBIT
4. Converts each selected value into their relevant data types.
5. Period of sending is defined in the ICD document (ICD-X).
6. User clicks on SendStatus() button. Emulator check if it is time to send the Status Data on the Ethernet through a Multicast message (ICD-X). If yes, the emulator sends the Multicast message. If no, the emulator wait for the cycle to check the time.

The use case ends when System Status data message is successfully send on the Ethernet.

**3.3 Requirement Traceability**

**3.3.1 ORS to SRS Traceability**

The following matrix is based on XX Operational Requirement Specifications for XXXXXX.

| **ORS Section No** | **SRS Section No** |
| --- | --- |
| 1 | Info |
| 1.1 | Info |
| 1.2 | Info |

**3.3.2 SRS to ORS Traceability**

The following matrix is based on XX Operational Requirement Specifications for XXXXX.

| **SRS Section No** | **ORS Section No** |
| --- | --- |
| 3.2.2.1.1 | 7.4.1 |
| 3.2.2.1.1 | 7.4.1.1 |
| 3.2.2.1.1 | 7.4.1.11 |
| 3.2.2.1.1 | 7.4.1.12 |

**4 PREPARATION FOR DELIVERY**

Version Description Document (VDD) will be prepared to describe the contents of the XXX software.

**5 NOTES**

**5.1 ACRONYMS AND ABBREVIATION**

| **S/N** | **Acronym** | **Definition** |
| --- | --- | --- |
|  | A-S | Air-to-Surface |
|  | A-A | Air-to-Air |