

NASA/CR-2017-219594



ATM Technology Demonstration-1 Phase II Boeing Configurable Graphical Display (CGD) Software Design Description

*George F. Wilber
Boeing, Seattle, Washington*

March 2017

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- **TECHNICAL PUBLICATION.** Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counter-part of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.
- **TECHNICAL MEMORANDUM.** Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.
- **CONTRACTOR REPORT.** Scientific and technical findings by NASA-sponsored contractors and grantees.

- **CONFERENCE PUBLICATION.** Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.
- **SPECIAL PUBLICATION.** Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.
- **TECHNICAL TRANSLATION.** English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at <http://www.sti.nasa.gov>
- E-mail your question to help@sti.nasa.gov
- Phone the NASA STI Information Desk at 757-864-9658
- Write to:
NASA STI Information Desk
Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199

NASA/CR-2017-219594



ATM Technology Demonstration-1 Phase II Boeing Configurable Graphical Display (CGD) Software Design Description

*George F. Wilber
Boeing, Seattle, Washington*

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Contract NNL13AA03B, NNL15AB46T

March 2017

Available from:

NASA STI Program / Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199
Fax: 757-864-6500



**ATM Technology Demonstration-1 (ATD) Phase II
Boeing Configurable Graphical Display (CGD)
Software Design Description**

Prepared for: Denise Scarce
NASA Langley Research Center
Hampton VA, 23681-2199

DOCUMENT NUMBER:

D780-10409-1

RELEASE/REVISION:

Rev A

RELEASE/REVISION DATE:

December 6, 2016

CONTENT OWNER:

Boeing (Airspace Operational Efficiency)

All revisions to this document must be approved by the content owner before release.



Document Information

Original Release Date April 1, 2016	Contract or CDRL Number (if required) NNL13AA03B-NNL15AB46T
Limitations	

Authorization for Release

AUTHOR:	Signature on File	9M-PW-ERCS	4/1/16
	George F. (Rick) Wilber	Org. Number	Date
APPROVAL:	Signature on File	9M-PW-ERCS	4/1/16
	Karl Rein-Weston	Org. Number	Date
DOCUMENT RELEASE:	Signature on File	9M-PW-ERCS	4/1/16
	Barbara L. Withers	Org. Number	Date

Table of Contents

List of Figures	v
List of Tables	v
1. Scope.....	6
1.1 Purpose.....	6
1.2 System Overview	6
2. References.....	8
3. Design Overview	9
3.1 CGD Display Overview	9
3.2 CGD Organization.....	10
3.3 CGD Software Implementation.....	11
4. Architecture.....	12
4.1 CGD System Architecture.....	12
4.2 CGD Software Architecture	13
4.2.1 Non-Normal Operations.....	15
4.3 Interface Design	16
5. Components	18
5.1 CGDServer.js	18
5.1.1 Interface Function.....	18
5.1.2 Message Processing Function	19
5.2 Cgd.html.....	20
5.2.1 HMI Design & Layout	21
5.2.2 HTML Document.....	29
5.2.3 Javascript Computation	30
5.3 FIMio.js.....	34
Appendix A: FIM IO to CGD Server Interface Definition (ICD).....	35
A.1 Fast-Slow Indicator Fields	35
A.2 Progress Indicator Fields.....	36
A.3 Commanded Speed Fields.....	37
A.4 FIM Mode Fields	37
A.5 FIM Status Message Field.....	38
A.6 Fully Populated Example.....	38
Appendix B: CGD Server to CGD Display Interface Definition (ICD).....	39
B.1 Commanded Speed Fields.....	40
B.2 Keep Alive Field	40
B.3 Connection Lost Field.....	40
B.4 Bad Data Field.....	41
B.5 Full Data Field.....	41

Appendix C: Boeing CGD Software Test Plan (STP).....	42
C.1 Approach	42
C1.1 Boeing-Honeywell Testing	44
C.2 CGD System Level Verification	44
C.3 Planned Verification Tests	44
C.3.1 Display Computation & Drawing Tests (DT).....	45
C.3.2 Connectivity and Link Management Tests (CT).....	46
C.3.3 Performance Tests (PT).....	47
C.3.4 Non-normal Operations Tests (AT)	48
C.4 Requirements to Verification Mapping.....	49
Revision Record.....	59

List of Figures

Figure 1 - Boeing CGD System Scope	7
Figure 2 - CGD System Design	9
Figure 3 - CGD Display Design.....	10
Figure 4 - CGD System Architecture.....	13
Figure 5 - CGD Software Architecture.....	14
Figure 6 - CGD Display Areas.....	22
Figure 7 - Non-Normal Operations Display	23
Figure 8 - Normal Non-FIM Operations.....	23
Figure 9 - FIM Operations Beyond 30 NMs.....	23
Figure 10 - FIM Operations Inside 30 NMs	24
Figure 11 - Fast Slow Indicator	25
Figure 12 - Fast-Slow Display Modes	25
Figure 13 - Progress Indicator.....	26
Figure 14 - Progress Indicator Display Modes	27
Figure 15 - Commanded Speed Display Modes	28
Figure 16 - FIM Mode Display Modes.....	28
Figure 17 - Status Message Display Modes.....	29

List of Tables

Table 1 Referenced Documents	8
Table 2 – CGD Requirements Testing Matrix.....	50

1. Scope

This Software Description Document (SDD) captures the design for developing the Flight Interval Management (FIM) system Configurable Graphics Display (CGD) software. Specifically this SDD describes aspects of the **Boeing** CGD software and the surrounding context and interfaces. It does not describe the Honeywell components of the CGD system.

The SDD provides the system overview, architectural design, and detailed design with all the necessary information to implement the Boeing components of the CGD software and integrate them into the CGD subsystem within the larger FIM system.

Overall system and CGD system-level requirements are derived from the CGD SRS (in turn derived from the Boeing System Requirements Design Document (SRDD)). Display and look-and-feel requirements are derived from Human Machine Interface (HMI) design documents and working group recommendations. This Boeing CGD SDD is required to support the upcoming Critical Design Review (CDR).

1.1 Purpose

The purpose of this Software Design Description (SDD) is to capture the architecture of the CGD System and the design of the Boeing CGD software subsystem and components.

This single document captures all aspects of the Boeing FIM CGD software design. It describes the display features, allocated requirements, architecture of CGD system and Boeing CGD subsystem. It contains the interfaces and protocol descriptions between all Boeing software components and external FIM system components.

The CGD display is to be used by the FIM equipped aircraft's crews to follow speed commands and status information that will cause the aircraft to move to the intended clearance location within the Interval Management (IM) arrival stream under Instrument Meteorological Conditions (IMC). All display information is provided via the EFB and no crew input is required on the CGD during IM operations.

1.2 System Overview

Figure 1 captures in graphical format the scope of the Boeing CGD subsystem and thus Boeing CGD software within the FIM display system (which includes both the CGD subsystem and the Cockpit Display of Traffic Information (CDTI) subsystem on the EFB) and the full FIM system. It illustrates that software will be hosted across multiple physical devices (the Master EFB and the CGD Display hardware. This is further addressed in follow-on architecture figures including the CGD functional architecture (i.e., Figure 2).

The scope of this SDD lies within the hashed circle in Figure 1, and includes the CGD Server that is hosted by the EFB and purpose built software executing CGD

displays on the CGD display devices. The CGD SDD scope is narrower than the MOPS CDTI requirements (which includes both EFB CDTI displays and primary field of view (PFOV) displays). In the MOPS, interface G requirements apply to both the CGD SDD and the EFB CDTI avionics SDD (i.e., Honeywell HMI/IO SDD).

Figure 1 depicts the scope of the FIM system, which includes the system requirements of both hardware and software which have been allocated to the Boeing CGD subsystem. The scope of this SDD only includes software components. It only includes the Boeing components of the CGD system.

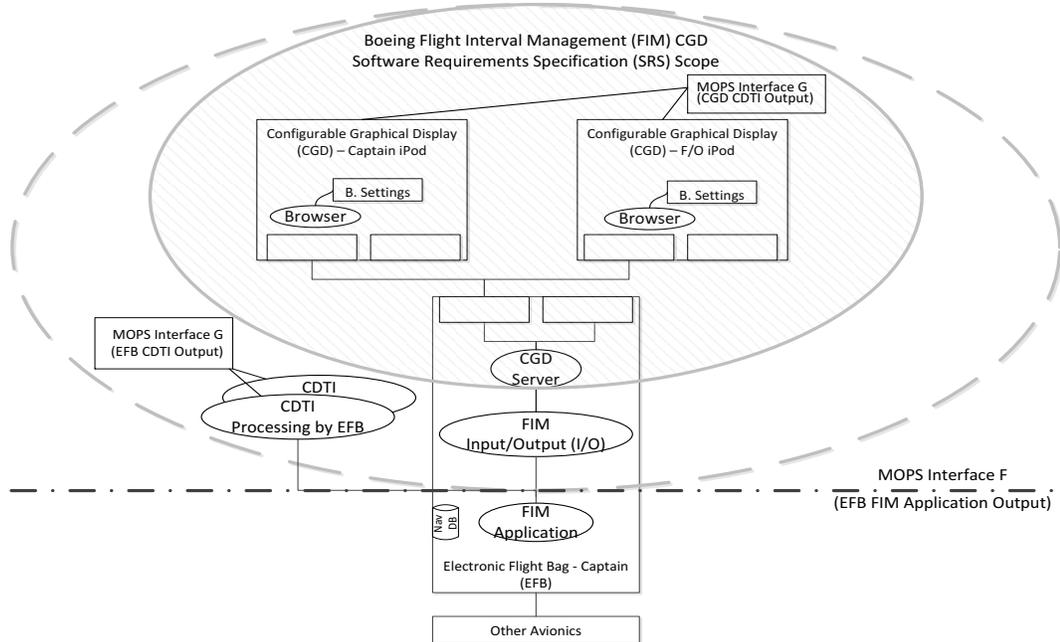


Figure 1 - Boeing CGD System Scope

2. References

Table 1 Referenced Documents

Document Title	Revision	Date	Organization
Boeing Air Traffic Management Technology Demonstration – 1 (ATD-1) Phase II, System Requirements Definition Document,	Rev A	10/30/2015	Boeing AOE
ATM Technology Demonstration-1 (ATD) Phase II Configurable Graphical Display (CGD) Software Requirements Specification	NEW	3/7/2016	Boeing AOE
The Honeywell and Boeing CGD HMI Interim Design Power Point Slides (i.e., wireframes)	V9	1/22/2016	Boeing AOE

3. Design Overview

The following figure shows a notional FIM system level functional design. It depicts two EFB's (captain and first officer) and two CGD displays (captain and first officer). Further it depicts wireless or wired connectivity between the Master EFB and CGD displays. Wireless is the baseline system design, but the software is designed to be agnostic to the physical network running over any TCP/IP network.

For flight Test PC will be executing a CGD display connected by Ethernet in cabin, while flight crew uses iPhone SE devices wirelessly. Flight test PC CGD display will support troubleshooting any non-normal behavior.

Figure 2 shows that the CGD Server is hosted on the Master EFB connected to the FIM IO. It is also connected to two CGD Display devices and the Test PC. Each CGD display executes the same application on the same data in a parallel, synchronous manner. The CGD Display software is hosted in the CGD Display hardware browser.

