Design Document for IMA System Integration and Verification Bench Reflect Memory Converter

Prepared by:

GE AVIC Civil Avionics Systems Company Limited (“AVIAGE SYSTEMS”)

Floor 1, Building 3, Digital Hub, 555 Dong Chuan Road, Min Hang District, Shanghai, 200241, PRC

|  |
| --- |
| GE AVIC Civil Avionics Systems (“AVIAGE SYSTEMS”) Proprietary Information 中航通用电气民用航电系统（“昂际航电”）专属信息 |
| The information in this document is GE AVIC Civil Avionics Systems Company Limited (“AVIAGE SYSTEMS”) Proprietary Information and is disclosed in confidence. It is the property of AVIAGE SYSTEMS and shall not be used, disclosed to others or reproduced without the express written consent of AVIAGE SYSTEMS. If consent is given for reproduction in whole or in part, this notice shall appear in any such reproduction in whole or in part. 本文件中所含信息是中航通用电气民用航电系统有限责任公司（以下简称“昂际航电”）的专有信息，属非公开信息。该信息为昂际航电财产。未经昂际航电明确书面同意，不得使用、泄露或复制该信息。允许复制全部或部分信息时，应在复制品上标明本提示。 |

© GE AVIC Civil Avionics Systems Company Limited (“AVIAGE SYSTEMS”), 2014

© 中航通用电气民用航电系统有限责任公司（“昂际航电”），2014

**Release Status**

**Disclaimer:** This document is not considered released unless the **Release Date** and **Released By** columns are filled out for this revision.

Revision Levels

Revision level is indicated in the document header and the **Rev.** column below, and is of one of the following types:

**Developmental:** Used for project drafts and not considered a “Formal Release”, and cannot be sent to a customer to fulfill a contractual delivery requirement. Revisions are indicated by X1, X2, X3, etc.

**Pre-Production:** Preliminary revisions, considered a “Formal Release”. Can support DRL deliveries to the customer if the contractual requirements are met. Pre-production documents undergo a pre-production checking and approval process prior to release. Revisions are indicated by 1, 2, 3, etc.

**Production:** AVIAGE SYSTEMS’ highest level of control and best effort as a company. Supports full-production efforts. Provides sufficient information to fabricate, assemble, and test deliverable production units produced in Engineering or Manufacturing. Revisions are indicated by –, A, B, C, etc.

Revision History

| **Change Order** | **Rev.** | **Rev. Date** | **Change Description** | **Release Date** | **Released By** |
| --- | --- | --- | --- | --- | --- |
| 1 | — | 140212 | Initial Creation |  | Julian.Xie |
| 2 | — | 140305 | Developmental |  | Julian.Xie |
|  |  |  |  |  |  |

Table of Contents

Section Page

1. Introduction 5

1.1 Purpose 5

1.2 Scope 5

1.3 Revision Level 5

1.4 Definition of Terms 5

1.5 Used Tools 5

2. Referenced Documents 5

3. Requirement Overview 6

4. Interface and Regulation 8

4.1 Configuration file 8

**4.1.1** VAIS configuration file 8

**4.1.2** I-SIVB configuration file 8

4.2 Reflect Memory Data Regulation 9

**4.2.1** Data format 9

**4.2.2** Data Read and Write Rules 9

**4.2.3** Data Status Setting 10

5. Detail design 10

5.1 Development environment 10

5.2 Main modules 10

**5.2.1** C# reflect memory card driver 10

**5.2.2** Configuration loader 10

**5.2.3** VAIS operator 11

**5.2.4** RFM operator 11

**5.2.5** Adapter 11

**5.2.6** Main Entry function 13

**5.2.7** Log Manager 13

5.3 Modules relationships 14

List of Figures

Figure Page

Figure 1. Software Startup process 6

Figure 2. Data Flow of I-SIVB reflect memory converter 7

Figure 3. Work Flow of I-SIVB reflect memory converter 7

Figure 4. Operations at each time cycle 12

Figure 5. Data exchange process work flow 13

Figure 6. Software components (Modules) 14

List of Tables

Table Page

Table 1. Reflect Memory Data Format with Location Address 9

# Introduction

## Purpose

This Design Document by AVIAGE SYSTEMS defines the top-level design and the details design to generate IMA system Integration and Verification Bench (I-SIVB) Reflect Memory Converter. The software is used to communicate with other reflect memory net nodes such as Iron bird, and read/Write the VAIS messages. It is part of COMAC integrated avionics test lab functionalities.

