Document Title:

G3.5 Generator Software Architecture and Design Description

Document Number:

100-00040-013

Revision:

B

Release Date:

11 Mar 2019

For:

Gala Therapeutics
155 Jefferson Dr., Suite 100
Menlo Park, CA 94025

RBC Medical Innovations
13715 W. 109th Street
Lenexa, KS 66215

Baseline 100-00040-013 RevB Exported from Jama Software by Dharmendra Kallur on 11 Mar 2019 under report “RBC Default Word Export”.

1 Introduction

1.1 Purpose

This document provides details as to how the Gala Control Processor CSCI and Display CSCI are implemented. The intended audience include software engineers, test engineers, and system engineers.

This document is responsible for the following:

- Formally document the software architecture including:
    - An overview of the software including its architectural drivers (e.g., quality requirements)
    - An overview of the overall architecture of the software
    - Relevant views of the software's architecture
    - How the software architecture will support the achievement of the architectural drivers
    - Ancillary information about the software architecture
- Improve stakeholder understanding of (and communication concerning) the most important, pervasive, top-level, strategic decisions and inventions concerning.
    - The overall software structure in terms of its major component elements and their relationships.
    - How these architectural elements will collaborate to provide the required characteristics and behavior (i.e., fulfill the architecturally-significant requirements and constraints).
- Provide a foundation upon which to:
    - Base the design, implementation, integration testing, deployment, and maintenance of the software.
    - Estimate the software's size.
- Support the analysis and verification of the software architecture.
- Support the reuse of the architecture on other software.
- Improve extensibility by providing a flexible foundation on which to extend the software as requirements change.
- Minimize the risk that necessary architectural decisions and inventions are either not made or made incorrectly.

- Ensure that the rationales (e.g., engineering tradeoffs) for architectural decisions are not lost, and therefore it lowers the risk that rejected architectural decisions will one day be revisited and selected after the original architects are gone.

1.2 Scope

The scope of this document encompasses the software utilized in the MK60FN1M0VLQ15 microcontroller and the Reach SLCD5+ plus for the Gala G3.5 Generator.

1.3 Revision History

Rev	**Author**	**Description**	**ECO Number**
1	D. Kallur	Initial Draft	N/A
A	D. Kallur	Initial Release	18-00261
B	D. Kallur	Replaced alarms and errors with fault	19-00051

1.4 Definitions

Phrase/Acronym	**Definition**
IFU	Instructions for Use
SLCD	Super Liquid Crystal Display
SOP	Standard Operating Procedure
PWM	Pulse Width Modulation
LED	Light Emitting Diode
IO	Input/Output- in respect to general purpose uses
CPU	Central Processing Unit
Functional unit	Smallest granularity of design. Functional units exist as part of a module.
ISR	Interrupt service routine.
Module	A group of code executing all or part of a task.
Process	An executing instance of an application that runs simultaneously with (or pseudo-

	simultaneously) with other processes in a multi-processing system. Each process operates within its own address space.
Task	A task is allocated to the SW by the system design. It is something the software has to do to meet the system requirements.
Thread	A unit of code that runs simultaneously with (or pseudo-simultaneously) with other threads in a multi-threading system. Threads execute within a process. Threads within a process share memory and operate within the process's address space.

2 Document References

SDP p/n: 100-00040-003, rev: See DHF	Gala G3.5 Generator Software Development Plan
PRD p/n: DCD006, rev: See DHF	Gala G3.5 Generator Product Requirements Document
GOT p/n: 100-00040-006, rev: See DHF	Gala G3.5 Generator Glossary of Terms
GHA p/n: 100-00040-009, rev: See DHF	Gala G3.5 Generator Hazard Analysis
Level of Concern and Software Safety Classification, Gala Generator, G3.5 p/n: RSK015, rev: See DHF	Level of Concern and Software Safety Classification, Gala Generator, G3.5
Software Build Procedure p/n: 100-00040-017, rev: See DHF	Software Build Procedure

3 SOFTWARE ARCHITECTURE

3.1 FEATURES

G3_5-DS-77

Intended Use

The generator, as configured within the Gala System is intended to provide a safe and effective means of delivering controlled pulsed electric field (PEF) energy output to the Gala RheOx Catheter (catheter) to achieve the intended function of the Gala System.

The Gala Generator software implements the following functionality:

- RF energy output regulation and control
- Fault monitoring
- User interface
- Calibration

RF Energy Output Regulation and Control

The software incorporates a means to deliver electrosurgical outputs to perform ablation and coagulation. The activation is executed using communication to the FPGA and drive logic, the commencement and cessation of which is controlled by the Main Controller's processing of the user interface and foot switch.

Fault Monitoring
The software incorporates monitoring of the system for suboptimal operating conditions. Faults are indicated to the user via the user interface.

User Interface
The software incorporates five methods of interfacing with the user:

- A speaker provides the user with audible feedback of the system's current state.
- A Super Liquid Crystal Display provides the user with visual feedback of the system's current state.
- A touch screen provides the user with the ability to acknowledge a message displayed on the screen.
- A foot switch provides the user with the ability to activate PEF energy.

Calibration

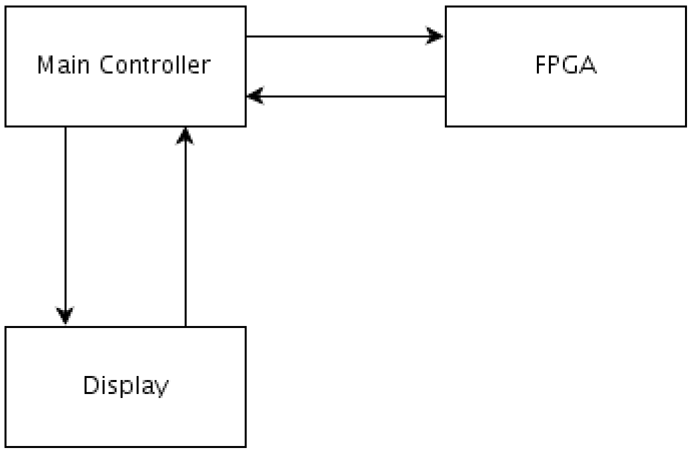
The software incorporates a method to calibrate RF output and feedback.

Rationale:

3.2 Architecture

G3_5-DS-78

The Gala Generator employs a combined microcontroller and FPGA architecture with the FPGA for control of output, and the microcontroller for user interaction, and fault indicating and handling.



The Main microcontroller (Main Controller), located on the Controller PCBA, processes inputs from the user interface facilitating RF output.
The FPGA, a third party assembly, located on the Main PCBA, provides interface to several circuits including the generation of the RF waveforms.
The SLCD, a third party assembly, provides interfaces to the Super Liquid Crystal Display and the touch screen.

RF Output
Information to control the power is sent from the Main Controller to the FPGA through a serial interface and GPIO input. The software contained in the Controller controls the output power using serial communication to a DAC which generates a reference analog voltage to control what voltage comes out of a pair of Ultravolt transformers. The FPGA is responsible for creating the drive waveforms for the different settings. As a result, when the Main Controller receives a request to deliver PEF energy through foot switch, messages are sent to the FPGA which generate the proper waveform.

