STAR MASTER CONTROL COMPUTER (MCC)

DESIGN DOCUMENT

INTERMAP

DOC-1009
Version 1.04
June 8, 2006

DOCUMENT HISTORY

Version	Date	Comment/Description of Change	Author(s)
1.00	May 15th, 2003	First Draft Release	TRM
1.01	July 9th, 2003	Final MCC Requirements	TRM, WRT
1.03	July 15, 2003	Update organization of requirements, add detail on state transitions and error conditions	TRM, WRT
1.04	July 17, 2003	Changed Mistake in State Machine	TRM, WRT
1.08	April 28th, 2004	Updated all sections	
1.09	June 8th, 2004	Add MCCProps.txt, and missiondb.txt config info	TRM

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Purpose

This document describes the requirements for the development of the STAR Master Control Computer (MCC) module as part of the STAR Core Technology. The STAR Core technology provides the basic subsystems required for an interferometric radar system.

1.2 Scope

This document outlines all aspects of the MCC including: functionality, design, and operation.

1.3 Definitions, Acronyms, and Abbreviations

MTTR	Mean Time To Repair
LRU	Line Replaceable Units
TCP/IP	Network Communication protocol
MCC	Master Control Computer
ANT	Antenna Module
RCVEX-RCAS	Receiver Exciter – Radar Control and Acquisition System Module
PWRDIST	Power Distribution Module
NAV	Navigation Module
WGASS-XTRANS	Wave Guide Assembly – X-Band Transmitter

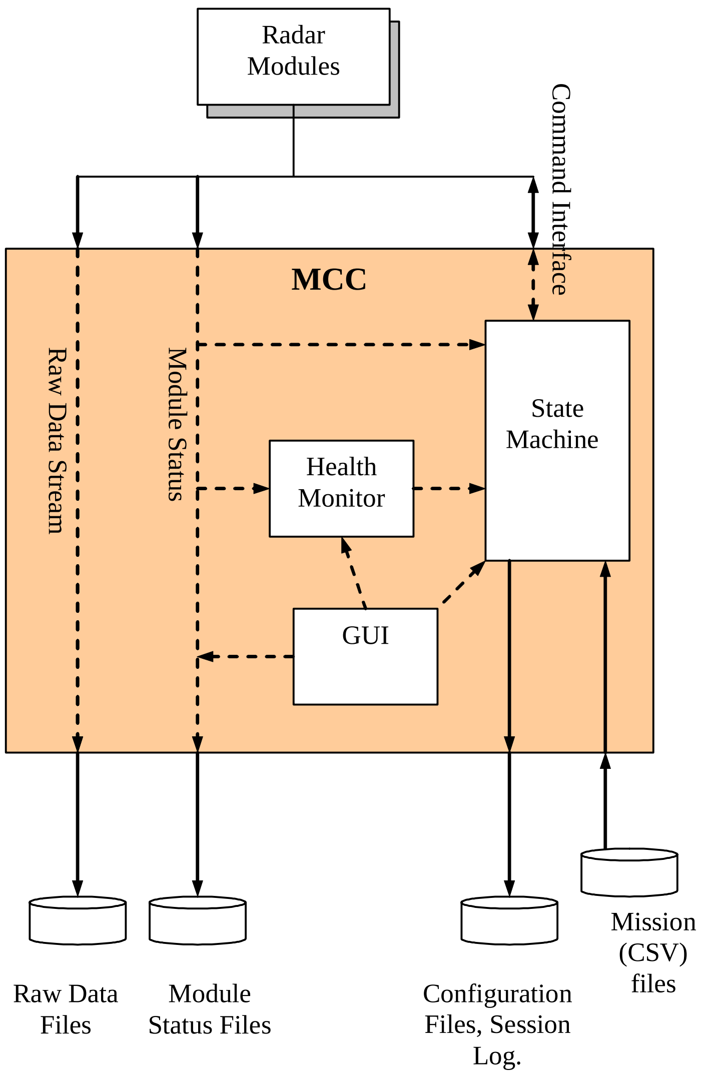
1.4 References

1. STAR System Requirements Specification (DOC–1000)
2. STAR System Interface Control Document (DOC–1002)
3. STAR Software ICD (DOC–1007)

2 FUNCTIONAL DESCRIPTION

The Master Control Computer (MCC) is the single central control of the radar system. It's primary functions include:

1. Control of the Radar System / Modules
2. User Interface for the Operator
3. Logging / Recording of data



The control of the system is broken up into states and controlled by the MCC's internal state machine. Each radar modules is controlled through the software command interface.

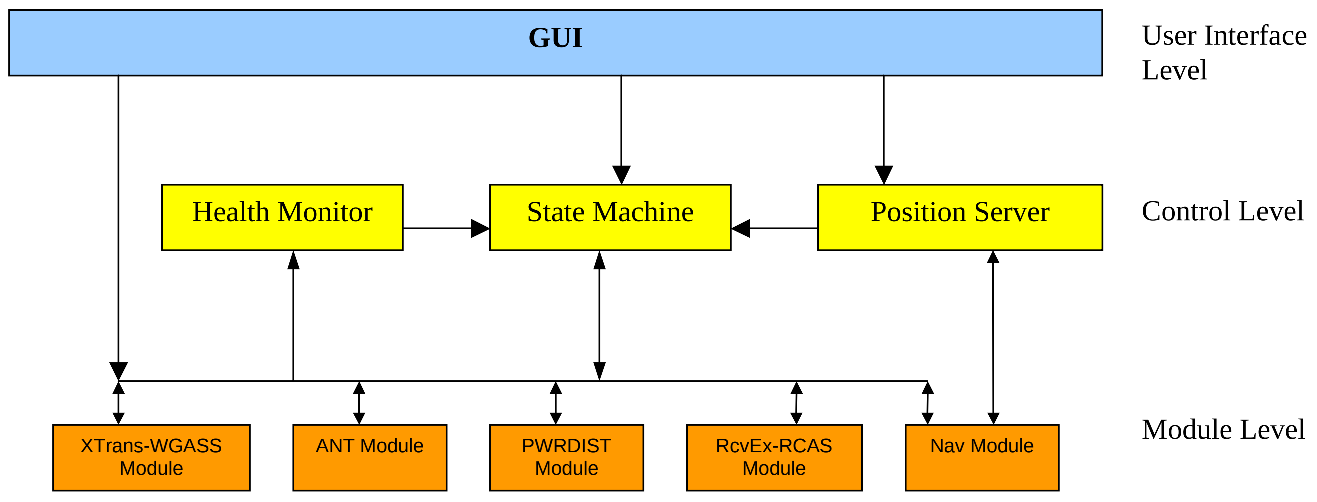
Radar System Status is received via the module status data streams. The MCC checks each module status, and writes out to a file. The Module Status provides feedback and real time information for each module. The Module Status is monitored by a 'Health Monitor', which checks and reports any errors to the State Machine.

