



<https://cosmiic.org>

Example - Early Feasibility Investigational Device Exemption

IDE Section:

Appendix H – Software Description, Verification and Validation

This material is provided by COSMIIC as an example of the contents of an EFS-IDE submission using the COSMIIC System. Regulatory requirements and information in this document may have been obsoleted since original submission of these documents. Please refer to the Read Me document for full background of the contents in this document set, reasoning for redactions, and licensing for these documents. For up-to-date technical information on the COSMIIC System, please visit the Docs site through cosmiic.org. This document is released by COSMIIC with the open source CC-BY-4.0 license.

1. Appendix H - Software Description, Verification and Validation	2
1.1 Level of Concern	3
1.2 Software Description	5
1.3 Device Hazard Analysis	11
1.4 Architecture Design Chart	12
1.4.1 CE Software Architecture	13
1.4.2 CT Software Architecture	16
1.4.3 PM Software Architecture	17
1.4.4 PG4 Software Architecture	18
1.4.5 BP2 Software Architecture	19
1.4.6 PM Bootloader Flowchart	20
1.4.7 RM Bootloader Flowchart	23
1.4.8 Charging Flowchart	26
1.4.9 CT Gateway	29
1.4.10 PM Gateway	31
1.4.11 PM Modes Diagram	33
1.4.12 NNP Distributed Processing	34
1.5 Software Requirements Specification (SRS)	35
1.6 Software Design Specification (SDS)	38
1.7 Software Development Environment Description	39
1.8 Verification and Validation Documentation	43
1.9 Revision Level History	44
1.10 Unresolved Anomalies (Bugs or Defects)	49
1.11 Software Testing	50
1.12 Traceability Analysis	51

Appendix H - Software Description, Verification and Validation

Level of Concern

Based on FDA's *Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices*, we have revised our analysis for Level of Concern, resulting in a **Moderate Level of Concern**. We provide our rationale, below, and have proceeded in documenting our software accordingly.

Table 1 Major Level of Concern

If the answer to any one question below is Yes, the Level of Concern for the Software Device is likely to be Major.

1. Does the Software Device qualify as Blood Establishment Computer Software? (Blood Establishment Computer Software is defined as software products intended for use in the manufacture of blood and blood components or for the maintenance of data that blood establishment personnel use in making decisions regarding the suitability of donors and the release of blood or blood components for transfusion or further manufacture.)

NO

2. Is the Software Device intended to be used in combination with a drug or biologic?

NO

3. Is the Software Device an accessory to a medical device that has a Major Level of Concern?

NO

4. Prior to mitigation of hazards, could a failure of the Software Device result in death or serious injury, either to a patient or to a user of the device? Examples of this include the following:

a. Does the Software Device control a life supporting or life sustaining function?

NO

b. Does the Software Device control the delivery of potentially harmful energy that could result in death or serious injury, such as radiation treatment systems, defibrillators, and ablation generators?

NO. The software controls the duration and occurrence of stimulation pulses, but the level of energy delivered is hardware limited and would be insufficient to result in death or serious injury. Stimulation at uncontrolled levels may result in temporary, reversible, localized tissue damage. Based on our Risk Analysis, issues dependent on software are limited to a severity of 3 and below.

c. Does the Software Device control the delivery of treatment or therapy such that an error or malfunction could result in death or serious injury?

NO. The software controls the duration and occurrence of stimulation pulses, but the level of energy delivered would be insufficient to result in death or serious injury. Based on our Risk Analysis, issues dependent on software are limited to a severity of 3 and below.

d. Does the Software Device provide diagnostic information that directly drives a decision regarding treatment or therapy, such that if misapplied it could result in serious injury or death?

NO

e. Does the Software Device provide vital signs monitoring and alarms for potentially life threatening situations in which medical intervention is necessary?

NO

CONCLUSION: The software is not a Major Level of Concern.

Table 2 Moderate Level of Concern

If the Software Device is not Major Level of Concern and the answer to any one question below is Yes, the Level of Concern is likely to be Moderate.

1. Is the Software Device an accessory to a medical device that has a Moderate Level of Concern?

NO

2. Prior to mitigation of hazards, could a failure of the Software Device result in Minor Injury, either to a patient or to a user of the device?

YES, stimulation levels are controlled via software, user intent is interpreted via software, but the level of energy delivered is hardware limited and would be insufficient to result in death or serious injury. For the hand grasp system, incorrect programming, user errors, or glitches in communication (radio or CAN) could cause held objects to fall potentially causing minor injury. For the trunk system, unintended stimulation of the trunk could potentially lead to an uncomfortable, but non-harmful loss of balance while sitting in the wheelchair.

YES, if nerve cuffs are used. Stimulation of nerve at muscle levels may result in temporary, reversible, localized tissue damage.

3. Could a malfunction of, or a latent design flaw in, the Software Device lead to an erroneous diagnosis or a delay in delivery of appropriate medical care that would likely lead to Minor Injury?

NO

If the answers to all of the questions in Tables 1 and 2 above are No, the Level of Concern is Minor

This regulation defines serious injury as an injury or illness that:

1. is life threatening;
2. results in permanent impairment of a body function or permanent damage to a body structure; or
3. necessitates medical or surgical intervention to preclude permanent impairment of a body function or permanent damage to a body structure. For the purposes of this document, the term permanent is defined as “irreversible impairment or damage to a body structure or function, excluding trivial impairment or damage.” 21 CFR 803.3(bb)(2).

CONCLUSION: The software is a MODERATE Level of Concern.

Software Description

Overview

Software is an essential part of the NNP system and resides in both implanted and external devices, allowing devices to be customized to individual users for various applications (e.g. Hand Grasp, Trunk Control). A high-level software block diagram is shown in Figure XX.

The solid gray regions represent the eight software components within in the system:

- Control Editor Application (CE App)
- Remote Tuner Application (RT App)
- Control Tower Application (CT App)
- Power Module Radio Bootloader (PM BL)
- Power Module Application (PM App)
- Remote Module CAN Bootloader (RM BL)
- Biopotential Module Application (BP2 App)
- Pulse Generator Application (PG4 App)

Broken lines in the schematic indicate that these connections are temporary for programming and user setup to be performed by a clinician or engineer. Solid lines require a wired connection. Dotted lines indicate a wireless connection. The implanted components can be used without interaction to the external components, however the CT is required for charging, querying system states, and starting the system after complete power down (batteries discharged, emergency shutoff). The PC is required for system programming and configuration. Each component is described briefly below to provide the reader with an understanding of each software component's purpose and intended users. The programming language, hardware platform, operating system, and use of Off-the-Shelf Software is summarized in [Software Requirements Specification \(SRS\)](#). The Software Architecture for each component is described in more detail in [Architecture Design Chart](#). CANFestival and Micrium are common to multiple components and are described briefly below. The software is classified as Moderate Level of Concern, and as such, **does not provide a critical role in protecting patient safety** as described in the [Device Hazard Analysis](#). Hardware design is sufficient to protect the patient from life threatening or permanent injury in the event of software malfunction. The [fail-safe](#) RM and PM Bootloaders are the only pieces of software whose anomalous behavior could result in a permanent inability to communicate with the implanted components as described in [Device Hazard Analysis](#).

Off-the-shelf Software Components

OTS Software is not present in either of the [fail-safe](#) software components (PM Bootloader and RM Bootloader). Other hazards related to OTS software are described in [Device Hazard Analysis](#) and are deemed to be only Minor Level of Concern. Therefore, only Basic documentation for OTS software components are provided. OTS software is included within the executable files themselves. Therefore any testing of software components containing OTS software also tests the OTS software itself. The end user of the software does not need to separately install any OTS software.

CANFestival

CanFestival provides an ANSI-C, platform independent, open source CANOpen® stack that can be built

as master or slave nodes on PCs, Real-time IPCs, and Microcontrollers. In the NNP System, it is implemented on every implanted module and the CT, on 3 different microcontrollers.

CANOpen is a communication protocol specification for embedded systems used in automation and other applications, including some medical and safety critical applications. According to the ISO-OSI communication standard, network protocols can be organized as seven layer models (physical, data link, network, transport, session, presentation, and application). In terms of the OSI model, CANOpen implements the layers above and including the network layer. The CANOpen standard consists of an addressing scheme, several small communication protocols and an application layer defined by a device profile or within the code implementation. The communication protocols have support for network management, device monitoring and communication between nodes, including a simple transport layer for message segmentation/desegmentation. The lower level protocol implementing the data link and physical layers is usually Controller Area Network (CAN), although devices using some other means of communication can run the CANOpen protocol stack. In the NNP System, the radio and USB communications both use some components of the CANOpen stack.

OSI Model		
7	Application	CANOpen
6	Presentation	
5	Session	
4	Transport	
3	Network	
2	Data Link	MCU
1	Physical	

The Object Dictionary (OD) is a key feature of CANOpen. It allows a global understanding of data transmitted in the system, and provides the integration point for both local and global resources. Data in the OD is organized via a 16bit index and an 8bit subindex. A local copy of the OD is stored in each device in the system (Remote Tuner and PC excluded).

Five important CANOpen communication objects are used in the NNP system:

- Network Management (NMT): NMTs follow a master/slave relationship. [In the NNP System, the PC-CT-PM chain is the master, and remote modules are slaves.](#)
- Service Data Object (SDO): SDOs follow a peer-to-peer client/server relationship. In most operations, the PM acts as the client and each RM acts as a server. The PM reads data from or writes data to a RM OD. The RM responds with the data from the OD or with confirmation that data has been written to the OD. The terms upload and download must be read as the action of the OD on the server/client. During an SDO upload, the OD gives data to the client via the server, i.e., the client reads from the server OD. During an SDO download, the OD receives data from the client via the server, i.e., the client writes to the server OD. In the NNP System, SDOs are also used to read or write OD entries on the PM or the CT. [In this scenario, the CT or PM may act as a server rather than a client.](#)
- Process Data Object (PDO): PDOs follow a producer/consumer model. Remote modules act as producers (of control information, e.g. MES features, accelerometer data) and the PM acts as the consumer. The PM maps incoming PDOs to specified locations within the PM OD forming "virtual

pipes" between RM data and the PM OD. PDOs are unsolicited and are not confirmed, resulting in minimal overhead.

- Heartbeat: Heartbeats also follow a producer/consumer model. Heartbeats are produced by each RM at regular intervals and consumed by the PM. If the PM has not received a heartbeat from a particular module in a specified amount of time, then the PM removes that module from the active node list.
- SYNC Object: The synchronization (SYNC) object is a short high priority message broadcast by the PM at a regular interval while in some modes. All remote modules receive the SYNC and can schedule stimulation or recording actions relative to the sync. CANOpen allows PDO transmission to be triggered automatically, directly controlled by SYNC objects. However, in the NNP system, the PDO transmission is always invoked by the application, indirectly scheduled via the SYNC object. In the NNP System, the SYNC object is used to specify the Stimulation frequency and MES recording interval.

Micrium

Micrium is the provider of a purchased RTOS (Real Time Operating System) known as C/OS-III. C/OS-III is a preemptive, multi-tasking OS that runs on a variety of microprocessor families. C/OS-III manages unlimited application tasks and an unlimited number of priority levels. C/OS-III allows for unlimited tasks, semaphores, mutexes, event flags, message queues, timers and memory partitions. The user allocates all kernel objects at run time. C/OS-III provides features to allow stack growth of tasks to be monitored. C/OS-III has a number of internal data structures and variables that it needs to access atomically. It protects these critical regions by disabling interrupts for almost zero clock cycles, ensuring that it is able to respond to some of the fastest interrupt sources. Interrupt response with C/OS-III is deterministic. C/OS-III ensures that NULL pointers are not passed, task level services from ISRs aren't called, arguments are within allowable range, and specified options are valid. Each API function provides an error code regarding the outcome of the function call. The specific application of C/OS-III is covered in the discussion of the task model. C/OS-III has been used in many medical device applications. Micrium also supplies the CAN and USB drivers for the communication subsystem and the file system drivers for the file IO subsystem.

Windows 7 and .Net API

The CE App is intended to run on Windows 7, a standard Microsoft product. The operating system services are available to programmers via the .Net API. The .Net API contains thousands of calls to various methods to provide functionality such as read and write to disk, using the USB link, and many others. Collectively, the Windows 7 operating system and its API are referred to here as the OS. Part of these services is the ability to partition the application into different threads, for example in the CE App, a foreground and background thread (<see CE Architecture>). The OS also allows the application to present the user certain selected views. In the CE app, these views are organized as tabs (identified and summarized below). An additional service set provides access to and management of the USB link using Windows 32bit (or 64bit) calls. Finally, the OS provides read-write access to disk to store and retrieves persistent data relating to the patient application (organized as an XML file).

Purpose and Intended Users

Control Editor Application

The CE App is the highest level component of the system and is used to configure and monitor the

system. Specifically the CE App provides the following services:

- Configuring the application values to adapt the NNP system to an individual patient, including stimulation parameters, profiles, command sources, and general sequences.
- Providing a troubleshooting tool for engineers engaged in optimizing specific system parameters.
- Maintaining a portable record of modifications and control settings for the individual patient.
- Monitoring battery charging, control, and reporting.
- Providing a method for collecting error records and other change-of-state information available in the system.

Although it is intended to operate connected to the Control Tower and actively engaged with the NNP system, the CE App can also operate in an off-line mode. In this case, certain features are not enabled and are “grayed out”.

The end user for the CE App is the clinical engineer who needs to configure and maintain an implanted NNP system. The CE App is not used directly by the patient.

Remote Tuner Application

The Remote Tuner is a simple user interface device to facilitate parameter setting by a clinical engineer. The device includes a knob, two buttons, and indicator LEDs. The device connects and communicates with the PC running the CE App via USB.

Control Tower Application

The CT App performs key communication, power, and application functions for the NNPS. It also provides hardware drivers for the CT hardware.