User Interaction
The Main Controller interfaces with the user through the user interface. This is comprised of front panel settings through the touch screen on the LCD display, foot switch input and numerous internal function settings to properly control settings and state adjustment. Signals from the user interface are presented to the Main Controller for processing. Information to activate RF output, Warning and Fault conditions and system start up contain an audio component that is sent via a PWM signal with specific frequencies to a speaker to provide the user with an indication of the system's current state. Information to activate RF output, Warning and Fault conditions, and system start up is sent via a serial signal to the SLCD to provide the user with an indication of the system's current state. Information to activate RF output is initiated from the foot switch to the Main Controller.

Fault Indicating and Handling
Information regarding a sub-optimal condition within the system evokes a fault condition. The fault condition contains an audible, visual and system response component. The audible component is indicated to the user through the speaker directly from the Main Controller. The visual component is indicated to the user through the SLCD from the Main Controller. In the case of the FPGA alerting the Main Controller to the condition, there is a cessation of output power. In the case of the Main Controller internally alerting itself to the condition, an internal fault condition is flagged.

Rationale:

4 Controller Software Design

4.1 Level of Concern

4.1.1 Level of Concern

G3_5-DS-1

Software Safety Classification Per IEC 62304:2006 Edition First, Medical device software -- Software Life Cycle Processes and Software Level of Concern Per FDA guidance (Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices issued May 11, 2005) is recorded in Risk Management File RSK015.

Rationale:

4.2 Processor Specifications

4.2.1 OS

G3_5-DS-2

This embedded system uses no general purpose operating system (OS).

Rationale:

4.2.2 Volatile Memory (RAM)

G3_5-DS-3

The MK60FN1M0VLQ15 has 128KB of RAM.

The on-chip RAM is split in two regions: SRAM_L and SRAM_U. The RAM is implemented such that the SRAM_L and SRAM_U ranges form a contiguous block in the memory map.

As such:

- SRAM_L is anchored at address 0x2000_0000 and occupies the space below this address, that is, addresses < 0x2000_0000 (0x1FFF_0000 – 0x1FFF_FFFF)
- SRAM_U is anchored at address 0x2000_0000 and occupies the space at and above this beginning address, that is, addresses ≥ 0x2000_0000 (0x2000_0000 – 0x2000_FFFF)

Rationale:

4.2.3 Non-Volatile Storage

G3_5-DS-4

The MK60FN1M0VLQ15 has 1024KB of Flash. No off-chip storage is used.

Rationale:

4.2.4 Boot Sequence

G3_5-DS-5

The code executions starts in file startup_MK60F12.s with PreInitialization function being the first call. This is followed by executing the RAM test, initializing peripherals, and then calling the Main function.

User inputs are ignored during boot up sequence.

Rationale:

4.2.5 Power On Self-Test (POST)

G3_5-DS-6

The following POST checks are performed to ensure correct device operation

- RAM check
- Register check
- ROM check
- Ultravolt HV supply check
- H-Bridge check
- Measurement circuit check

Detailed explanation of the above is provided in the POST Module (G3_5-DS-38)

Rationale:

4.2.6 Safety Mitigations

G3_5-DS-7

An internal Watchdog timer is enabled to ensure that the processor is operating properly. Under normal operations, the watchdog timer is continuously strobed in the main loop after all the tasks finish running.

If the watchdog timer is not strobed for more than 300 ms, the watchdog timer will detect this as a fault and reset the processor.

On each boot-up, the internal registers are monitored to check if the reset was performed by the watchdog timer. If the reset source was the watchdog timer, the system enters fault state.

Other POST checks are explained in POST module (G3_5-DS-38).

Rationale:

4.2.7 Threads

G3_5-DS-8

The control processor does not use threads anywhere. All modules mentioned within this document operate within the same thread in a super-loop. All modules run to completion.

Rationale:

4.2.8 Interrupt Service Routines (ISRs)

G3_5-DS-9

The Controller software implements the following ISRs:

ISR Type	Module	Details
GPIO	Heart Sync Manager	Gets interrupt on the rising edge of cardiac signal
GPIO	FPGA Manager	Gets interrupt on the rising edge when overcurrent is detected by the FPGA
UART0	FPGA Manager	Gets interrupt upon receiving data from FPGA
UART3	Display Manager	Gets interrupt upon receiving data from SLCD
UART4	Debug Manager	Gets interrupt upon receiving data from Debug port (PC)
GPIO	ADC Manager	Gets interrupt on the falling edge after data is received on the SPI port. The GPIO is controlled by FPGA

All the interrupts are assigned with same priority level.

Rationale:

4.2.9 Updating Software

G3_5-DS-10

The software of both Controller and UI cannot be updated while the software is running.

Software has to be installed or updated as explained in the Gala Generator G3.5 Software Build Procedure (100-00040-017).

Rationale:

4.3 MODULES

4.3.1 Main Program Loop

G3_5-DS-84
Rationale:

4.3.1.1 Mode of Operation

G3_5-DS-11

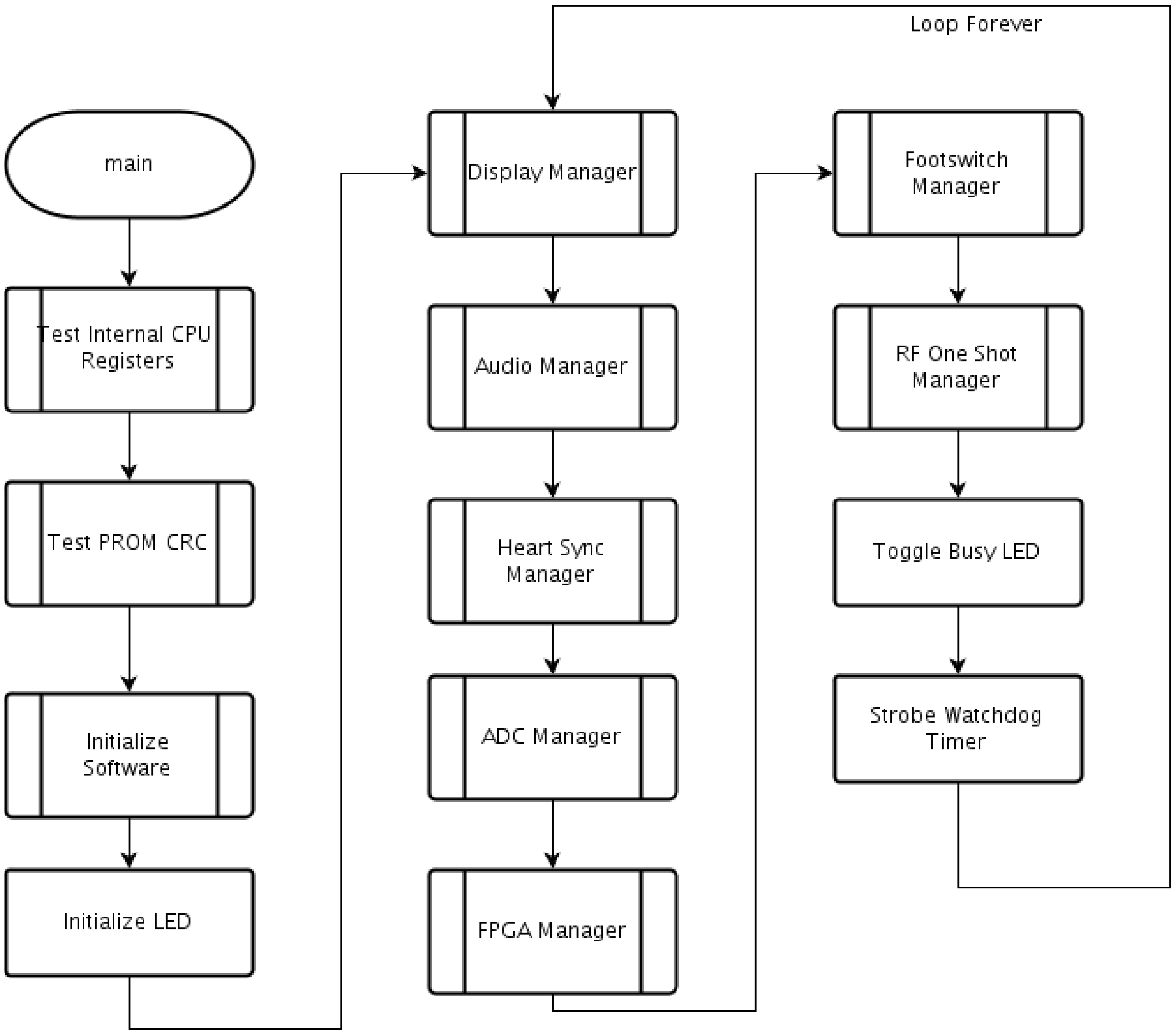
The driver loop for the Main Controller is contained in the function named main.