## Scope

This document applies only to software generated for the I-SIVB Reflect Memory Converter (I-SIVB RMC).

## Revision Level

This is a Developmental revision of the document and is provided as an Engineering snapshot for the purpose of the generation of the I-SIVB Reflect Memory Converter needed by I-SIVB test platform.

## Definition of Terms

|  |  |
| --- | --- |
| I-SIVB | System Integration and Verification Bench for  IMA system |
| I-SIVB RMC | I-SIVB Reflect Memory Converter |
| VAIS | Virtual Aircraft Integration Software |

## Used Tools

The following tools or software are used or to be used together with I-SIVB RMC software.

|  |  |
| --- | --- |
| VAI Control Panel | VAI Tool provide by GE. |
| General Model Bridge | Software developed by AVIAGE SYSTEMS, to convert VAIS parameters in different protocol (ARINC664 to ARINC429, for example). |

# Referenced Documents

The following documents of the exact issue shown form a part of this document to the extent specified herein. For those documents showing no date of issue, the latest issue applies.

COMAC

|  |  |
| --- | --- |
| E-C289JB054 | C919 Aircraft Avionics Laboratory Reflective Memory Network User Standard. |

# Requirement Overview

This software is a data converter between VAIS and reflect memory network. Currently, as the details of parameters protocol is not decided, we use Non Protocol Data type to read and write from the VAIS.

We shall use General Model Bridge to match each parameter after the final ICD released. Using General Model Bridge we do not to change I-SIVB RMC software, only need to write a configuration file matching each parameter’s protocol. Details of General Model Bridge are not described here.

We use VAI Control Panel to start this application by adding the execution path into the VAIS configuration file. The VAIS configuration file defines the NPD type parameters’ definition. Once the VAIS configuration file loaded, all the parameters defined in the file can be setup as VAIS parameters in the VAIS data pool. The I-SIVB RMC can get all the VAIS parameters’ handle and then read or write values.

Following is the Chart showing the startup of this application:



Figure 1. Software Startup process

Once the I-SIVB RMC starts up, it will load I-SIVB configuration file to initialize. After all the preparation work done, it begins exchanging data between VAIS and reflect memory network at a defined time cycle.Figure2 shows the data flow and Figure3 illustrates the work flow.

‘Data1’ and ‘Data4’ are defined in Document E-C289JB054.

In all the following charts, ‘RFM’ is short for reflect memory.



Figure 2. Data Flow of I-SIVB reflect memory converter



Figure 3. Work Flow of I-SIVB reflect memory converter

# Interface and Regulation

## Configuration file

Except for the VAIS configuration file which defines the NPD Parameters, there is an I-SIVB configuration file which will be read by I-SIVB RMC at the program entry.

The configuration file contains reflect memory network configures, VAIS configuration file path and others.

### VAIS configuration file

VAIS configuration file is to be loaded by VAI Control Panel at the beginning. And the SIVB parameters will be created and handles ready for I-SIVB RMC to use.

I-SIVB configuration file contains VAIS configuration file path and will read the contents after starting up.

Reading the VAIS configuration file, I-SIVB RMC will get the Parameters information defined and based on it to exchange data between reflect memory addresses and the VAIS parameters’ handles.

As what we discussed above, all the VAIS parameters are defined as Non-Protocol-Parameter and the property ‘DataFormatType’ to be ‘Double’. Our software also supported several other data types but currently we use ‘Double’.

Here are the examples:

<NonProtocolMessage MsgName="Data1" DisplayName="Data1" TransmissionPeriod="50">

<NonProtocolParam ParamName="Data1\_1" DisplayName="Static Pressure" DataFormatType="Double” />

<NonProtocolParam ParamName="Data1\_2" DisplayName="Altitude (29.92)" DataFormatType="Double” />

The Startup execution file (I-SIVB RMC) path is also defined in the configuration file.

Here is an example:

<Utility UtilityName="I-SIVB RMC" StartupFilePath="d://I-SIVB RMC.exe"/>

### I-SIVB configuration file

It includes the refresh rate; reflect memory device card id information and the application descriptions for participant definitions.

It includes the data selection and the data segment addresses in the reflect memory.

There are specific parameter addresses for data1 as well.