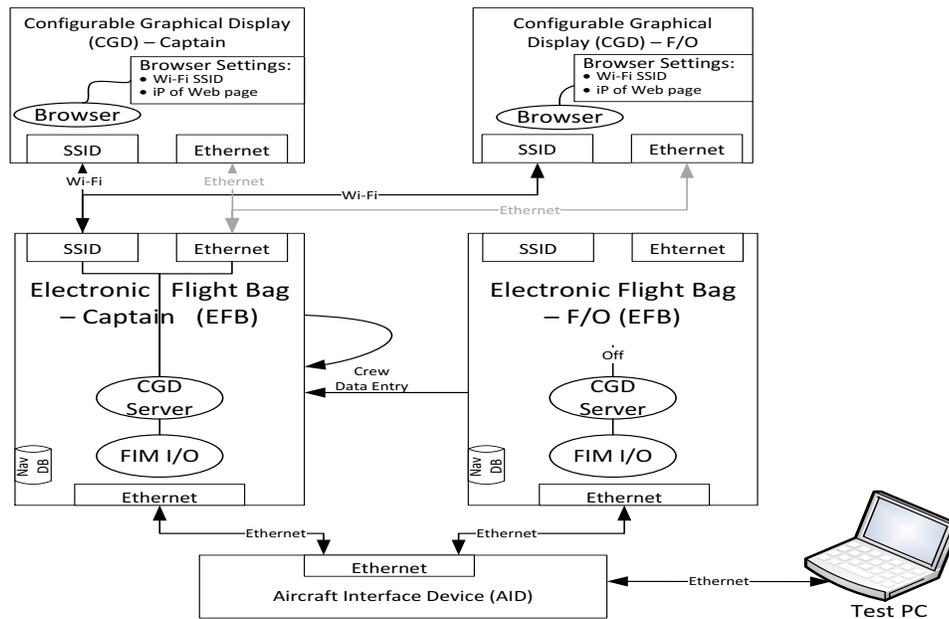


Figure 2 - CGD System Design

3.1 CGD Display Overview

The purpose of the CGD system is to display critical FIM information to the flight crew in their PFOV. The CGD displays must therefore perform display functions in a manner to ensure only valid information is provided and misleading and hazardous information is not displayed in the PFOV. Information shall be displayed on the CGD according to the layout illustrated in Figure 3. The CGD information required is shown in the example layout in Figure 3:

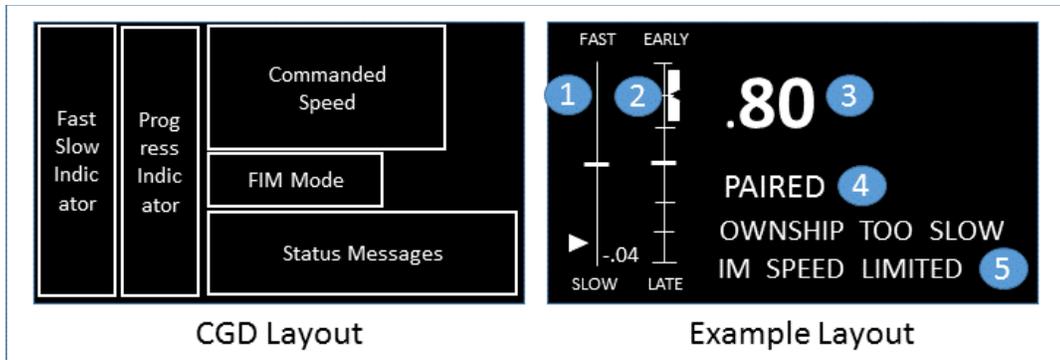


Figure 3 - CGD Display Design

1. FAST/SLOW indicator
2. Progress Indicator
3. FIM commanded speed
4. FIM mode
5. FIM alert message (s)

3.2 CGD Organization

The operational goal for the CGD is to display aircraft speed guidance and deviation indications in the flight crew's PFOV. The CGD provides the crew with the necessary information to safely conduct airborne IM spacing:

- Display IM commanded speeds within the crew's PFOV
- Display IM state changes and speed changes in the PFOV
- Monitor EFB outputs and alert the crew to keep them fully aware of changes
- Display information consistent with the EFB IM pages
- Auto-populate information when received from the EFB with no perceivable EFB-CGD latency
- Display data consistent with the EFB data and formats.

To achieve this a very simple design with simple data flows and minimum logic processing on the CGD displays has been developed. The FIM application on the Master EFB generates the detailed EFB content and displays. This application makes all of the display decisions and passes the already processed information to the CGD system. The FIM application passes only the subset of data needed for display processing to the FIM IO function. The FIM IO function then sends only the specific CGD display commands and associated information to the Boeing CGD subsystem. In parallel the FIM IO passes consistent display information to both EFB displays. FIM IO does not filter CGD specific data but sends all of the data each cycle.

All possible CGD logic processing is done on the EFB computer by CGD server, leaving only display commands and drawing constructs required on the CGD Display device. In fact, no resident software is hosted or executed on the CGD display

hardware. The only operational CGD software executed directly on the CGD display is the commercial HTTP Browser. The HTTP browser uses the latest HTML-5 display capabilities to render the CGD display. This minimizes the latency, ensures same information is displayed on both CGD display devices at the same time and reduces display drawing complexity, latency and jitter.

The EFB is a hardened Microsoft Windows device allowing development and deployment on common operating system architecture. The CGD device is an iPhone SE IOS based device. Further the design leverages existing and emerging network and communication designs based on Ethernet, IP, and Wi-Fi built into the host devices using standard software interfaces, protocols and device drivers.

3.3 CGD Software Implementation

CGD Software is written in Javascript and HTML-5. CGD display software consists of a single HTML document using latest HTML-5 constructs for document management, screen layout, drawing and animation. Computing software and logic on both CGD display and EFB are written in Javascript. Javascript is interpreted natively and is not compiled.

The Javascript embedded in the HTML document on CGD display is executed by the browser in browser context as part of the document (DOM). The browser is fully HTML-5 compliant.

The Javascript executed on the EFB is executed by a small but powerful, advanced runtime executable called “node.js”. Node.js is built on Chrome’s V8 Javascript engine. Node.js uses an event-driven, non-blocking IO model that makes it lightweight and efficient. Node.js’ package ecosystem, is the largest ecosystem of open source libraries in the world, according to their writing. The executable is under 10 MB and self-contained with all required features. This is important due to EFB hardware and resource constraints.

Node.js supports three basic networking capabilities used for component interfaces. (1) TCP/IP for socket-to-socket communications, (2) HTTP communications, and (3) Web Socket interfaces and services. No code is directly written to develop these interfaces and services. All that is required is library inclusion and configurations. This allows common communication software between CGD components whether on same device or on separate IP devices connected by either hardwired IP/Ethernet or IP/Wi-Fi.

4. Architecture

The CGD architecture fits into the overall FIM architecture as depicted in Figure 4. That is the CGD system communicates with the FIM applications on EFB. This communication is via FIM IO function in Honeywell FIM application. The FIM applications provides display commands and messages to the overall display system (CGD and CTDI) as shown in Figure 4. The FIM application ensures consistent and timely display management for both EFB displays and CGD displays. The interface between the FIM application and FIM IO is external to CGD design and is defined by Honeywell. It is defined in Honeywell SDD and is out of scope for this Boeing CGD SDD document.

While there are two EFB's, each hosting a copy of EFB FIM and CGD EFB software; and one Wi-Fi source (dedicated Wi-Fi switch/router), the overall design is that only one EFB will provide FIM guidance at time and only one Wi-Fi will transmit at any time. This allows simplified flight crew management of the CGD display device settings. The Master EFB will provide FIM guidance for both EFB displays and for all CGD displays. That EFB is declared the Master EFB. The Master EFB will command the CGD functions on that device to execute. The CGD functions on the Slave EFB will be dormant. Upon certain failure modes a switch over to backup EFB may be required. If so, a full reset is assumed, which will switch the slave EFB to the Master state.

4.1 CGD System Architecture

CGD system architecture consists of components developed by Boeing and by Honeywell. All CGD components, except for the FIM IO are developed by Boeing and documented in this SDD. The FIM IO is documented in the Honeywell FIM application SDD. However, the interface between the FIM IO and the Boeing CGD software components is defined in this document, in Appendix A.

CGD System Architecture consists of a central dual server based solution. The dual server concept is where a single CGD component, the "CGD Server" executes on the Master EFB as a server to both the EFB FIM IO function to accept data from FIM application and as CGD display server for the CGD display devices. The FIM IO becomes a TCP/IP client on the same device. The CGD displays become HTTP and Web Socket clients on different devices.

Application data flows primarily in one direction. Given that TCP/IP is being used for both CGD Server interfaces, there is low-level bi-directional communication. Application layer information flows only in one direction between FIM IO and CGD Server and from CGD Server to CGD Displays. This is true for the CGD Server to CGD Display with one exception. The exception is that a non-acknowledgement ("NAK") message is returned from CGD Display to CGD Server over Web Socket in the even an error in the incoming data is detected. When an incoming error is detected all ICD defined data (vs. deltas only) will be sent on the next display update cycle to

ensure consistent and accurate data is displayed. Also full (non-delta) JSON data will be commanded (via NAK) right after initial cgd.html initialization and after any “CONNECTION LOST” condition.

The CGD design is based on using non-real time computing and networking resources. However, there are some tight time based requirements in SRS. To that end the EFB FIM application (via FIM IO) will send data to the CGD system at precisely a two hertz rate. This data rate will be the main timing element used by normal operations for drawing and animating the CGD displays. Also, the display hardware selected is capable of game player animation performance.

See Figure 4 for CGD system architecture in context of the overall FIM system and components. It also shows the Boeing CGD subsystem architecture and system level connectivity.

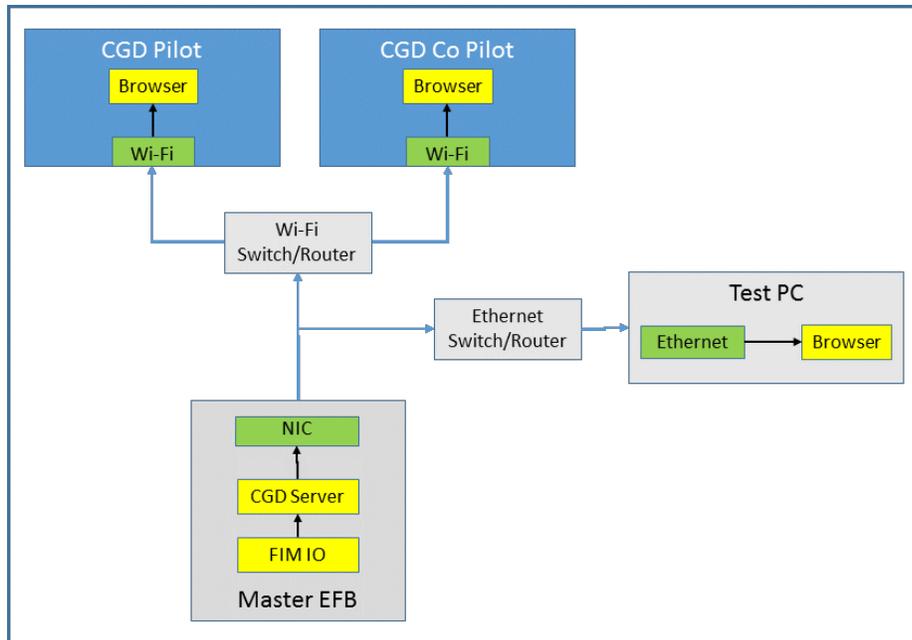


Figure 4 - CGD System Architecture

4.2 CGD Software Architecture

The CGD software architecture is depicted in Figure 5 below. Yellow items are Boeing CGD components and cream are Honeywell components. CGD Software Architecture consists of:

1. FIM IO – on EFB (Honeywell), component called “FIMio”
2. CGD Server – on EFB (Boeing), component named “CGDServer.js”

3. CDG Displays – on CGD display hardware (Boeing), component named “cgd.html”.

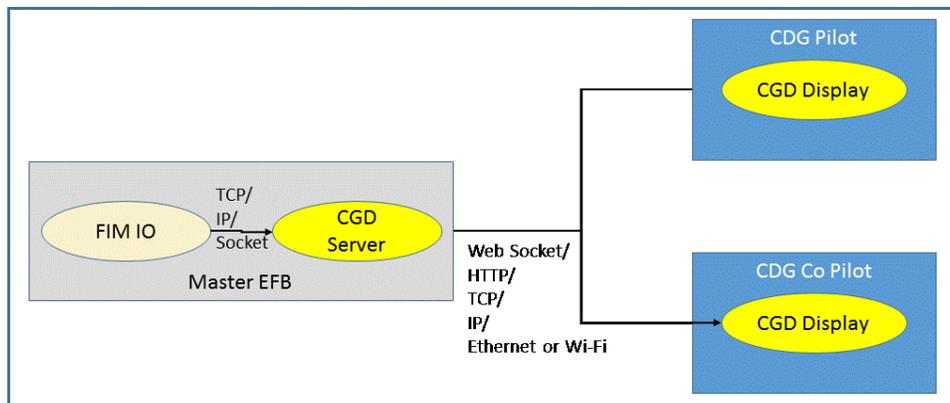


Figure 5 - CGD Software Architecture

The architecture and design is such that FIM IO (FIMio) data is passed as text strings in JSON format to the CGDServer over TCP/IP using socket interface. The CGDServer is the TCP server and the FIMio is the TCP client. Therefore, the CGDServer must be started first and listen for the FIMio to connect. The CGDServer is persistent and continues to run even if the FIMio connects, disconnects and reconnects to the TCP socket.

The CGDServer sends text strings in JSON format to all connected CGD displays over a HTTP and Web Socket protocols where the CGDServer is the server and the CGD display browsers are Web Socket and HTTP clients. The browser links to the Master EFB and requests the cgd.html page. The Web Socket is a protocol and service that allows connection oriented communications between HTTP server and HTTP clients. This allows an HTTP server to continually send updates REST-fully to the HTTP clients. This is in contrast to typical web-based HTTP services where client security is paramount and server-to-client pushed communications are tightly controlled. The CGDServer provides the Web Socket over HTTP services, which are built into the HTML-5 and node.js systems. Further, the HTTP service is layered on TCP/IP over either wired Ethernet or Wi-Fi.

Again the CGDServer is persistent and listens for CGD display connections. Multiple CGD displays may connect, disconnect and reconnect at will. CGDServer manages all these connections and protocol layers.

The execution of the CGD system is defined in a sequential manner with minimum latency to allow all data being displayed on the CTDI and CGD PFOV displays as consistently and timely as possible. For the main data path (normal operation) the data flow and control flow are the same. The incoming FIMio data drives CGDServer and CGD display execution (event driven). There is only a single thread for all normal data and control flow (beyond background threads managing communications

services and protocols). While the devices are not real-time, the processing system is designed to minimize latency and appear as near real time as possible. This is doable using fast computing and networking designed for animation and gaming.