After entering main, the CPU internal registers are checked. If a fault is detected, the device will not progress beyond to initialization. If it's successful the PROM CRC is checked to test the integrity of the PROM. The results of the check is saved in static memory and is later checked during POST check in Display Manager.

Hardware and software will be initialized. The single main task loop that runs continuously to perform all system operations will run forever, strobing the watchdog at the beginning of each cycle. After all system operations have been executed for a cycle of the main task loop, the CPU busy LED is initialized for toggling.

The CPU busy LED is toggled On for 25 milliseconds and then off for 750 milliseconds, to indicate that the main task loop is running.

In addition, the Main Controller will allow three types of interrupts, to process SPI communications, UART communications, and IO communications.



Main Task Loop Flow Diagram

Rationale:

4.3.1.2 Input / Output

G3_5-DS-12

Main executes the functions in Display, Audio, Heart Sync, ADC, FPGA, Footswitch and RF One Shot manager. At the completion of the Main loop, the WDT is strobed and the BUSY LED is toggled.

Rationale:

4.3.1.3 Functional Units

G3_5-DS-28

Function	**Description**
main	Main task loop
static void RunningLed (BOOLEAN initialize)	Flash CPU running led on PCB and Strobe WDT

Rationale:

4.3.2 Display Module

G3_5-DS-22

Rationale:

4.3.2.1 Mode of Operation

G3_5-DS-98

Display module is responsible for handling the UI displayed on SLCD5 Plus display. SLCD5 Plus displays the current state of the device and also warning and fault conditions.

Display module communicates with SLCD5 Plus display through UART3 port. It communicates at 115200 baud rate, one stop bit and zero parity bits. UART receive interrupt is enabled and on receiving data from SLCD, the interrupt service routine stores the data in receive buffer.

The receive buffer is processed in the function that is called in the main loop.

On power on, SLCD5 displays splash screen. When the splash screen is displayed, all the POST checks are performed. RAM and ROM check failure will result in screen remaining in Splash screen. Any other POST failure will result in the display module entering Fault mode and displaying fault screen.

Fault code and text displayed on fault screen is shown in the table below.

Fault Description Table

Code	**Fault Description**	**Fault Description**	**Recommended Action**
F1	Watchdog timer fault	Controller reset from the Watchdog timer	Turn off unit and turn back on. Contact Gala Therapeutics if message reappears.
F2	RAM test fault	RAM test failure	Remains in Splash screen

F3	SW image corrupt	PROM CRC check failure	Remains in Splash screen
F4	Discharge fault	Capacitors are not discharged completely	Turn off unit and turn back on. Contact Gala Therapeutics if message reappears.
F5	Charge fault	Capacitors are not charging	
F6	Internal fault	Bank A top stuck	
F7	Internal fault	Bank A bottom stuck	
F8	Internal fault	Bank B top stuck	
F9	Internal fault	Bank B bottom stuck	
F10	Stuck foot switch	Stuck footswitch during POST	Ensure foot switch is not pressed. Turn off unit and turn back on. Contact Gala Therapeutics if message reappears.
F11	Calibration data corrupt	Calibration data corrupted	Turn off unit and turn back on. Contact Gala Therapeutics if message reappears.
F12	Self-test fault	Calibration load check failure.	
F13	Flash driver fault	Flash driver initialization failure. (Can only occur in TEST mode)	
F14	Measurement packet fault	Measurement packet V or I out of limit	
F15	Therapy packet fault	Therapy packet V or I out of limit	

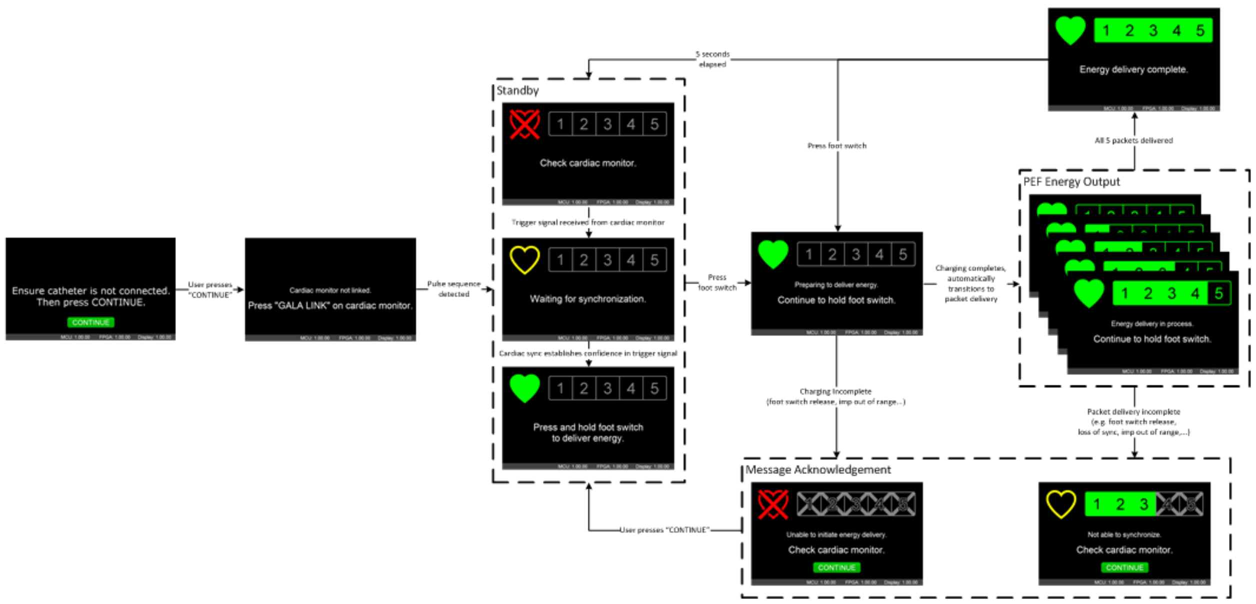
If POST is successful, the module enters Calibration self test mode. Here the module waits for users input through the touch screen. On receiving users input, RF calibration pulse is initiated to test the calibration load.