Following is one example:

<?xml version="1.0" encoding="utf-8"?>

<I-SIVBconfig>

<RefreshRate value="50"/>

<RfmDevice id= "0" byteSwap="true" />

<SourceSelection firstSelection="IronBird">

<IronBirdData1>

<Data offset="B09000" end="B09FFF"/>

<ParameterAddress>

<Parameter name="Static Pressure" from="0x1504E69" to="0x1504E70"/>

….

</ParameterAddress>

</IronBirdData1>

<FCSMiniRigData1>

<Data offset="B07000" end="B07FFF"/>

<ParameterAddress>

<Parameter name="Static Pressure" from="0xF04E69" to="0xF04E70"/>

…

</ ParameterAddress >

</FCSMiniRigData1>

</SourceSelection>

<Data4Message>

<Data4 offset="B0C000" end="B0CFFF"/>

</Data4Message>

<VaisConfigPath value="D:\\Test System\\Works\\I-SIVB-APP\\I-SIVB\\Config\\VAIS\_Config.icduser"/>

<I-SIVBParticipant name = "I-SIVB\_SIM\_APP" description = "AVIAGE" partNumber = "1" version = "1"/>

</I-SIVBconfig>

When ‘SourceSelection firstSelection’ is set to be ‘"IronBird’’, I-SIVB RMC will read data at the ‘IronBirdData1’ at the memory address as ‘<Data offset="B09000" end="B09FFF"/>’ and map the bytes to the right fields by using the specific parameter address.

When ‘SourceSelection firstSelection’ is set to be ‘FCSMiniRig’, I-SIVB RMC will read data at the ‘<FCSMiniRigData1>’ at the memory address as ‘<Data offset="B07000" end="B07FFF"/>’ and map the bytes to the right fields by using the specific parameter address.

‘RefreshRate’ is to configure the I-SIVB RMC software data operating time cycle (ms).

‘RfmDevice id’ is to configure the reflect memory card diver.

## Reflect Memory Data Regulation

### Data format

According to E-C289JB054, the Data1 or Data4 is a continous data segment contains all the parameters updated by data source. Each cycle all the parameters shall be updated and written on to the reflectmory network as a whole entity.

For data reader, there is no way to know whether the data is ready or not. So we design the data format by adding a head data segment indicating the data (parameters package) status before the real Data1 or Data4 data(parameters).

Table 1 is the data format example.In the Table, ’Data offset’ is the defined start address of the data segment in I-SIVB configuration file.

Table . Reflect Memory Data Format with Location Address

|  |  |  |
| --- | --- | --- |
| **Data Contents** | **Start Offset (Hex Bytes)** | **End Offset(Hex Bytes)** |
| Data Status | 00000 + Data offset | 00007 + Data offset |
| Data1(or Data4) Paramter1 | 00008 + Data offset | 0000F + Data offset |
| Data1(or Data4) Paramter2 | 00010 + Data offset | 00017 + Data offset |
| Data1(or Data4) Paramter3 | 00018 + Data offset | 0001F + Data offset |
| Data1(or Data4) Paramter4 | 00020 + Data offset | 00027 + Data offset |

### Data Read and Write Rules

The Data1 or Data4 parameters data shall be written at the address 8 bytes after the ‘Data offset’ which was defined in the I-SIVB configuration file for each data segment.

All the parameters value data shall be written in the same byte orders, Big Endian or Little Endian.

In the data segment, the parameters’ value shall be written in the same orders as the parameters defined at VAIS configuration files, otherwise it will lead non-match data and error happens. The data shall be written one by one (8 bytes for each) with no space between. Although I-SIVB RMC software supports multi-type data reading and writing, the Iron Bird and FCS Mini-Rig are using the only data format, 8 byte double float data type.

### Data Status Setting

As is mentioned above, we save the first 8 byte of the data segment marking the data segment status. For example, when data source has not generated the data yet, the status can tell the data is not ready for others to use.

Currently, we define two statuses for each data segment.

Status1: No Data, Use 0x0000 0000 0000 0000

Status2: Data in working, Use 0xFFFF FFFF FFFF FFFF

Each time when data generator write the data into reflect memory network, it shall write the Status ‘Data in working’ as the head data to the data segment (all the data segment is defined longer at least 8bytes than the data in use).

Each time when data generator pauses for reasons, it shall write the Status ‘No Data’ on to the head 8bytes of the data segment. When data generator would stop sending out data, it shall write the ‘No Data’ status onto the same data address.