The MCC provides a GUI for the operator to monitor and control the state of the radar system and the radar modules. The GUI monitors the Module status, the Health Monitor and State Machine and displays the information to the user. The GUI allows the operator to load mission files and abort data acquisition. Any error messages that are detected by the State Machine, or health Monitor are reported to the operator through the GUI.

The MCC reads in the raw data stream from each module and outputs it to file. The MCC will output a system configuration file collected from the modules, and a Session log containing maintenance information.

3 SOFTWARE DESIGN

3.1 Overview



The MCC Software is divided into several logical layers. At the highest level is the User Interface, where the operator interacts with the system. Next, the Control Level, maintains the state and monitors the health of the system. The lowest level is the Module Level, which consists of a software wrapper around each radar module.

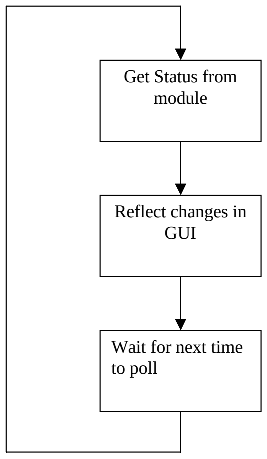
For specific design details, please refer to the Javadocs.

3.2 User Interface Level

The MCC GUI will be designed such as it is completely decoupled from all other layers. This is achieved by having no direct methods calls to the GUI layer by any other layer. This GUI is system aware, by polling the other layers for information.

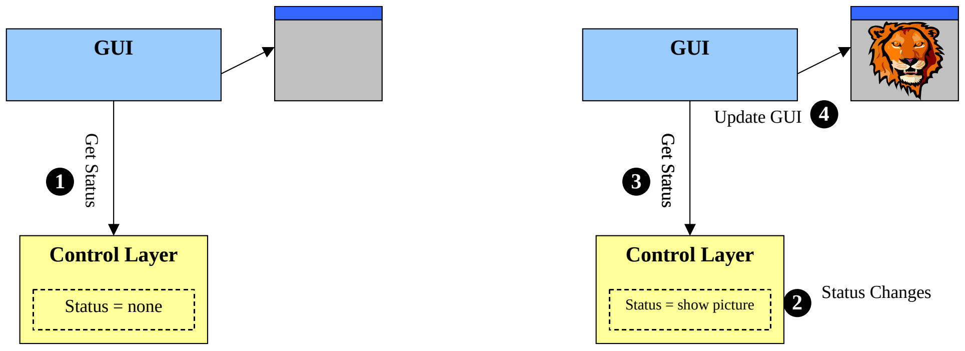
3.2.1 Theory of operation:

The GUI is controlled through a single thread which does the following:



In the MCC the GUI polls at 5Hz, which results in a wait time between polls of 200ms. This provides a quick enough response that a human operator will not notice the delay.

Refer to the following illustration:



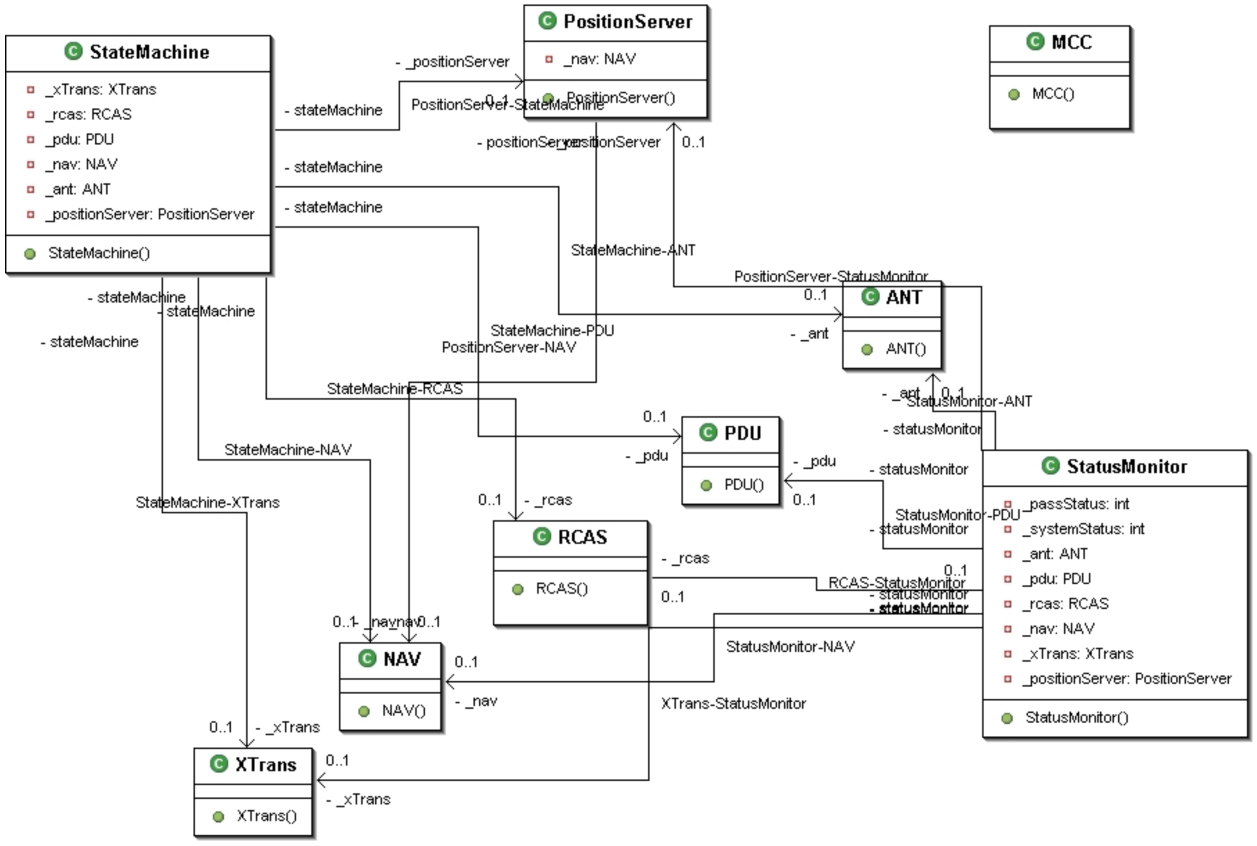
In a polling GUI, the GUI contains a thread that continually monitors the status from the other layers (1). The GUI then reflects the status on the GUI, if required.