On success the module enters Standby state. Standby screen displays the current Heart state, Packet counter and a message. The heart state information is provided by the Heart sync module.

The packet counter is provided by the RF One Shot manager. The message displayed is driven by Display module, Heart sync module and RF One shot manager.

The screen output information is provided by RF One Shot manager during therapy delivery.

The operational screen work flow is shown below


Ensure catheter is not connected. Then press CONTINUE.
CONTINUE
User presses "CONTINUE"
Cardiac monitor not linked.
Press "GALA LINK" on cardiac monitor.
Pulse sequence detected
Standby
Check cardiac monitor.
Trigger signal received from cardiac monitor
Waiting for synchronization.
Cardiac sync establishes confidence in trigger signal
Press and hold foot switch to deliver energy.
5 seconds elapsed
Press foot switch
Press foot switch
Preparing to deliver energy.
Continue to hold foot switch.
Charging completes, automatically transitions to packet delivery
PEF Energy Output
Energy delivery in process.
Continue to hold foot switch.
All 5 packets delivered
Energy delivery complete.
Charging Incomplete (foot switch release, imp out of range...)
Packet delivery incomplete (e.g. foot switch release, loss of sync, imp out of range,...)
Message Acknowledgement
Unable to initiate energy delivery.
Check cardiac monitor.
CONTINUE
Not able to synchronize.
Check cardiac monitor.
CONTINUE
User presses "CONTINUE"
MCU 1.00.00 FPGA 1.00.00 Display 1.00.00

Operational Screen Work Flow

Rationale:

4.3.2.2 Input / Output

G3_5-DS-23

Display modules gets its input for POST from POST module, Calibration and PEF delivery inputs and faults from RF One shot manager. Touch input from SLCD+ though UART port.

Display module outputs to the SLCD+ display through the UART port.

Rationale:

4.3.2.3 Functional Units

G3_5-DS-29

External Interface	**Description**
void InitDisplay (void);	Function to initialize front panel display serial interface
void DisplayManager (void);	Manage front panel display interface

enum RFState GetRFState (void);	Return RF state
void SetRFState (enum RFState);	Set RF State
BOOLEAN DisplayReady (void);	Return status of display state
BOOLEAN IsDisplayCardiacSync (void);	Return status of display state
BOOLEAN IsDisplayCalCheck (void);	Return status of display state
void ISRDisplayRX (void);	Interrupt service routine for the front panel display
enum HeartGraphic GetHeartState(void);	Gives current heart color
void ShowRemainingPackets (BOOLEAN, UInt8);	Update front panel display to remaining packets
void ShowSingleMessage(const Int8 * messageParam);	Update front panel display with the given message.
void ShowDoubleMessage(Int8 * messageParam1, Int8 * messageParam2);	Update front panel display with the given message.
BOOLEAN DisplayBusy (void);	Returns true if the display has bytes left to transmit
void SetFinishedMessageDisplay (BOOLEAN finishMessage);	Setter for the finished message being displayed and set timer for message to be cleared
BOOLEAN GetFinishedMessageDisplay(void);	Getter for the finished message being displayed and set timer for message to be cleared
void SetMessageAckFlag (BOOLEAN state);	Updates the flag that holds the state of Message Acknowledgement
BOOLEAN GetMessageAckFlag (void);	Fetches the state of Message Acknowledgement flag
void ShowTerminatedPackets (UInt8 count);	Update front panel display to show terminated packets
void ShowDeliveryNotStarted(void);	Update front panel display to show delivery has not started
void DisplayMessage(Int8 * messageParam1, Int8 * messageParam2);	Updates screen with either One or Two message with a continue button
void DisplayContinueButton(void);	Updates screen with a continue button
void DisplayHeart(enum HeartGraphic heartGraphic);	Update front panel display to show heart state
void EnterFaultScreen(Fault_Type_t faultType);	Function to enter fault screen state and display appropriate fault type

Rationale:

4.3.3 Heart Sync Module

G3_5-DS-30

Rationale:

4.3.3.1 Mode of Operation

G3_5-DS-99

Heart sync module is responsible for processing the heart beat for heart stability.

HeartSyncManager function is called in the main loop. Each time the loop starts, it checks to see if there was an interrupt flag from the TTL interrupt line. If the flag is set, the loop resets the interrupt flag and processes the heart beat for heart stability information. Next, it checks the heart state, and updates sends information to the SLCD accordingly with a red solid heart, yellow or green blinking heart depending on the state.

The interrupt is triggered on the rising edge of Cardiac Sync line.

Heart sync module is also responsible for verifying the Cardiac monitor during Cardiac monitor state. Cardiac monitor is verified after detecting five trigger inputs from the cardiac monitor within a 45msec time period

Rationale:

4.3.3.2 Input / Output

G3_5-DS-31

Input to this module come from Cardiac Sync line connected to the GPIO pin with ISR enabled.

The module output the heart state (Red Heart, Yellow and Green) to the UI display.

Rationale:

4.3.3.3 Functional Units

G3_5-DS-33

External Interface	**Description**
void InitHeartSync (void);	Initialize heart sync hardware interface
void HeartSyncManager (void);	Handles heart beat input
BOOLEAN CardiacMonitorVerified(void);	Return Cardic monitor verification status

BOOLEAN HeartStable (void);	Return TRUE if heart beat is stable
UInt32 GetLastMarkTime (void);	Return value of high speed timer when last mark was recorded
UInt32 GetLastValidMarkTime (void);	Return value of high speed timer when last "valid" mark was recorded
UInt8 GetHeartBeatCount (void);	Return heart beat counter
BOOLEAN HeartDetected (void);	Return TRUE if any heart beat is detected
void SkipCardiacMonitorCheck(void);	Skips Cardiac Monitor Check
void SetLastPacketMarkTime (void);	Sets the lastPacketMarkTime with lastMarkTime to use it for blanking period

Rationale:

4.3.4 RF One Shot Manager

G3_5-DS-87
Rationale:

4.3.4.1 Mode of Operation

G3_5-DS-34

The driver loop for the RF one shot manager is contained in the function named rfoneshot.
Each time through the loop, it checks the state of RF delivery and proceeds with the logic related to preparing, firing or finishing the RF waveform once ready. Inside this loop, audio indicators are performed for the different states, charging, ready to fire, and firing. This loop is tasked with the control of the FPGA fire state and the Ultravolt voltage levels dependent on user input from the SLCD and foot switch.

RF One Shot manager sends the PEF packet settings to FPGA through UART port.

