A Generic Robustness and Stress Testing Tool for Radar Software

Inder Singh Chauhan, Jang Bahadur Singh, M Justin Sagayaraj
Electronics and Radar Development Establishment, CV Raman Nagar, Bengaluru, India

***Abstract*—Software that are controlling radar are real time in nature. These software implement mission critical and safety critical requirements. Any single failure could result in major mission or operational failure. The software in radar consists of multiple interacting subsystems. Proper functioning and interactions of these subsystems in real time is very important. Therefore the interfaces between these subsystems have to be tested thoroughly for robustness, stress and boundary conditions. The interface interaction among the subsystems involve signal detection, multiple tracks tracking, Dwell scheduling, waveform parameters setting, array calibration, radar health and safety parameters monitoring etc.**

A Generic Robustness and Stress Testing Tool (GRST) was developed to perform interface stress testing. The tool scans through the interface parameters and perform stress test at interface level which will put the subsystems for robustness testing to detect major functional failure before the system is integrated into final system.

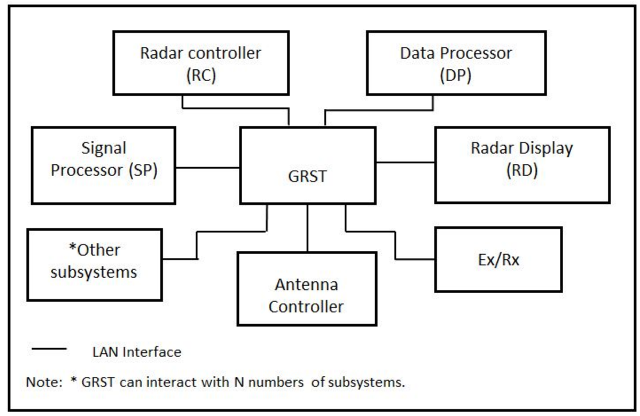

Fig. 1. Block Diagram of GRST

I. Introduction

A Radar system is a real time and mission critical system and its backend consists software modules to perform resource management (Radar controller), Signal Processor, Data Processor, Radar Display, Synchronization, etc. Radar's backend is software intensive, where multiple Radar subsystems have to interact with each other in a synchronized manner to perform the Radar's operational requirements. Interface, robustness, boundary and stress testing of the Radar subsystems are indispensable before their integration. Moreover, the interfaces between the Radar subsystems have to be verified for Radar operation requirement. The main objective behind the development of this tool is to make a generic testing tool that can take all kind of interface structures through configurations files. Configuration files can be generated through the Graphical User Interface (GUI) of the Generic Robustness and Stress Testing Tool (GRST).

Radars Interface messages parameters play vital role in subsystem communication and they contain very critical data like signal detection, multiple targets tracking, wave form parameter setting, Radar beam control parameters, health parameters, Dwell scheduling, and IQ data, any single failure of these parameters data could result the failure of mission or operation.

Section II gives an overview of GRST and section III describes its implementation. Some of the results obtained by using the tool during testing are given in section IV.

II. Generic Robustness and Stress Testing Tool

GRST is a generic software testing tool for checking interface functionality, boundaries and robustness conditions and stress testing of Radars subsystems. It has the capability to handle all kind of interface structures through tools configuration files and perform the testing on the same without changing the tool code. This tool will run on a network environment and send all interface data/parameters to target Radar subsystems and monitor the outcome of the same. This tool is Configurable based on subsystem message structures and can log the messages. A number of simulators are required to test each of these Radar subsystems. In general, these simulators are tightly coupled with the associated Radar subsystems since they are developed to test the exact functionalities of the Radar subsystems. Usually, Radar requirements are highly volatile, which leads to frequent changes in the software, as the simulators are tightly coupled with the software, this necessitates changes in the simulators as well. A block diagram showing the interaction between GRST and the Radar subsystems is shown in Fig. 1.

III. Implementation

This tool has two major modules, one for Radars subsystems interface, robustness and stress testing and the other for tool configuration through GUI. In GRST testing module user can test Radar's subsystems interfaces, robustness and stress with other related subsystem.For interface testing user can select both source and destination subsystems then all its related messages will be listed,now user can send any message to

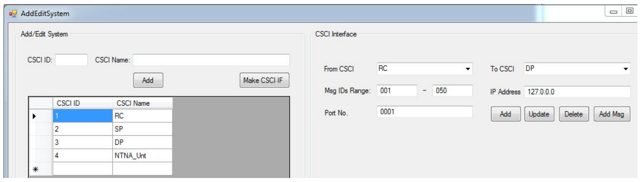

Fig. 2. Subsystem Configuration Through GUI

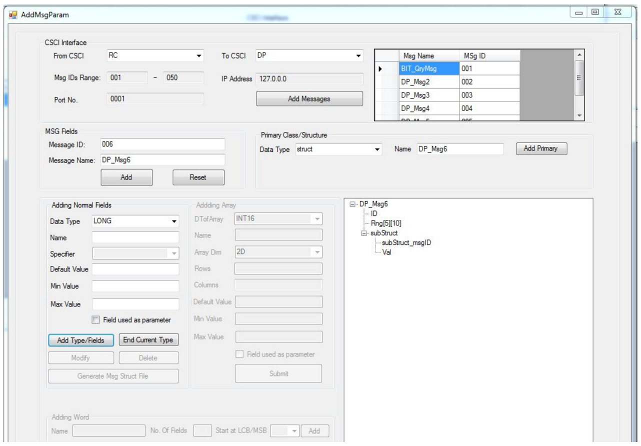

Fig. 3. Subsystem Message Configuration Through GUI

check the interrace.Boundaries and robustness test can be done by setting each message field value with min. value, max value, mid. Value, out of boundary or a user defined value and send to the target subsystem. For stress test user can set a loop limit and the tool will keep sending that message to the target subsystem continuously, till the limit is reached.In GRST GUI configuration module user can add/Edit/Delete/View the Radar subsystems and set the interface with other Radar subsystems by setting IP address, port number and messages range. After setting interface between two radar subsystem user can add messages between them by setting message name, message id,other fields like range rate, Elevation, Azimuth,number of PRF, array fields (1D and 2D) and substructure.(Fig. 2 & Fig. 3). These data will be saved in two configuration file. One configuration file contains messages list of all radar subsystem and the other contains message declarations. These configuration files can be configured manually too. The messages between a Radar's Back-end subsystems may be classified as:

i. Point to Point Messages
ii. Broadcast Messages
iii. Information Messages
iv. Logging Messages
v. Multi-cast Messages

The messages will have a message body and a common header format. The message body contains the message specific parameters. The messages include

1. BIT Query
2. BIT Response
3. Thermal Status

```
# MsgName         : MsgId : P_Id  : Port  : IP                      : IF   : minVal : maxVal

RC-->             : 1     : 1
RC-->DP           : 13    : 1     : 0001  : 127.0.0.0               : 3     : 001   : 050
BIT_QryMsg        : 001   : 13
DP_Msg2           : 002   : 13
DP_Msg3           : 003   : 13
DP_Msg4           : 004   : 13
DP_Msg5           : 005   : 13
RC-->SP           : 12    : 1     : 0002  : 127.0.0.1               : 2     : 51    : 100
SP_Msg1           : 51    : 12
SP_Msg2           : 52    : 12
RC-->NTNA_Unt     : 14    : 1     : 0003  : 127.0.0.2               : 4     : 500   : 600
SP-->             : 2     : 2
SP-->DP           : 23    : 2     : 0004  : 127.0.0.3               : 3     : 601   : 700
SP-->RC           : 21    : 2     : 0005  : 127.0.0.4               : 1     : 701   : 800
```