Specifically, the CT App provides the following services:

- Routing messages from the USB link locally or to the rest of the implanted NNPS system via the Radio.
- Managing error conditions and generating replies to the above .
- Monitoring battery charging, control, and reporting.
- Providing persistent (or non-volatile) memory for the implantable system to store recorded data, error messages, and state information through power down conditions.
- Providing a user interface to the patient to monitor and change operating states, implemented as configurable scripts.
- Maintain a real-time-clock and manage PM clock for data logging/error reporting

The intended users include the clinical engineer and the patient. In the lab setting, the CT acts as a bridge between the CE App and PM for loading new applications to the PM or RM and to configure patient settings. The patient will use the CT without the CE App to can change major NNP operating states, such as standby, exercise, and active control, to view system status, and to recharge the PM

Power Module Bootloader

The PM BL performs key fail-safe and upgrading functions for the NNPS. It also provides a minimal set of hardware drivers to access important application data and [load new application images to the PM](#).

Specifically, the PM BL provides the following services:

- Configures default pin settings (input/output, mux, high/low, pullups)
- Minimal radio interface with dedicated address, channel number, and transmit power
- Loading new PM App images
- Read PM App radio addresses, channel, and transmit power from Flash

- Reports module Serial Number
- Invoke the PM App after a delay time (20s) if no entry into PM BL from CE-CT
- Prevents overwriting itself

There is no direct end user of the PM BL, The PM BL can only be utilized through the CT via the CE App PM Bootloader Tool, whose end user is the engineer who loads PM App into devices before or after implantation or needs to determine the PM App radio settings. The patient will not have access to the PM BL functionality through the CT interface. The PM Bootloader serves an important function in mitigating risk by allowing system upgrades in the case of malfunctioning software

Power Module Application

The PM App performs key communication, power, and application functions for the NNPS. It also provides hardware drivers for the PM hardware.

Specifically, the PM software provides the following services:

- Routing messages from the radio locally and to the rest of the implanted NNPS system via the wired implanted network (FESCAN).
- Managing error conditions and generating replies to the above .
- Hosting the specific patient application in form of scripts composed in the CE App
- Supervising battery charging, control, and reporting.
- Managing power distribution to the other implanted modules in the system including providing the control point for the fail-safe shutdown through a magnetic switch.
- Providing persistent (or non-volatile) memory for the implantable system to store recorded data, error messages, and state information through power down conditions.
- Reporting IO information such as temperature and accelerometer data.

Interfacing with the PM App in the lab/clinic is always done through the CT and CE App so the engineer is not a direct end user of the PM App. However, the patient is a direct end user of PM App, as the system can be used without a connection to the CT. In this case the patient must be aware of how to change system modes using only implanted command signals (MES, accelerometers) without any monitoring/display capability.

Remote Module Bootloader

The RM BL allows a new image of the BP2 App or PG4 App to be loaded using the RM Bootloader Window in the CE.

Specifically, the RM BL provides the following services:

- Configures default pin settings (input/output, mux, high/low, pullups)
- Minimal CAN interface to PM
- Change Node number
- Selects node uniquely
- Reports module Serial Number
- Loading new BP2 App and PG4 App images
- Invoke the BP2 App or PG4 App after an **NMT_Wake_Remote_Module** command
- Prevents overwriting itself

There is no direct end user of the RM BL, The RM BL can only be utilized through the CT or CT-PM via the CE App RM Bootloader Tool. The end user of the CE App RM Bootloader Tool is the engineer who loads the RM App into devices before or after implantation or needs to modify RM node numbers in the system.

Biopotential Application

The BP2 App residing in the BP2 hardware performs key communication, recording, and processing functions for the NNPS. It also provides drivers for the BP2 sensors.

Specifically, the BP2 App provides the following services:

- Routing messages from the BP-2 to and from the wired network (FESCAN).
- Records and outputs raw full bandwidth (1KHz) MES data from 1 channel at a time
- Records MES features from programmable length discrete time windows
- Programmable control of stimulus artifact rejection
- Drivers for accelerometer, temperature sensing

The intended user of the BP2 App is the patient. The software is intended to obtain command signals (e.g. continuous commands for proportional control of a hand grasp and logic signals for changing system states or scripts) from the patient's voluntary EMG signals.

Pulse Generator Application

The PG4 App residing in the PG4 hardware performs key communication, recording, and processing functions for the NNPS. It also provides drivers for the PG4 sensors.

Specifically, the PG4 App provides the following services:

- Routing messages from the PG4 to and from the wired network (FESCAN).
- Calculates stimulation values (pulse width and amplitude) for command inputs via preloaded patterns or by direct settings.
- Programmable control of stimulus pulse scheduling .
- Generates stimulus pulse logic commands.
- Drivers for accelerometer, temperature sensing.

The intended user of the system is the patient. The software controls stimulation of the user's paralyzed muscles based on an input signal generated by scripts run within the PM App.

Device Hazard Analysis

From the system-level NNP hazard analysis, those issues dependent on software are limited to a severity of 3 and below

Anomalous unforeseen behavior in the PM or RM bootloader could result in the permanent inability to communicate with some or all of the implanted components. This would require revision surgery to restore functionality to the system. This is the worst outcome due to any software malfunction in the system and does not affect the safety of the patient directly. The bootloaders will all be exercised before implantation to verify that the devices can be reprogrammed. Further, the bootloaders have been tested extensively on test modules under various conditions (See Testing Summary). No condition was found that caused a permanent inability to communicate with the modules.

Anomalous behavior in the PG4 App could result in unintended stimulation or stimulus parameters that exceed the allowed pulse width. Pulse amplitude is hardware limited. As discussed in the Risk Analysis, the severity of such an event would be limited to a 2 or below. Anomalies in OTS Software (CANFestival) could result in unintended stimulation values, but only within specified limits. Pulse Width cannot exceed 255 due to a communication error because it is represented in a single byte.

Anomalous behavior in the CT App or PM App could result in inaccurate reporting of temperature or continue charging without regard for temperature. However, the hardware limits charging coil field strength such that the PM capsule temperature and external coil temperature do not exceed our maximum temperature rise regardless of software operation. The PM has also been tested under expected operating conditions and was shown not to exceed our maximal temperature rise. Therefore, the software is not a critical safety path, but is rather used as a mitigation of risk in the case of hardware failure.

Anomalies in OTS software (CANFestival or Micrium) could affect the reporting of temperature in implanted modules and the external charging coil. However, this does not pose any additional risk to that already described within the CT App and PM App that utilize these OTS software components.

In no other parts of the system can software affect energy flow to the patient to elicit harm whether via inadvertent or intentional misuse.

For the purposes of the Early Feasibility IDE, it is recognized that ongoing improvement of the software will be necessary. Prior to the start of the IDE, all requirements with priority levels marked "Critical" and "Major" must be verified and validated through simulated use testing. No subjects will be entered into the study until these requirements are met.

Architecture Design Chart

Overview

As described in the [Software Description](#), there are several software components in the NNP system. Refer to <Figure H1> for a brief explanation of each component.

In this section we provide an Architectural Design Chart for the main application components. These charts illustrate the use of OTS components (Micrium and CANFestival) within the applications:

- [CE Software Architecture](#)
- [CT Software Architecture](#)
- [PM Software Architecture](#)
- [BP2 Software Architecture](#)
- [PG4 Software Architecture](#)

The implant bootloaders are the most critical software components and therefore follow a straightforward, limited execution path as shown in the following diagrams:

- [PM Bootloader Flowchart](#)
- [RM Bootloader Flowchart](#)

The CT mitigates the risk of overheating the implant and external coil during the charging process in the case of a hardware failure. The following diagram illustrates the process by which this is achieved:

- [Charging Flowchart](#)

The routing of data from external devices (CT directly, or PC via CT) to and from the implant (PM) as well as within the implant (PM to RM) is an important capability of the NNP system. The following diagrams illustrate—in more detail than the CT and PM Software Architecture Diagrams—the interaction between tasks and the communication methods between modules.:

- [PM Gateway](#)
- [CT Gateway](#)

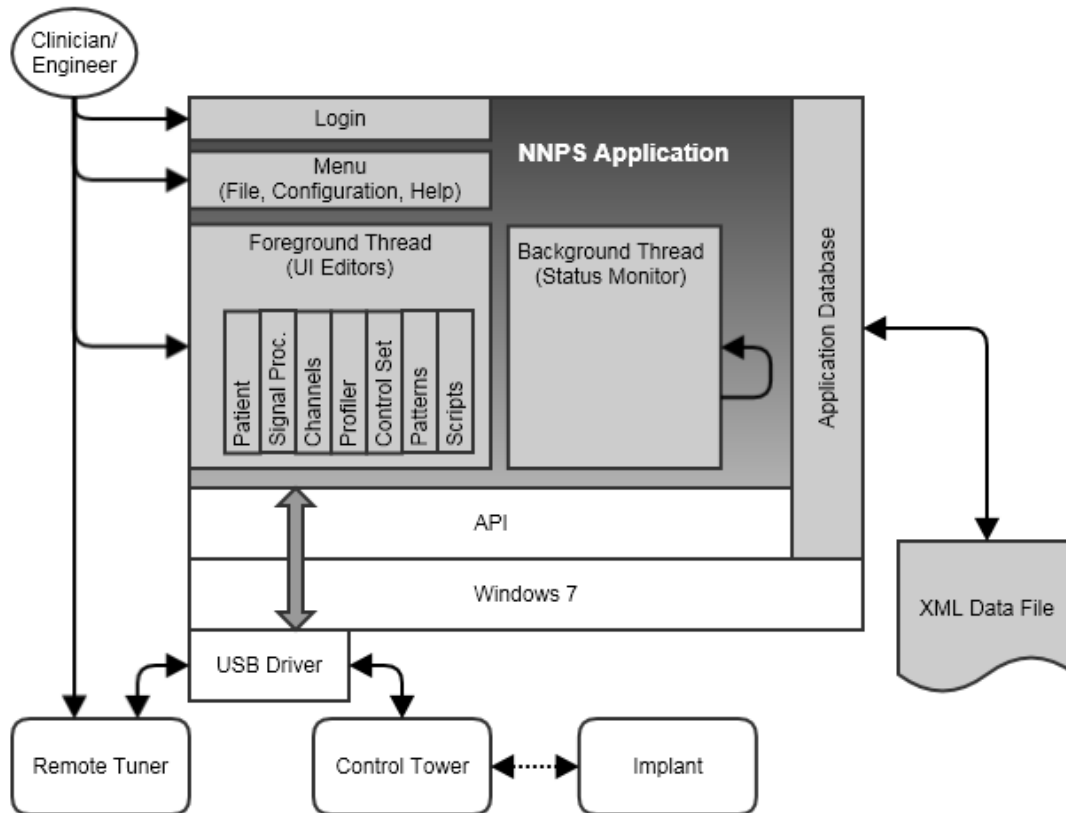
The concept of system modes is also important within the NNP system. The following diagram shows the progression of the PM through various modes:

- [PM Modes Diagram](#)

The NNP uses distributed processing to minimize network bandwidth. Typically, EMG will be processed locally by the BP2s as features. These features will be used by the PM within configurable scripts to determine command values. The command values are then used by the PG4s to determine stimulus parameters via predefined patterns stored within the module. This process is illustrated here:

- [NNP Distributed Processing](#)

CE Software Architecture



Foreground Thread

The foreground thread is controlled by an implicit state machine task based on user interaction with what's visible at any given time. By selecting a given control and handle within the control, the system calls a specific method that then invokes changes in behavior related to the context and data available at that moment. So the application can be organized as a collection of methods based on what's visible and available. Based on this observation, the following discussion summarizes the behavior available after the application is opened. The discussion assumes all features are available and the software is online.

Patient Tab

This menu item is meant to present to the clinician an immediate representation of the current state of the system, any relevant medical history, and access to prior visits. Any critical error messages are also immediately displayed.

Channels Tab

The channels table is an annotation of the topology of the system. Each channel (sensor and stimulation) is identified by its module parent, its location in the body, and relevant parameters. There is a field to allow the clinician to enter specific notes regarding function, placement, etc. The channel table can also be sorted by module type.

Sensors Tab

The sensors tab provide a visual representation of the various sensors in the system: temperature, motion, and EMG signals.

Profiler Tab

The profiler tab sets parameters for the stimulation channels (Threshold, Saturation, Maximum, and Frequency). Values are established by interacting with the patient using a slider bar (or external remote tuner) that modifies the selected stimulation parameter. These values are stored in the patient application file and used in determining the range of values in the pattern tab. They are not downloaded to the stimulation module and are not fail-safe ranges.

Control Set Tab

The control set tab organizes the application by relating inputs to stimulation setpoints. Inputs can be derived from MES features, accelerometer values, time, or any other values resident in the Object Dictionary. The control strategy may involve calculations applied to one or multiple values, advance filtering techniques, and other combinatorial strategies. The tab provides an interface to generate the scripts necessary to invoke pattern sets for different applications (e.g. Palmar Grasp, Lateral Grasp, Trunk Positioning, Exercising) and determines their method of being invoked. Different scripts might be invoked by an MES logic signal, a time base, or an external event such as a button push on the CT.

Patterns Tab

A pattern is a look-up table that controls a stimulus pulse (pulse width and/or amplitude) based on the command value being sent to the stimulus module. This tab allows the user to graphically manipulate the look-up table functions via a collection of points that – once downloaded to the stimulation module – are linearly interpolated to provide amplitude and pulse-width parameters to the individual stimulation channels for any command value.

Background Thread

The background thread maintains a dictionary of active modules in the system and can display the status of them automatically. The foreground thread user actions are frequently checked against the state of various modules to determine allowable actions. The background thread also periodically (every two seconds) checks certain system states such as battery power and network connections and displays them in the application status bar at the bottom of the screen. The background thread only runs if the system is online.

Menu Items

File

File contains the standard UI actions associated with file operations. These are: New, Open, Save, Save As, Close, Print, and Exit. They conform to Microsoft's user interface guidelines.

Configuration

Configuration is a collection of mini-apps that provide one-time configuration and engineering access to the system. These include: Module configuration (for establishing parameters on a per module basis), Network Testing, Radio Configuration, Read Records, Flash Download, Users, Update Library, and Options. These functions are discussed in more detail in the Software Design Specifications.

Help

Help lists the NNPS user's guide, the Clinical Programmers user's guide, and the standard About box.

Login

The login function solicits a user name and password before it grants access to the system to limit access to authorized clinical engineers.