There may be situations such as system crash; it may not able to set the data segment status to be No Data. These situations are ignored here.

For I-SIVB, the Iron Bird and FCS-Mini-Rig generate the Data1 parameters for I-SIVB to use; they shall also maintain the Data Status Value following the rules above.

For I-SIVB, it shall generate Data4 parameters for reflect memory network; it shall update the Data Status according to its own running status (VAIS running/Pause or shutdown).

When I-SIVB is to read the data from reflect memory network, it shall read the Status first, and then behave according to the status.

Other systems as Iron Bird, FCS-Mini-Rig shall read the Status as well if need.

# Detail design

## Development environment

The I-SIVB RMC software is mainly developed in C# language, using the VAI CommonSimDotNet.dll to handle the VAIS operations. As the reflect memory card driver is written in C language, we need to develop a C #dynamic link library as card driver first.

The whole project is developed in Visual Studio 2010 environment.

## Main modules

Following are the main modules in I-SIVB RMC software.

### C# reflect memory card driver

Firstly, we use C++ language to encapsulate the data structures defined in the c head file, and generate the C++ dynamic link library using the original C driver.

Secondly, we develop C# dynamic link library based on the C++ dynamic link library above for I-SIVB RMC software to use.

### Configuration loader

The I-SIVB Configuration contains reflect memory network configurations and VAIS information.

This function is to read the xml files and initialize the software configurations.

### VAIS operator

This module is to get the VAIS parameter information and initialize the VAIS handles for use.

It provides all the useful VAIS operations based on API provided by CommonSimDotNet.dll.

### RFM operator

This module is to get the reflect memory configuration data and initialize the Reflect memory card.

It provides all the useful Reflect memory card operations based on API provided by C# Reflect Memory Card Driver.

### Adapter

This module controls the work flow, defines and handles the refresh cycle and at each cycle it calls VAIS operator and RFM operator functions. The functions include the reflect memory card data read/write and status setting and reading, and also the VAIS parameters updating and reading. The adapter use the VAIS callback function frame work, and each at each callback function(‘ Compute Call back’), it check the VAIS status and operates differently at each status. Following charts describes the operations:

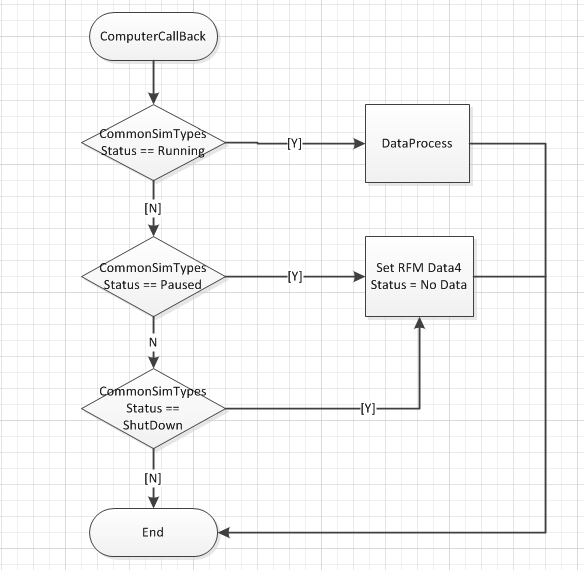


Figure 4. Operations at each time cycle

The data process contains the reflect memory data operation and VAIS data operation, which can be seen in the following chart.



Figure 5. Data exchange process work flow

### Main Entry function

It is the entry function of I-SIVB RMC; it executes the configuration loader function and goes to the Adapter module. It calls the VAIS and Reflect memory card initialization functions, and goes to the loop circle to exchange and process data if no error happens.

### Log Manager

This module provides the error logging function.

Configuration files reading error, parameters reading and matching error, also the reflect memory card operation error will be logged. There are two way of logging, console print and saved in txt file. When error happens in logging into txt file, it only logs on the console window. When error happens in configuration stage, the console window will be visible, and show the error information, give user selections, and wait user to quit or continue.

## Modules relationships

The modules listed above is combined together to build the I-SIVB RMC software. Main Entry function is the entrance and triggered by VAI Control Panel; then it open the Log Manager and ready for error messages; in the same time, it begins to load and parse the configuration files.

After the entire initialization process is ready, it calls the Adapter which will execute the data exchange process.



Figure 6. Software components (Modules)

# 