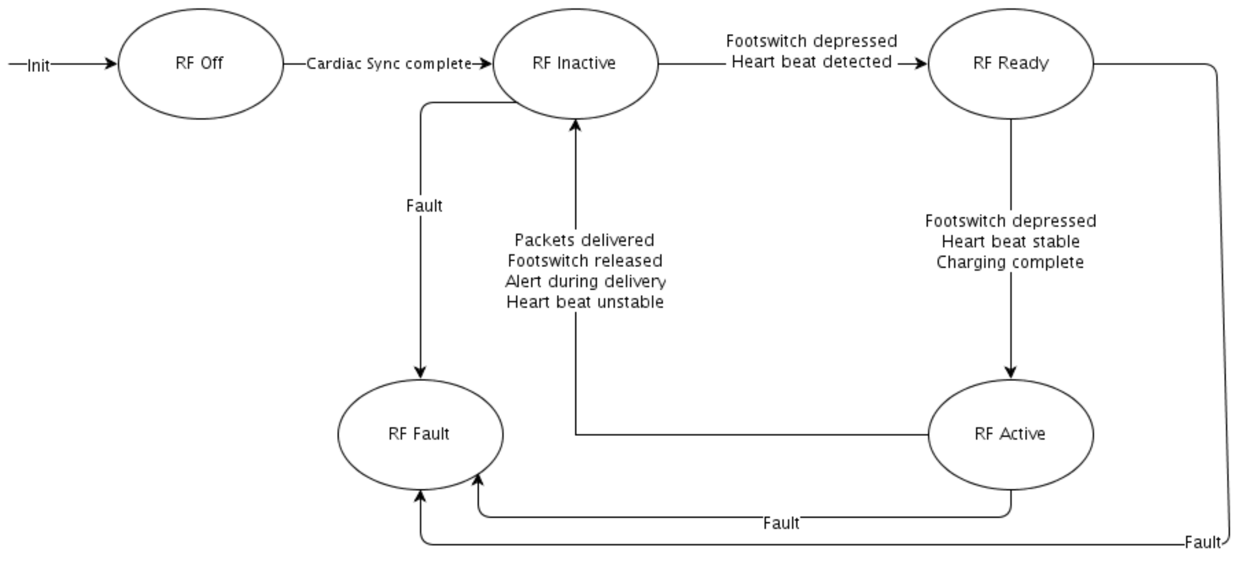
Summary of Pulsed Electric Field Voltage Output Waveform Parameters

Waveform Parameter	**Value**
Frequency (kHz)	600
+Vout = |-Vout| (V)	2,500
Cycle Count	60
Packet Duration (µsec)	100
Packet Count/Max Number of Packets per Activation	5

RF One Shot Manager has 5 major states

- RF Off
  - On power on, the manager starts with RF Off state where no RF delivery is performed. On successful POST completion, Calibration check and Cardiac sync completion the module enters RF Inactive state.
  - During Calibration Check a low voltage PEF energy is delivered across an internal calibration resistor to perform a calibration check. If the voltage, current and impedance is not within the limits, Fault state is entered.
- RF Inactive
  - During this state the manager monitors the Foot switch and Cardiac state. If the foot switch is depressed and cardiac heart beat is detected then the manager enters RF Ready state.
- RF Ready
  - In this state the manager commands the DAC to charge the capacitors to either 100 V if it's measure pulse else to 2500 V equivalent voltage based on impedance measured if it's a Therapy pulse.
    The audio module is commanded to output Charging tone. The manager remains in this state for a minimum of 2 seconds to make sure the capacitors are completely charged to deliver therapy pulse.
    After charging is complete, the module enters RF Active state.
- RF Active
  - In this state the manager commands the FPGA to deliver the PEF packet based on the cardiac input. PEF packet is delivered after 50 ms of cardiac input and prior to 55 ms. The display manager is commanded to update the packet counter after each PEF packet. The Audio manager is commanded to output period treatment tone once every second. During this state the packets voltage, current and impedance are also monitored. If the measured values are out of limit, based on the type of error either Temporary messaging screen or Fault screen is displayed.
- RF Fault
  - RF Fault state is entered anytime there are any faults in the system. No RF delivery will be done once in this state. RF Manager would remain in this state forever and would require a power reset of the system to reenter RF Off state.

The image below provides a brief description of state flow between the RF states.



Rationale:

4.3.4.2 Input / Output

G3_5-DS-35

RF One Shot manager gets its inputs from Display manager, Foot switch manager, Heart Sync module, ADC manager, Calibration module. It outputs to Ultravolt manager, FPGA manager, Audio manager and Display module.

Rationale:

4.3.4.3 Functional Units

G3_5-DS-37

External Interface	**Description**
BOOLEAN CalibrationSelfTest(void);	Function to perform calibration self test
BOOLEAN GetCalibrationSelfTestResult(void);	Fetches calibration self test result
void RFOneShotManager (void);	Function to prepare and deliver one RF shot
BOOLEAN RfDeliveredOnLastMark(UInt8);	Given a beat number, the function returns whether or not the input beat number is the same as the beat number of the last RF packet was delivered on. This function is used for the "Fast Recovery" portion of cardiac sync.
void SetRFDeliveryType (RFDeliveryType_t	Range check input parameter and set rf delivery type

deliveryType);	
RFDeliveryType_t GetRFDeliveryType (void);	Return RF delivery type
void EnableSkipCalCheck(void);	Enables skipCalCheck flag
UInt8 GetCurrentPacketCount(void);	Provides the current packet number delivered
Fault_Type_t GetRFFaultType(void);	Returns fault type
UInt16 RF_GetVoltageGainCal(void);	Fetch Voltage gain calibration value
void RF_SetVoltageGainCal(UInt16 voltGainCal);	Set Voltage gain calibration value
UInt16 RF_GetCalVoltageGainCal(void);	Fetch Calibration check Voltage gain calibration value
void RF_SetCalVoltageGainCal(UInt16 calVoltGainCal);	Set calibration check Voltage gain calibration value

Rationale:

4.3.5 POST Module

G3_5-DS-88
Rationale:

4.3.5.1 Mode of Operation

G3_5-DS-38

Power-On Self-Test is perform as soon as the unit is powered On to check the sanity of the system.

Checks mentioned in the table below are performed as a part of POST check.

In case of failure, the appropriate fault code is sent to the Display module to display on the screen.

Test	Description	Failure (Fault displayed)
Initialize WDT	Initialize Watchdog timer. Timeout period set to 300ms. If the Watchdog is not strobed for more than 300ms after initialization, WDT will reset the processor. After the reboot, the processor enters fault state.	F1 Watchdog timer fault
RAM Check	RAM check performed. Will stay here if there is a failure	Stays in Splash screen
Register	R0 to R12 registers checked	Stays in Splash