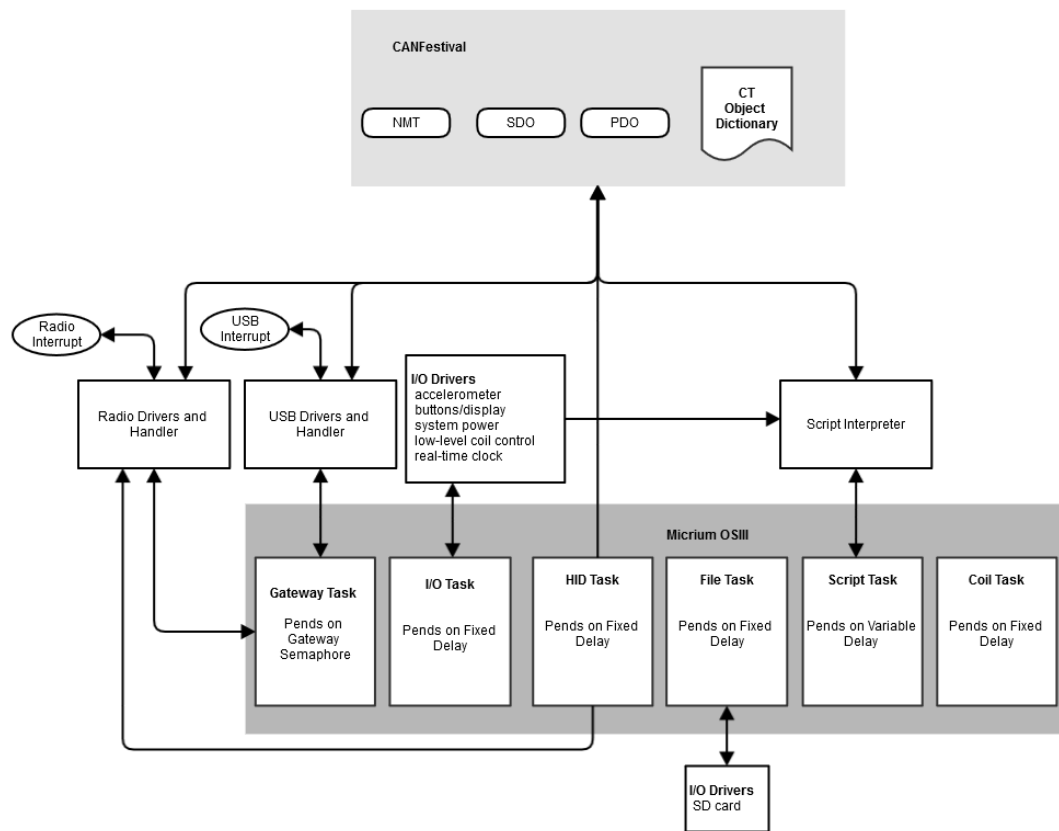
Data Storage on disk

Data storage on disk is an XML format that reflects the system schema encoded in the application. Only deidentified data will be stored.

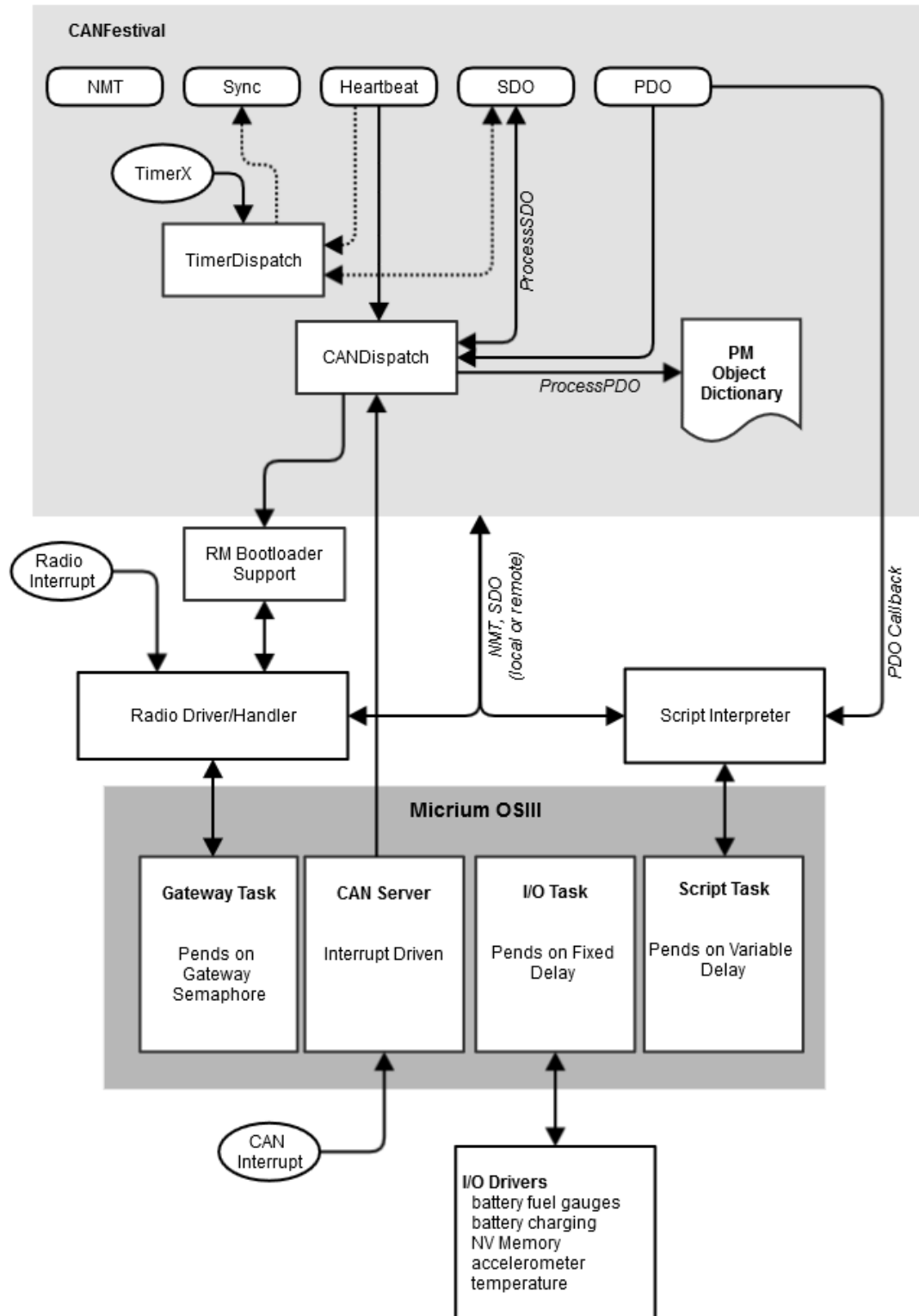
USB messaging

The USB messaging mechanism follows a standard ISO protocol stack and is based on the CANOpen protocol. See the protocol documents related to the Control Tower for a discussion of the meaning of the various layers represented in the protocol stack. Note that the CE App does not include an Object Dictionary. The application is considered a client-only instance in the CANOpen architecture, and as such, does not serve Object Dictionary information to the rest of the system. However, it does have an intimate knowledge of the system syntax and provides a help file to the user detailing all the Object Dictionary addresses available.