All possible logic processing happens either in or before the FIMio or in the CGDServer. This minimizes downstream latency and jitter and makes graphics and drawing quick and simple on the CGD display devices. Every effort is made to minimize the amount of drawing and redrawing by only requiring drawing of changed display items.

Note that while goal is to keep CGD Server and cgd.html from computing complex logic, various charts are added to determine and log errors encountered from FIM IO or by bad data in transit. This includes checking numeric values, value ranges and defaults.

4.2.1 Non-Normal Operations

There are system requirements that state that the information displayed to the flight crew must be timely and not provide hazardous information. To accomplish this with one-way communications, watchdog timers are implemented in minor threads in both the CGDServer and CGD display software design.

Honeywell's FIMio ensures the data exiting its interface is timely and display information correct. Then a certain time is allocated to Boeing CGD subsystem to process and render the display commands from FIMio socket interface to CGD display for crew viewing. The CGD system has no way to determine if the data incoming is correct, so some portion of the system level CGD requirements have been allocated to aspects of the FIM application and FIMio processing. The CGD system however, handles timeliness of the data within its domain and ensures the flight crew is alerted if normal operations are not functioning properly. That is, when data is not being received at the prescribed rate, when current data becomes stale, or connectivity is lost.

Non-normal operations that result in loss of connectivity provide a single message to the flight crew (on all active CGD displays) stating "CONNECTION LOST" and an error reason. The following may cause this condition.

1. CGDServer lost connection to FIMio "LOST FIM CONNECTION".
2. CGDServer has not received good data from a connected FIMio for too long "FIM IO TIMEOUT".
3. CGD display has no initial connection to FIMio "NO INITIAL FIM CONNECTION".
4. CGD display has lost connection to CGDServer "LOST CGD SERVER CONNECTION".

5. CGD display has not received good data from a connected CGDServer for too long “CGD SERVER TIMEOUT”.

The CGDServer sets a watchdog timer and monitors connection to FIMio over TCP/IP. It sends appropriate message to CGD display when it discovers non-normal (too long between messages or lost connection) operation and data flow. The CGD display software sets a watchdog timer and monitors connection errors on Web Socket, and HTTP connection as well as looking for long delays between messages. When the connection is lost and not re-established in a timely manner, or when data is not received from CGDServer in a timely manner even when a connection is established; the system declares non-normal operation.

Any time non-normal connectivity is detected the current display information in all display areas is removed from the display and the “CONNECTION LOST” message is displayed.

Another non-normal operation is receiving corrupted or lost data. This is a transient problem that is reset on the next one half second cycle and should not be visible or require operator action.

1. Missing or corrupted data in a FIM IO message will cause a JSON parsing error. This will cause corrupted data update to be delayed for this one half second cycle. FIM IO automatically sends all updated data on the next cycle.
2. Missing or corrupted data in a CGD Server message will cause a JSON parsing error. This will cause corrupted data to be delayed for this one half second cycle. The CGD Display will send a NAK message to CGD Server over Web Socket, which will trigger a full update of all data (to all CGD Displays) on the next display cycle.

4.3 Interface Design

There are two interfaces in the CGD system.

1. FIMio to CGDServer
2. CGDServer to CGD display

Each uses a slightly different low-level protocol but uses same data and message format structure and mostly identical message formats. The messages are defined in Javascript object notation or “JSON”. JSON messages are text based and have a simple syntax. For this effort the key elements of the JSON message are the container “{ JSON data }” in curly brackets and the data elements. Each data element has a defined name and is followed by a data value (or values), separated by a colon. Each message field is comma separated. See Appendix A and Appendix B for details.

The Javascript engine provides a built in feature to parse, analyze and process JSON fields. Fields are then requested by name and a value or “Undefined” is returned. The

overall JSON message is contained in a single “{}” enclosed text string with first and last characters being “{“ and “}” respectively.

Note that each field in JSON message is comma separated by a “,”. Spaces are optional.

It is required that JSON data be properly formatted. JSON error formats will cause the entire message to be ignored and discarded in operation.

Appendix A contains the FIM IO to CGD Server interface definition between Honeywell and Boeing software. And Appendix B contains the CGD Server to CGD Display interface.

5. Components

5.1 CGDServer.js

CGDServer.js is the software that executes on the Master EFB and performs the CGD display server functions. It sets up, listens for and accepts TCP connections from FIMio client. It sets up, listens to and accepts HTTP and Web Socket connections from CDG displays. After establishing TCP connectivity to FIMio, it accepts JSON based message data at 2 hertz rate. It processes all fields in the message and sends updated JSON message data to all connected CGD displays over Web Sockets.

The CGDServer is persistent and continues execution even when some or all CGD displays and/or FIMio becomes disconnected. It awaits reconnection. If FIMio connection is lost it is in non-normal operational mode.

CGDServer.js execution is controlled by the EFB FIM application or similar EFB executive. CGDServer does not monitor its own execution. It must be started, stopped and restarted by an outside (non-Boeing) developed component.

CGDServer is a single Javascript program. It may be kicked off by batch file or in a MS DOS command window. It is executed as “node CGDServer”. There are several command line options for CGServer.js. (1) a “-l” for logging, (2) “-s” for show output on console. It may be entered such as “-sl”. The “-s” flag indicates to show command line output of the processed message. The “-l” flag indicates to capture CGD messages in log files for debugging, testing and playback. When logs are turned on, three files are captured. First, the raw incoming FIMio JSON messages. Second, the outgoing CGD Server processed JSON messages. Third, connection and error log. All three logs capture each line with a Unix timestamp (end of line or on next line as separate JSON message).

Major CGDServer computation involves managing the connectivity interfaces and protocols, and processing messages for incoming and outgoing transmission.

One data flow change added during CGD integration and testing is a CGD Server timer. The timer is set to time out prior to five seconds minimum for stale data. The timer goes off if FIM IO does not send timely JSON message to CGD Server. This allows CGD Server to tell CGD display that it has a FIM IO time out situation. Also added are some warning messages if FIM IO timing is delayed.

5.1.1 Interface Function

The CGDServer creates a TCP socket connection and listens for FIMio to connect. It acts as a TCP server. When the FIMio connects it waits to receive incoming messages on the IP socket. When an incoming message is received, it is processed and handled accordingly. If the FIMio connection is lost, non-normal messaging is invoked and the CGDServer waits for the FIMio to reconnect.

The CGDServer also implements a simple HTTP server function and serves HTTP protocol. This HTTP server opens a Web Socket interface and listens for Web Socket connections. Multiple Web Socket connections are allowed. Web Sockets are allowed to come and go as they please. CGDServer is continually listening for new CGD display clients to join. All outgoing data is broadcast to all connected CGD display clients over their assigned Web Socket port.

Connection errors on either end are caught and ignored. The CGDServer simply returns to listening mode and continues operation as if nothing happened (other than possibly entering non-normal mode after a specified wait period).

CGDServer also accepts incoming NAK messages from CGD display clients when any one receives corrupted data and processes the request.

5.1.2 Message Processing Function

The primary function of the CGDServer is to accept incoming JSON messages from FIMio, process and store them, and send any changed parameters to the connected CGD displays. To keep CGD display processing simple, as much logic processing as possible is completed by the FIM application, FIMio or CGDServer. All incoming JSON messages are fully populated with all potential fields. This allows simple logic for FIMio.

The CGDServer accepts and stores all incoming data. It notes each field that is changed and ensures the changes are stored and forwarded to CGD displays. It only sends changed parameters. This reduces computation and drawing on the CGD display devices. Exceptions are when any new CGD display joins the connectivity group or any CGD display has received corrupted JSON data, then a fully populated message is sent from CGDServer to all CGD displays. This ensures instant synchronization and display of same information to both crew members.

All incoming messages are JSON parsed. If the incoming message is corrupted, i.e. JSON parsing returns an error, the entire message is ignored. A warning message may be recorded (depending on command line parameters) and processing will continue with the next incoming JSON message. If any JSON message field is missing, the field is ignored and other fields are sent. No consistency check is made. No JSON typos or errors should exist in operation. If so, they are ignored and all are handled normally on the next incoming message cycle.

Progress Indicator and Fast-Slow Indicator constrained values are checked for valid values and ranges. If the values are not valid, then the old value remains for the next display cycle. This is true for enabling, value and scale fields in Progress Indicator and the enabling and VAC in Fast-Slow Indicator.

Invalid JSON field values are displayed on the console log and in action log file.

5.1.2.1 Command Speed Computations

The CGDServer accepts commanded speed color from FIMio. The commanded speed color may be a normal mode (white) or be commanded to get the flight crew's attention via alert (reverse video color) or blinking by alternating reverse video and normal video at one hertz rate. The CGDServer controls the blinking and rate. It sets the "actual" right now colors (drawing and background) for each cycle. Therefore, the CGDServer modifies the FIMio commanded speed color as needed and creates an "actual" commanded speed color and a background field color. The background color is normally black, but goes to white in reverse video and alternates between white and black during every other blinking cycle. Internally the CGD computes a "commanded speed display mode" of normal, alert or reverse video, along with a blinking component of on-or-off that alternates each cycle. Commanded speed incoming value is checked for validity, if not valid, then new value is not updated.

5.1.2.2 Keep Alive Field

CGDServer adds an "alive" field to each outgoing message. Technically this message is only required when no other data changes, however, in practice this message field is attached to each outgoing JSON message. There is no effect unless it is the only message field contained in the entire JSON message during the cycle.

5.1.2.3 Array Computations

CGDServer performs some computations and processing for the fields that are arrays. Arrays are more complicated than single valued fields as both the number of entries and values in the arrays may change. These must be handled to ensure display continuity and ensure no exceptions are raised when array length or data entries vary.

5.2 Cgd.html

The file "cgd.html" is the software component executed on the CGD display. It draws the CGD display on the CGD display hardware device. The CGD hardware device is defined as a mobile device (iPhone SE) running either IOS operating system. It hosts a native HTML-5 compatible browser designed for at least minimal animation. No native code is developed for the CGD display hardware. Only the HTML-5 browser is required to be resident on the device.

The only software executing CGD specific functions is the "cgd.html" file. This file is downloaded each time the browser is directed to the "cgd.html" file at the IP address of the Master EFB and Web Socket port the CGDServer is listening on. This is a standard HTTP client-server architecture and design with Web Socket application layer data exchange on top.

Once the code is downloaded, the display fields will be updated via Web Socket interface over JSON formatted messages between the CGDServer and cgd.html, now temporally resident on the CGD display hardware inside the HTML-5 browser

document object domain (DOM). The browser automatically redraws the display areas and elements using HTML-5 and internal Javascript constructs and callback mechanisms.

When processing incoming data and a JSON parsing failure is detected, a NAK message is sent to the CGDServer to ensure a full update on the next display cycle.

There are three major design criteria for the CGD display software.

1. HMI design and layout.
2. HTML document design.
3. Javascript computation and drawing design.

5.2.1 HMI Design & Layout

The current CDG display design is based on an iPhone SE for the measurements and numbers below. **However**, the design is NOT **required** to be hosted on this specific device or fixed to the current dimensions. It is easily mapped to other sizes and layouts and hosted on other HTML-5 capable mobile devices.

Note: the numbers and references to specific device below are for design clarity. And later changes to other devices with same design will not obviate this version of this SDD. Tweaking dimensions slightly between the five display areas is considered implementation not design.

One major task then is to ensure the principles and design constraints defined by the HMI Working Group (WG) are implemented in the CGD display design. In the end the HMI WG will have final say on whether or not the CGD HMI design meets the utility and look-and-feel requirements. Specifically they determine that the text and graphics are readable and acceptable. The current display is defined to be provided in landscape mode with 1136 pixels width and 640 pixels in height.

Note: While there is an attempt to keep the EFB HMI and CGD HMI same, differences are designed in. Not always are colors same, (all CGD data is displayed in white) not all EFB data is displayed on CGD. CGD status messages are a subset of EFB status messages, a blank CGD display is not good as pilot could confuse it with powered off CGD device.

Five fields are defined in the CDG display. Figure 6 depicts a typical CGD display and denotes the five areas. Two of the five areas require graphics drawing and three require text only processing, although one requires text animation. The layout shows three columns with the right hand column separated into three rows. All of the five areas are designed as HTML-5 drawing canvases.

Fully Populated CGD Display

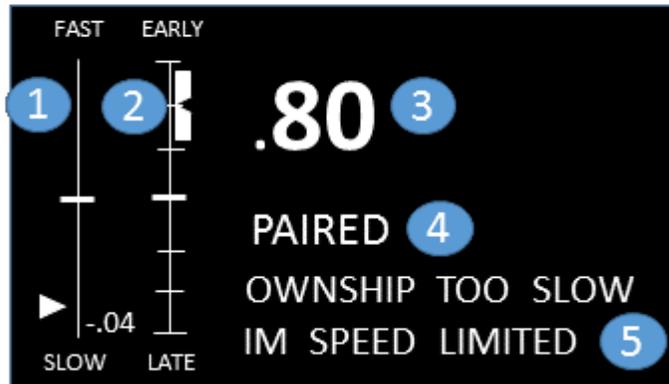


Figure 6 - CGD Display Areas

The five graphic areas contain:

1. Fast/slow indicator
2. Progress indicator
3. Commanded speed
4. Current FIM mode
5. Status messages

The FIM application (via FIMio) commands values and display graphics in the five areas. Any or all of which may be blank. However, the HMI WG has determined that a fully blank screen is not acceptable (appears same as if powered off) and the minimum normal display should contain at the very least the FAST-SLOW and EARLY-LATE labels. Further if non-normal operations are encountered, all information is pulled from the screen and the “CONNECTION LOST” message is displayed in the first line of the status message area as shown in Figure 7 below.

In addition to the “CONNECTION LOST” message, a sub-message below (and in smaller font) provides a reason for the error. This information is used by flight engineer to debug the error. Error reasons include: “NO INITIAL CONNECTION”, “LOST FIM IO”, “LOST CGD SERVER CONNECTION”, “LOST CGD SERVER TIMEOUT”, or “BAD DATA”.

Example of No Valid Data

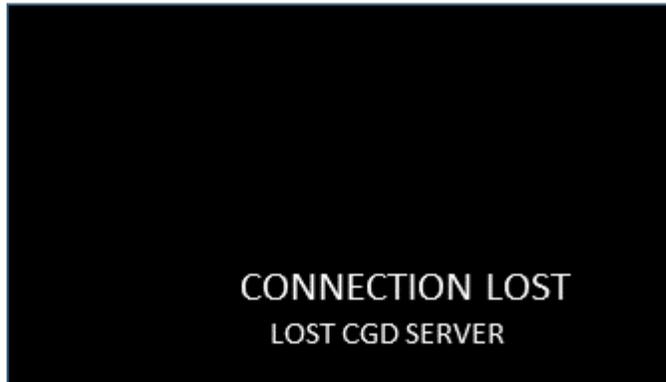


Figure 7 - Non-Normal Operations Display

In normal operations when not performing FIM operations but when the system is set up and running normally, the display will contain the FAST-SLOW and EARLY-LATE labels only as depicted below in Figure 8. These labels are displayed to ensure a blank CGD screen is not a normal operationally display, which is what will appear if the CGD device loses power. It is possible for a FIM status message “IM SYS FAIL” to be displayed during non-procedural operations.

No FIM Operation



Figure 8 - Normal Non-FIM Operations

During normal FIM operations outside 30 NM from target the following display will be seen with no Progress Indicator displayed as shown in Figure 9.

During FIM operations within 30 NMs a fully populated display will normally be shown as depicted in Figure 10, perhaps with additional Fast-slow and Progress indicator graphics.

Typical Beyond 30 NMs

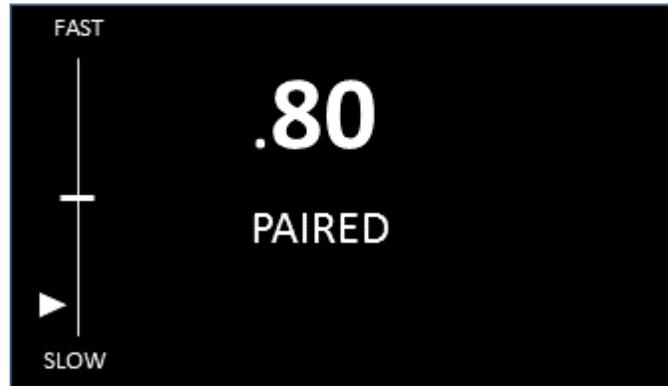


Figure 10 - FIM Operations Inside 30 NMs

The following sections describe normal display graphics for each of the five CGD display areas.

Note: it is possible for the CGD to display only labels and the “IM SYS FAIL” status message. While not earlier envisioned by HMI WG this has been added as a useful PFOV display element. Pilot will look to EFB for details of cause.

5.2.1.1 Fast-Slow Indicator Display Area

The Fast-Slow indicator display area is on the left of the screen and encompasses all vertical pixels. It contains top and bottom labels and a vertical bar and tick mark representing the FIM commanded ESP (Estimated Speed)). VAC (Velocity aircraft) is displayed as a white triangle along the left side of the vertical bar. Positive and negative speed deltas may also be shown (operationally only one at a time) as shown in Figure 11.

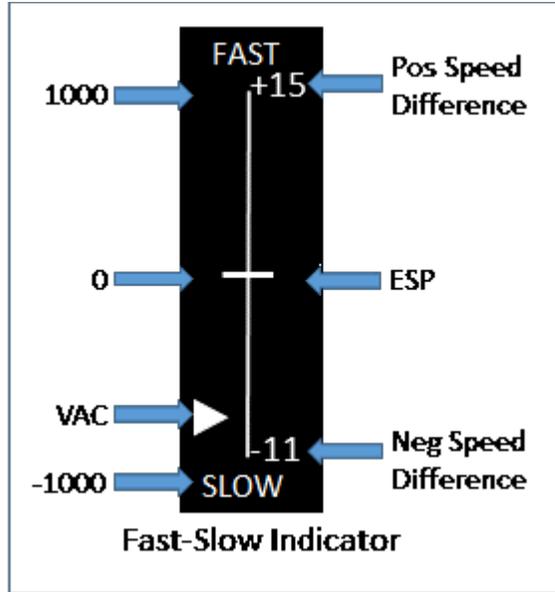


Figure 11 - Fast Slow Indicator

The Fast-Slow indicator may be displayed in any of the following display modes, all controlled by the FIMio. Not all modes are planned for operational usage. They are however, potential results of FIMio inputs. See notes below for which modes should not be encountered in operation.

Note: the Fast/Slow OFF mode should never be commanded by FIM IO.

The different display views for the display modes are depicted in Figure 12.

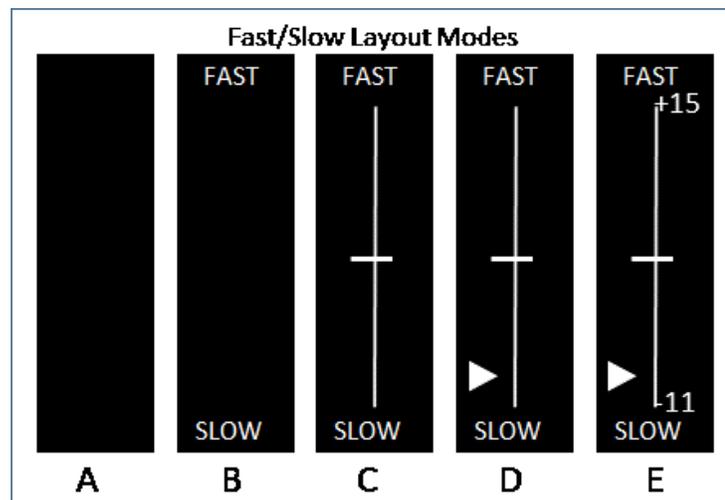


Figure 12 - Fast-Slow Display Modes

A. Off (not a normal operational mode)

- B. Labels only (normal operational mode for between FIM procedures)
- C. Enabled – no data (not a normal operational mode)
- D. Enabled with VAC
- E. Enabled with VAC & speed differentials

5.2.1.2 Progress Indicator Display Area

The Progress indicator display area is just to the right of the Fast-Slow indicator and encompasses all vertical pixels. It contains top and bottom labels and a vertical bar and tick marks. Progress indicator is displayed as a white caliper along the right side of the vertical bar. Caliper size depends on the scale which may be large or small as shown in Figure 13.

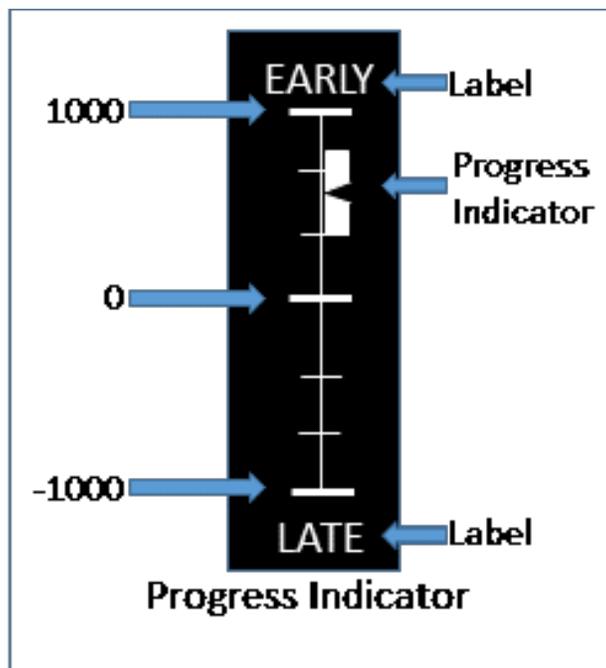


Figure 13 - Progress Indicator

The Progress indicator may be displayed in any of the following display modes, all controlled by the FIMio. Not all modes are planned for operational usage. They are however, potential results of FIMio inputs.

Progress indicator has modes for time-based operations (with EARLY-LATE labels) and distance-based operations (with NEAR-FAR labels). Progress indicator supports two scales, with difference size calipers and different tick marks. See notes below for which modes should not be encountered in operation.

Note: the Progress Indicator OFF mode should never be commanded by FIM IO.

The different display views for the display modes are depicted in Figure 14.

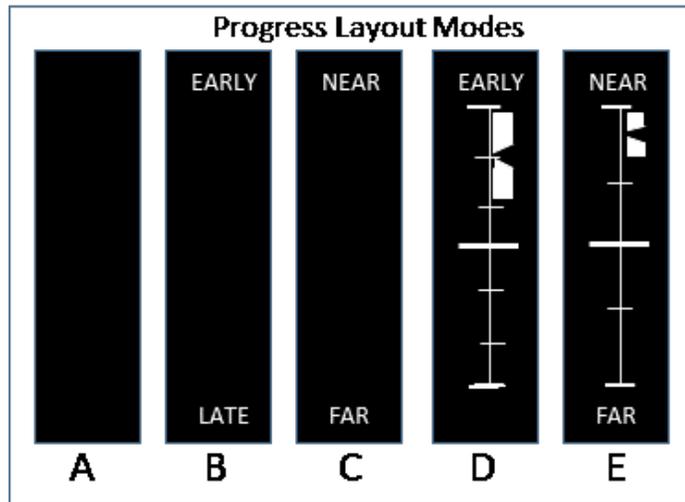


Figure 14 - Progress Indicator Display Modes

- A. Off - (not a normal operational mode)
- B. Time based labels only
- C. Distance based labels only
- D. Time based enabled with caliper (shown with large caliper)
- E. Distance based enabled with caliper (shown with small caliper)

5.2.1.3 Commanded Speed Display Area

The commanded speed display area is to the right of the Fast-Slow indicator and encompasses top third of the vertical pixels. It contains a single line of text, possibly an empty string. The commanded speed may be drawn in white for normal display on a black background or in black on a white background (reverse video). It may blink by alternating between white-on-black and black-on-white at a one hertz rate to get flight crew attention.

The following display modes are allowed and depicted in Figure 15.

- A. Off or no value
- B. Normal white
- C. Attention – reverse video
- D. Non-conformance – alternating normal & reverse video

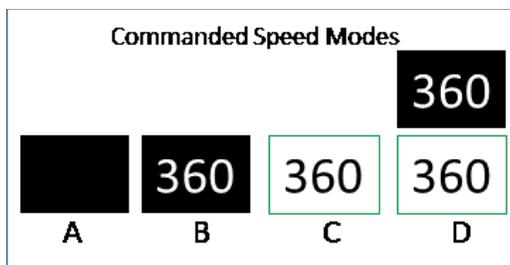


Figure 15 - Commanded Speed Display Modes

5.2.1.4 FIM Mode Display Area

The FIM Mode display area is to the right of the Fast-Slow indicator and below the commanded speed using the middle third of the vertical pixels. It contains a single line of text, possibly an empty string. The FIM mode may be drawn in white. While any text may be displayed the display is designed for following text values.

- UNABLE
- AVAILABLE
- ARMED
- SUSPENDED
- PAIRED

The following display modes are allowed and depicted in Figure 16. Text can be in white or empty.



Figure 16 - FIM Mode Display Modes

5.2.1.5 FIM Status Message Display Area

The FIM Status Message display area is to the right of the Fast-Slow indicator and below the FIM mode using the bottom third of the vertical pixels. It contains up to three lines of text, possibly only a single empty string. The status messages are only drawn in white. While any text may be displayed the display is designed for a set of text values, the longest message being “OWNSHIP OFF ROUTE”.

Note: the CGD FIM status message list is a subset of EFB FIM message list.

The following are expected FIM status messages:

1. TGT OFF ROUTE
2. OWNERSHIP OFF ROUTE
3. IM SPEED LIMITED
4. IM SYS FAIL
5. TGT DATA LOST

The following display modes are allowed and depicted in Figure 17, showing zero, one, two or three simultaneous messages. .

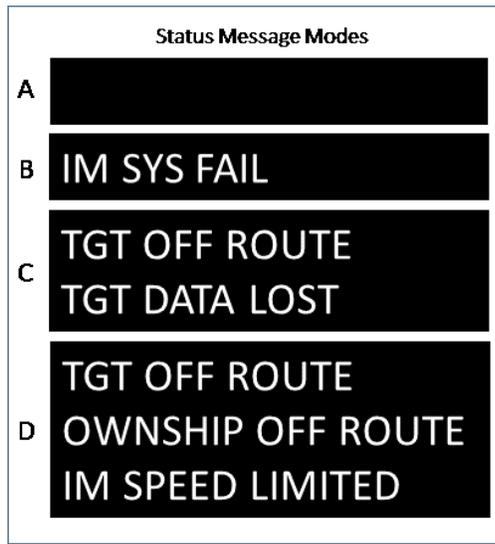


Figure 17 - Status Message Display Modes

5.2.2 HTML Document

The HTML document has a head and a body. Only the Javascript described in the next section exists in the head. The document definition is contained in the HTML body.

On initialization the document creates the five graphics display areas by generating and initializing canvases for each area. Each canvas has an “id” that is available for the Javascript to draw into. The drawing graphics areas have multiple layered canvases. The canvas locations and pixel sizes are defined in each canvas division definition, along with default style for text size, color, and font. The following canvases are created for the five display areas. The layered canvases are transparent and overlay a dimensionally equivalent base canvas.

1. Overall CGD canvas “cgdCanvas” occupying full 1136 by 640 pixels. This is the base canvas and has a black background color. No drawing is performed in this canvas. However, it contains all other canvases.
2. Fast-Slow base canvas “fsCanvas” occupying left 210 pixels by 640 high is generated. It contains the labels, bar and tick mark. It has a black background. This canvas need only be updated when the fast-slow indicator mode is updated (very rarely).
3. Fast-Slow layer canvas “fsLayerCanvas” occupies the same area as the fsCanvas. It has a transparent background. It contains the VAC triangle and positive and negative difference text values. It is clipped to ensure writing does not impinge the labels. This area is redrawn (regularly in operation) as VAC or difference values update.
4. Progress indicator base canvas “piCanvas” occupying left 240 pixels by 640 high is generated. It contains the labels, bar and tick marks. It has a black background. This canvas need only be updated when the progress indicator mode is updated (very rarely).
5. Progress indicator layer canvas “piLayerCanvas” occupies the same area as the piCanvas. It has a transparent background. It contains the progress indicator caliper. It is clipped to ensure writing does not impinge on the labels. This area is redrawn (regularly during operation) as progress indicator or scale updates.
6. Commanded speed canvas “spdCanvas” occupies remaining right 660 pixels and top 300 pixels of the cgdCanvas area. It contains a single line of text. This canvas is updated when the commanded speed is updated (regularly during operation).
7. FIM mode canvas “modeCanvas” occupies remaining right 660 pixels and the middle 200 pixel right below the spdCanvas. It contains a single line of text. This canvas is updated when the FIM mode is updated (regularly during operation).
8. FIM Message Status canvas “statusCanvas” occupies remaining right 660 pixels below the modeCanvas with 300 vertical pixels. It contains up to three lines of text. This canvas is updated when the status messages are updated (regularly during operation).

5.2.3 Javascript Computation

The cgd.html Javascript is contained in the HTML head. It computes and stores all of the computational variables. It manages the canvases and performs drawing logic. It draws the graphics and text in the canvases and implements the limited animation. All of the data is managed in the document DOM and accessed by the canvas’ handles.

The document sections are located by their canvas Id's and mapped to document and canvas contexts.

Each time a JSON message is received, the new information is processed. Updated text and graphics computed and then drawn in the appropriate display areas and canvases. Only necessary drawing is accomplished to minimize graphics drawing time and impacts on latency, delay and jitter.

5.2.3.1 Initialization

The Javascript initializes internal variables and display parameters. The initial display is completely blank or black. All text values are empty and no labels or graphics are drawn. Values that have defined defaults in the FIMio to CGDServer interface are set to default values but will be overwritten on initial JSON data submittal.

The Javascript makes a Web Socket connection to the CGDServer by creating a new Web Socket client and connecting with the CGDServer's IP address and Web Socket Port (at this time 9099). The Web Socket sets up callback functions for "onopen" and "onerror" messages as well as the main callback of "onsubmit" which is defined below in the next section.

Document contexts are built for each canvas using the canvas Ids. Default canvas parameters such as font type, font size, font color, etc... are set for each canvas.

Both the piLayerCanvas and the fsLayerCanvas obtain clipping regions to limit the drawing area and keep symbols away from labels and boundaries.

CGDServer is commanded for a full JSON update. Lastly a timeout is set. If the CGDServer does not respond with new data within the timeout, a "CONNECTION LOST" message will be generated right out of the gate. The initial timeout is longer than the in-operation timeout as it is ok to assume there is no valid operation upon initialization.

5.2.3.2 On Submit

The main method of the Javascript is the callback for "onsubmit". This function is called and passed incoming JSON message. The Web Socket function directly calls the "drawDisplay (string)" function with the JSON message as the "string" value. The drawDisplay function performs all of the operational display computation and drawing.

First thing the function does is clear the incoming timeout. The last thing it does is reset the timeout for an operational value (at this time five seconds).

Incoming string data is JSON and is parsed using JSON.parse(string) function. If an error is discovered in parsing execution terminates for this cycle and waits for the next cycle of JSON input. A corrupted JSON message results in a NAK message being sent to CGD Server, and the incoming JSON data is ignored.

If the incoming string data is successfully parsed by the JSON parser then it returns a response that contains the JSON text parsed and accessible as individual JSON fields. Each JSON field is accessed using a simple command “var x = response.fieldName” where the parsed JSON response field is requested by name is set to a value in variable x, alternatively, the JSON field may be used directly in a logical operation. One example is the “if (response.fieldName != undefined)” This construct is used to determine if the field exists at all in the JSON response. It is important to handle this situation to avoid accidental Javascript exceptions and crashing.

Processing for each specific area in CGD graphics display is defined below:

5.2.3.2.1 Fast-Slow Computations

If the Fast-slow enable message is sent then Fast-slow enable variable is set to the incoming value. When the incoming value is set to 0 (off) or 1 (labels only) both the fsCanvas and fsLayerCanvas are cleared. If the new mode is greater than 0 (off) then the labels are drawn on the fsCanvas. If the new mode is 2 (enabled) then the bar and tick marks are drawn in light gray and white.

If an incoming Fast-slow VAC message and/or Fast-slow differences message is received then the variables are updated.

If the current Fast-slow enabled mode is 2 (enabled) then the upper layer canvas fsLayerCanvas is cleared. And Fast-slow differences and VAC are drawn on the upper layer canvas. The VAC vertical position is computed by taking the pixel limits and mapping them to the ICD scale limits of +/- 1000. The FIMio commands vertical location to +/-1000 with 0 in center. Javascript converts that to map the actual pixel ratio and offset and draws the VAC triangle to the left of the bar with right point centered on the commanded vertical pixel row. Clipping insures the top and bottom of the triangle don't obscure the labels. Further, no error checking is done on the incoming values and any inputs above 1000 or below -1000 will appear as no VAC has been commanded.

5.2.3.2.2 Progress Indicator Computations

If the Progress indicator enable message is sent then Progress enable variable is set to the incoming value. When the incoming value is set to less than 3 (labels only) both piCanvas and piLayerCanvas are cleared. When the indicator is changed to any value above 0 (off) the labels are drawn on the base canvas. Value of 1 or 2 labels only. Value of 3 or 4 indicates fully enabled indicator. When mode is set to fully-enabled the bar and tick marks are drawn on the base canvas in light gray and white.

If an incoming progress value or progress scale message is received then the variables are updated.

If the current indicator enabled mode is 2 or 4 (enabled) then the upper layer canvas piLayerCanvas is cleared. The progress indicator caliper is drawn on the upper layer canvas. The progress indicator vertical position is computed by taking the pixel limits

and mapping them to the ICD scale limits of +/- 1000. The FIMio commands vertical location to +/-1000 with 0 in center. Javascript converts that to map the actual pixel ratio and offset and draws the caliper to the right of the bar with middle gap centered on the commanded vertical pixel row. Clipping insures the top and bottom of the caliper don't obscure the labels. The scale controls the caliper size. Small scale is small caliper and large scale is large caliper.

5.2.3.2.3 Commanded Speed Computations

If the commanded speed color message is sent then the spdCanvas text color is set to the appropriate incoming command (0 – white, 1 – black).

If the commanded speed background color message is sent then the speed background value is set to the appropriate incoming command (0 – black, 1 - white).

If the commanded speed value message is sent then the speed value parameter is set to the incoming text value.

When commanded to change the speed value or color or background color the spdCanvas is cleared the background and text color are set and the speed value is drawn.

5.2.3.2.4 FIM Mode Computations

The FIM mode color message is ignored and text is always white.

If FIM mode value message is sent then the FIM mode value parameter is set to the incoming text value.

When commanded to change the FIM mode value the modeCanvas is cleared the mode value is drawn.

5.2.3.2.5 FIM Status Message Computations

If the FIM status message is sent then the status value parameter is set to the incoming text values (array).

When commanded to change the status messages the statusCanvas is cleared and the new text messages are drawn with the (up to) three text strings aligned left justified above and below each other.

5.2.3.2.6 Lost Connection Computations

This message halts normal operations and calls the displayError function with “server lost” parameter. This triggers “CONNECTION LOST” message (and reason). It clears all other information from display until normal operations are resumed at which time it commands a full JSON update from CGDServer.

5.2.3.3 Display Error

The displayError (msg) function is called anytime there is an error in processing. In operation JSON parsing errors are logged and current cycle data processing is discontinued. The other error messages result in connection lost messages. They cause all of the canvases to be cleared and the “CONNECTION LOST” message to be displayed in the status message display area/canvas with a sub-message giving reason for the error.

5.3 FIMio.js

FIMio.js is a software package provided to simulate the actual Honeywell FIM IO function. It is a Javascript program that runs on either the EFB or developmental laptop and executes the interface exactly the same as the Honeywell FIMio. It is used for development and testing. It may also function as a playback tool to playback captured CGD FIM sessions. This may be useful for debugging, testing and software or HMI validation or to reconstruct operational displays.

This software is included in this discussion to allow CGD system to perform standalone execution (without FIM application and Honeywell FIMio). It is invoked in a MS DOS command window as “node FIMio” or with any of the following command line options. In default mode the program makes only one pass through the “data.txt” JSON input file, delays 500 msec between messages and executes on the “data.txt” input file.

The program accepts command line parameters, loads the input file, and opens a TCP connection to the CGDServer. It then steps through each JSON message in order, sending it over the TCP interface.

Command line options include the following:

1. -s to show command line output and chatter
2. -l to loop infinitely through the JSON messages in data file
3. -d xxx to set the delay to xxx msec between messages
4. -f xxx to set the input file name to xxx (i.e. data.txt).

Appendix A: FIM IO to CGD Server Interface Definition (ICD)

The protocol between the FIMio and the CGDServer is based on TCP/IP sockets. The CGDServer sets up a listening socket and waits for the FIMio to connect. Javascript Object Notation (JSON) is used as message format protocol. A set of CGD display message (in JSON format) are defined to provide the message and data level interface.

There are two interfaces in the CGD system.

1. FIMio to CGDServer
2. CGDServer to CGD display

Each uses a slightly different protocol but uses same data and message format structure and mostly identical message formats. The messages are defined in Javascript object notation or “JSON”. JSON messages are text based and have a simple syntax. For the effort the key elements of the JSON message are the container “{ JSON data }” in curly brackets and the data elements. Each data element has a defined name and is followed by data value (or values), separated by a colon. Multiple fields are comma delimited. The format is “data Name”:”data Value” or for multi-value sets, arrays are used. Such as: “data Name”:[‘0’,’1’,’2’]. Example is {“Pies”:[“apple”,”peach”,”banana”], “Cookies”:[“Chip”, ”Sugar”, “Oreo”], “Cost”：“\$1.00”}.

The Javascript engine provides a built in feature to parse, analyze and process JSON fields. Fields are then requested by name and a value or “Undefined” is returned.

It is required that JSON data be properly formatted. JSON error formats will cause the entire message to be discarded in operation, resulting in no update during this display cycle.

The FIMio is required to send the fully populated JSON message every timed display cycle event to CGDServer to ensure ability to quickly synchronize coming and going CGD display devices in the one-way communications architecture. The interface is for a single JSON message for each event cycle. Each field is described below, along with display examples and default values. The following sections are defined as the five elements of the CGD display layout, but are simply a single text string.

A.1 Fast-Slow Indicator Fields

JSON Fields:

- Enable fast/slow indicator “fsenable”
 - Values = (1-labels only, 2-on/enabled)

-
- Purpose: on enables indicator: off – blank, labels only – “FAST & SLOW”, on – labels, bar & ESP with difference values & VAC as commanded
 - Default = “1”
 - Example: “fsenable”:”2”
 - VAC value “fsvac”
 - Values = range from -1000 to +1000 or “” (no white triangle to display)
 - Purpose: sets vertical center of white triangle relative to top & bottom of CGD scale when not “”
 - Default = “”
 - Example: “fsvac”:”500”
 - Speed difference “fsdiff”
 - Values = [“positive value”, “negative value”]
 - Purpose: displays positive or negative speed difference values when exceeds defined value
 - Default = []
 - Examples: [“+15”, ”-11”], [“+15”] or [“”, “-11”]

Note: only first or second entry should be non “”.

A.2 Progress Indicator Fields

JSON Fields:

- Enable Progress indicator “pienable”
 - Values = (1- distance labels, 2-time labels only, 3-distance on, 4-time on)
 - Purpose: on enables indicator: off – blank, labels only “EARLY” & “LATE” or “NEAR” & “FAR”, on adds bar & tick marks & caliper if commanded
 - Default = “2”
 - Example: “pienable”:”4”
- Progress value “pival”

- Values = range from -1000 to +1000 or ""
- Purpose: sets vertical center of symbol relative to top & bottom of CGD scale or (no caliper displayed)
- Default = ""
- Example: "pival": "500"
- Scale "piscale"
 - Values = 0-small, 1-large
 - Purpose: set size of caliper
 - Default = "0"
 - Example: "piscale": "1"

A.3 Commanded Speed Fields

JSON Fields:

- Commanded speed value "spdval"
 - Values = any text string, number or ""
 - Purpose: provides commanded speed in knots or mach
 - Default = ""
 - Example: "spdval": "350"
- Speed color "spdcolor"
 - Values = 1-white, 3-attention, 4-nonconformance
 - Purpose: set commanded speed display mode
 - Default = "1" (white)
 - Example: "spdcolor": "3"

A.4 FIM Mode Fields

JSON Fields:

- Commanded speed value "mode"

- Values = any text string including ""
- Purpose: provides FIM mode
- Default = ""
- Example: "mode": "PAIRED"

A.5 FIM Status Message Field

JSON Fields:

- Status messages "status"
 - Values = zero to three strings of text
 - Purpose: provides FIM annunciations
 - Default = []
 - Example: "status": ["IM SPEED LIMITED", "OWNSHIP OFF ROUTE"]

A.6 Fully Populated Example

The following is an example of a fully populated JSON message from FIMio.

```
{
  "fsenable": "2",
  "fsvac": "500",
  "fsdiff": ["+15", "-5"],
  "pienable": "4",
  "pival": "500",
  "piscale": "1",
  "spdval": "350",
  "spdcolor": "1",
  "mode": "PAIRED",
  "status": ["IM SPEED LIMITED", "", ""]
}
```

Sent as one entry with no linefeeds:

```
{"fsenable": "2", "fsvac": "500", "fsdiff": ["+15", ""], "pienable": "4", "pival": "500", "piscale": "1", "spdval": "350", "spdcolor": "1", "mode": "PAIRED", "status": ["IM SPEED LIMITED", "", ""]}
```

Appendix B: CGD Server to CGD Display Interface Definition (ICD)

The CGDServer to CGD display interface is based on communications between a single CGDServer and multiple CGD displays. The CGD displays are not co-resident with CGDServer (though they may be in development) and use an HTTP interface between the CGDServer's HTTP server host and the CGD display's HTTP client in an HTML-5 compatible browser.

The concept is to use a "REST-ful" interface and allow efficient server updates to the HTTP clients. This is not standard web practice. Security issues usually require the HTTP client to request data. HTTP servers pushing data is usually considered risky. However, Web Sockets have been designed to eliminate the risk and provide server-side data push efficiently and securely. Web Sockets are layered over the HTTP protocol on the browser. This allows a persistent connection and alleviates security concerns about the server driving the client. The peer-per-message formats are JSON text strings.

The CGDServer to CGD display message data is based on the same concept and message format as the FIMio to CGDServer. In fact, almost all of the message fields are identical. With a few changes required to allow simplified CGD display logic and simplified FIMio interface. Below is a list of the message differences:

1. The CGDServer only sends data to CGD display that changes from previous message (fully populated JSON not required).
2. CGDServer does send fully populated messages to newly arriving CGD display clients to allow instant synchronization. A fully populated message is also sent when any CGD displays reports NAK message indicating corrupted message was received.
3. CGDServer must send a message every cycle, even if no data is changed, thus using the keep-alive message field. The only time no message is sent is when incoming data from FIMio is corrupted.
4. CGDServer may detect a lost connection, so we added a lost connection message field.
5. The Commanded Speed color from FIMio contains color mode (normal, reverse video or blinking) which is mapped to an instant commanded speed color and instant background color. This allows normal display, reverse video and blinking (controlled by CGDServer) without CGD display logic or knowledge of FIM operational mode, so a Commanded speed background color message field is transmitted when needed as well.

B.1 Commanded Speed Fields

JSON Fields:

- Speed color “spdcolor”
 - Values = 1-white, 3-black
 - Purpose: modified to actual “Now” commanded speed color (in context of commanded speed mode and background color)
 - Default = “1” (white)
 - Example: “spdcolor”:”1”
- Commanded speed background color “spdbg”
 - Values = 0- black or 1- white
 - Purpose: reverse video or blink
 - Example: “spdbg”:”1”

B.2 Keep Alive Field

When no computational data has changed (and CGDServer does not detect incoming message error) a keep alive message field is required. In practice the keep alive message is sent every cycle but is ignored and simply triggers update and watchdog timer reset when it is the only field in the entire JSON message.

JSON Fields:

- Keep alive field “alive”
 - Value = “”
 - Purpose: continuous CGD messaging
 - Example: “alive”:””

B.3 Connection Lost Field

When the CGDServer detects connection lost between CGDServer and FIMio or loss of TCP connection, it sends the connection lost message.

JSON Fields:

- Status messages “lost”

- Values = “timer”, “conn”
- Purpose: provides CGD display with lost connection or time out status
- Example: “lost”:”timer”

B.4 Bad Data Field

When the CGD Display detects bad data from CGDServer, it sends the bad data or NAK message due to lost connection/CGD time out or corrupt JSON message.

JSON Fields:

- Bad data messages “nak”
 - Values = “json” or “lost”
 - Purpose: provides CGD server with bad data non-acknowledgement
 - Example: “nak”:”json”

B.5 Full Data Field

When the CGD Server determines to send a full JSON message vs. deltas only, it sends this data field. This allows CGD server to recognize this is a full update due to new arriving client, too long of a time delay or response to NAK error message request for full data.

JSON Fields:

- Full messages “full”
 - Values = “”
 - Purpose: provides CGD server with knowledge of full JSON update
 - Example: “full”:””

Appendix C: Boeing CGD Software Test Plan (STP)

The Boeing CGD Software Test Plan (STP) is captured in this section. The STP describes the plan for testing Boeing CGD software to all of allocated CGD SRS (SRDD and CGD SRS derived) and CGD SDD derived requirements. The test approach taken will be as follows:

1. List all allocated requirements.
2. Describe which requirements are allocated to Boeing CGD software system or components.
3. For requirements where only partial verification can be accomplished in the standalone Boeing CGD system, describe what external aspect must be verified at a higher system level or at a peer non-Boeing CGD software test.
4. Describe methods that will be used to test and verify all Boeing allocated CGD system capabilities and requirements.
5. Define and describe a group of collected tests for accomplishing detailed requirements and capability verification.
6. Allocate capabilities and requirements to the appropriate test group(s) for testing.
7. Document requirements summary, test group allocation and verification level.
8. Describe the process for testing, verifying, and documenting test results.
9. Discuss built-in capabilities for future regression testing (should downstream changes be made to the design or requirements after formal testing) and to support peer-level software and CGD system level verification testing (not directly part of this document's scope).

Note: This section contains test plan. It is accurate for test approach and test names. HOWEVER, implementation details and system test mapping will cause minor changes. The actual tests and results are contained in the "CGD Test Report" (along with updated matrix) and in the "CGD Systems Test Plan". Please refer to these documents for final and accurate information related to software unit testing implementation.

C.1 Approach

The general approach to testing and requirements verification is based on the fact that the primary purposes of the CGD software is (1) to display information to flight crew correctly and in a timely manner AND (2) ensure consistency of PFOV information

displayed on CGD with CTDI displays. The second, not being in the scope of this Boeing CGD software but requiring CGD system contribution to complete system level requirements.

Display oriented requirements will be tested and demonstrated using the CGD Simulator concept. The FIMio.js simulation will be used in place of Honeywell FIM software. It will drive the actual CGDServer and cgd.html software components together in an integrated CGD system. Test procedures will be developed, primarily using test scripts of JSON messages to simulate and stimulate CGDServer and cgd.html execution, processing and rendering. The test scripts will be attached to test procedures along with test results and log files. They will be organized into logical testing groups to execute similar CGD display features. These tests will execute:

1. Realistic operational scenarios to verify HMI design requirements and criteria.
2. Detailed test scripts for each of the five display areas to verify computing and graphics of each display area.
3. Detailed test scripts to verify connectivity management, timing and latency requirements.
4. Detailed test scripts to verify non-normal operations.
5. Non-test script unit testing to exercise and validate internally complex and non-accessible logic paths and conditions.

As many code elements will be verified by the simulation and test script approach as possible. However, some error and timing conditions will require modification to the non-operational driver software or even operational code for low-level functional testing. Where possible and when not affecting operational performance, flags will be used to alter and control the software execution and logic paths. This approach facilitates regression testing if and when later re-verification is required. Also, it minimizes the risk of altering code functionality by intrusive code modifications.

As much testing as possible will be performed on Windows laptop. All of the functional software features can be tested on a generic Windows PC running a generic HTML-5/Javascript enabled browser. HMI design and performance verification will require testing on actual EFB and CGD display devices for actual timing, sizing and exercising real browser in operational context. In any case all scripted verification testing will be completed on actual hardware at least once for the final verification effort.

Test groups defined will be basis for allocating specific test procedures to each script to ensure all mapped requirements and capabilities are verified and executed. Test Procedures will be documented for each test group, test script and test line item. The test procedures will either be a script or process description. They will describe how to perform the test. What the results for test are and where the logs are recorded. The log files may be used as a comparison for additional future evaluation and

verification. For each test, the logs will be analyzed and recorded for test procedure compliance. They will be saved and made available to team to potentially be used later to support regression or system level testing and verification. This later sharing and reuse applies ONLY to tests executed by non-invasive scripting techniques.

C1.1 Boeing-Honeywell Testing

It is Boeing's intent to coordinate with Honeywell's efforts at the CGD system and FIM system levels. To this end Boeing will make accessible to Honeywell final test procedures and scripts. This will allow Honeywell to drive the Boeing CGD software in their CGD and FIM system testing environment. If and when conflicts, confusion or errors arise they can refer to and compare results with Boeing CGD scripted tests. This works for software verification to requirements as well as verifying HMI design criteria.

Further, it is Boeing's desire that Honeywell capture and provide Boeing with operational scripts from FIM application execution that Boeing can use at the unit and subsystem level to verify correct operation performance and HMI design. To this end the CGDServer has a built in capture function. When turned on the CGDServer captures a log of commands that may be used for playback of the CGD displays in the FIM application. This will be useful for system debugging and verification in later stages of CGD system and FIM system testing.

C.2 CGD System Level Verification

While not in the scope of the Boeing CGD STP, support is provided for CGD system level testing and peer level Honeywell CGD unit testing (non-Boeing FIM system components). Supporting this is important as the FIM MOPS, SRDD, SRS all capture common display system and PFOV requirements. These requirements are allocated to CTDI and CGD PFOV systems, very few of which are fully verifiable in any single software unit test or Boeing CGD subsystem testing. Many PVOF requirements are allocated and processed by several software components, systems and subsystems. See Figure 1. Provisions for supporting non-Boeing CGD subsystem testing includes the following:

1. Description of which requirements allocated to CGD are NOT fully verified in this testing.
2. Description of potential system level testing support provided by purpose-built Boeing CGD test scripts.
3. Sharing of test scripts, results and log files with Honeywell.

C.3 Planned Verification Tests

The following is a list of planned tests that will be mapped to the necessary test scripts and alternative testing methods.

C.3.1 Display Computation & Drawing Tests (DT)

The five drawing areas will be verified for all graphics drawing and text drawing correctness. This includes correctly handling drawing locations, colors, text attributes, font sized, text animations and graphics drawings and animations.

These tests use the scripted approach and verify both CGDServer and cgd.html software simultaneously. They are used to verify HMI design and PFOV display requirements.

1. DT 0 - Overall CGD Display Test

Overall CGD Display Test will validate, test, log and demonstrate all top-level CGD display capabilities and requirements. When combined with DT 1 through DT 5 will validate ALL CGD display capabilities and requirements for CGDServer and cgd.html normal display operations. This includes processing JSON message fields, normal communications, Javascript parameter processing (CGDServer and cgd.html), cgd.html graphics and text drawing/rendering and logic processing and normal timing capabilities. It will verify general HMI design criteria and operational utility along with all CGD display requirements. It is a CGD system level script driven test for top-level CGD capabilities.

2. DT 1 – Fast-Slow Indicator Test

Fast-slow Indicator (FSI) Test will validate, test, log and demonstrate all FSI display capabilities and requirements. This includes processing FSI JSON message fields, normal communications, Javascript parameter processing (CGDServer and cgd.html), cgd.html graphics and text drawing/rendering and logic processing and normal timing capabilities. It will verify general HMI design criteria and operational utility along with all FSI display requirements. It is a FSI script driven test for FSI CGD capabilities.

3. DT 2 – Progress Indicator Test

Progress Indicator (PI) Test will validate, test, log and demonstrate all PI display capabilities and requirements. This includes processing PI JSON message fields, normal communications, Javascript parameter processing (CGDServer and cgd.html), cgd.html graphics and text drawing/rendering and logic processing and normal timing capabilities. It will verify general HMI design criteria and operational utility along with all PI display requirements. It is a PI script driven test for PI CGD capabilities.

4. DT 3 – Commanded Speed Test

Commanded Speed (CS) Test will validate, test, log and demonstrate all CS display capabilities and requirements. This includes processing CS JSON message fields, normal communications, Javascript parameter processing

(CGDServer and cgd.html), cgd.html text drawing/rendering and logic processing and normal timing capabilities. It will verify general HMI design criteria and operational utility along with all CS display requirements. It is a CS script driven test for CS CGD capabilities.

5. DT 4 – FIM Mode Test

FIM Mode (FM) Test will validate, test, log and demonstrate all FM display capabilities and requirements. This includes processing FM JSON message fields, normal communications, Javascript parameter processing (CGDServer and cgd.html), cgd.html text drawing/rendering and logic processing and normal timing capabilities. It will verify general HMI design criteria and operational utility along with all FM display requirements. It is a FM script driven test for FM CGD capabilities.

6. DT 5 – FIM Status Test

FIM Status (FS) Test will validate, test, log and demonstrate all FS display capabilities and requirements. This includes processing FS JSON message fields, normal communications, Javascript parameter processing (CGDServer and cgd.html), cgd.html text drawing/rendering and logic processing and normal timing capabilities. It will verify general HMI design criteria and operational utility along with all FS display requirements. It is a FS script driven test for FS CGD capabilities.

7. DT 6 – Operational HMI Test

Operational HMI (OH) Test will validate, test, log and demonstrate all CGD HMI display capabilities and requirements for one or more operationally oriented demonstrations. This includes demonstrating cgd.html drawing, rendering and animations for both static HMI design acceptance and dynamic HMI sequential operations. This test is different than the other tests in that it will NOT attempt to execute logic paths to test code points. It is designed instead to demonstrate natural (though perhaps sped up) sequence of CGD displays and HMI sequences to allow HMI evaluations and perhaps training and orientation of operators. It will verify general HMI design criteria and operational utility. It is an OH script driven test for OH CGD HMI capabilities.

C.3.2 Connectivity and Link Management Tests (CT)

The different normal and failure modes relative to connectivity management of TCP, HTTP and Web Socket connections are verified by the tests below. The major concerns are to create, manage and effectively handle anticipated failure conditions of the communications protocols and services between the FIMio and the CGD display.

These tests use the scripted approach and verify both CGDServer and cgd.html software simultaneously. They are used to verify HMI design for lost connections and PFOV display requirements, along with unit level derived requirements.

Where possible corrupted test scripts will be used to test failure modes. Alternatively, flags will be sent to non-operational driver code (FIMio.js). In some cases external triggers will be implemented to induce failure conditions (interrupt FIMio.js messaging, break FIMio-CGDServer TCP connection, interrupt CGDServer functional processing, break HTTP and/or Web Socket connection, restart or kill browser session, break Ethernet connection, etc...). Any tests that require code manipulation will be captured in the non-normal test subsection C.3.4 below.

1. CT 1 – Normal Operations Connectivity Test

Normal Operations Connectivity (NOC) Test will validate, test, log and demonstrate all normal communications setup, management and service operations capabilities and requirements. This includes CGDServer connecting with FIMio over TCP/IP and CGD Displays over HTTP & Web Sockets. It also includes cgd.html connectivity to CGDServer. It is a NOC script driven test for NOC CGD communications capabilities.

2. CT 2 – Failure Modes Connectivity Test

Failure Modes Operations Connectivity (FMC) Test will validate, test, log and demonstrate all non-normal communications setup, management and service operations capabilities and requirements. This includes CGDServer listening for and re-connecting with FIMio over TCP/IP and CGD Displays over HTTP & Web Sockets. It includes meeting and missing timeout schedules, and handling starts and restarts. It also includes cgd.html re-connectivity to CGDServer. It is a FMC script driven test for FMC CGD communications capabilities.

C.3.3 Performance Tests (PT)

The latency and performance requirements are verified by analysis using metrics collected using non-real time methods. These tests will be conducted using both the scripted approach and software driver (FIMio.js delay) manipulation. That is it will exercise temporally long computationally loaded test scripts. The FIMio.js simulator will be commanded to execute with a very short delay. Execution will be timed. Output will be capture to log files and analyzed to ensure no data is lost in display processing. Estimated time per cycle will be calculated and checked against timing requirements. The tests will verify typical, loaded and anticipated worst case performance for latency and timing.

Note: there will be no attempt to verify or measure actual timing performance of each time step, processing cycle or message processing.

These tests will be conducted on the actual hardware devices using one CGDServer in EFB platform and two CGD display devices.

Example: a test script may be developed for a loaded condition over 1000 seconds of data (at 2 hertz, 2000 JSON message cycles). The FIMio.js will be configured for a 100 msec delay. A stopwatch will be used to determine total computation time. The result will be divided by the number of steps to compute average latency and execution time. The result should be well less than the requirement to allow some statistical variation.

1. PT 1 – Typical Performance Load Test

Typical Performance Load (TPL) Test will validate, test, log and demonstrate near-real time execution of Boeing CGD system (CGDServer TCP input to final CGD display rendering) capabilities and requirements. It is a TPL script driven test for TPL CGD communications capabilities. The TPL script is typical operational performance loaded.

2. PT 2 – Heavy Performance Load Test

Heavy Performance Load (HPL) Test will validate, test, log and demonstrate near-real time execution of Boeing CGD system (CGDServer TCP input to final CGD display rendering) capabilities and requirements. It is a HPL script driven test for HPL CGD communications capabilities. The HPL script is heavily operational performance loaded (at least 200% of typical operational CGD graphics load).

3. PT 3 – Worst Case Performance Load Test

Worst Case Performance Load (WCPL) Test will validate, test, log and demonstrate near-real time execution of Boeing CGD system (CGDServer TCP input to final CGD display rendering) capabilities and requirements. It is a WCPL script driven test for WCPL CGD communications capabilities. The WCPL script is heavily operational performance loaded (as close to possible worst case operational CGD graphics load).

C.3.4 Non-normal Operations Tests (AT)

There are a number of software unit test conditions that must execute either non-normal operational conditions or execute complex logic paths that are never executed in normal operational or even typical failure conditions. These tests require fabricating either broken data messages (missing fields, unparsable JSON messages), or modifying operational or driver software to command non-operationally accessible alternative software execution paths between internal functions within the two software components. These are classified as non-normal operations tests.

Where possible corrupted test scripts will be used to drive non-normal behavior. First fallback will be modifying non-operational test driver (FIMio.js) code or input

parameters. Lastly, non-operational code breaks will be introduced for testing, documented then removed from the operational code after verification. Typical non-operationally connectivity tests are described above in the connectivity test subsection and not repeated here.

1. AT 1 – Bad Message Test

Bad Message (BM) Test will validate, test, log and demonstrate Boeing CGD system resiliency features when bad or corrupted FIMio to CGDServer or CGDServer to cgd.html messages are processed, detected and handled. The objective is to ignore the bad data and continue normal operations (unless timeout occurs) and handle (recover from) all potential software stopping exceptions. The BM script contains a variety of corrupted JSON messages and JSON messages with potential errors (bad values, missing fields, misspelled values, etc...).

2. AT 2 - CGDServer Non-normal Logic Test

CGDServer Non-normal Logic (CAL) Test will validate, test, log and demonstrate CGDServer resiliency features when non-normal logic paths are executed in the CGDServer.js code. These tests are defined bottom up from potential error conditions and complex logic flows defined in the software component design. The CAL script contains a variety of corrupted JSON messages and JSON messages with potential errors or unexpected capabilities. Further, this test will require procedures that are beyond just scripting capabilities to test and are limited to testing just this component. For example, this may require changing the code in or between several Javascript methods to execute all of the code in the Javascript method.

3. AT 3 – cgd.html Non-normal Logic Test

The cgd.html Non-normal Logic (CHAL) Test will validate, test, log and demonstrate cgd.html resiliency features when non-normal logic paths are executed in the cgd.html code. These tests are defined bottom up from potential error conditions and complex logic flows defined in the software component design. The CHAL script contains a variety of corrupted JSON messages and JSON messages with potential errors or unexpected capabilities. Further, this test will require procedures that are beyond just scripting capabilities to test and are limited to testing just this component. For example, this may require changing the code in or between several Javascript methods to execute all of the code in the Javascript method.

C.4 Requirements to Verification Mapping

The following table (Table 2) lists all requirements to be verified as part of the Boeing CGD software testing scope. The table briefly describes each requirement and provides a summary of what aspects are tested in this scope and what are expected to

be tested outside this scope of testing. Further, it maps the requirement to one of the defined test groups listed in the previous section. See CGD SRS for full requirement description and access to requirement allocated in SRS to CGD but not allocated to Boeing CGD software per table below.

Verification column key:

NA – verification not applicable.

NO – no, planned verification by Boeing CGD software tests.

PB – verification partially in compliance by Boeing CGD software tests but requires Honeywell unit and/or FIM systems tests to complete full requirement verification. Most common reason is requirement specifies drawing value by Boeing CGD and determining value and/or resolution by Honeywell FIM.

YC – yes, verification completed by Boeing CGD software tests.

Table 2 – CGD Requirements Testing Matrix

ID	Requirement Synopsis	Verification Notes	Ver	Test
R-BOE-SRDD-95	General ASA System latency requirements and related guidance.	Not Allocated to Boeing CGD Software	NA	
R-BOE-SRDD-685	Progress Indicator: The spacing deviation shall be indicated on the progress indicator on a time scale with the center of the scale representing zero spacing error	PI as defined in SDD with progress indication in time mode & commanded by FIM	PB	DT 2
R-BOE-SRDD-686	Progress Indicator: The time scale range shall change to accommodate varying spacing deviation magnitudes.	PI as defined in SDD with PI scale & commanded by FIM.	PB	DT 2
R-BOE-SRDD-1224	The requirements in the ASA MOPS DO-317B / ED-194A §2.3.3, and its sub-sections apply to these MOPS, as applicable to the AIRB application.	General requirement NOT verified. This requirement is met by all lower level display design requirements.	NO	
R-BOE-SRDD-1228	The HMI/CDTI Display Elements Requirements defined in DO-317B / ED-194A, §2.3.4 apply. This section lists the additional HMI/CDTI Display Elements specific to IM, called IM Information Elements.	General requirement NOT verified. This requirement is met by all lower level display design requirements.	NO	

R-BOE-SRDD-1504	When the Distance-based resolution of 0.1 NM	Not allocated to Boeing CGD software as it is controlled by Honeywell FIM	NA	
R-BOE-SRDD-1507	When the Time-based ASG is between 0 and 999 seconds, it shall be displayed to the Flight Crew in seconds, with a resolution of one (1) second.	PI as defined in SDD & commanded by FIM with fine resolution vertical scale. In distance based mode. Resolution accuracy allocated to non-Boeing CGD software	PB	DT 2
R-BOE-SRDD-1544	The HMI/CDTI shall provide the IM Speed Change Visual Advisory to the Flight Crew when IM Speed has changed.	Commanded speed design in SDD & commanded by FIM to change.	PB	DT 3
R-BOE-SRDD-1551	IM Speed resolution in Knots	Not allocated to Boeing CGD software as it is controlled by Honeywell FIM	NA	
R-BOE-SRDD-1554	IM Speed resolution in Mach	Not allocated to Boeing CGD software as it is controlled by Honeywell FIM	NA	
R-BOE-SRDD-1560	In aircraft that do not have automated means to inform the Flight Crew of the safe maneuvering speed.	Not allocated to this test flight due to flight test aircraft selection	NA	
R-BOE-SRDD-1564	The HMI/CDTI shall transition to UNABLE State when the ASA Application Status is set to "Unavailable – Fault" or "Unavailable to Run."	Not allocated to Boeing CGD software as it is controlled by Honeywell FIM	NA	
R-BOE-SRDD-1568	The HMI/CDTI shall transition to UNABLE State when the IM Traffic Application Capability is set to "Invalid."	Not allocated to Boeing CGD software as it is controlled by Honeywell FIM	NA	
R-BOE-SRDD-1584	When the HMI/CDTI is in the OFF/TERMINATE State, the HMI/CDTI shall remove the displayed IM Speed.	Commanded speed is removed as defined in SDD & commanded by FIM. Command to remove on condition is allocated to non-Boeing CGD software	PB	DT 3

R- BOE- SRDD- 1706	When the HMI/CDTI is in the SUSPENDED States, the HMI/CDTI shall remove the following IM Information Elements from display:	Commanded speed is removed as defined in SDD & commanded by FIM. 1. Progress Indicator from all displays 2. IM Speed from CGD Command to remove on condition is allocated to non-Boeing CGD software	PB	DT 2 DT 3
R- BOE- SRDD- 1759	The PFOV shall display the following elements:	Display as defined in SDD the following: 1. Commanded IM Speed 2. Commanded IM Speed Change Indication 3. IM Annunciation a. IM Speed is Limited b. Non-normal IM Termination c. IM Equipment Unqualified d. IFPI conformance indication e. IM designated traffic unqualified f. Commanded IM Speed Conformance Monitoring Status	YC	DT 3 DT 5 DT 3
R- BOE- SRDD- 1760	•Commanded IM Speed	Commanded speed display per SDD	YC	DT 3
R- BOE- SRDD- 1761	•Commanded IM Speed Change Indication	Commanded speed display per SDD Normal, colors and reverse video	YC	DT 3
R- BOE- SRDD- 1762	•IM Annunciation	FIM status message display per SDD	YC	DT 5
R- BOE- SRDD- 1763	•IM Speed is Limited	FIM status message display per SDD & as commanded by FIM.	PB	DT 5
R- BOE- SRDD- 1764	•Non-normal IM Termination	FIM status message display per SDD & as commanded by FIM.	PB	DT 5

R-BOE-SRDD-1765	•IM Equipment Unqualified	FIM status message display per SDD & as commanded by FIM.	PB	DT 5
R-BOE-SRDD-1766	•IFPI conformance indication	Commanded speed display per SDD & as commanded by FIM. Blinking for non-conformance	PB	DT 3
R-BOE-SRDD-1767	•IM Target unqualified	FIM status message display per SDD & as commanded by FIM.	PB	DT 5
R-BOE-SRDD-1768	•Commanded IM Speed Conformance Monitoring Status	Commanded speed display per SDD & as commanded by FIM. Blinking for non-conformance	PB	DT 3
R-BOE-SRDD-1790	When the Assigned Spacing Goal is Distance-based, and when the Designated Traffic reaches the Planned Termination Point, the HMI shall remove the Measured Spacing Interval or Predicted Spacing Interval from display.	Information is removed as defined in SDD & commanded by FIM. Command to remove on condition is allocated to non-Boeing CGD software	PB	DT 0 DT 1 DT 2 DT 3 DT 4 DT 5
R-BOE-SRDD-1796	If the Progress Indicator is displayed textually Distance-based resolution	Not allocated to Boeing CGD software as CGD only displays PI data graphically	NA	
R-BOE-SRDD-1799	If the Progress Indicator is displayed textually Time-based resolution	Not allocated to Boeing CGD software as CGD only displays PI data graphically	NA	
R-BOE-SRDD-1802	If the Progress Indicator is displayed textually Distance-based resolution	Not allocated to Boeing CGD software as CGD only displays PI data graphically	NA	
R-BOE-SRDD-1805	If the Progress Indicator is displayed textually Time-based resolution	Not allocated to Boeing CGD software as CGD only displays PI data graphically	NA	

R-BOE-SRDD-1809	If the Progress Indicator is displayed graphically, during the Maintain Stage, it shall include ASG and MSI.	PI as defined in SDD & commanded by FIM with the features defined in SDD. PI display contents allocated to non-Boeing CGD software	PB	DT 2
R-BOE-SRDD-1812	If the Progress Indicator is displayed graphically, during the Achieve Stage, it shall include ASG and PSI.	PI as defined in SDD & commanded by FIM with the features defined in SDD. PI display contents allocated to non-Boeing CGD software	PB	DT 2
R-BOE-SRDD-1815	The Progress Indicator shall limit the Spacing Deviation Indicator between upper and lower bounds.	PI as defined in SDD & commanded by FIM with the features defined in SDD. PI display contents allocated and located to non-Boeing CGD software	PB	DT 2
R-BOE-SRDD-2113	The requirements defined in ASA MOPS DO-317B/ED-194A §2.3.8 for Flight Deck Integration apply.	Not allocated to Boeing CGD software	NA	
R-BOE-SRDD-2121	The Total Latency, from time of applicability (at interface F) to actual time of display (at interface G), shall be less than 0.5 seconds for the following IM information:	0.25 seconds of latency allocated to Boeing CGD subsystem from CGDServer interface to flight crew view	PB	PT 1 PT 2 PT 3
R-BOE-SRDD-2399	The requirements defined in the ASA MOPS DO-317B/ED-194A §2.3.11 apply.	Color definitions and range for graphics and text as defined in SDD & commanded by FIM. Command to set variable colors is allocated to non-Boeing CGD software	PB	DT 0 DT 1 DT 2 DT 3 DT 4 DT 5
R-BOE-SRDD-2413	The requirements in the ASA MOPS DO-317B / ED-194A §3.1 apply.	Not allocated to Boeing CGD software	NA	
SRS 2.2.1	The CGD shall display in landscape mode using full display capability	CGD display layout is verified in combined display testing	YC	DT 0 DT 1 DT 2 DT 3 DT 4 DT 5
SRS 2.2.2	CGD design shall execute on mobile graphics display device with maximum latency of TBD (i.e., < 0.25) seconds.	0.25 seconds of latency allocated to Boeing CGD subsystem from	YC	PT 1 PT 2 PT 3

		CGDServer interface to flight crew view		
SRS 2.2.3	CGD shall execute at 2 hertz rate.	Demonstrate running at 2 hertz rate CGD rate is commanded by external (Honeywell) message rate and is not allocated to Boeing CGD software	YC	PT 1 PT 2 PT 3
SRS 2.2.4	CGD shall inhibit the display of stale information TBD (i.e., > 5) seconds	CGD software will detect overdue data and lost connections from CGDServer TCP connection to CGD crew viewing Data age is managed by FIM updates and that aspect is externally controlled and is not allocated to Boeing CGD software	YC	CT 2 AT 2 AT 3
SRS 2.2.5	CGD shall receive information distributed by EFB FIM over network interface using message formats per the Boeing Interface Control Document (ICD)	Test CGDServer processing data for CDG displays	YC	DT 0 DT 1 DT 2 DT 3 DT 4 DT 5
SRS 2.2.6	CGD shall display multiple font sizes and colors as needed for visibility and to match similar EFB display formats per SAE ARP 4102 V7.	Color definitions for graphics and text size and formats are as defined in SDD & commanded by FIM. Command to set variable colors is allocated to non-Boeing CGD software	YC	DT 1 DT 2 DT 3 DT 4 DT 5
SRS 2.2.7	CGD shall execute with minimal operator control complexity by automation and simple interface for startup, restart and reset operations.	CGD will handle non-normal operations seamlessly. *System level aspects allocated to CGD System Hardware	PB*	CT 2 AT 2 AT 3
SRS 2.2.8	CGD shall display the following five elements as commanded by EFB FIM and applicable to operational state (i.e., when non EFB FIM connectivity) in format described in the Boeing CGD SDD (SDD): 1. Fast-Slow indicator 2. Progress indicator 3. Commanded speed indicator 4. Current FIM mode indicator	CGD graphics and text drawing and animations as defined in SDD & commanded by FIM.	YC	DT 0 DT 1 DT 2 DT 3 DT 4 DT 5

	5. Status messages.			
SRS 2.2.9	<p>CGD Fast-Slow Indicator shall conform to the SDD definition:</p> <ol style="list-style-type: none"> 1. Labels (FAST and SLOW) shall be fixed, static and displayed when commanded by EFB FIM 2. Not displayed when not commanded. 3. Vertical bar and Expected Speed Profile (ESP) (white bar) shall be fixed and displayed only when commanded by EFB FIM 4. Not displayed when not commanded. 5. The Positive and Negative Speed Differences shall be displayed only when commanded by EFB FIM as depicted with positive above and negative below ESP 6. Empty string or blank is a normal state for either or both speed differences 7. The current vehicle airspeed (VAC -- white triangle) shall be displayed at a vertical level only when commanded by EFB FIM at the commanded location relative to ESP (centered) and upper and lower scale bounds in a color commanded by EFB FIM. 	Fast-slow graphics and text drawing and animations as defined in SDD & commanded by FIM.	YC	DT 1
SRS 2.2.10	<p>CGD Progress Indicator shall conform to the following definition: Layout shall be as described in SDD.</p> <ol style="list-style-type: none"> 1. Labels (EARLY and LATE) shall be displayed above and below the vertical bar, to 	Progress indicator graphics and text drawing and animations as defined in SDD & commanded by FIM.	YC	DT 2

	<p>duplicate the EFB FIM display with (EARLY and LATE) as default when commanded by EFB FIM</p> <ol style="list-style-type: none"> 2. Not displayed when not commanded. 3. Vertical bar and tick marks shall be fixed and displayed only when commanded by the EFB FIM. 4. The progress symbol (caliper) shall be displayed at a vertical level only when commanded by EFB FIM relative to center tick mark and upper and lower scale bounds as commanded by EFB FIM with at least 250 pixels of resolution. 5. Progress symbol (caliper) shall be set to (BIG or SMALL) to scale as commanded by EFB FIM with BIG as the default. 6. The progress symbol (caliper) shall be displayed in a color commanded by the EFB FIM 			
<p>SRS 2.2.11</p>	<p>CGD Commanded Speed Indicator shall conform to the definition in the SDD:</p> <ol style="list-style-type: none"> 1. Commanded speed shall be displayed as text as commanded by EFB FIM only when commanded by EFB FIM. 2. Commanded speed shall be displayed in color as commanded by EFB FIM (white). 3. Commanded speed shall be displayed as text as commanded by EFB FIM as reverse video color only 	<p>Commanded speed graphics and text drawing and animations as defined in SDD & commanded by FIM.</p>	<p>YC</p>	<p>DT 3</p>

	<p>when commanded by EFB FIM as a speed change.</p> <p>4. Commanded speed shall be displayed as text as commanded by EFB FIM as alternating normal and reverse video at one hertz only when commanded by EFB FIM for speed non-conformance.</p>			
SRS 2.2.12	<p>CGD Current Mode Indicator shall conform to the SDD definition:</p> <ol style="list-style-type: none"> 1. Current mode shall be displayed as text as commanded by EFB FIM only when commanded by EFB FIM. 2. Current mode shall be displayed in color as commanded by EFB FIM (white). 	FIM mode text drawing as defined in SDD & commanded by FIM.	YC	DT 4
SRS 2.2.13	<p>CGD Status Message Indicator shall conform to the SDD definition:</p> <ol style="list-style-type: none"> 1. Status message shall be displayed as text as commanded by EFB FIM only when commanded by EFB FIM. 2. Status message shall display zero to three lines of status messages as commanded by EFB FIM. 	FIM status messages text drawing as defined in SDD & commanded by FIM.	YC	DT 5

Revision Record

Revision Letter**A****Changes in this Revision**

Updates made throughout this document reflect design changes due to HMI and system architecture updates throughout the development process.

Signatures

AUTHOR:	Signature on File	9M-PW-ERCS	12/6/16
	George F. (Rick) Wilber	Org. Number	Date
APPROVAL:	Signature on File	9M-PW-ERCS	12/6/16
	Karl Rein-Weston	Org. Number	Date
DOCUMENT RELEASE:	Signature on File	9M-PW-ERCS	12/6/16
	Barbara L. Withers	Org. Number	Date

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE (DD-MM-YYYY) 01- 03- 2017	2. REPORT TYPE Contractor Report	3. DATES COVERED (From - To)
--	--	-------------------------------------

4. TITLE AND SUBTITLE ATD-1 Phase II: Boeing Configurable Graphical Display (CGD) SDD	5a. CONTRACT NUMBER NNL13AA003B
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S) Wilber, George	5d. PROJECT NUMBER
	5e. TASK NUMBER NNL15AB46T
	5f. WORK UNIT NUMBER 330693.04.10.07.09

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) NASA Langley Research Center Hampton, VA 23681-2199	8. PERFORMING ORGANIZATION REPORT NUMBER
---	---

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001	10. SPONSOR/MONITOR'S ACRONYM(S) NASA
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) NASA-CR-2017-219594

12. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 03
Availability: NASA STI Program (757) 864-9658

13. SUPPLEMENTARY NOTES
Langley Technical Monitor: Denise K. Scarce

14. ABSTRACT

This Software Description Document (SDD) captures the design for developing the Flight Interval Management (FIM) system Configurable Graphics Display (CGD) software. Specifically this SDD describes aspects of the Boeing CGD software and the surrounding context and interfaces. It does not describe the Honeywell components of the CGD system. The SDD provides the system overview, architectural design, and detailed design with all the necessary information to implement the Boeing components of the CGD software and integrate them into the CGD subsystem within the larger FIM system. Overall system and CGD system-level requirements are derived from the CGD SRS (in turn derived from the Boeing System Requirements Design Document (SRDD)). Display and look-and-feel requirements are derived from Human Machine Interface (HMI) design documents and working group recommendations. This Boeing CGD SDD is required to support the upcoming Critical Design Review (CDR).

15. SUBJECT TERMS

ATD-1; Airspace management; Aviation; Software design description

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email: help@sti.nasa.gov)
U	U	U	UU	64	19b. TELEPHONE NUMBER (Include area code) (757) 864-9658