Check		screen
ROM check (CRC)	CRC of program ROM is calculated and checked against the value stored. If there is a mismatch, a fault is raised.	F3 SW image corrupt
Cap Discharge Check	Checks if the capacitor voltage is above 80V. If it is then a fault is raised.	F4 Discharge fault
Cap Charge Check	Ultravolt HVPS are command to 100V. 150ms wait time before polling voltage from Ultravolt. If UltraVolt voltage (VMON_A and VMON_B) is less than 80V, a fault is raised.	F5 Charge fault
Bank A, top on, bottom off	FPGA is command to turn +Ph1Drv On and rest is turned off. HVCap is monitored and if voltage read is less than 80V then fault is raised.	F6 Internal fault Bank A bottom stuck
Bank A, top On, bottom On	FPGA is command to turn +Ph1Drv and –Ph2Drv On and rest is turned off. HVCap is monitored and if voltage read is greater than 50V then fault is raised.	F7 Internal fault Bank A top stuck
Bank B, top on, bottom off	FPGA is command to turn +Ph2Drv On and rest is turned off. HVCap is monitored and if voltage read is less than 80V then fault is raised.	F8 Internal fault Bank B bottom stuck
Bank B, top On, bottom On	FPGA is command to turn +Ph2Drv and –Ph1Drv On and rest is turned off. HVCap is monitored and if voltage read is greater than 50V then fault is raised.	F9 Internal fault Bank B top stuck
Foot switch check	If foot switch is pressed during POST then a fault is raised	F10 Stuck foot switch
Calibration data check	Calibration values are stored along with its compliment in program memory during system checkout. If the stored calibration values are corrupted, then fault is raised.	F11 Calibration data corrupt
Calibration load self-test	100V voltage pulse is put across the 200Ω load calibration resistor. If the resistance measured is less than 190Ω or greater than 210Ω a fault is raised. If the impedance is within limits then the Voltage and Current	F12 Self-Test fault

	measured is checked against the limits. Measured Voltage should be 94V ≥ Vmeas ≤ 106V Measured current should be 0.47A ≥ Imeas ≤ 0.53A	
Flash driver fault (Only in Test mode)	Flash driver failures are checked during test mode. Flash drivers are not enabled/Initialized during normal operations.	F13 Flash driver fault
Measurement packet fault	If measure pulse voltage or current is outside limits, then a fault is raised. 50V ≥ Vmeas ≤ 140V 0.1A ≥ Imeas ≤ 5A	F14 Measurement packet fault
Therapy packet fault	If therapy pulse voltage outside limits, then a fault is raised. The following conditions should satisfy to raise the alert The impedance of each PEF packet is within the range of 50 to 450 ohms, the difference in impedance between each PEF packet and the low voltage PEF packet (ref DCD006-0041) is less than 10%. If the average measured voltage amplitude of the 5 delivered PEF packet differs from the commanded amplitude by more than +10% / -5% an alert is raised.	F15 Therapy packet fault

Rationale:

4.3.5.2 Input / Output

G3_5-DS-39

POST is initiated after the processor boots up. It gets its input information from processor registers, CRC module and ADC module.

POST outputs the status to the Display module.

Rationale:

4.3.5.3 Functional Units

G3_5-DS-41

External Interface	Description

BOOLEAN PostCheck (void);	Run POST with FPGA drive to test drive voltage
BOOLEAN PostResults (void);	Return post results
Fault_Type_t GetErrorType(void);	Return Post error type

Rationale:

4.3.6 FPGA manager

G3_5-DS-89
Rationale:

4.3.6.1 Mode of Operation

G3_5-DS-42

The FPGA's contains all the logic for communication with the controller card and controlling the sequencing and timing of the switches. The FPGA used first contains logic for serial communication, and second, contains the logic to run the timing and switching sequences specified by the user.

Communication is through a RS485 interface at 115,200 baud. All data transferred is with ASCII characters. Binary data is transferred with hexadecimal characters. The protocol for send data from the controller to the FPGA is as follows:

- Start of packet "@"
- Letter representing what command to use
- Data for the command sent – (5 bytes)
- CRC for the data – (4 bytes)
- End of packet "#"

The data sent to the FPGA is echoed back followed by the following sequence:

- Start of packet "@"
- ACK or NACK
- CRC for ACK or NACK (6 bytes)
- Software version number (2 bytes)
- End of packet "#"

FPGA Serial Communications

- "B" – Send waveform frequency setting
- "C" – Send waveform rest period setting
- "D" – Send waveform count cycle setting
- "E" – Send waveform dead time setting

- “F” – Send waveform packet count setting
- "S” – Send POST command information

FPGA LED Configuration

The FPGA has 4 on-board LEDs, each with its own useful information to display.
LEDs:
0 – FPGA heartbeat
1 – Safety triggered
2 – RX or TX is active
3 – Waveform is running

Rationale:

4.3.6.2 Input / Output

G3_5-DS-43

FPGA manager gets its inputs from the POST module for the POST check and from the RF One Shot Manager for calibration and therapy packets.

FPGA manager outputs to the FPGA through the UART port.

Rationale:

4.3.6.3 Functional Units

G3_5-DS-45

External Interface	**Description**
void InitFPGA (void);	Initialize FPGA serial interface
void FPGAManager (void);	Manage messaging to and from FPGA
void ISR_FPGARX (void);	Interrupt service routine for the RF FPGA drive
BOOLEAN GetOverCurrentDetect (void);	Return Overcurrent status
void ClearOverCurrentDetect (void);	Clears overCurrentDetected
void EnableOverCurrentDetect (void);	Enables hardware to detect overcurrent
BOOLEAN GetFPGADriveOn (void);	Return status of FPGA RF drive
void SetRFDrive (BOOLEAN);	Return status of FPGA RF drive
UInt16 GetFPGAVersionNumber (void);	Returns the FPGA version number
void BuildPacket(Int8 cmd, Int8 *payload);	Setup a message packet to send to the FPGA

void FPGATransmitBlocking (void);	This function will block until all bytes in the FPGA Uart transmit buffer are transmitted
void FPGATransmitTreatmentSettings (void);	This function adds treatment settings to Tx buffer

Rationale:

4.3.7 Ultravolt Manager

G3_5-DS-90
Rationale:

4.3.7.1 Mode of Operation

G3_5-DS-46

Ultravolt manager is responsible for setting the output voltage of Ultravolt high voltage power supply.

Ultravolt manager is connected to a DAC LTC2632ACTS8-LZ12. DAC output is connected to Ultravolt HVPS.

DAC is commanded by bit banging the GPIO and sending data.

Rationale:

4.3.7.2 Input / Output

G3_5-DS-47

Ultravolt manager gets voltage set point from RF One Shot Manager. It outputs the commands to the DAC through the GPIO by bit banging.

Rationale:

4.3.7.3 Functional Units

G3_5-DS-49

External Interface	**Description**
void InitUltravolts (void);	Initialize ultravolt power supply interface
void SetUltraVoltage (enum VoltageSelectState, double);	Set selected DAC to requested setting
void EnableUltraVolts (void);	Enable both ultravolt power supplies
void DisableUltraVolts (void);	Disable both ultravolt power supplies
void ResetReference (void);	Reset DAC

Rationale:

4.3.8 Foot switch Module

G3_5-DS-91
Rationale:

4.3.8.1 Mode of Operation

G3_5-DS-50

Foot switch manager handles foot switch input. Foot switch is polled each time it's called from main loop.

Debouncing is performed on the foot switch input to remove noise.

Rationale:

4.3.8.2 Input / Output

G3_5-DS-51

Foot switch input is through a GPIO pin. Foot switch manager provides this information to the RF One shot manager and POST module.

Rationale:

4.3.8.3 Functional Units

G3_5-DS-53

External Interface	**Description**
void FootswitchManager (void);	This function initializes default software parameters
BOOLEAN GetFootswState (void);	Return status of footswitch press
void InitFootswitch (void);	Initialize footswitch hardware

Rationale:

4.3.9 GPIO Manager

G3_5-DS-92
Rationale:

4.3.9.1 Mode of Operation

G3_5-DS-54

GPIO manager provides interface to handles controller GPIO. GPIO initializes GPIO as input or output pins.