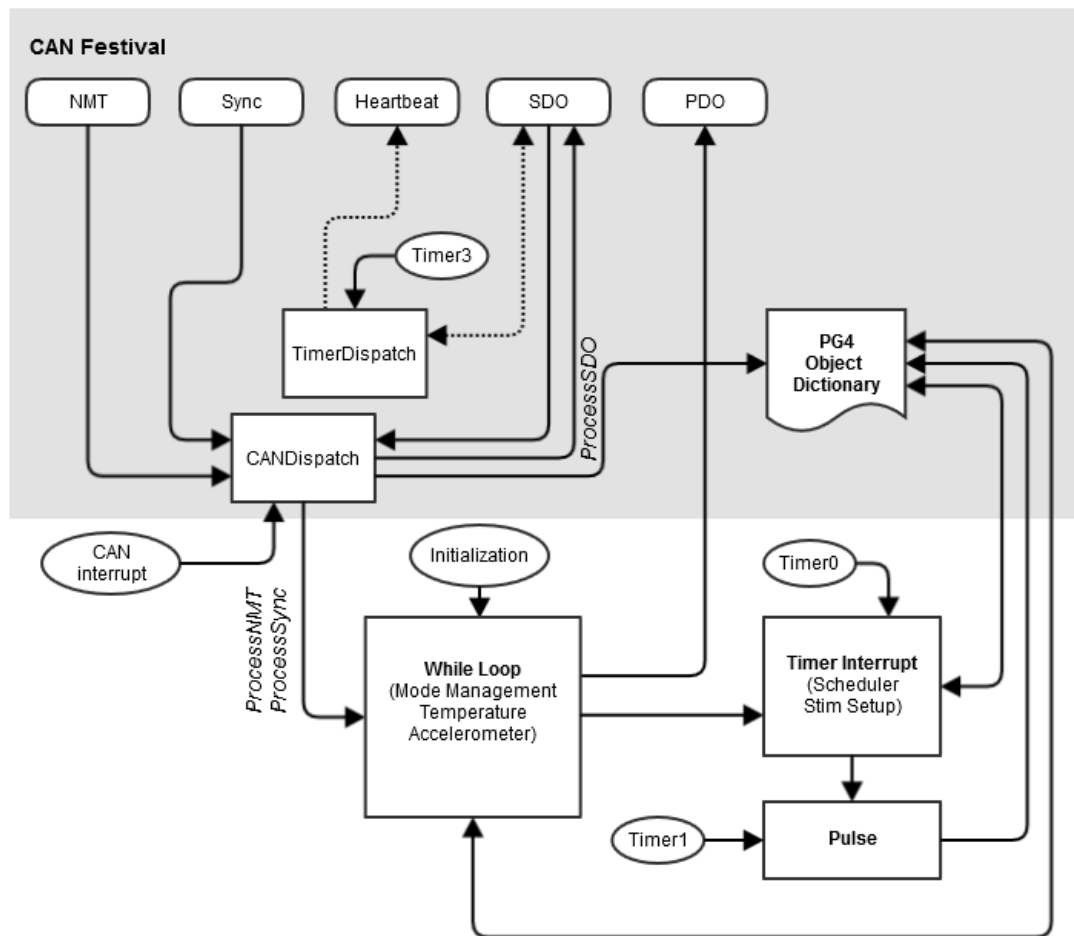
CT Software Architecture



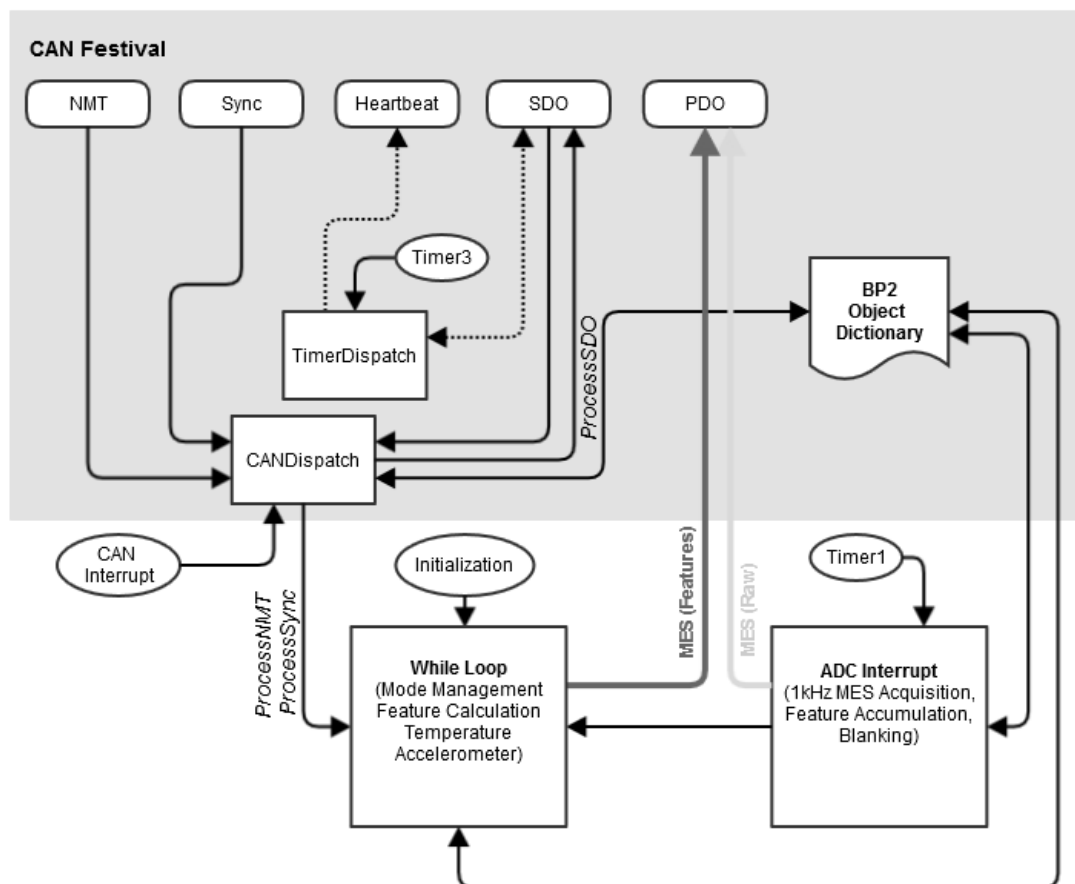
PM Software Architecture



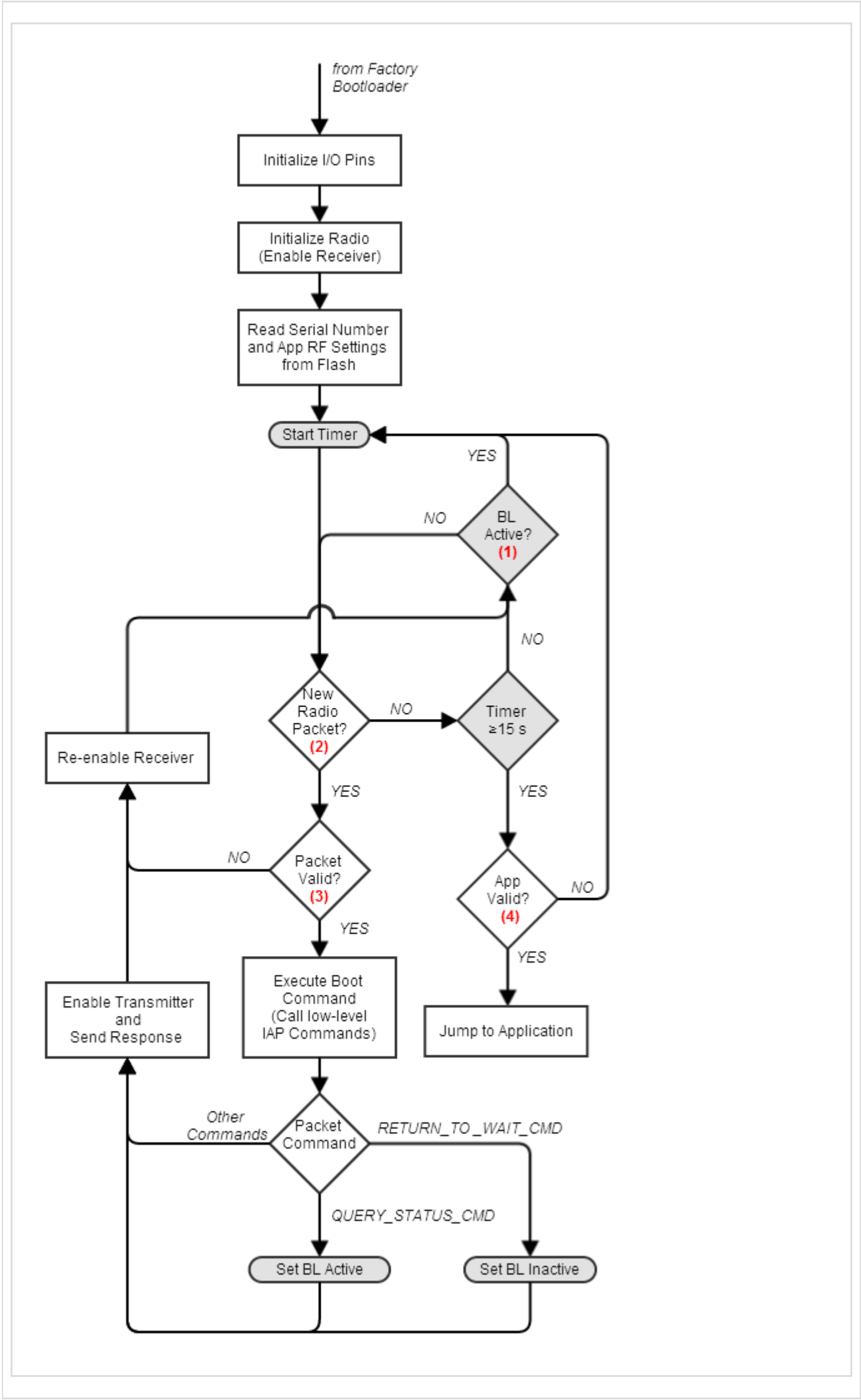
PG4 Software Architecture



BP2 Software Architecture



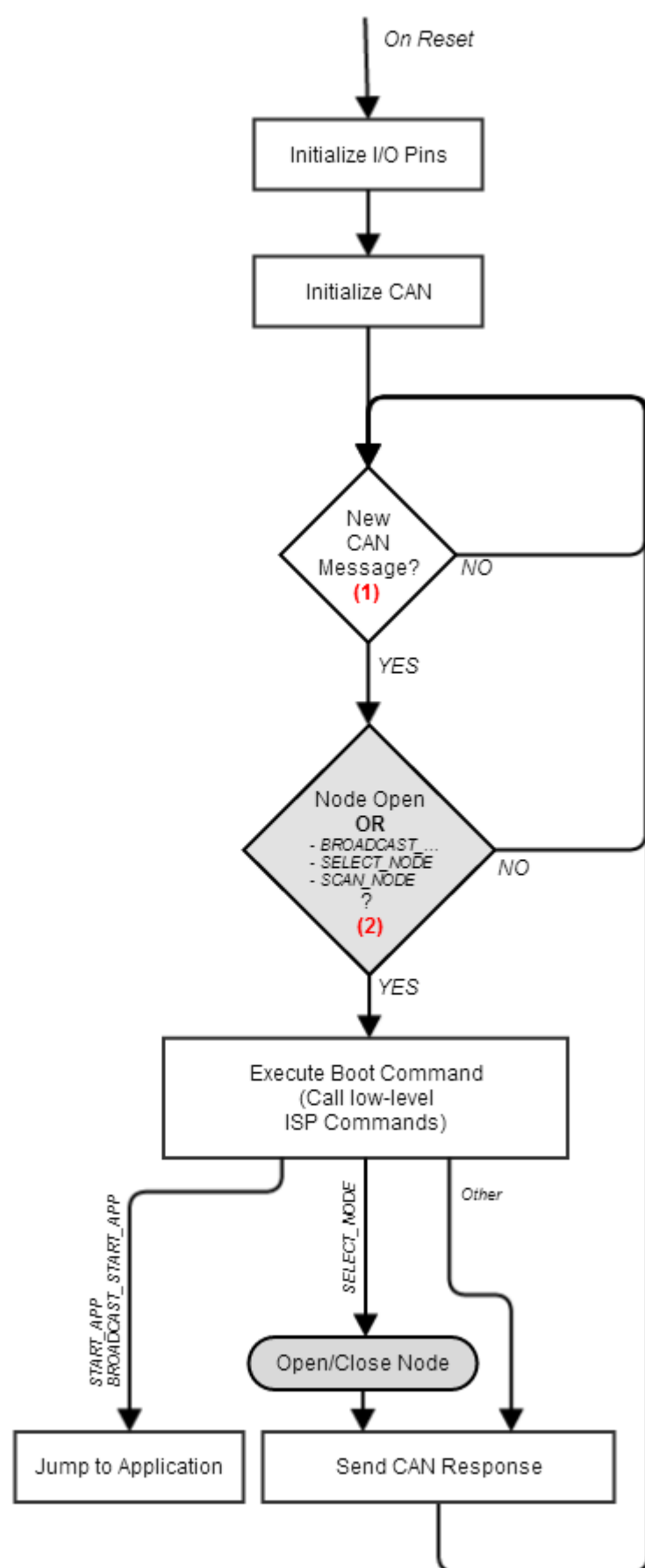
PM Bootloader Flowchart



1. BL status is inactive at startup
2. Radio packets with a bad CRC are discarded before arriving in the FIFO and are not detected here
3. The packet is considered valid if it exceeds the minimum length to contain a command byte
4. The application is considered valid if the Interrupt Vector Sum = 0

This page intentionally left blank.

RM Bootloader Flowchart



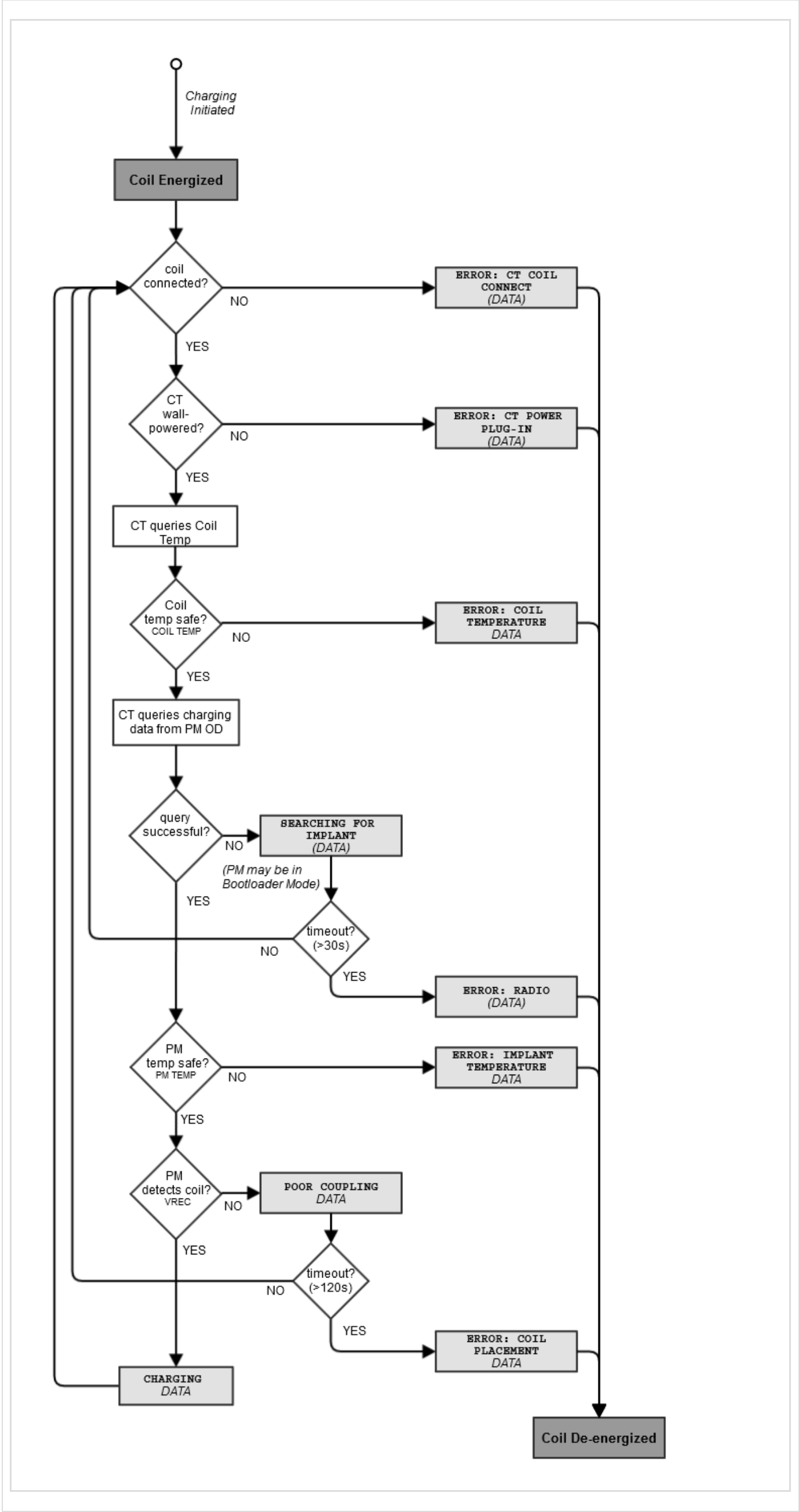
1. The RM BL uses a CAN acceptance filter. Only messages with an identifier of 0x14X are seen here. Messages with a bad CRC are discarded by hardware.
2. Most commands require a unique node to be opened. A unique node is obtained even if multiple nodes have the same node number by matching the Serial Number prior to opening.

This page intentionally left blank.

Charging Flowchart

This diagram shows the CT management of the charging process. The lightly shaded boxes show what the patient will see on the LCD display on the CT. DATA refers to pertinent charging data (battery voltages, charging current, implant and coil temperatures, near-field link received voltage, etc.). In some cases not all data will be available. If the CT fails to receive a valid response from the PM, the CT will enter the Radio Fault condition and de-energize the coil. If the PM was turned off (battery discharge or emergency shutdown), the coil is required to energize the PM. It will take 20 s until the PM exits Bootloader Mode and can be polled by the CT via radio. The patient/caretaker has up to 2 minutes to move the coil into position if the PM does not detect the coil. Note that the error conditions shown here are a risk mitigation strategy. Even without any software limits in place, the outer surface of a PM capsule did not increase by more than 2°C above ambient (See [Summary of Testing & Evaluation](#)).

The Charging Task loop is completed roughly every 200ms. During each loop, the task feeds a watchdog timer. If the watchdog goes for a specified amount of time without being fed, it resets the processor. The reset I/O settings of the processor result in the coil being de-energized. Thus, even under anomalous software conditions the coil cannot be left on unless actively charging an implant.



Name	Limit
PM detects coil	VREC > 2V
PM temp safe	PM Thermistor Temp < 40°C
Coil temp safe	Coil Thermistor Temp < 41°C

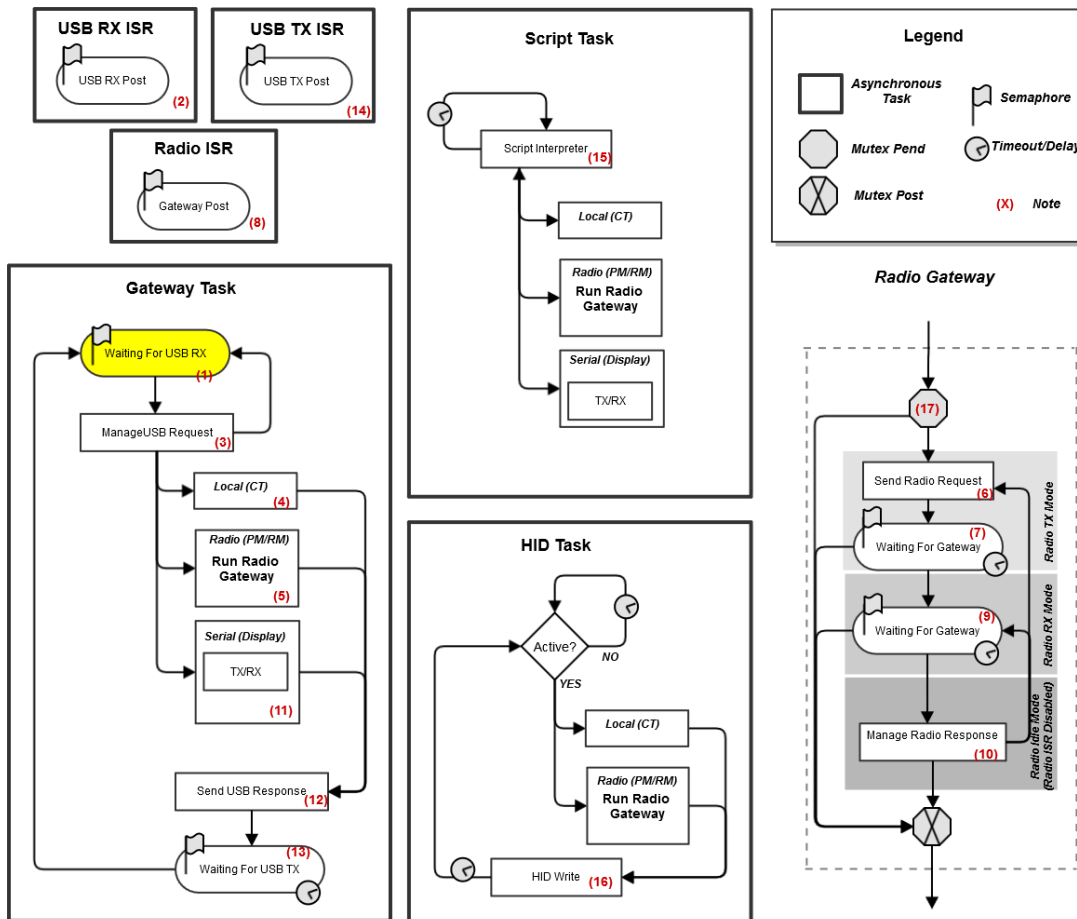
This page intentionally left blank.

CT Gateway

CT Gateway

The following Diagram shows the CT tasks involved in routing data to and from the CT.

- The Gateway Task supports routing data between the PC and the implant, CT, or display when used in the lab for parameter setting, programming, and system diagnostics.
- The HID Task supports routing data between the implant or CT and the PC for real-time display of data (e.g. raw full-bandwidth EMG)
- The Script Task supports routing data between the CT and implant or display when the CT is used in stand-alone operation during patient use.



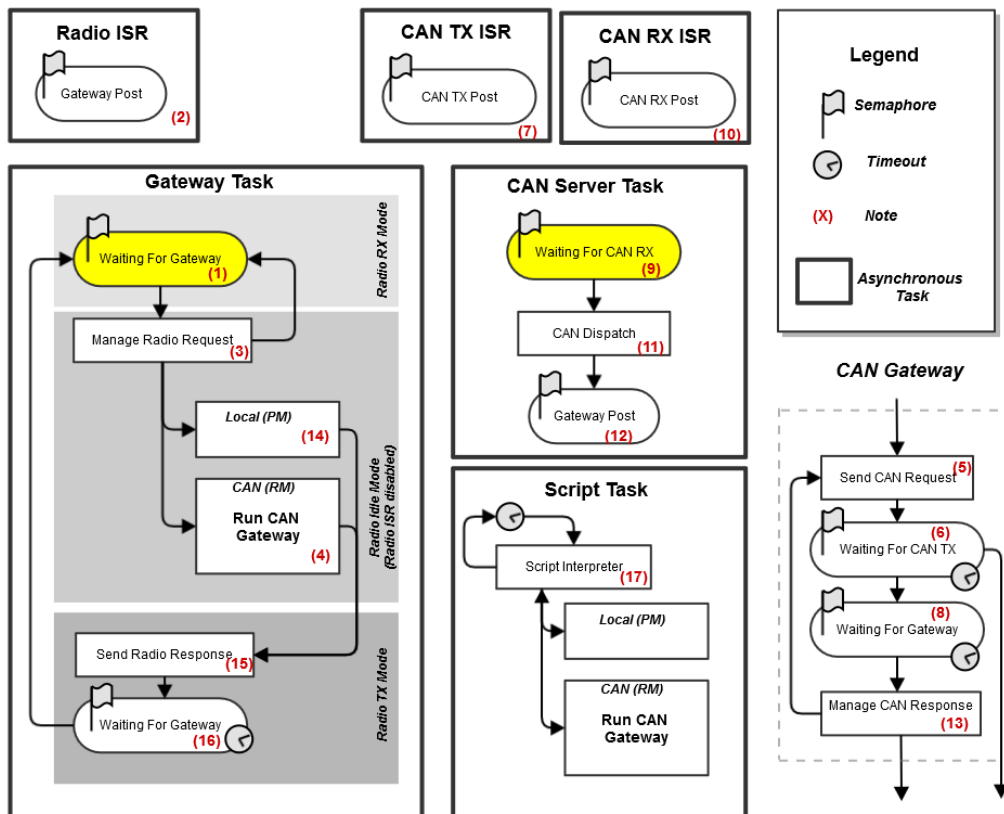
1. The Gateway Task will wait indefinitely for the arrival of a USB packet.
2. The USB Semaphore and ISR are handled within the Micrium BSP
3. The Gateway Task will then process the USB request, either locally, via Serial port, or via radio to PMs or connected RMs. If the message cannot be interpreted, the Gateway Task again waits for the next USB packet.
4. If local, the Gateway Task gets or sets the appropriate data from the CT
5. If radio, the Radio Gateway is entered
6. The Radio Gateway sets the radio to TX mode and sends the radio request
7. The Radio Gateway pends on the Gateway Semaphore, waiting until the packet has been sent. A timeout prevents waiting indefinitely if there is an unexpected error
8. The Radio ISR is triggered on the falling edge of GDO0 (indicates the receipt or completion of transmission of a radio packet if IOCFG0=0x06).

9. The Radio Gateway will wait again until the Gateway Semaphore is posted, indicating receipt of a new radio packet from the CT. The timeout here is dependent on the request.
10. The Radio Gateway then manages the radio response, either transmitting additional messages, receiving additional messages, or continuing.
11. If serial, the Gateway task reads or writes to or from the requested serial port
12. The Gateway Task completes by sending a USB response
13. The task waits until the USB response has been sent indicated by the USB TX semaphore
14. The USB TX semaphore is posted by the USB TX ISR
15. The Script Task asynchronously runs any scheduled scripts which also have access to the Gateway functionality of the Gateway Task. Note that the Script Interpreter may also be called directly by the Gateway Task in response to an event (e.g. button press polled from Display via Serial)
16. The HID Task also runs asynchronously and allows high speed USB writes (via the HID USB protocol) to the PC without the need for the PC to request packets as in the Gateway Task. The HID task also has access to the Gateway functionality except for the Serial Port.
17. The Radio Gateway is protected by a Mutex so that the Gateway, Script, and HID task cannot use it simultaneously.

PM Gateway

PM Gateway

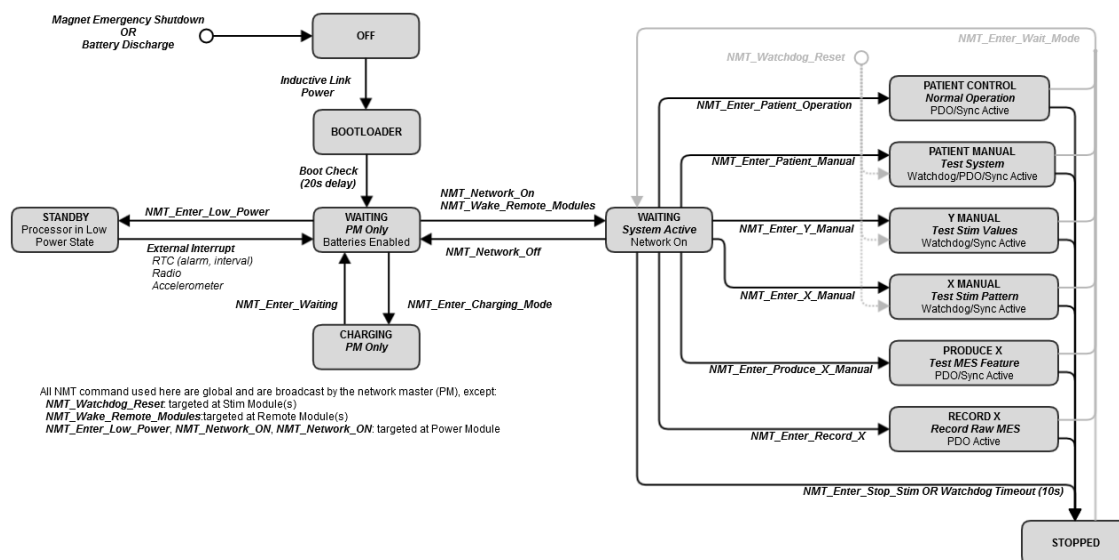
- The Gateway Task supports routing data between the CT and PM or RMs (via CAN). This may be done with the CT in stand-alone mode or connected to the PC.
- The CAN Server Task handles all incoming CAN packets including SDOS, NMTs, Heartbeats, PDOs, and bootloader commands
- The Script Task supports routing data between the PM and RMs. This allows the PM to function independently from the CT under certain conditions



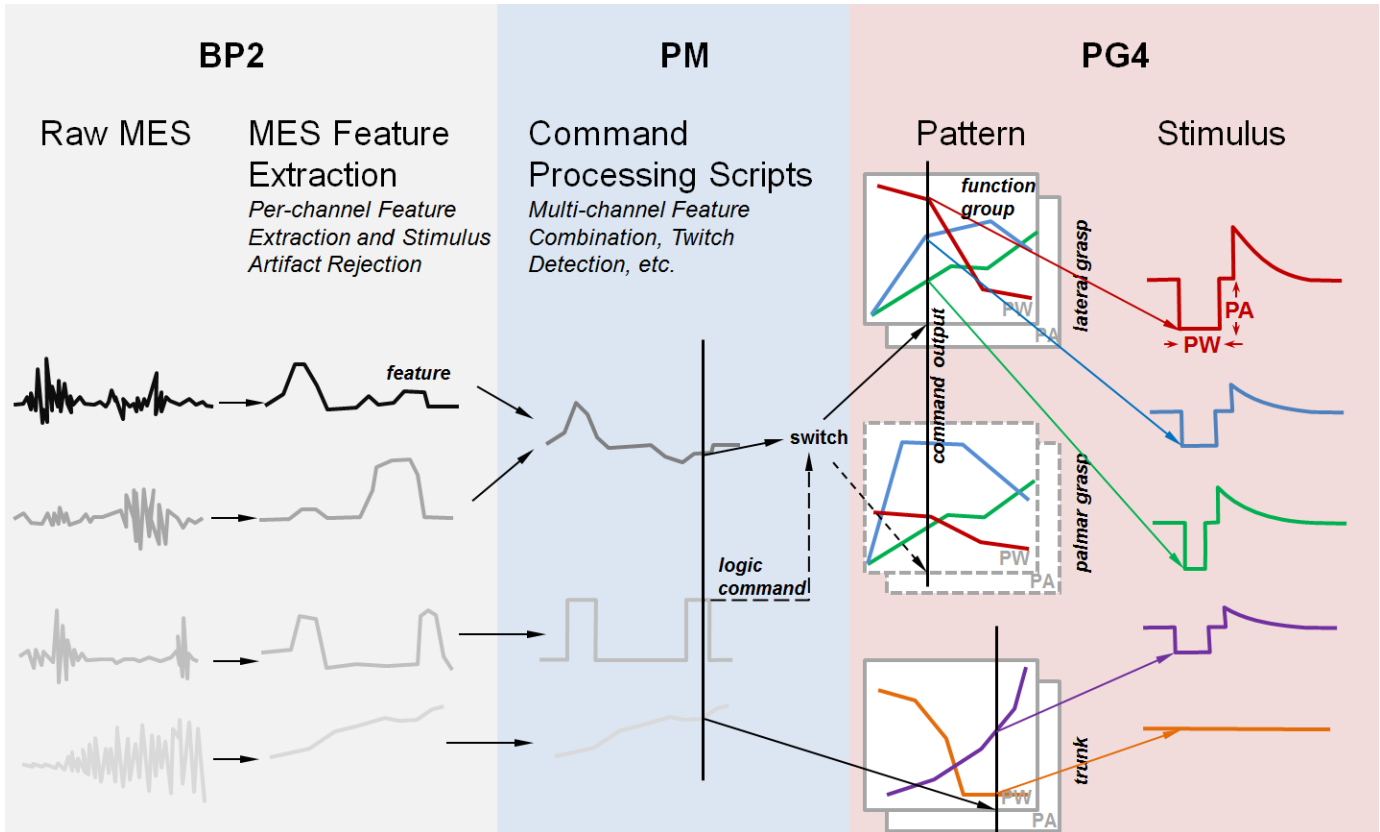
1. The Gateway Task will wait indefinitely until the Gateway Semaphore is posted, indicating receipt of a new radio packet from the CT.
2. The Radio ISR is triggered on the falling edge of GDO0 (indicates the receipt or completion of transmission of a radio packet if IOCFG0=0x06).
3. The Gateway Task will then process the request in the radio packet, either locally or via the CAN port. If the message cannot be interpreted, the radio is restored to RX mode, and the Gateway Task again pends on the Gateway Semaphore waiting for the next radio packet.
4. If CAN, the CAN Gateway is entered
5. The CAN Gateway starts the process of sending out an SDO, NMT, or Bootloader command
6. If the CAN transmitter is busy, it must wait until the transmitter is ready signaled by the CAN TX semaphore, or until the semaphore times out in the case that the network never returns to an idle state
7. The CAN TX semaphore is posted by the CAN TX ISR
8. If the CAN Gateway expects a CAN response, it will pend on the Gateway Semaphore. Otherwise it will exit the CAN Gateway. While waiting for a CAN message, the Gateway Task uses a timeout in the case that no CAN response is received. The timeout time is dependent on the CAN request. The timeout may occur because the node to which the CAN message is issued does

9. The CAN Server task pends on the CAN RX Semaphore
10. The CAN ISR posts the CAN RX Semaphore
11. CAN Dispatch handles the CAN message appropriately. Only some messages (SDO and RM Bootloader commands) post to the Gateway Semaphore
12. The Gateway Semaphore is only posted by the CAN Server Task when the Gateway Task is waiting on a CAN response, because this is the only situation in which a bootloader or SDO response should be issued by a remote module.
13. In some cases the Gateway will continue sending or receiving additional CAN messages as necessary
14. If local, the Gateway Task simply gets or sets the appropriate data on the PM
15. The Gateway task sends the radio response
16. The Gateway task waits for completion of the sent message indicated by the Gateway Semaphore
17. The Script Task can also use the CAN Gateway (asynchronously from the Gateway Task) to route data to and from RMs. Note that the script interpreter can also be called directly by the CAN Server Task upon receipt of PDOs.

PM Modes Diagram



NNP Distributed Processing



- A **function group** is a group of stimulus channels controlled by a single **command value** (via a pattern—or look-up table—for PW and PA, stored in each PG4). Logic commands can be used to determine which **function groups** are active. The **profiler** helps the clinician set the minimum effective (threshold) stimulation and maximal stimulation (without significant spillover) values. These values are displayed in the **pattern tab** (See [CE Software Architecture](#)) for reference, but the profiler range can be exceeded within the patterns if desired. Safe stimulation parameters (determined by electrode type) are insured by hardware design.
- Command Processing Scripts in the PM typically use **features** from the BP2 to generate **command values**. However, other system data such as time and accelerometer values can be utilized by the scripts to generate **command values** as well. **Features** from MES may include Mean Absolute Value, Mean Value, Zero Crossings, Slope Sign Changes, and Waveform Length.
- The stimulation frequency and recording interval is set for all channels based on a sync pulse period. Stimulus artifact rejection simply ignores samples within a pre-specified time window when stimulation is expected to cause an artifact.

Software Requirements Specification (SRS)

Overview

The target hardware including microprocessor and memory constraints as well as required programming language for each software component is shown in [Table H1](#). The purpose and intended user of each software component is described in [Software Description](#). Management of Memory Leaks is discussed in the [Software Development Environment Description](#).

The Software Requirements Specification for each software component as well as general functional, interface, and performance requirements are included as a column within the Traceability Matrix.

Table H1

		Clinical Programmer		Control Tower	Power Module		Remote Modules		
		RT App	CE App	CT App	PM BL	PM App	RM BL	PG4 App	BP2 App
Hardware Platform	Processor	ATmega328 (Arduino Nano 3.0)	PC	Phillips LPC2388	Phillips LPC2129		Atmel AT90CAN128		
	Architecture	8-bit AVR RISC Microcontroller	32 or 64bit Intel Processor	32-bit ARM7 RISC Microcontroller	32-bit ARM7 RISC Microcontroller		8-bit AVR RISC Microcontroller		
	Speed	up to 16 MHz	variable	up to 72 MHz (48 MHz used)	up to 60 MHz (10 MHz used)		up to 8MHz at 3.3V (1 and 4 MHz used)		
	Flash	32 KB	variable	512 KB	256 KB		128 KB		
	RAM	2 KB	variable	98 KB	16 KB		4 KB		
	EEPROM	1 KB	-	-	-		4 KB		
	CAN	-	-	2 (1 used)	2 (1 used)		1		
	I2C	1	-	3	1		1		
	UART	1	-	4 (2 used)	1		2 (not used)		
	SPI	1	-	1	2		1		
	USB	-	variable	1	-		-		
	Additional Flash	-	-	SD Card	4 MB		-		
	Additional RAM	-	-	-	256 KB		-		
Operating System		-	Windows 7/8 (32 or 64 bit)	Micrium uC-OSIII	-	Micrium uC-OSIII	-	-	-
Programming Language		C	C#	C	C		C		
Development Environment		??	Microsoft Visual Studio 2010	IAR Embedded Workbench for ARM	IAR Embedded Workbench for ARM		IAR Embedded Workbench for AVR		
Off-the-Shelf Software		-	.NET (v4) National Instruments Libraries	CanFestival	-	CanFestival	-	CanFestival	CanFestival
Software Architecture		-	CE Software Architecture	CT Software Architecture	PM Bootloader Flowchart	PM Software Architecture	RM Bootloader Flowchart	PG4 Software Architecture	BP2 Software Architecture

This page intentionally left blank.

Software Design Specification (SDS)

The Software Design Specification for each software component as well as general functional, interface, and performance requirements are included as a column within the Traceability Matrix, where each design specification is tied directly to a requirement.

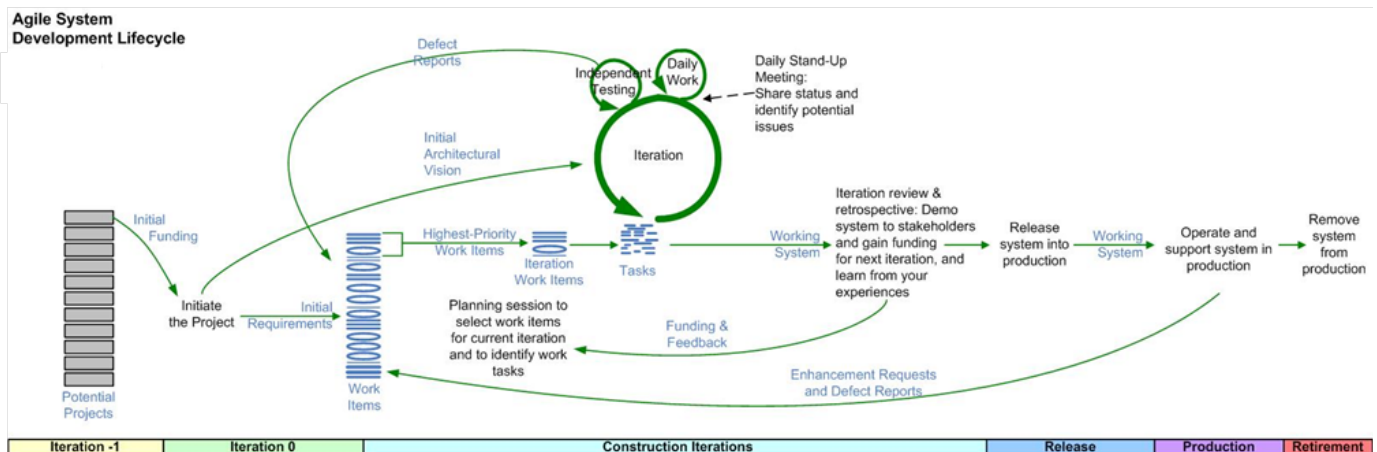
Software Development Environment Description

Software Development Life Cycle Plan

The software development life cycle model followed by the NNPS software development project most closely resembles the industry standard AGILE model. However, since the development team is small and the task has many separate components, some of the activities will also incorporate elements of the waterfall life cycle model. This combination of models will be achieved by appropriately scaling tasks and documentation so that each activity generates a document record.

This project will entail the following:

- Software Development Plan
- System Requirements Definition
 - Clinical and Patient Use Cases
 - Detailed Software Requirements Specifications
 - Derived from NNPS requirements and use cases
 - Includes design, functional, implementation, interface, performance, and physical requirements
 - Follows 21 CFR para 820.3(y)
- Software Architectural Design
- Detailed Software Design Specifications
- Phased software construction/coding/implementation and unit testing according to AGILE model
 - See diagram below
- Testing
 - Integration Testing
 - Verification Testing
 - Validation Testing
 - Software Validation Plan
 - Software Validation Procedures
 - Software Validation Records
- Release
- Installation
- Support
- Maintenance
- Retirement



The following table breaks down activities in relation to the first animal implant and first human implant. The initial planning horizon for this project ends with the IDE submission. Following phases include support for the first human implant and development of investigational tools for the principal investigators.

Phase	Summary of Activities	Deliverable(s)
0	Develop and plan NNPS SW project	Software development plan; system requirements; software architecture
1	Select embedded OS; identify development tools; develop first functional prototypes	Network architecture; detailed design specifications; working drivers and first pass object dictionaries
2	Identify thin application functionality; implement PC and OS based networking; begin functional test cycle	Specification of first major functional deliverables; test plans and testing for network capability
3	Identify IDE functionality; animal/bench testing	Thin application client for animal/bench testing
4	Implement IDE functionality	Code freeze and start of verification and validation testing
5	IDE release	Final testing and IDE submission
6	IDE support	Installation and Support ongoing IDE trial, Bug Fixes,
7	Identify and implement	New Features within anticipated changes

Bi-weekly software development meetings are held with all team members present to discuss project progress, changes to documents, changes to source code, and task assignments. Meeting notes are used to document and communicate the content of the software development meetings and are maintained on a Wiki (Confluence Server).

Ongoing financial support for the Software team?

Configuration Management

Issue Tracking

Bugs are identified by stakeholders (clinician, engineer) via an online bug reporting system (JIRA). Patients will be requested to report any unexpected behavior to the clinical team. New features will also be requested in this manner.

The software can be broken up into several categories:

- Clinician configured patient-specific scripts
- User Interface (CE GUI, CT display/button management)
- Database (underlying XML that stores system configuration information)

- Off-the-shelf software (Micrium, CANFestival)
- Embedded Application code (CT, PM, RM)
- Embedded Bootloader code (PM, RM)
- Low-level drivers (CT, PM, RM)

The software team is responsible for identifying the category or categories into which the issue falls and select the appropriate team member(s) to address the issue.

Revision Control

This project will use the Subversion revision control system for code storage. Subversion automatically maintains a history of changes made to files such that a complete historical record is created. Software development team members will “check in” (i.e. commit) source code as it is developed into the designated Subversion repository. Repositories are located at jdc-consulting.unfuddle.com.

The software revision history maintained by Unfuddle identifies a given Subversion revision that is used for a release of the software and firmware components for distribution, clinical investigation, or other validation testing. A separate build management system will be used to record and release all components for a given software release.

The following information must be identified for each release:

- Software/firmware version number
- Release status and date
- Test status
- Overview of major software/firmware changes

Build and Source Files

Daily work directories will be backed up to Unfuddle. A separate build management system will be used to record and release all components for a given software release. It's URL is release.jdc-consulting.us

Maintenance

For the purposes of this IDE, the underlying operating systems will not change. Within the scope of this IDE, maintenance is limited to bug fixes and new features outlined within Anticipated Changes

Test Code and Test Stubs

Test tools, including software testing artifacts, will be stored in separate Subversion directories on Unfuddle. Any software release will identify the corresponding revisions of the test tools and artifacts.

Integration, Verification, and Validation Testing

Testing must be performed to ensure that the delivered software and firmware components performing according to their specifications. Specifically, each requirement in the respective Software Requirements Specification of each software and firmware component must be verified.

Integration, verification, and validation testing shall be performed on target hardware on the bench or in simulated use testing. Normal operating conditions, unexpected inputs, and simulated fault/error conditions must be tested. Final software and firmware with no special provisions must be used. A software requirements tracing matrix must be used to record all links between requirements, specifications, test procedures, and test results.

Executable Code

A separate build management system will be used to record and release all components for a given software release. Individual executables will be delivered to the Product Project Leader, who will be responsible for downloading executables to the appropriate hardware. The Product Project Leader will also be responsible for verifying that software and hardware revisions are compatible.

Software Development

Tools

The Software Development team uses Microsoft Visual Studio to develop the PC application (Control Editor) and IAR Embedded Workbench for the embedded applications (CT, PM, bootloaders). Both of these tools mainstream and consistently used for commercial products and provide exceptional debug and product support.

Source Code Documentation

Source code is commented in-line. We have chosen to implement a documentation package called Doxygen to assemble these in-line comments into standalone documents. This documentation is highly useful for generating function call maps and listing global variables.

Design Considerations

- Micrium was chosen as the operating system for ARM processors (PM and CT). Micrium has a track record in safety critical systems and provides source code. A primary benefit of the Micrium choice is its pre-emptive capability, which means that it has a deterministic response to changing process conditions.
- No operating system was chosen for AVR processors (PG4 and BP2). Because of the low complexity of the applications targeted to this processor, an operating system would have resulted in unnecessary overhead in processing time and memory.
- We have chosen to not use any compiler optimization. This reduces the object code complexity and minimizes the possibility of undetected race conditions and reduces some of the troubleshooting issues.
- We have chosen not to use dynamic allocation of memory in any of the embedded applications. Because all memory is pre-allocated by the compiler, there is no possibility of memory leaks.

Verification and Validation Documentation

Verification

Verification documentation is supplied within the NNP design database in a Traceability Matrix format alongside the SRS and SDS.

Validation

During the EFS IDE trial, the clinicians and software engineers will continue to work closely together to assure that the software meets the patients' and clinicians' needs. Validation documentation is captured within our bug/issue/improvement tracking database and through the addition or modification of the the software requirements specification, software design specification, and testing entries within the NNP design database

Revision Level History

Overview

Revisions are managed via subversion (SVN). Source code is hosted on a secure Unfuddle Server as described in the [Software Development Environment Description](#).

The ability to modify the software is a critical to the success of this Early Feasibility Study. New revisions will include bug fixes and new or revised features based on the information learned through the first-in-human study. New revisions will only be implemented in-human as part of a certified release (See [Software Development Environment Description](#)) that has undergone basic functionality and compatibility testing between its components. As described in the Hazard analysis, erroneous applications can only impact efficacy of the device, not safety. If problems are found with a certain release in-human, either an older working release or an updated tested release that addresses the problems can be downloaded to the implant.

The PM and RM bootloaders are the only software components that cannot be modified after implantation and are also the only safety critical components. Erroneous behavior of the bootloaders could result in an inability to communicate with the implant or to reprogram the implant (which would require an elective replacement surgery to replace the affected module)

Below, we include the revision level history for the bootloaders. All bootloader tests were performed on the latest revision. Application (non-bootloader) code revision level history can be provided on request.

PM Bootloader

rev	date	change log
46	April18,2014	Added Serial Number JIT module
45	April15,2014	added commandbyte cmd to allow reset to od defaults
44	April02,2014	forced IDLE mode before TX, updated SVN rev
43	April02,2014	Set MCSM1.RXOFF_MODE to 3 ("stay in RX mode") to fix bug of going into unrecoverable IDLE mode after receiving a message with bad CRC (Also changed IOCFG0 mode to 7 during RX). updated SVN number.
42	March30,2014	changed SN address
41	March24,2014	corrected SN write. added radio response if read request for 0 bytes. increased tx power to 0x60 (~ 0 dB). added Functional Spec document
40	March21,2014	moved SN to 0x0003DF00. some error checking added. all commands now result in a radio response.
39	March07,2014	Added doxygen comments to each function, renamed some functions/variables for consistency, updated wrlImageData to report failure to calling function
38	March06,2014	enabled CRC autoflush and set PKTLEN to 0x3F
37	February17,2014	removed radio response from wrlImageData. Corrected initial pin settings for SPI1 in PINSEL1.

36	September02,2013	fixed channel reporting
35	August31,2013	added SN lockout.
34	August29,2013	changed channel number to 5
33	August14,2013	fixed blank check and serial number functions. errata found on documentation on number of bytes returned from IAP calls.
32	August13,2013	added RF values
31	May08,2013	added serial number write/read
30	May06,2013	everything except for non-volatile storage of s/n
29	April30,2013	modified main.c for new io definitions based PM1B
28	April20,2013	medradio
27	June10,2011	added icf to svn
26	June10,2011	added eww
25	June10,2011	changed return address for radio to Tower, 0xbd. changed max data return len to 59.
24	June08,2011	changes to ewd, ewp, and .hex file from 18months ago.. don't recall specific changes. sorry.
23	September08,2009	
22	July31,2009	removed unused files
21	July31,2009	changed /cs from p0.22 to p1.23. incorpd new default ioinit settings for p0&p1. converted most io ops to iar-style and located most hardware defines at top of cc file.
20	July08,2009	added iopinrw.h file to proj
19	July08,2009	added timer0 reset before running the app to simulate more a hardware reset. changed default clock to 10MHz for power module, not eval boards.
18	December12,2008	upgraded fosc macros to handle new freqs. changed timeout to 20 secs.
17	December10,2008	added startup to change interrupt vectors to 0x2000 section.
16	December05,2008	MOVED boot2 segments from 0-3 to 0
15	December02,2008	converted to run IAR, saved keil to tag..
14	December02,2008	saving keil version
13	December02,2008	deleted unused code/comments

12	November21,2008	updated wr/rd commands to multipkt and added query. tested for correct params, not for errors yet.
11	November18,2008	tested tx,rx using read command @250KBd.
10	November17,2008	made tx/rx routines to handle 250khz w/variable buffers <64 bytes. still testing.
9	November10,2008	added Rx routines, untested.
8	November10,2008	make routines to handle fifo when txing more then 64 bytes
7	November08,2008	Boot is talking over Radio to base. Configured Addressed and variable length messages to 196 bytes.
6	November04,2008	finished boot commands, modified radio structures, tested flash erase/write/read.
5	October28,2008	added iap command basics
4	October27,2008	ADDED radio files, readregs from radio, handshake commands w/radioSO pin-tested.
3	October27,2008	Set/measure time base, toggle io pins, turn off PLL,
2	October24,2008	config and testing of Spi port.
1	October22,2008	

RM Bootloader























rev	date	change log
56	June02,2014	changed notes on Bootloader Lock Bit
55	June02,2014	Corrected build options
54	June02,2014	Workaround for writes on page boundaries (0xe000-0xffff). Changed MEM_FLASHPROTECT for improved consistency with higher level programming pieces. Enforced BL write protect lock bit within code.
53	May28,2014	Fixed bug in isp_write_block
52	May23,2014	moved clock prescaler code to first line in init code. Added not on CKDIV8 fuse
51	May22,2014	Fixed checksum initialization and clock
50	May19,2014	Restored (reduced) delay in can_display_data (~46ms delay between packets reduced to ~5ms)
49	May19,2014	Removed SCAN_NODE command (redundant with SELECT_NODE). Removed delay timeout in can_display_data

48	May14,2014	doxygen comments added, NODE_SELECT modified in preparation for removing SCAN_NODE, some unnecessary variables removed
47	May07,2014	Node Open/Close modified. SN=0, returns SN, SN=SNmod opens Node
46	May06,2014	Changed CRC to simple 1 byte checksum, allowed bypass if sizeApp =0 or 0xFFFF (never programmed), added DEFAULT_OD command
45	May01,2014	Partial (requires modifications to RMBootloader.c in PM/CT). Node change and CRC set Commands added.
44	April30,2014	partial (not fully functional)- stripped SSB, setup protected flash space for NNB and potential CRC/checksum
43	February17,2014	corrected CAN bittiming (to match application settings at 1MHz), updated SVN rev
42	October08,2013	built RMBootloader.hex from latest source, update rev
41	October08,2013	fixed multiple node start issue (node 3)
40	October07,2013	added WHO function
39	October07,2013	rebuild with correct project settings
38	October07,2013	build with updated pin settings, rev 37 did not update RMBootloader.hex, so pin settings were not corrected. LED is now off during bootloader. nSTIMLE now set low (active) in Bootloader
37	September05,2013	toggle PORTD.5 (CAN_TX) in AT90CAN_GPIO_NA_InitStatic(), updated rev# in config.h
36	September04,2013	changed configuration data on static init.
35	September02,2013	
34	August15,2013	added more security for duplicate node.
33	August14,2013	UPDATED HEX FILE.
32	August11,2013	documentation update, no functional modifications
31	August11,2013	documentation update, no functional modifications
30	May08,2013	updated SVN rev number
29	May08,2013	renamed output file to RMBootloader.hex
28	April20,2013	
27	April17,2013	added SVN reference
26	April11,2013	added delay on read reply

25	April09,2013	NNP adds for serial number and broadcast restart.
24	March24,2013	added SLIM bootloader files
23	May19,2012	updated bootloader notes and files
22	February14,2011	The changes for revision 21 were not checked into unfuddle correctly. This version implements these changes:
21	March20,2009	- Removed hardware condition checking
20	March12,2009	Version sent to independent tester for intermediate test
19	March12,2009	Removed test LED's
18	March12,2009	Update CAN parameters for 1MHz clock and 100K CAN
17	March12,2009	Removed test LED's
16	March12,2009	Removed test LED's
15	February06,2009	Meeting minutes from the Remote module bootloader specification review on 1-27-09
14	February06,2009	Added diagrams for startup conditions and a new state diagram showing the timeout exit from bootloader
13	February06,2009	First edits of specification based on the design review
12	January28,2009	
11	January15,2009	initial version
10	January15,2009	initial version
9	January15,2009	initial version
8	January15,2009	Created new file for isp functions.
7	January15,2009	Removed the scheduler
6	January15,2009	Removed references to old protocol
5	January09,2009	Removed UART references
4	January09,2009	Removed UART references
3	January09,2009	Removed UART references and some test code
2	January09,2009	This version fixes the bugs from the original Atmel bootloader.
1	January09,2009	Initial version of Remote Module bootloader from Atmel

Unresolved Anomalies (Bugs or Defects)

As described in the [Software Development Environment Description](#), the source code for all Software components is maintained under revision control. Software bugs are tracked within an online database. Unresolved bugs are listed below:

Key	Summary	T	Created	Updated	Due	P	Status	Resolution
XS-36	CT reverts to "Test Produce X" after RM Bootloader		Aug 11, 2014	Aug 11, 2014			OPEN	Unresolved
XS-35	New Value Control Entry X range		Jul 31, 2014	Jul 31, 2014			OPEN	Unresolved
XS-32	Cannot send messages to COM0 Display		Jul 28, 2014	Jul 29, 2014			OPEN	Unresolved
XS-31	Channels Tab doesn't update when Module added		Jul 24, 2014	Jul 24, 2014			OPEN	Unresolved
XS-28	Script Task and Gateway Conflict in PM		Jul 09, 2014	Jul 09, 2014			OPEN	Unresolved
XS-18	Sensors View Graph doesn't work in 64-bit		Dec 17, 2013	Jul 24, 2014			OPEN	Unresolved
XS-10	Pattern YAxis on refresh		Dec 12, 2013	Aug 01, 2014			OPEN	Unresolved
XS-8	Command Mode Slider Issues		Dec 12, 2013	Jul 24, 2014			OPEN	Unresolved
XS-7	Pattern Amp not correct		Dec 12, 2013	Aug 01, 2014			OPEN	Unresolved
XS-6	Pattern Mode fails to update		Dec 12, 2013	Jul 24, 2014			OPEN	Unresolved
XS-5	Pattern Amp interpolated Wrong		Dec 12, 2013	Jul 30, 2014			OPEN	Unresolved

11 issues

Software Testing

The Software Testing for each software component as well as general functional, interface, and performance requirements are included as a column within the Traceability Matrix, where each test may be tied to several requirements. The pass/fail criteria are specified within the requirement. Note that much of the software testing is actually system testing, relying on multiple software and hardware components.

For the purposes of the Early Feasibility IDE, it is recognized that ongoing improvement of the software will be necessary. Prior to the start of the IDE, all requirements with priority levels marked "Critical" and "Major" must be verified. No subjects will be entered into the study until these requirements are met.

Traceability Analysis

The traceability of requirements to design and testing is maintained within our NNP design database and shown in the following pages. Each row shows the originating requirement (SRS), the rationale for this requirement, the importance for this requirement (safety and basic functioning), how the requirement was implemented (SDS), the testing protocol document ID for the requirement, and the results of the test.

Key	P	ReqType	Summary	Requirement (SRS)	Min	Value	Max	ResNum	Value	Rationale	Design (SDS)	Verification	Test Results
Software													
PM Bootloader													
	Critical	Other Interface	Radio	PM Bootloader shall provide basic radio functionality to receive and send radio messages to/from programming device. Interface parameters (e.g. channel, addresses, TX power) shall be hardcoded.						Radio is only means of communication between external programmer and implant	Custom drivers		PASS
XH-103	Critical	Other Interface	pin configuration	PM Bootloader shall initialize processor pins so that they are not floating or contending						Reduce operating current	See PRJ-NNP-SYS-SPEC-20 Bootloader Specification, Power Module, P	Verified by inspection.	21365199
XH-106	Critical	Functional	battery	PM Bootloader shall not enable charging/discharging of battery pack.						Safety: Charging Coil provides sufficient power for bootloader operation	Battery connection disabled under reset (default) conditions		21365225
XH-108	Critical	Functional	entry	PM Bootloader shall always be entered on reset						Guarantee access to BL from known state	Code begins at 0x0000		21365199
XH-104	Critical	Performance	exit	PM Bootloader shall start application after inactivity timeout		15				No user input required to jump to application	Jump to 0x2000 ("Start Application")		21365199
XH-105	Critical	Functional	erase/write memory	PM Bootloader shall support erasing and writing Application Flash Sector's other than where PM Bootloader code is located						Must not overwrite itself	Precompiled Flash library protects sector 0 (0x0000-0x2000)		21365225
XH-107	Critical	Functional	read memory	PM Bootloader shall support reading from all on-board Flash						Includes App radio settings	Precompiled Flash library		21365225
XH-109	Critical	Functional	Serial Number	Unique SW Serial Number shall be stored in same sector as PM Bootloader in PM processor. Serial Number shall match HW Serial Number marked on PCB.	1		65535	1		Identification of module in situ	[See PM Bootloader installation procedure]165 15239]		21365199
XH-139													
HW Application													
HW drivers													
	Minor	Other Interface		PM App shall provide driver support for accelerometer, thermistor, battery fuel gauges, radio, off-board RAM, and off-board Flash						Required to utilize HW	Custom Drivers		14582725
XH-121	Trivial	Other Interface	CANOpen	PM App shall use CANOpen as communication protocol: operate as NMT Master, consume Heartbeat messages, produce SYNC messages, consume PDOs, and operate as server and client when handling and initiating SDOs						Support node list of all correct nodes, set Implant operational modes and respond to system mode changes, route messages to/from RMs, receive sensor data from RMs, and synchronize activity of RMs	CANFestival stack	Verified by inspection.	Open
XH-125	Trivial	Functional	Entry	PM App shall always be entered from PM bootloader						Known state on startup	See PM Bootloader (App starts at 0x2000)	Verified by inspection.	Open
XH-122	Trivial	Functional	RTOS	PM App may use RTOS to manage tasks						Simplifies programming of asynchronous tasks, protects against certain programming errors			Open
XH-126	Trivial	Functional	Startup	PM App shall execute a startup procedure to configure settings from non-volatile memory						Patent specific settings are preserved while powered down	RestoreList in OD 0x2900, data stored in Flash		Open
XH-129	Trivial	Functional	Scripts	PM App shall store and execute scripts						Configurable command/control methods to support research applications	Stored in Flash, executed via ScriptInterpreter		Open
XH-130	Minor	Other Interface	Radio	PM App shall store and allow modifying Radio settings						Allows different systems to be on different channels and/or different addresses to eliminate crosstalk between systems	PM OD 0x2600.1-4		21365239
XH-131	Trivial	Functional	Data logging	PM App shall support data logging to off-board Flash						?			Open
XH-132	Major	Functional	RM Bootload	PM App shall support routing RM Bootloader commands from programmer to RM						Provides data for research and for design iteration	PM Gateway		21365199
XH-133	Major	Functional	FESCAN	PM App shall manage FESCAN network, set Network Voltage:	4.6		9.5	0.1	V	External programmer cannot connect to RMs directly while in situ so commands must be routed through PM	PM OD 0x3010.1, 0x3000.1		21365299
XH-134	Minor	Performance	VNET default	Begin switching at a low VNET voltage setting to avoid the EL7156 High power mode.	4.5		6.3		V	Enables turning on/off or adjusting Network Voltage to conserve power.	See 7569707		21365299
XH-49	Trivial	Functional	RTC	PM App shall manage real-time clock						Used to timestamp events for data logging and to trigger scheduled scripts.	?		Open
XH-135	Trivial	Functional	Battery	PM App shall manage battery pack, enable/disable charge/discharge						PM must be powered off of batteries if external coil is removed. Charging current is managed by PM to maximize coupling efficiency.	PM OD 0x3000		14582725
XH-136	Trivial	Functional	Low Power	PM App shall support low power mode and waking via radio and/or accelerometer and/or RTC						Conserve power	Microcontroller supports low power modes. Radio and Accelerometer tied into microcontroller exten	Open	Open
XH-159	Major	Functional	Charging	PM shall support reading temperature and charge level while charging/discharging						Additional safety precaution to monitor implant heating	Protected Charging Mode		22347843
XH-140													Open
XH-160	Trivial	Functional	OD							?			Open
RM Bootloader													
	Critical	Other Interface	CAN	RM Bootloader shall support basic CAN functionality to receive and send CAN messages to/from programming device. Interface parameters (e.g. bit timing, acceptance filter, identifiers) shall be hardcoded						CAN is only method of communication between RM and other devices	Atmel CAN drivers from original Bootloader.	Verified by inspection.	PASS
XH-57	Critical	Other Interface	pin configuration	RM Bootloader shall initialize processor pins so that they are not floating or contending						Reduce operation current	Common pin settings for BP2 and PG4 defined in <RMBL Pin Settings>	Verified by inspection.	PASS
XH-67	Critical	Functional	module select	RM Bootloader shall provide a method to uniquely select a module within a network for performing bootloader operations individually						Allows programming, querying, or starting App independently	Module must be selected (using node number and serial number) for most!		21365199
XH-155	Critical	Functional	clock	RM Bootloader shall operate with a minimum clockspeed for CAN operation	1					Reduce operating current	CLKPS set to "divide by eight" from 8MHz oscillator within initialization routine	Verified by inspection.	PASS
XH-62	Critical	Functional	entry	RM Bootloader shall always be entered on reset of the processor						Guarantee access to BL from known state	BOOTRST fuse programmed. (See RM Bootloader installation procedure)1		21365199
XH-58	Critical	Functional	exit	Application" command.						BL mode is maintained unless expressly exited.	Jump to 0x0000 ("Start Application") is initiated by a specific command via		21365199
XH-61	Critical	Functional	erase/write memory	RM Bootloader shall support erasing and writing Application Flash and EEPROM						Load new binary images and non-volatile settings to module	Flash and EEPROM drivers from original Bootloader		18088026
XH-59	Critical	Functional	read memory	RM Bootloader shall support reading Application Flash, Boot Flash, and EEPROM						System diagnostics	Flash and EEPROM drivers from original Bootloader		18088026
XH-60	Critical	Functional	protected	RM Bootloader code shall reside in write protected Boot Flash section						Cannot be overwritten without JTAG connection (removed during manufacturing)	BLB1 (Boot section lock bit) set to SPM_Disable after Bootloader is instalk	Verified by inspection.	PASS
XH-63													

Key	P	ReqType	Summary	Requirement (SRS)	Min	Value	Max	ResNum	Value Rationale	Design (SDS)	Verification	Test Results
XH-64	Critical	Functional	serial number	Unique SW Serial Number shall be stored in write protected Boot Flash section in RM processor. Serial Number shall match HW Serial Number marked on PCB.	1		65535	1	Record keeping. Also utilized to select unique module in the case of duplicate node numbers	Stored at 0x1FFFD-0x1FFF. [See RM Bootloader installation procedure]	21365199	PASS
XH-65	Critical	Functional	module type	Module type shall be stored in write protected Boot Flash section in RM processor					BP2 and PG4 share common bootloader. Module type insures programmer loads correct Application.	Stored at 0x1FFFF. [See RM Bootloader installation procedure]	21365199	PASS
XH-66	Major	Functional	node number	RM Bootloader shall support changing the CAN node number	1		127	1	Each node in system must be unique for CAN Festival	Node stored in last page of Application Flash (0x1DF00).	18088026	In Progress
XH-68	Minor	Other Interface	BP2 Application HW drivers	BP2 App shall provide drivers for temp sensor, accelerometer, and programmable gain wiper.					Required to utilize HW	Custom drivers	16515185	Open
XH-69	Trivial	Other Interface	CANOpen	BP2 App shall use CANOpen as communication protocol. operate as NMT Slave, produce Heartbeat messages, consume SYNC messages, produce PDOs when required, and operate as server when handling SDOs					Respond to system mode changes, synchronize MES recording interval, deliver sensor data to PM, and route messages to/from PM	CANFestival stack		Open
XH-123	Trivial	Functional	Entry Startup	BP2 App shall always be entered from RM Bootloader					Known state on startup	See RM Bootloader		Open
XH-71	Trivial	Functional	Sample Features	BP2 App shall execute a startup procedure to configure PDO settings and other configuration settings from nonvolatile memory					Patient specific settings are preserved while powered down	RestoreList in OD 0x2900, data stored in EEPROM		Open
XH-72	Minor	Functional	Raw	BP2 App shall sample MES on two channels	1				Two MES channels in HW	PDOs transmitted every 8 samples at 1kHz		Open
XH-73	Trivial	Functional	OD	BP2 App shall support MES feature calculation. Synchronization with PG4 shall be supported to allow ignoring samples corrupted by stimulus artifact					Local feature calculation minimizes required network bandwidth	5 optional blanking windows per MES channel are scheduled off of SYNC message		Open
XH-74	Trivial	Functional	PatternSim	BP2 App shall support Raw sampling of one channel at full bandwidth					System diagnostics, research purposes	PDO transmitted every 8 samples (containing 8 samples) at 1kHz		Open
XH-75	Trivial	Functional	PG4 Application HW drivers	PG4 App shall provide drivers for temp sensor, accelerometer, and compliance voltage DAC, and CAN					?			Open
XH-76	Trivial	Other Interface	CANOpen	PG4 App shall use CANOpen as communication protocol. operate as NMT Slave, produce Heartbeat messages, consume SYNC messages, produce PDOs when required, and operate as server when handling SDOs					Required to utilize HW	Custom drivers	18088070	Open
XH-77	Trivial	Functional	Entry Startup	PG4 App shall always be entered from RM Bootloader					Respond to system mode changes, synchronize stimulus interval, deliver sensor data to PM, and route messages to/from PM	CANFestival stack		Open
XH-79	Trivial	Functional	Watchdog	PG4 App shall execute a startup procedure to configure settings from nonvolatile memory					Known state on startup	See RM Bootloader		Open
XH-78	Trivial	Performance	PatternSim	PG4 App shall support pattern stim operation, in which a command can map to one or more channels PulseWidth and PulseAmplitude					Patient specific settings are preserved while powered down	RestoreList in OD 0x2900, data stored in EEPROM		Open
XH-80	Minor	Performance	Waveform	PG4 App shall support configuring stimulus parameters within range and resolution determined by HW.					Protect patient from nuisance stimulation during lab use, clinical setup	watchdog timer configured in certain operational modes <see ref>		Open
XH-84	Trivial	Performance	DirectSim	PG4 App shall support direct stim operation, in which PulseWidth and PulseAmplitude for a channel are directly specified.					Local patterns minimize required network bandwidth	Patterns stored in EEPROM, active patterns copied to RAM. PW and PA linearly interpolated from c		Open
XH-81	Trivial	Performance	VOS set	PG4 App shall support adjusting the compliance voltage	10		34	1	Profiling channels, system diagnostics	PW, PA configurable directly or via pattern. Frequency configurable via sch	21004528	Open
XH-82	Trivial	Performance	VOS test	PG4 App shall support testing if pulse is in compliance for PulseWidth	10		255		Minimize operating current by reducing compliance voltage when possible	PG4 OD 0x3210.3, VOS	18088070	Open
XH-83	Trivial	Performance	CT Application HW drivers	PG4 App shall support testing if pulse is in compliance for PulseWidth					System diagnostic. Used for impedance measurements.	PG4 OD 0x3210.4, STCC	18088070	Open
XH-85	Minor	Other Interface	HW drivers	CT App shall provide drivers for radio, accelerometer, external coil thermistor, LCD display, user interface buttons, and USB					Required to utilize HW	Custom drivers		Open
XH-86	Trivial	Other Interface	CANOpen	CT App shall use CANOpen as communication protocol. operate as NMT Master and operate as server and client in handling and initialing SDOs					Respond to and trigger system mode changes. Route messages to/from PM and PC	CANFestival stack		Open
XH-127	Trivial	Functional	Entry RTOS	CT App shall be entered upon reset					Known state on startup	App starts at 0x0000		Open
XH-128	Trivial	Functional	Startup	CT App may use RTOS to manage tasks					Simplifies programming of asynchronous tasks, protects against certain programming errors			Open
XH-87	Trivial	Functional	Scripts	CT App shall execute a startup procedure to configure settings from nonvolatile memory					Patient specific settings are preserved while powered down	RestoreList in OD 0x2900, data stored on SD Card		Open
XH-88	Major	Functional	Radio	CT App shall store and execute scripts					Configurable command/control methods to support research applications. Menu options and display control	Stored in Flash, executed via ScriptInterpreter		Open
XH-112	Major	Functional	Stand-alone Operation PM Battery	CT App shall store and allow modifying Radio settings and support hardcoded PM Bootloader Radio settings					Allows different systems to be on different channels and/or different addresses to eliminate crosstalk between systems.	CT OD 0x2600	21365239	Open
XH-98	Major	Functional	CT Battery	CT App shall manage PM Battery charging without connection to PC					Take home charger for patient	See Charging Flow	22347843	In Progress
XH-97	Trivial	Functional	Mode Monitoring	CT App shall support enabling/disabling CT Li Ion battery charging and monitoring charging status					System diagnostics	CT OD 0x5015		Open
XH-99	Trivial	Functional	Wake PM	CT App shall support monitoring operational modes within PM without connection to PC					Take home patient interface	?		Open
XH-100	Trivial	Functional	Pipe	CT App shall support waking the PM from Low-power mode					Extend implant battery life between charges without powering down implant.	wake-on-radio polling		Open
XH-101	Minor	Functional	Pass-through Operation	CT App shall serve as a pipe between CE and PM for SDOs, NMTs, and PDOs					Radio is only means of communication between external programmer and implant	CT Gateway		Open

Key	P	ReqType	Summary	Requirement (SRS)	Min	Value	Max	ResNum	Value	Rationale	Design (SDS)	Verification	Test Results
XH-187	Major	Functional	PM Bootload	CT App shall support routing PM bootloader commands from programmer to PM						Radio is only means of communication between external programmer and implant	CT Gateway	21365199	PASS
XH-102	Trivial	Functional	RTC	CT App shall update Real-Time-Clock on CT						Used to timestamp events for data logging and to trigger scheduled scripts.	?		Open
Control Editor													
System Configuration													
XH-143	Major	User Interface	PM Bootload	CE shall provide UI to PM bootload process						Download new application (binary file) to PM. Access fail-safe communication to implant.	PM Bootload Tool	21365199	PASS
XH-144	Major	User Interface	RM Bootload	CE shall provide UI to RM Bootload process. CE shall guarantee unique open node and support changing node number						Download new application (binary file) to RM. Access fail-safe communication with RM.	RM Bootload Tool	21365199	PASS
XH-145	Trivial	User Interface	Module Config	CE shall provide UI to configure modules' settings						Setup PDO mapping, set recording and sim intervals, set charging parameters	Module Configuration Tool		Open
XH-152	Trivial	Functional	Network Config	CE shall provide UI to turn on/off network voltage and set VNET						System configuration	"Net Diagnostics" Tool	21365295	Open
XH-152	Minor	User Interface	Radio Config	CE shall provide UI to change Radio parameters in CT and PM						Allows different systems to be on different channels and/or different addresses to eliminate crosstalk between systems.	"Radio Configuration" Tool	21365239	Open
XH-146													
Monitoring													
XH-148	Trivial	Node List		CE should display the mode for each active node						System diagnostics	"Network Display" in Monitor Bar		Open
XH-149	Trivial	PM VSYS		CE should display VSYS from the PM						System diagnostics	Monitor Bar		Open
XH-150	Trivial	Network Status		CE should display the status of the PM FESCAN network						System diagnostics		21365295	Open
XH-151	Trivial	Radio Status		CE should display status and link quality of radio						System diagnostics		21365239	Open
Patient Configuration													
XH-141	Trivial	User Interface	Privacy Modes	CE shall store no patient-identifiable information						HPAA/IRB requirement	Patient Code used	Verified by inspection.	Open
XH-142	Trivial			CE should manage patient modes						System configuration		Open	Open
NNPS													
Networked Neural Prosthesis System													
Radio													
App Radio Settings													
XH-91	Trivial	Other Interface	Channel	NNPS shall support Radio channels	0	9	1			Multiple channels allow choosing most clear channel	300KHz channel separation, channel set by [CC1101 Radio Settings]14582;	21365239	Open
XH-92	Trivial	Other Interface	Addresses	NNPS shall support Radio addresses	0x01	0xFE	1			Address filtering allows different systems to be on the same channel without crosstalk	Addresses and address filtering set by [CC1101 Radio Settings]14582790]	21365239	Open
XH-95	Trivial	Other Interface	TX power	NNPS shall set transmit power						Transmit power affects transmission distance, required receiver sensitivity, and packet error rate	?	21365239	Open
XH-114	Minor	Other Interface	MedRadio Coexistence	NNPS shall use MedRadio compatible settings						FCC requirement	[CC1101 Radio Settings]14582790] for NNPS	Verified by inspection.	Open
XH-201	Minor	Other Interface		Multiple NNPS systems should be usable within 1m of each other						Multiple users may be in close proximity	Multiple channels and addressing	21365239	Open
BL Radio Settings													
XH-93	Critical	Other Interface	Channel Address	NNPS shall support fixed PM BL Radio channel			5			fail-safe communication with implant	hardcoded	Verified by inspection.	Open
XH-94	Critical	Other Interface		NNPS shall support fixed PM BL Radio address (local and remote)		0xBC				fail-safe communication with implant	hardcoded	Verified by inspection.	Open
XH-96	Critical	Other Interface	PM TX power	PM radiated power limit			25		uW	fail-safe communication with implant	ccValue=96 on PM sets TX power below 25uW	21365239	Open
CAN													
XH-115	Major	Other Interface	Bit Timing	NNPS shall use compatible bit timing parameters on all Modules that require CAN interface. CAN Baudrate:			100		Kbit/s	Modules with different MCU clock rates must be able to communicate at common CAN bit rate	[CAN bit timing]16515624] for NNPS	21365295	Open
XH-137	Trivial	Other Interface	Errors	NNPS shall log CAN errors in each module OD						System diagnostics	OD: 0x2500	21365295	Open
Protocols													
XH-116	Trivial	Other Interface	CANOpen	NNPS shall implement CANOpen as communication protocol across system						Common API insures compatibility	CANFestival stack	Verified by inspection.	Open
XH-138	Trivial	Other Interface	OD	NNPS shall store Object Dictionary in each module						Part of CANOpen	CANFestival stack	Verified by inspection.	Open
Scripting													
XH-120	Trivial	Functional	scripts	NNPS shall support SDO, MMT, write to File, and basic math and branching operations on modules with scripting capability						Support autonomous system diagnostics, data logging, and configurable command/control algorithms to support research use	OpCode List, Script Interpreter	Verified by inspection.	Open
Environment													
XH-117	Trivial	Environmental	Code Revisions	NNPS code revisions shall be maintained in a code repository						Track changes, merge changes by multiple developers	Unfuddle	Verified by inspection.	Open
XH-119	Trivial	Environmental	Documentation	NNPS code shall use source code comments to automatically generate code documentation						Provide automatic low-level documentation	Doxygen	Verified by inspection.	Open
XH-118	Trivial		Guidelines	NNPS code should follow applicable coding guidelines						Reduce coding errors and readability among developers			Open