Fig. 4. Snippet From Message List Configuration File

```
File  Edit  Format  View  Help
#------------------------------ Message's Declaration File------------------------#

#------------------------------RC-->DP MSGs --------------------------------------#
START_MSG
MSG_ID : 001
MSG_NAME : BIT_QryMsg
MSG_LEN : 36
START_PARAM
#Type         : Name        : ExtraInfo : offset: Note  : Value : MinVal : MaxVal
START_STRUCT  : BIT_QryMsg  : 0         : 0     : 0     : 0
START_STRUCT  : MsgHdr      : 0         : 0     : 0     : 0
UINT32        : MsgId       : 0         : 0     : 0     : 001
UINT32        : Reserved    : 0         : 4     : 0     : 0
UINT32        : Reserved    : 0         : 8     : 0     : 0
UINT32        : MsgLen      : 0         : 12    : 0     : 36
UINT32        : Reserved    : 0         : 16    : 0     : 0
UINT32        : Reserved    : 0         : 20    : 0     : 0
UINT32        : Reserved    : 0         : 24    : 0     : 0
END_STRUCT    : MsgHdr      : 0         : 28    : 0     : 0
UINT32        : DP_BitTyp   : 0         : 28    : 0     : 0     : 1      : 2
UINT32        : DPSubsys_ID : 0         : 32    : 0     : 0     : 1      : 6
END_STRUCT    : BIT_QryMsg  : 0         : 32    : 0     : 0
END_PARAM
START_MSG
```

Fig. 5. Snippet From Message Declaration Configuration File

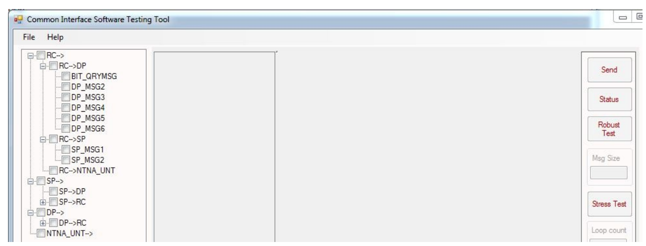

Fig. 6. Interface and Test Selection in GRST

4. Calibration Status
5. Dwell Command
6. Mode Command
7. Detection Report
8. Beam Request
9. Track Info
10. Radar State
11. Beam Position
12. Track Selection

Interface messages requirements are captured from Interface Requirement Specification (IRS) [1] and Interface Detailed Design (IDD) [2] and based on that two configuration file messages are created (Fig. 4 & Fig. 5). GRST reads these configuration files and lists all the messages. After setting the interface between Radar subsystems user can add / Edit / Delete the messages as well.(Fig. 6).The software is implemented in C# programming language under the Windows Operating System.

IV. Results

Some of the results obtained while using GRST for the interface testing of the Data Processor (DP) of one of the Radars is described in this section. Fig. 7 shows the configuration of BIT Query message between the Radar Controller (RC)

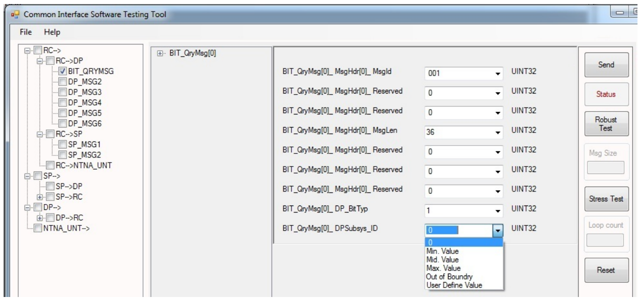

Fig. 7. Sending a BIT Query message from Radar Controller to Data Processor

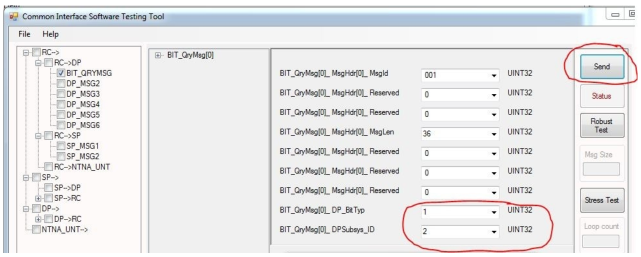

Fig. 8. Status After Sending BIT Query message from Radar Controller to Data Processor

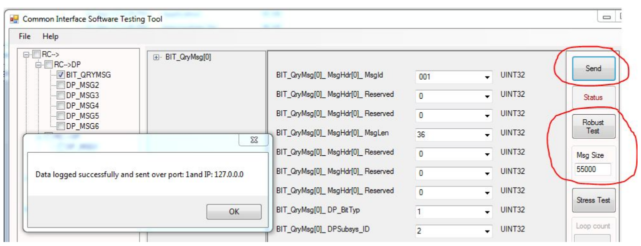

Fig. 9. A Robustness Test Scenario for Data Processor

and the DP. This message is configured through GRSTs Main Window. The field values are then set with min. / max. / mid / out of boundary / user defined values and the message is sent across. The communication status is shown in Fig. 8. A robustness test scenario for DP is shown in Fig. 9. Here the BIT Query message size has been increased to 55000 bytes and then sent to the DP from the RC. For the stress testing of this subsystem, the loop count (number of times the message has to be sent) has been increased to 10000 as in Fig. 10. The BIT Query message from RC is then sent to the DP 10000 times. These tests were performed on DP with other messages. Interfaces between other subsystems were also tested. Quite a number of failures were observed during this testing, including occasional subsystem crashes. The tool helped in bringing out the flaws quite early, thus helped in meeting the development time schedules.

V. Conclusion

GRST is a very powerful tool for checking radar subsystems interface functionalities. GRST has been used in multiple radar development and found to be very effective in early detection and removal of critical errors.

Further it is planned to enhance the functionalities of GRST for generate the messages structures and classes, make a generic interface with other radar simulators / tools, real time data analysis (snooping) integration with real, calibration and BIT failure.

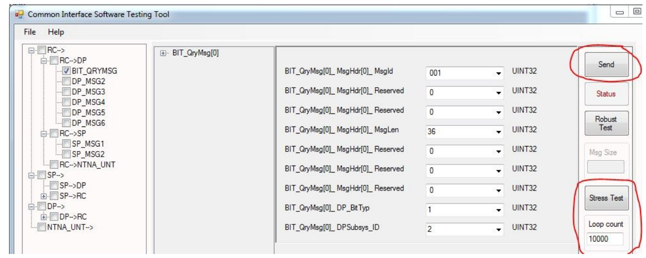

Fig. 10. A Stress Test Scenario for Data Processor

References

[1] *LRDE Airborne Radar Interface Requirement Specifications,* Internal Technical Document.

[2] *LRDE Airborne Radar Interface Design Description,* Internal TechnicalDocument.

Inder Singh Chauhan did his B.Tech Degree in Computer Science & Engineering from the Uttarakhand Technical University (2013) and joined Electronics and Radar Development Establishment, Bangalore in the year 2015. His areas of interest include Simulator Development and Radar Software IV & V.

Jang Bahadur Singh did his B.E degree in Computer Science & Engineering from Kumaum Engineering College, Dwarahat in 1996 and joined Electronics and Radar Development Establishment, Bangalore in the year 2000 as a Scientist. He has worked for the development of Radar Data Processors and is currently working in the area of Radar Simulators and IV & V of Radar Software.

Justin Sagayaraj M is a scientist in Electronics and Radar Development Establishment, Bangalore. He has received M.Tech in Computer science & Engineering from Indian Institute of Technology, Madras, MSc (computer Science) From Madurai Kamaraj University, and BSc (Special Mathematics) from Madurai Kamaraj University. He has worked for many radar tracking systems and communication systems.