It also provides APIs to set, clear and toggle GPIO if it's set as an output and read API if it's set as an input.

Rationale:

4.3.9.2 Input / Output

G3_5-DS-55

GPIO gets it's input from RF One shot manager, FPGA manager, Footswitch manager, Display and ADC manager.

It outputs to the controllers GPIO registers.

Rationale:

4.3.9.3 Functional Units

G3_5-DS-57

External Interface	**Description**
void GPIO_Init(const GPIO_TABLE_T * ioTable, int num);	Initializes the GPIO driver
bool GPIO_GetActiveState(int signal);	Returns the state of the GPIO signal based on it's active state
void GPIO_SetActive(int signal);	Sets the state of the GPIO signal based on it's active state
void GPIO_ClearActive(int signal);	Clears the state of the GPIO signal based on it's active state
void GPIO_Toggle(int signal);	Toggles the GPIO signal

Rationale:

4.3.10 Audio Manager

G3_5-DS-93
Rationale:

4.3.10.1Mode of Operation

G3_5-DS-58

This module is responsible for interacting with the speaker. It uses Flex Timer module to produce PWM output. It handles setting the frequency of the PWM signal, turning on/off speaker and playing all the tones.

Rationale:

4.3.10.2Input / Output

G3_5-DS-59

It receives it's inputs from Display module and RF One shot manager. It outputs PWM signal through a GPIO pin using the FlexTimer module (FTM).

Rationale:

4.3.10.3Functional Units

G3_5-DS-61

External Interface	Description
void InitAudio(void);	Sets up the FlexTimer3 module to output to pin E10 for the speaker
void TurnOnSpeaker(void);	Turns on the speaker by enabling the clock for the flextimer3
void TurnOffSpeaker(void);	Turns off the speaker by disabling the clock for the flextimer3
void SetAudioFrequency (UInt16 frequency);	Sets the frequency for the speaker to output when it is turned on.
BOOLEAN IsSpeakerOn(void);	Return status of speaker
void AudioManager(void);	Handles treatment success and failure audio tones
void SetTreatmentComplete(BOOLEAN setTreatmentComplete);	Setter of local variable for if the treatment is over
void SetHFSuccess(BOOLEAN setHFSuccess);	Setter of local variable for if the treatment succeeded or not
void SetSoundTimer(void);	Set sound timer to current high speed timer

Rationale:

4.3.11 Debug Module

G3_5-DS-94
Rationale:

4.3.11.1Mode of Operation

G3_5-DS-62

Debug module implements the interface for debug UART port. Debug UART port is used to perform communication between the controller and PC.

Controller outputs the Voltage, Current and Impedance calculated for each PEF delivered through the Debug port.

This information is used to perform Voltage and current measurement calibration and output voltage calibration.

Voltage and current measure calibration is to ensure that the voltage and current read back using the ADC is accurate.

Output voltage calibration is to ensure that the output voltage is accurate to the commanded voltage.

Measurement and output voltage calibration values are entered through the Debug port and is saved on the controllers nonvolatile program flash memory.

UART port operates at 115200 baud rate, one stop bit and no parity bit.

Calibration values can be entered and saved to the controllers memory only in TEST mode.

Rationale:

4.3.11.2Input / Output

G3_5-DS-63

Debug module receives it's input from RF One shot Manager and PC through the UART port. It outputs it's data through the UART port.

Rationale:

4.3.11.3Functional Units

G3_5-DS-64

External Interface	**Description**
void Debug_Init (void);	Function to initialize debug serial interface
void Debug_SendBuffer (Int8 * message, UInt16 size);	Transfer message to serial transmit buffer
void Debug_Execute(void);	Routine to transmit data through UART Port

Rationale:

4.3.12 Flash Manager

G3_5-DS-95
Rationale:

4.3.12.1Mode of Operation

G3_5-DS-65

Flash Manager enables access to flash memory blocks. It uses the Standard Software Driver (SSD) for C90TFS/FTFx flash to perform high-speed program/erase operations.
Flash manager handles writing calibration values to the flash memory. Flash manager is initialized only during TEST mode.

Rationale:

4.3.12.2Input / Output

G3_5-DS-66

Flash manager gets it's input from the Debug module. It outputs to the Flash Memory Module (FTFE) to write data to the controllers flash memory.

Rationale:

4.3.12.3Functional Units

G3_5-DS-67

External Interface	Description
extern uint32_t RelocateFunction(uint32_t dest, uint32_t size, uint32_t src);	This function provides users a facility to relocate a function from one location to another location in RAM.
extern uint32_t FlashInit(PFLASH_SSD_CONFIG pSSDConfig);	This API will initialize flash module by clearing status error bit and reporting the memory configuration via SSD configuration structure.
extern uint32_t FlashCommandSequence(PFLASH_SSD_CONFIG pSSDConfig);	This API is used to perform command write sequence on the flash. It is internal function, called by driver APIs only
extern uint32_t FlashEraseSector(PFLASH_SSD_CONFIG pSSDConfig, uint32_t dest, uint32_t size, pFLASHCOMMANDSEQUENCE pFlashCommandSequence);	This API will erase one or more sectors in P-Flash or D-Flash memory. This API always returns FTFx_OK if size provided by user is zero regardless of the input validation.
extern uint32_t FlashVerifySection(PFLASH_SSD_CONFIG pSSDConfig, uint32_t dest, uint16_t number, uint8_t marginLevel, pFLASHCOMMANDSEQUENCE pFlashCommandSequence);	This API will check to see if a section of P-Flash or D-Flash memory is erased to the specified read margin level
extern uint32_t FlashProgram(PFLASH_SSD_CONFIG pSSDConfig, uint32_t dest, uint32_t size, uint8_t* pData,	This API is used to program 4 consecutive bytes (for program long word command) and 8 consecutive bytes (for program phrase command) on P-flash or D-Flash block. This API always returns FTFx_OK if size provided by user is

pFLASHCOMMANDSEQUENCE pFlashCommandSequence);	zero regardless of the input validation
extern uint32_t FlashProgramCheck(PFLASH_SSD_CONFIG pSSDConfig, uint32_t dest, uint32_t size, uint8_t* pExpectedData, uint32_t* pFailAddr, uint8_t marginLevel, pFLASHCOMMANDSEQUENCE pFlashCommandSequence);	This API tests a previously programmed P-Flash or D-Flash long word to see if it reads correctly at the specified margin level. This API always returns FTFx_OK if size provided by user is zero regardless of the input validation
extern uint32_t FlashCheckSum(PFLASH_SSD_CONFIG pSSDConfig, uint32_t dest, uint32_t size, uint32_t* pSum);	This API will perform 32 bit sum of each byte data over specified flash memory range without carry, which provides rapid method for checking data integrity.
extern uint32_t FlashProgramSection(PFLASH_SSD_CONFIG pSSDConfig, uint32_t dest, uint16_t number, pFLASHCOMMANDSEQUENCE pFlashCommandSequence);	This API will program the data found in the Section Program Buffer to previously erased locations in the Flash memory.