If the status changes in the lower layer (2), the next time the GUI polls status (3) it will see the new status change and update the GUI

3.3 Control Level

The MCC control level prima consists of a state Machine, System health monitor, and Position Server.

- *Monitor* - collects the status from all of the software modules, and collects all error or warning conditions in the system.
- *StateMachine* - Controls the current system configuration and implements all of the state transition functions.
- *Position Server* - interprets the line configuration to an SCH frame, and compare the intercept points to the current GPS position. The Position Server will notify the state machine when a state change is required based on intercepting a way point.



3.3.1 *State Machine*

The state machine is the main implementation of the state transitions described in section. It is able to receive state change requests from the GUI. State changes that are based on way point intercepts originate at the Position Server, which needs to be configured with the current line and way point to intercept.

3.3.2 Position Server

The position server provides all the required position and transform data to the system in order to intercept way points. This object get (re-)initialized with the current line information and perform the SCH Transform. It polls the *NAV* module for navigation data and calculates the distance between the current position and all the way points on the current line. The state machine has access to this object to set or reset the waypoint to intercept on. When an intercept occurs the position server will notify the state machine, so that it can perform the system function to change the state.

3.3.3 Status Monitor

The status monitor continually polls (at a configurable interval) all the modules for the current status of the system. This information may be accessed by the state machine for determining whether the state may be changed. This information is also accessed by the user interface for display to the user. The Status Monitor is state aware in order to determine whether a certain condition constitutes a system fault. (Time not synchronized is not a System Fault during initialization, but it is during mapping).

3.3.4 MCC

The MCC object contains the main method for starting the application. This object is responsible for reading the MCC configuration, instantiating all the objects and starting all the threads.

3.4 Module Level

The MCC wraps each radar module in a software / client wrapper, which are referred to in the MCC as modules. Each module is responsible for:

- Controlling all IO with the physical module.
- Recording the collected data.
- Inspecting the status and making the module status available to the health monitor.

The module provides all the functionality needed by the control level (state machine). The low level TCP/IP and UDP interface is completely insulated from the Control level.

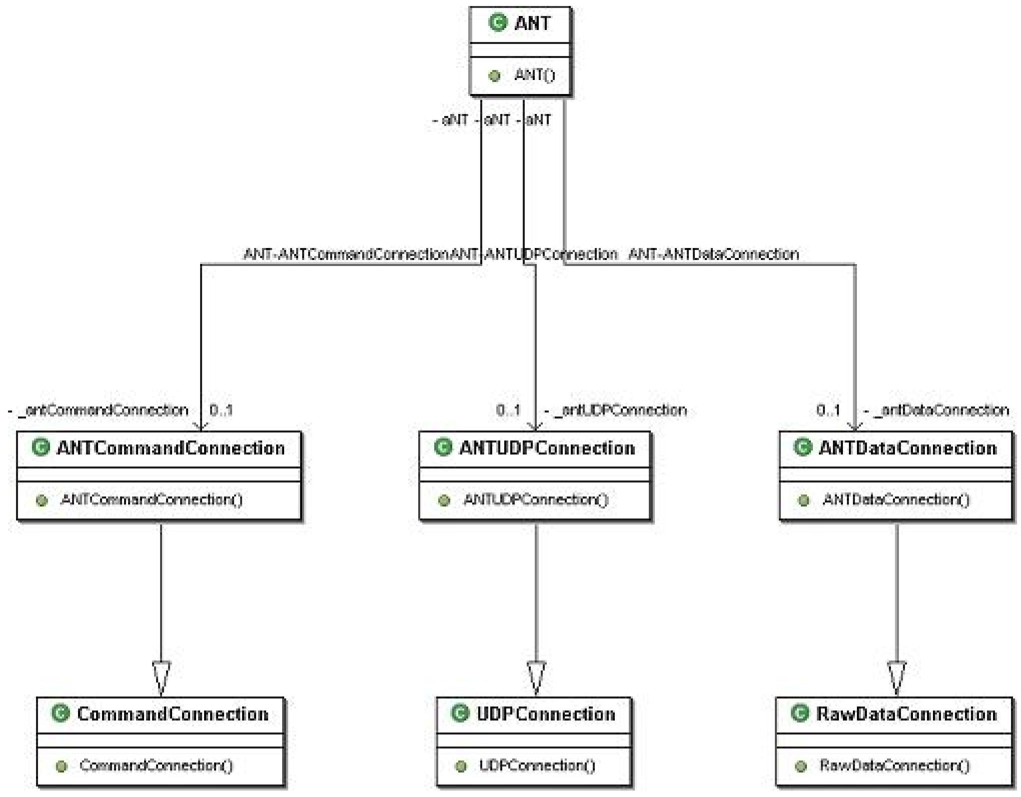
The 5 radar modules are wrapped in the following classes:

1. *NAV* – Navigation Module
2. *ANT* – Antenna Modules
3. *RCAS* – The RCAS Module. Note: The RCAS Modules contains an MDR submodule, since it's functionality is large and in some places treated as a separate module.
4. *PDU* – Power Distribution Module
5. *XTrans* – WaveGuide Assembly and X-Band TWTA

Due to the similarities between the various radar modules, many of the classes are reused and common to each. These common elements include the *CommandConnection, UDPConnection,* and *RawDataConnection.*

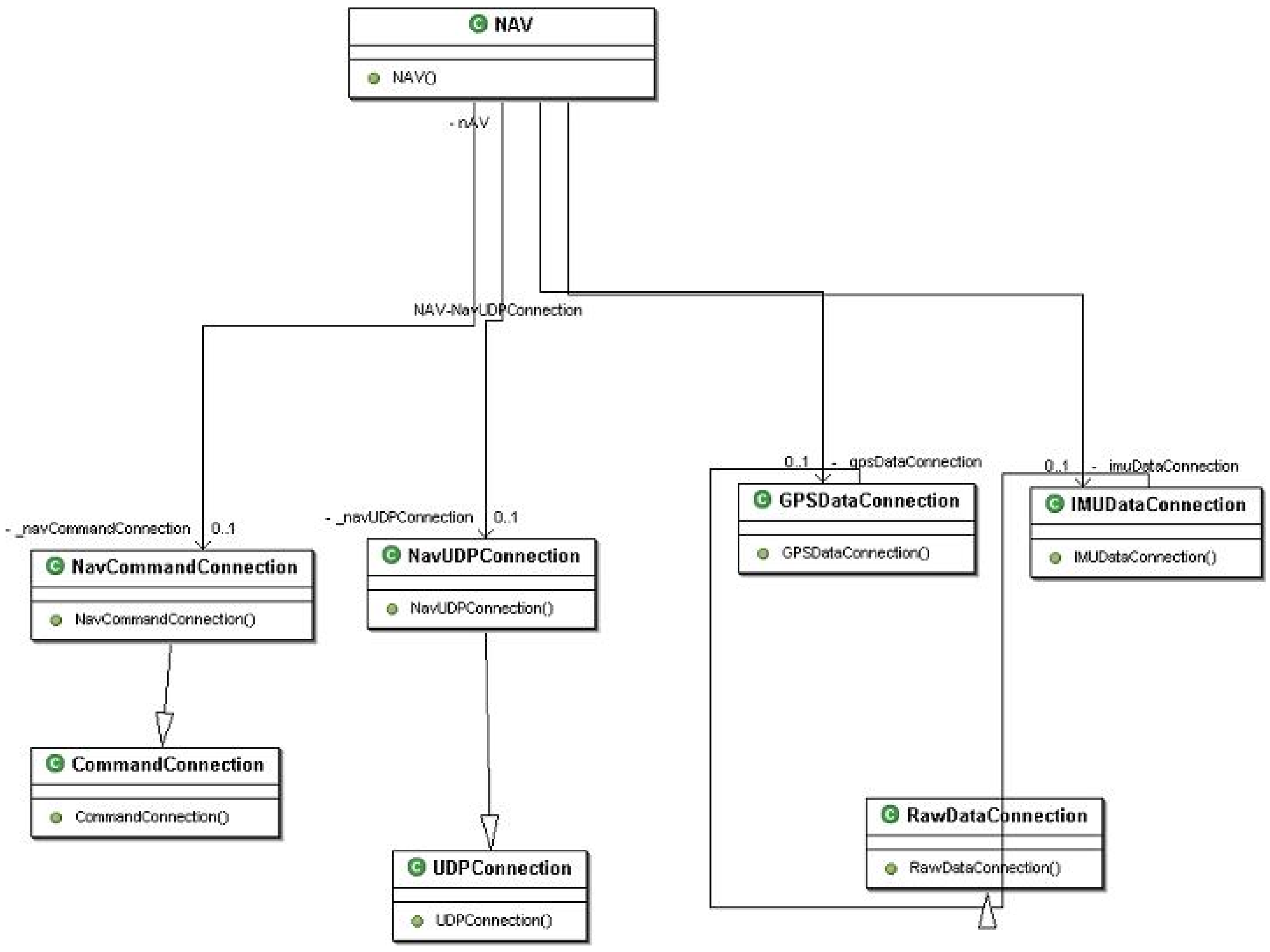
3.4.1 ANT

The ANT Module contains the Antenna UDP Status connection, command connection and ANT Raw data connection. This object implements all the error monitoring for the antenna module, while recording the antenna data. This class makes the antenna high level interface available to the state machine.



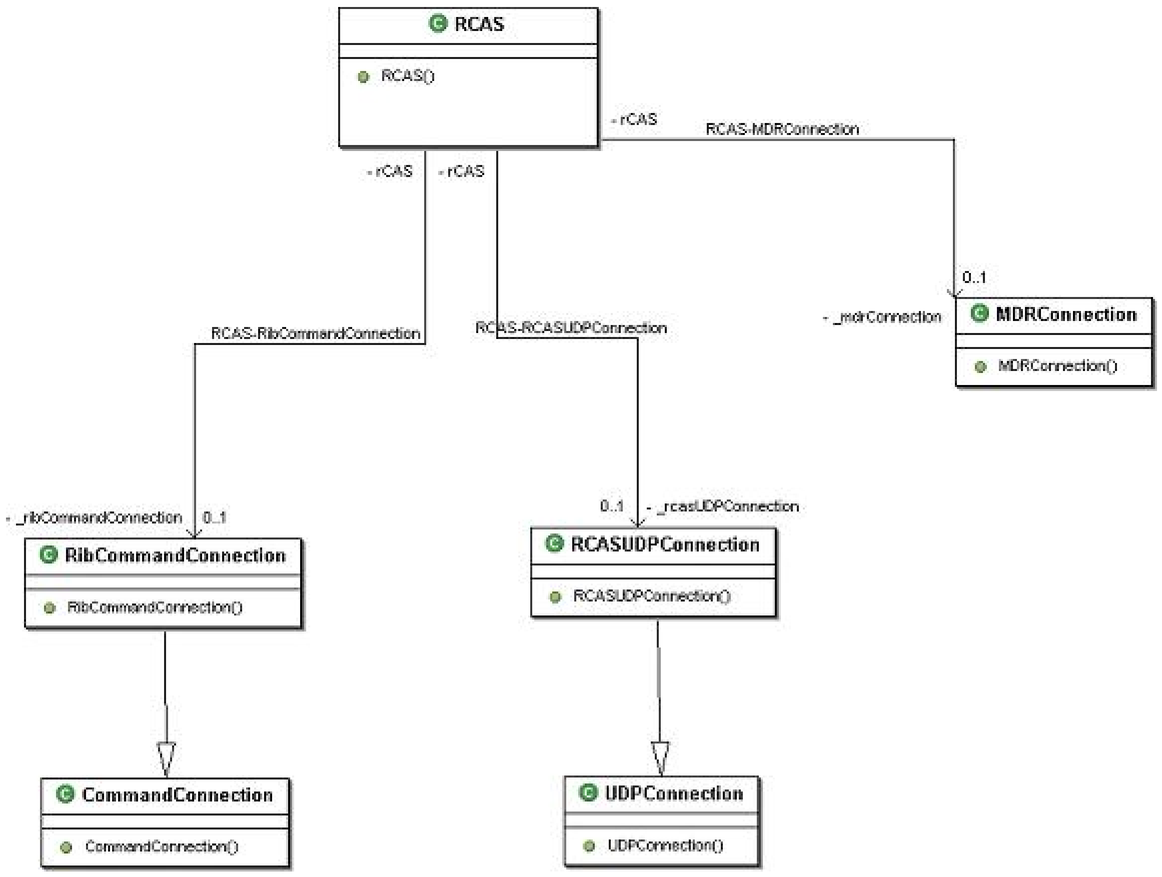
3.4.2 NAV

The NAV class contains the NAV UDP Status connection, command connection and Raw data connections for the GPS and IMU. This object implements all the error monitoring for the NAV, while recording the raw data. This class makes the NAV high level interface available to the state machine.



3.4.3 RCVEX-RCAS

The RCAS Module contains the RCAS UDP Status connection, command connection and the MDR API Implementation. This object implements all the error monitoring for the module, while recording the RCAS Status data. This class makes the RCAS high level interface available to the state machine.

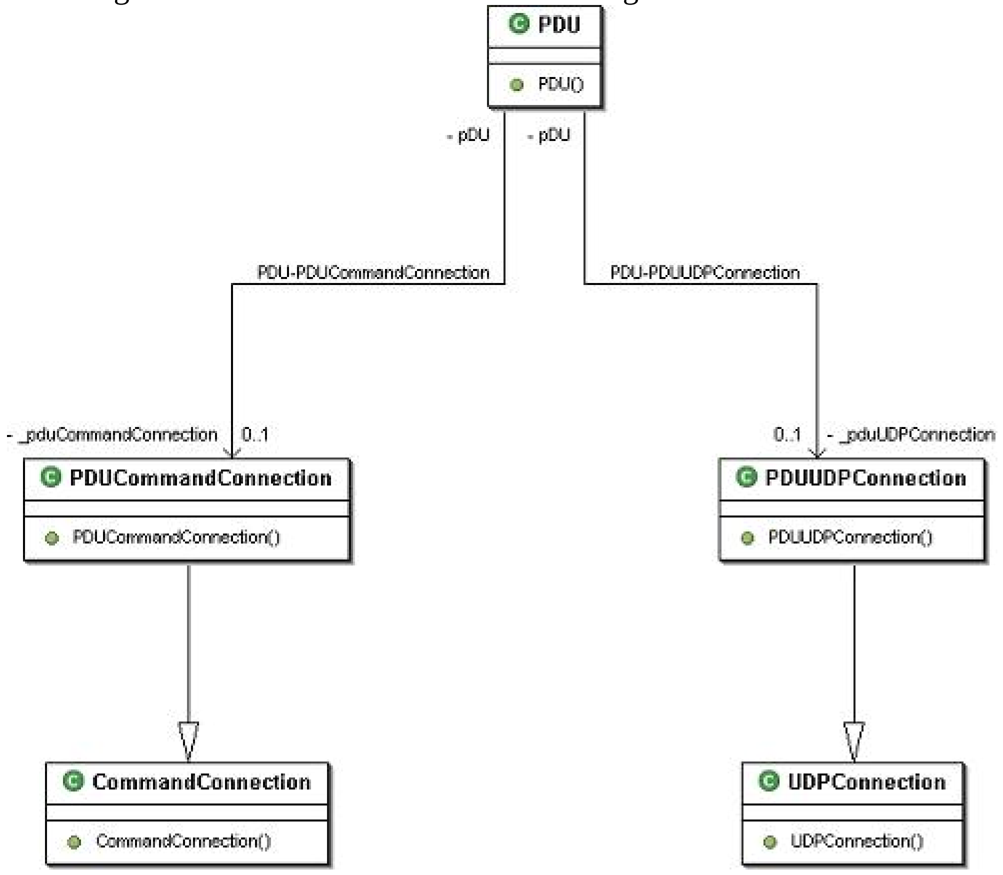


3.4.4 MDR

The MDR is a subset of the RCAS, and contains a wrapper around the MDR hardware.

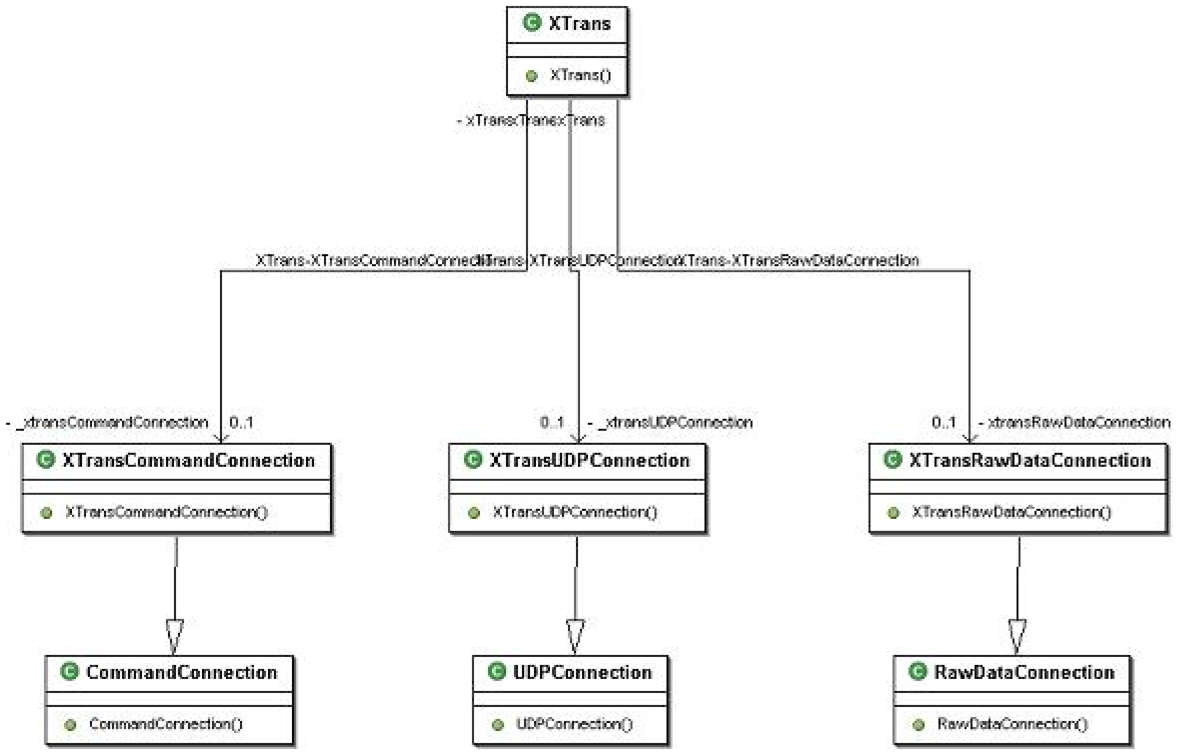
3.4.5 PWRDIST

The PDU Object contains the Power distribution UDP Status connection and command connection. This object implements all the error monitoring for the PDU module, while recording the antenna data. It makes the PDU high level interface available to the state machine.



3.4.6 XTRANS-WGASS

The XTrans object contains the XTrans UDP Status connection, command connection and XTtrans Raw data connection. This object implements all the error monitoring for the XTRANS-WGASS module, while recording the raw data. This object makes the high level interface available to the state machine.



3.4.7 Configuring Radar

The MCC must ensure that the signal source setting of each module is correct, as it is indicated in the raw phase header. Thus the RCAS signal source must be set only after the system is in the correct signal source state.

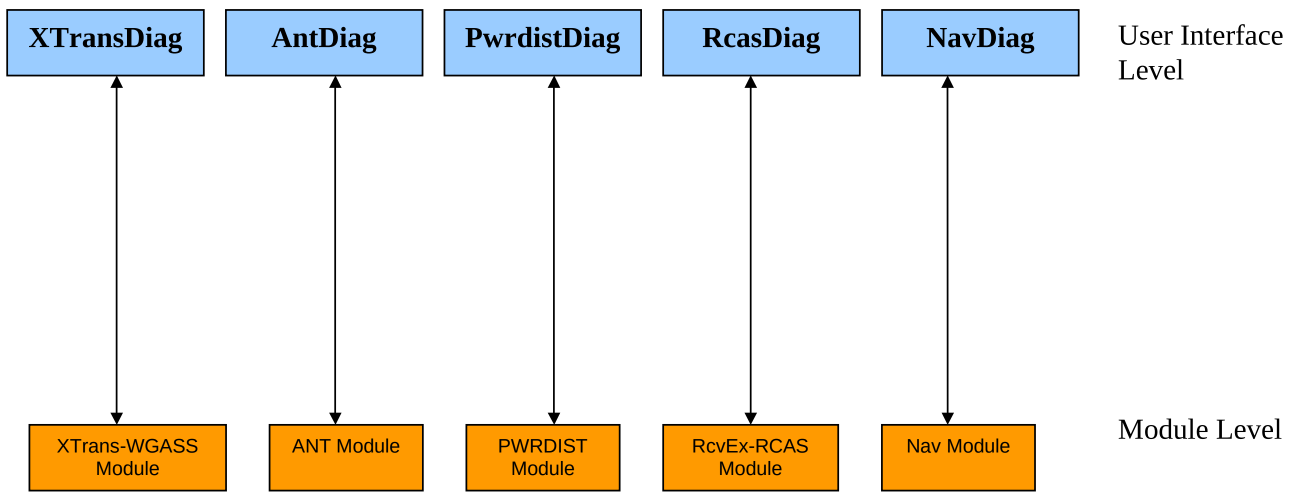
For example, when Configuring the radar parameters (Timing, or mode), the MCC must do the following sequence:

1) The MCC instructs the RCAS to set its signal source to Transition by sending a setSignalSource() command.
2) The MCC waits for the CommandResponse, and confirms that there was 'no error'
3) The MCC waits for an RCAS status packet to arrive with the Transition Signal Source.
4) The Rx Status packet is written to file
5) The MCC issues any configuration change commands to the modules (RCAS, WGASS, etc.)
6) The MCC instructs the RCAS to set its signal source to the new Source, by sending a setSignalSource() command
7) The MCC waits for the CommandResponse, and confirms that there was 'no error'
8) The MCC waits for an RCAS status packet to arrive with the correct Signal Source

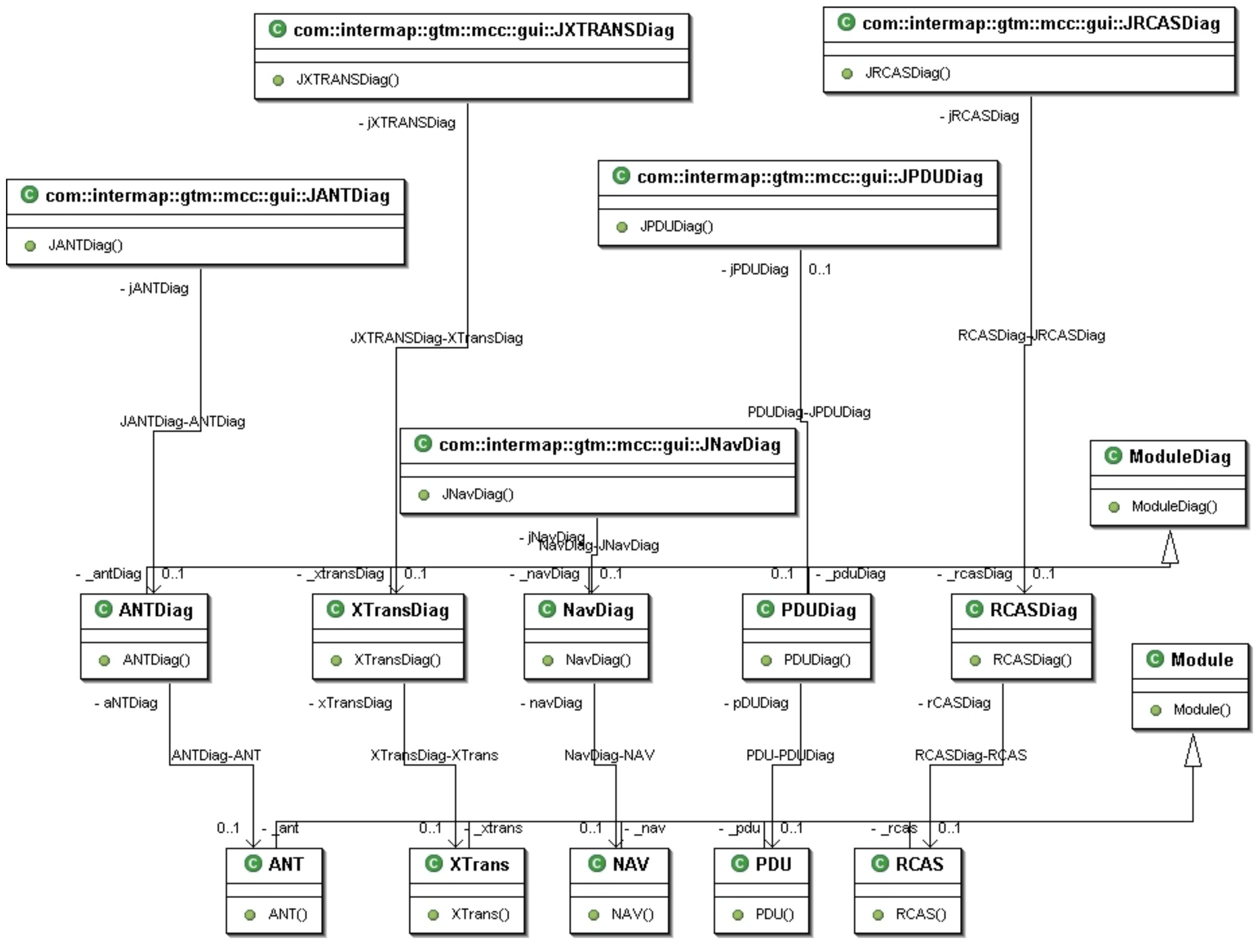
9) The Rx Status Packet is written to file

The MCC writes all received Status packets to file. Thus, there will be at least one status packet with the required transition information.

3.5 Diagnostic Tools

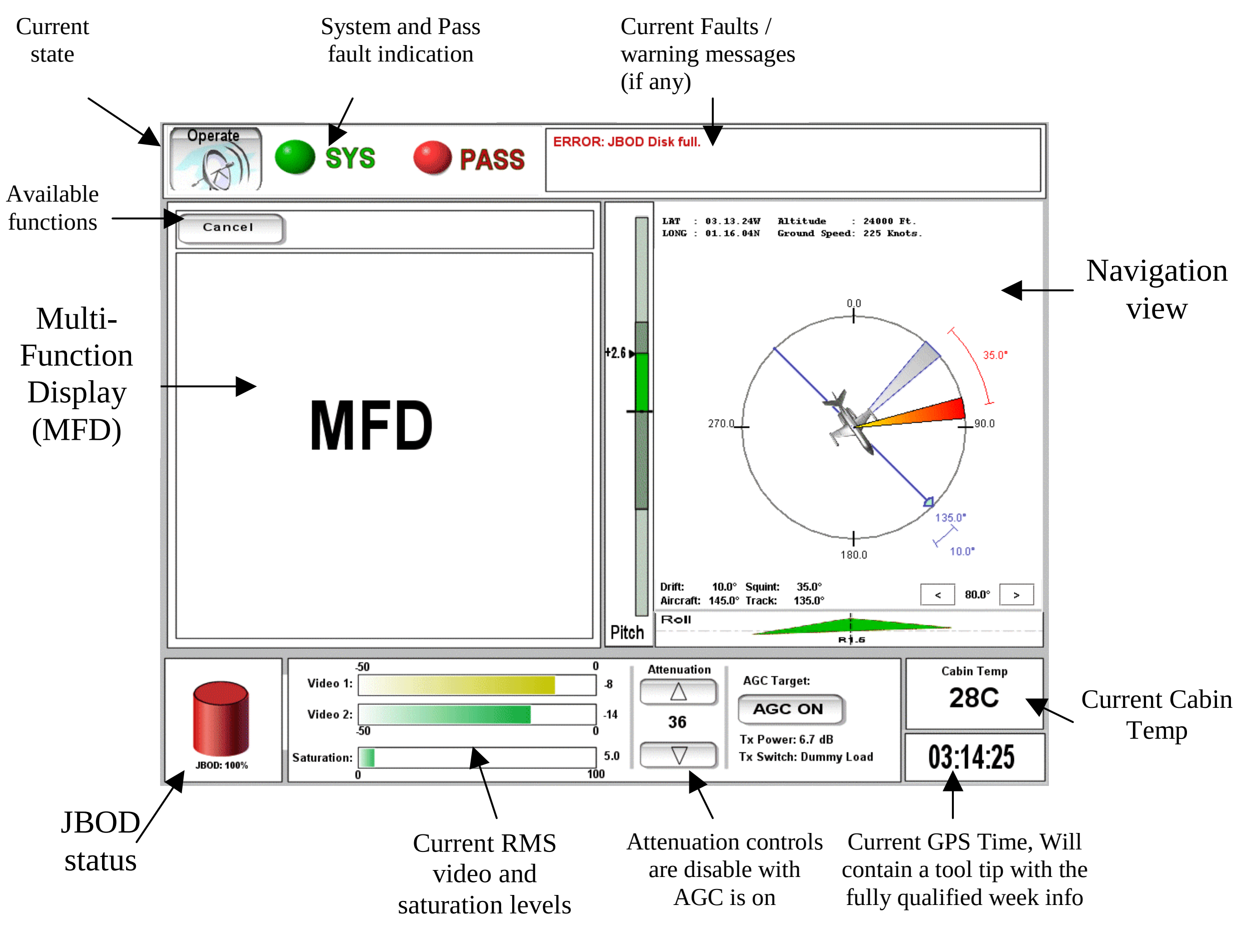


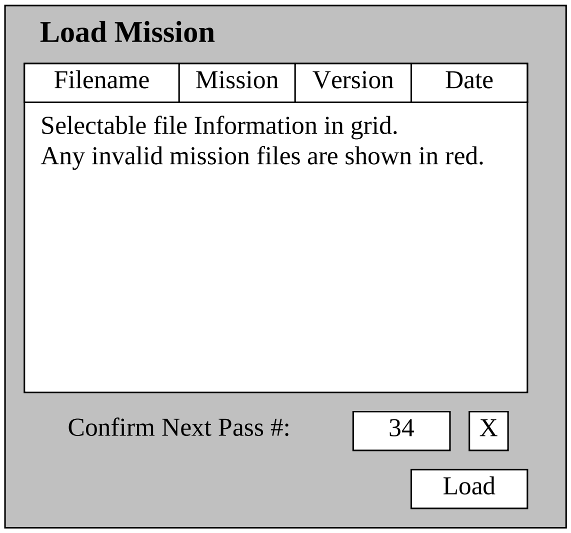
The Diagnostic tools are built using the same GUI framework as the MCC, but only interacting with the Module Level. Each module has a unique diagnostic tool associated with it. Like the MCC the diagnostic tools poll the modules for status, and issues commands to it. Since all of the module functionality is only in the module layer, the Diagnostic tools are capable of displaying status and logs, recording data, and issuing all commands (and more) that the MCC Control layer would use. Thus with all of the Diagnostic tools, the system could be run independently.



3.6 GUI

The MCC GUI design is based on the following illustration:


Current state
System and Pass fault indication
Current Faults / warning messages (if any)
Operate
SYS
PASS
ERROR: JBOD Disk full.
Available functions
Cancel
Multi-Function Display (MFD)
MFD
+2.6
Pitch
LAT : 03.13.24W Altitude : 24000 Ft.
LONG : 01.16.04N Ground Speed: 225 Knots.
Navigation view
0.0
90.0
180.0
270.0
35.0°
135.0°
10.0°
Drift: 10.0° Squint: 35.0°
Aircraft: 145.0° Track: 135.0°
<
80.0°
>
Roll
R1.6
JBOD: 100%
Video 1:
Video 2:
Saturation:
-50
0
-8
-14
5.0
100
Attenuation
36
AGC Target:
AGC ON
Tx Power: 6.7 dB
Tx Switch: Dummy Load
Cabin Temp
28C
03:14:25
Current Cabin Temp
JBOD status
Current RMS video and saturation levels
Attenuation controls are disable with AGC is on
Current GPS Time, Will contain a tool tip with the fully qualified week info


Load Mission

Filename	Mission	Version	Date

Selectable file Information in grid.
Any invalid mission files are shown in red.

Confirm Next Pass #: 34 X

Load

Figure 1