Rationale:

4.3.13 ADC Manager

G3_5-DS-96
Rationale:

4.3.13.1Mode of Operation

G3_5-DS-68

ADC manager provides an interface to read external ADCs. There are two external ADCs used LTC1859 and ADS8363.

LTC1859 is used to read voltage output at Ultravolt and voltage across the capacitors. SPI2 is used to communicate with LTC1859 and is set as master while accessing it.
ADS8363 is used to read the voltage and current delivered to the patient. SPI1 and SPI2 is used to communicate with ADS8363. SPI1 is used to read voltage and SPI2 is sued to read current. Both SPI1 and SPI2 are set as slaves while accessing ADS8363. In slave mode the FPGA provides the clock and CS lines to ADS8363, SPI1 and SPI2.

Ultravolt output voltage and capacitor output voltage are read during POST checks to verify the hardware.

The output therapy voltage and current is read every time a PEF is delivered. A PEF packet has 60 pulses at 600 KHz which is has a length of 100 µs.

The FPGA provides the conversion trigger. ADC is triggered 10 times on the positive cycle and 10 times on the negative cycle. Of the 10 samples, the ADC manager uses 5 samples and generate average positive and negative voltage and current measurements. It calculates the output impedance using the measured voltage and current.

ADC manager outputs average positive and negative voltage, current and impedance though the debug port after each delivered pulse.

The calculated impedance measurements are used by the RF One Shot manager as input to the control loop. The voltage and current measurements are monitored to make sure they are always within tolerance.

Rationale:

4.3.13.2Input / Output

G3_5-DS-69

ADC manager reads the SPI data when it receives an interrupt on the falling edge of ADC read line. It gets its input data from LTC1859 and ADS8363.

The PEF voltage, current measurement and the calculated output impedance values are used by RF One Shot manager for control loop. It also outputs these values to the debug UART port.

Rationale:

4.3.13.3Functional Units

G3_5-DS-70

External Interface	**Description**
void InitializeSPIToAccessLTC1859(void);	Initialize GPIOs and SPI port to access LTC1859
void InitializeSPIToAccessADS8363(void);	Initialize GPIOs and SPI port to access ADS8363
double MonitorVSenseBlocking (void);	This function configures the ADC to read Vsense, then returns the latest reading from the ADC.
double MonitorUltravoltBlocking (enum ADCExternalChannel);	This function configures the ADC to read ultravolt A, then returns the latest reading from the ADC.
void ADCManager(void);	Processes ADC readings
double GetAvgCurrentRead(void);	Returns measured current
double GetAvgVoltRead(void);	Returns measured volt
double GetImpedance(void);	Returns measured impedance
void ResetSPI(void);	Resets flags and SPI bus after end of therapy

Rationale:

4.3.14 Calibration Module

G3_5-DS-97
Rationale:

4.3.14.1Mode of Operation

G3_5-DS-71

This module is responsible for managing storing calibration values and reading the stored calibration values. To store the calibration values, the module uses Flash Manager to write the values to the Program flash memory.

The values are stored as 32 bit unsigned integers. Along with the value, it's complement is also stored in the memory next to it. This is to ensure the integrity of the data when it is read.
When the data is read back, if the stored value and it's complement do not match, then fault state is entered. Calibration values can only be stored during Test mode.

The calibration data is ready during POST and is provided to RF One Shot manager and ADC manager.

Rationale:

4.3.14.2Input / Output

G3_5-DS-72

Calibration modules gets it's inputs from the Debug module. It outputs the read calibration values to ADC Manager and RF One Shot manager.

Rationale:

4.3.14.3Functional Units

G3_5-DS-73

External Interface	**Description**
void InitCalibration (void);	Initialize Calibration
double VSenseCal (void);	Read Voltage sense calibration
double ISenseCal (void);	Read Current sense calibration
void Cal_SetVoltageCal(UInt16 voltCal);	Set Voltage calibration value
UInt16 Cal_GetVoltageCal(void);	Fetch Voltage calibration value
void Cal_SetCurrentCal(UInt16 currentCal);	Set Current calibration value
UInt16 Cal_GetCurrentCal(void);	Fetch Current calibration value
void Cal_SetVoltageGainCal(UInt16 voltGainCal);	Set Voltage gain calibration value

void Cal_SetCalVoltageGainCal(UInt16 calVoltGainCal);	Set Cal check Voltage gain calibration value
void InitializeFlash(void);	Initialize Flash driver
BOOLEAN IsCalValueCorrupted(void);	Returns state of calibration values
BOOLEAN IsFlashInitSuccessful(void);	Returns Flash initialization state

Rationale:

5 UI Software Design

5.1 Level of Concern

5.1.1 Level of Concern

G3_5-DS-79

Software Safety Classification Per IEC 62304:2006 Edition First, Medical device software -- Software Life Cycle Processes and Software Level of Concern Per FDA guidance (Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices issued May 11, 2005) is recorded in Risk Management File RSK015.

Rationale:

5.2 Processor Specifications

5.2.1 General Description

G3_5-DS-83

The software contained in the SLCD controls the graphic information presented to and respondent from the user. These signals are presented to the Main Controller for processing. Information to notify the user is sent from the Main Controller to the SLCD. The SLCD processes the information and updates the displayed graphic user interface. The software when mentioned strictly refers to the macro commands reliant upon the third party firmware. Communication with the SLCD is through an RS232 interface.

Operating Environmental Constraints

Hardware

- 16-bit color (565 mapping - same as PC / Mac)
- Touch controller (4 wire resistive) on board
- High speed ARM9 processor (200+MHz)
- On-board RS232, RS485, CMOS level interfaces up to 230,400 baud
- Up to 28MB highly reliable NOR data flash for user downloadable bitmaps with optional RLE compression
- SD card slot for firmware upgrades and bitmap / macro storage

Rationale:

5.3 Modules

5.3.1 Bitmap and List file

5.3.1.1 Mode of Operation

G3_5-DS-74

To implement a GUI, a set of graphic images are needed for backgrounds, logos, buttons, switches, and so forth. These are created on a PC as bitmaps (.bmp files) in 24-bit color. The bitmaps are compressed into a binary file which is then downloaded directly into the SLCD5 flash memory via the serial port.

Bitmaps are numbered from 1 to n, depending on how many bitmaps are loaded (with BMPload). The first bitmap loaded will be bitmap #1, the second bitmap loaded will be #2, etc.

The file names of all the bitmaps are stored in a file in the order it needs to be numbered. It's a better way to load all the image files using BMPLoad software.

Rationale:

5.3.1.2 Input / Output

G3_5-DS-75

Bitmap files and BMP List file is provided as input to the BMPLoad software.

Rationale:

5.3.1.3 Functional Units

G3_5-DS-76

There are no functional units in this module.

Rationale:

5.3.2 Macro File

5.3.2.1 Mode of Operation

G3_5-DS-80

A macro is a set of commands that can be invoked with a single command. It is also linked to the buttons used on the display.

Several macros are used to draw the graphical elements on the screen. These macros are invoked by the Controller's Display module by sending command through the serial port.

Rationale:

5.3.2.2 Input / Output

G3_5-DS-81

The Controller's Display module provides it's input to invoke the macros and the macro outputs the commands on the screen.

Rationale:

5.3.2.3 Functional Units

G3_5-DS-82

There are no functional units in this module.

Rationale: