REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**			
1. REPORT DATE ***(DD-MM-YYYY)*** 27032017	**2. REPORT TYPE** Technical		**3. DATES COVERED** ***(From - To)*** 10032017 - 20032017
4. TITLE AND SUBTITLE Bootloader-Application Transition Failure Investigation – GVR-BOT Technical Report			**5a. CONTRACT NUMBER** **5b. GRANT NUMBER** **5c. PROGRAM ELEMENT NUMBER**
6. AUTHOR(S) Ty Valascho			**5d. PROJECT NUMBER** **5e. TASK NUMBER** **5f. WORK UNIT NUMBER**
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) US Army, Tank Automotive Research Development and Engineering Command (TARDEC) Warren, MI 48397			**8. PERFORMING ORGANIZATION REPORT**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) US Army, Tank Automotive Research Development and Engineering Command (TARDEC) Warren, MI 48397			**10. SPONSOR/MONITOR'S ACRONYM(S)** **11. SPONSOR/MONITOR'S REPORT NUMBER(S)**
12. DISTRIBUTION / AVAILABILITY STATEMENT UNCLASSIFIED: Distribution Statement A. Approved for public release; distribution is unlimited.			
13. SUPPLEMENTARY NOTES The views, opinions, and/or findings contained in this report are those of the authors and should not be construed as an Official Department of the Army position, policy, or decision, unless so designated by other documents.			
14. ABSTRACT GVR-BOT customers have been having issues with internal circuit boards getting stuck in bootloader mode, waiting for reflash. An investigation was made and the root cause was found to be the loss of power to internal boards during the bootloader to application software transition. Four categories of failure mode were found and explained. A software solution is proposed.			
15. SUBJECT TERMS GVR-BOT, robot, bootloader, software, system			
16. SECURITY CLASSIFICATION OF: **UNCLAS / DIST A**			**17. LIMITATION OF ABSTRACT** A **18. NUMBER OF PAGES** 19 **19a. NAME OF RESPONSIBLE PERSON** Ty Valascho
a. REPORT DIST A	**b. ABSTRACT** DIST A	**c. THIS PAGE** DIST A	**19b. TELEPHONE NUMBER** ***(include area code)*** (586) 459-7485

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

US Army - TARDEC

Bootloader-Application Transition Failure Investigation

GVR-BOT Technical Report

Valascho, Ty
3-27-2017

Executive Summary: GVR-BOT customers have been having issues with internal circuit boards getting stuck in bootloader mode waiting for reflash. An investigation was made and the root cause was found to be the loss of power to internal boards during the bootloader to application software transition. Four categories of failure mode were found and explained. A software solution is proposed.

Contents

Introduction

Numerous instances of GVR-BOT robots getting “stuck in bootloader mode” have been observed. When this happens, the robot will not act on movement commands. An attempt was made in March 2017 to discover the root cause of these occurrences.

There are three embedded circuit boards in the GVR-BOT Main Electrical Housing: Pwr Mgt (PN: 0003), Mobility (PN: 0007) and Flipper (PN: 0008). The software of these three circuit boards each consists of bootloader firmware and application code software. Upon first powering up the processor in each board, execution control passes first to its bootloader. The bootloader interrogates an EEPROM location – the “Signature Byte” - to determine if the user requested the ability to reflash the application software the last time the processor was powered up. If reflashing is desired, the bootloader waits for the reflash command and execution control is not passed to the application software. This is called “bootloader mode.” Once the board has entered bootloader mode, it will remain in bootloader mode - even if the circuit board is reset again – until a reflash event has occurred or the board is commanded back into application mode. If reflashing was not desired, execution control is passed to the application software.

This sequence of instructions is the same for all three embedded circuit boards. A simplified flow diagram of execution is presented in Figure 1.

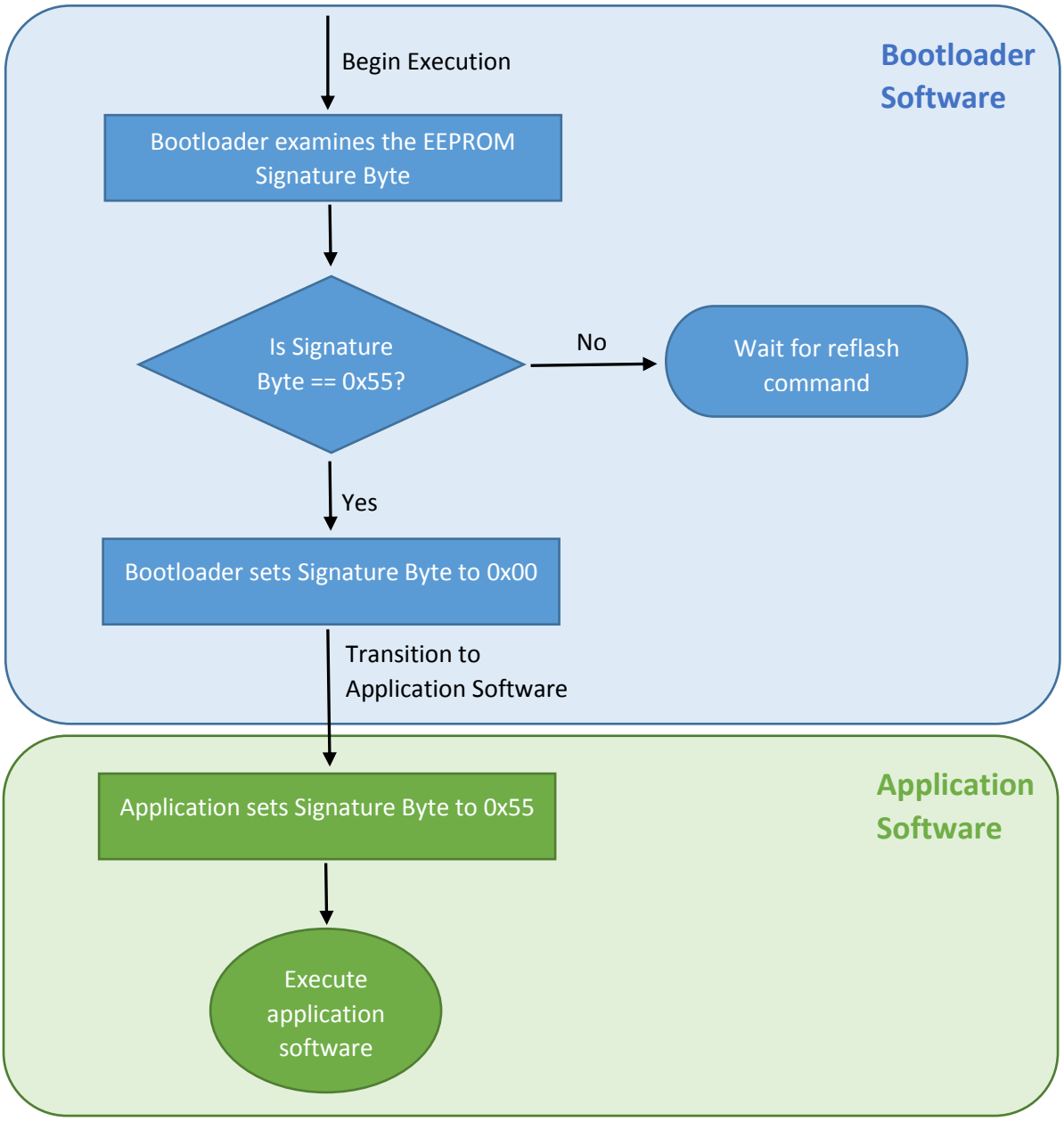

Figure 1 – Simplified Execution Flow of GVR-BOT Bootloader to Application Transition

The issue that customers have experienced is that some of the circuit boards are entering bootloader mode spontaneously. They remain in bootloader mode until action is taken by the customer to put it back into application mode. The issue often occurs when the robot's batteries are less than 10% SOC. The Mobility board was almost always the circuit board reported to be stuck in bootloader mode. The Flipper board was reported as being in bootloader mode less than 5% of the time. There are no known reports of the Pwr Mgt board becoming stuck in bootloader spontaneously.

Issue Investigation

Investigation of Software

To understand exactly what is happening when a GVR-BOT embedded circuit board becomes "stuck" in bootloader mode, a more in-depth look at the software must be taken.

The purpose of the bootloader software is to enable the application software to be updated without opening the robot to physically access the JTAG connector on the embedded circuit boards. The bootloader software can only be updated by using the JTAG connector.

For the software to incorrectly remain in bootloader mode, a corruption of the Signature Byte must be occurring. It would be unlikely that so many robots would be having random hardware corruptions of the exact same EEPROM byte. No other EEPROM locations have been reported as corrupted and the failures can be recovered from by manually setting the software back to application mode. Furthermore, all three embedded boards use an Atmel ATXMEGA256A3U-AU-ND. If hardware corruption of EEPROM was the root cause, it is logical to expect the failure to be happening in roughly equal amounts on all three embedded circuit boards. This was not the case – almost all reported failures were happening on the Mobility board.

Since a hardware issue was determined to be unlikely, the behavior of the software was examined next. Low battery power was reported to be a factor which led the team to suspect that the Mobility board was browning out or being reset during the boot sequence. It was theorized that if the Mobility processor was powered off after the bootloader software set the Signature Byte to 0x00, but before the application software overwrote this value with 0x55, the failure mode symptoms would match what was being reported.

Reproducing the Issue

The first challenge was to find a way to reliably cause the robot's embedded circuit boards to enter bootloader mode un-commanded by the user. Attempts to power the robot on and off from batteries that were below 10% SOC were not found to cause the issue under laboratory conditions.

It was discovered that the issue could be consistently induced by powering electrical hardware with a high inrush current from any of the robot's payload ports. GVR-BOT software 1.00 and 1.01 would cause the issue 100% of the time when the hardware was attached to any of the robot's payload ports. It was unknown whether this was the same failure mode the customers had been experiencing.

The issue was caused by the interaction of several components within the Main Electrical Housing of the GVR-BOT. It is explained in a report entitled "Payload Inrush Investigation – GVR-BOT Technical Report" from March 10, 2017.

Figure 2 shows the normal power up sequence of the GVR-BOT, where the abbreviations "BL" and "APPL" represent the bootloader and application sections of the software respectively.

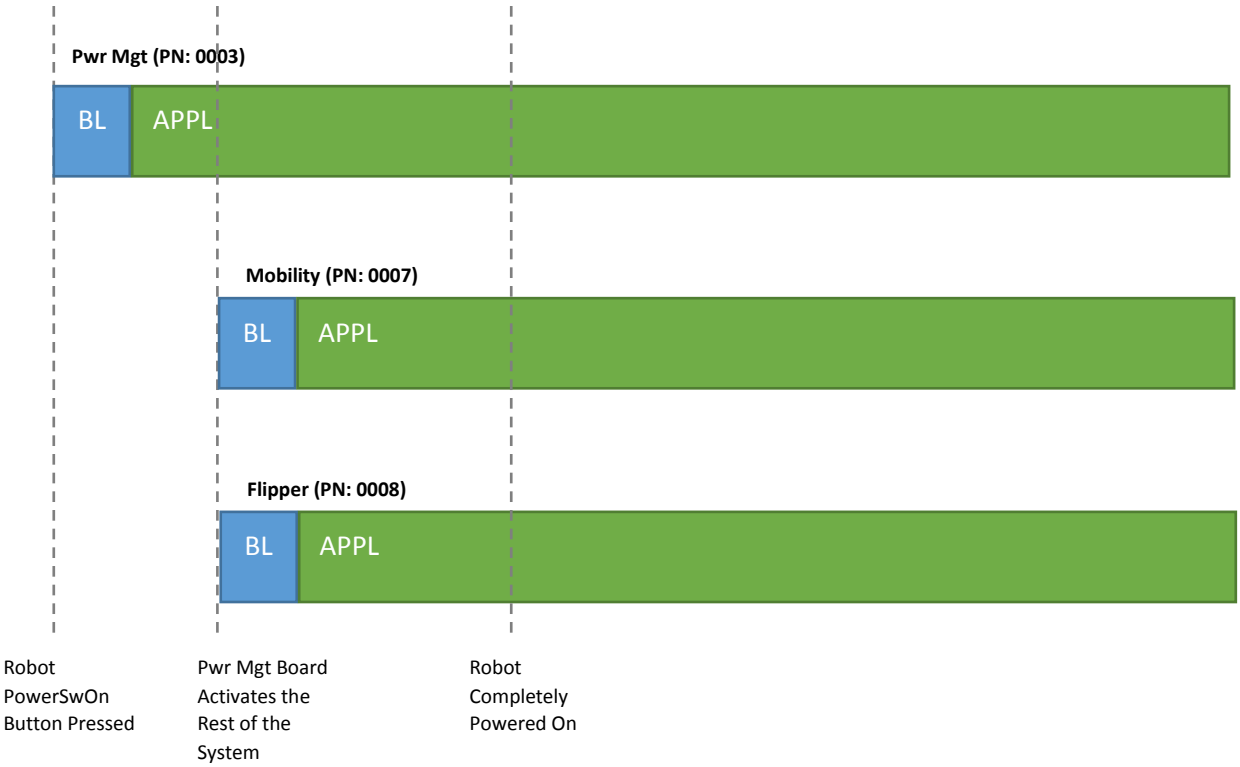

Figure 2 – Simplified Normal Power-On Sequence of GVR-BOT System

As can be seen in the sequence diagram, the Pwr Mgt circuit board powers up first and initializes its software passing through the bootloader code, then to the application code where it energizes MOSFETs that provide power to the rest of the circuit boards in the system. This diagram is simplified in that it shows a subset of all the circuit boards in the GVR-BOT system. Note that the diagram shows the sequence and approximate relationship of events but is not necessarily to scale temporally.

When the inrush event happens, the system bus voltage dips from 34 VDC down to 5 VDC for 0.4 milliseconds and the power-on sequence of the Mobility board is interrupted because its main processor loses power. If this interruption happens at the wrong moment in time – between the bootloader software setting the Signature Byte to 0x00 and the application software setting it to 0x55 – the circuit board will power-on with the Signature Byte still set to 0x00 and remain in bootloader indefinitely.

A representation of this sequence is provided in Figure 3, with the location in time of the power interruption denoted as Power Interruption Event.

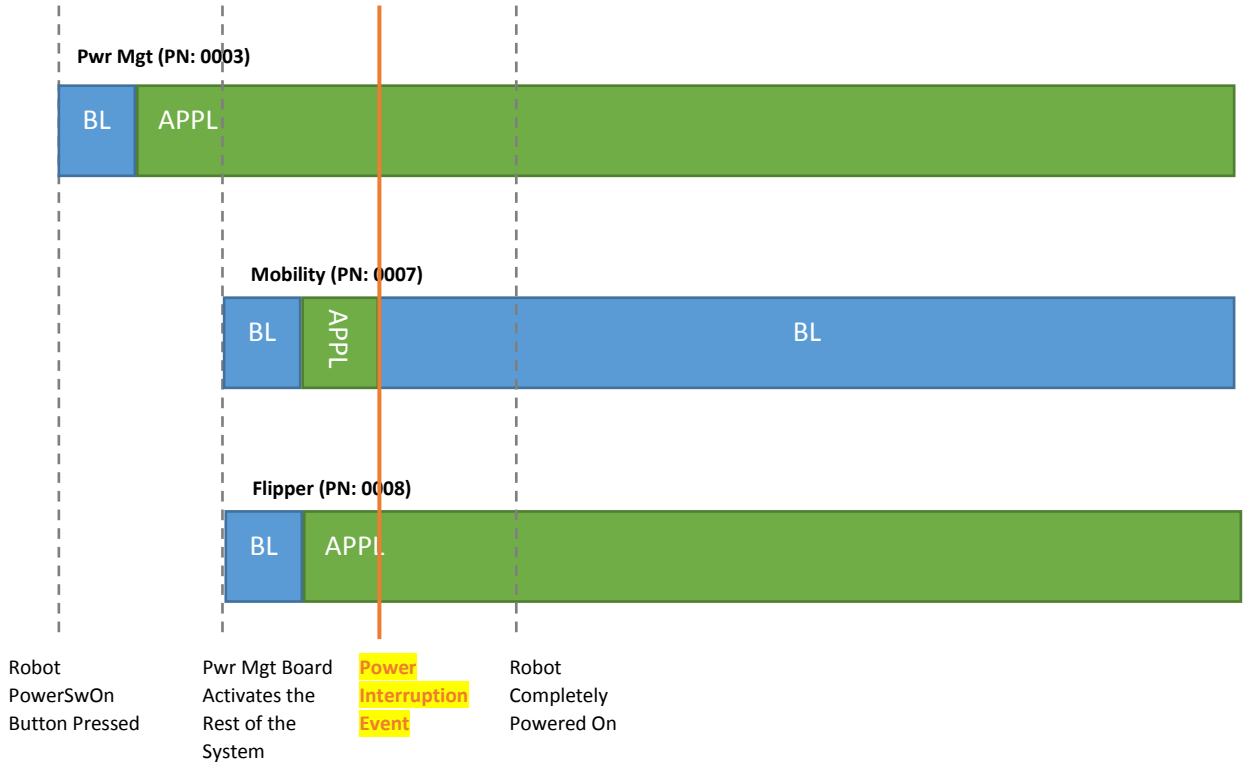

Figure 3 - Interrupted GVR-BOT Power-On Sequence Resulting in Bootloader Issue

A Power Interruption Event could be caused by batteries at 0% SOC or some other disturbance of the power system that causes the internal circuit boards to lose power while the robot is booting up.

Root Cause Investigation

"Window of Vulnerability" Theory

Once a way was found to reliably cause the issue, an attempt was made to prove or disprove the theory that the root cause was the Mobility board being reset between the bootloader software setting the Signature Byte to 0x00 and the application software setting this value to 0x55.

If the theory was correct, there would be a small window of time that the Mobility board was vulnerable to a power off event. Any reset before this "window of vulnerability" would not cause a change to the Signature Byte, so subsequent power-on events would be correct. Any reset that happened after the window of vulnerability would allow the application code to set the Signature Byte correctly and the system would not experience the problem.

It was decided to add small delays to the Pwr Mgt application software so that the payload ports – with the high inrush current payload attached - would be activated at later and later times during each subsequent trial. If a window of vulnerability existed, this test should show failures when the reset occurred within this window and no failures outside this window.

Figure 4 shows the window of vulnerability and where it would exist in the power-on sequence of the robot.

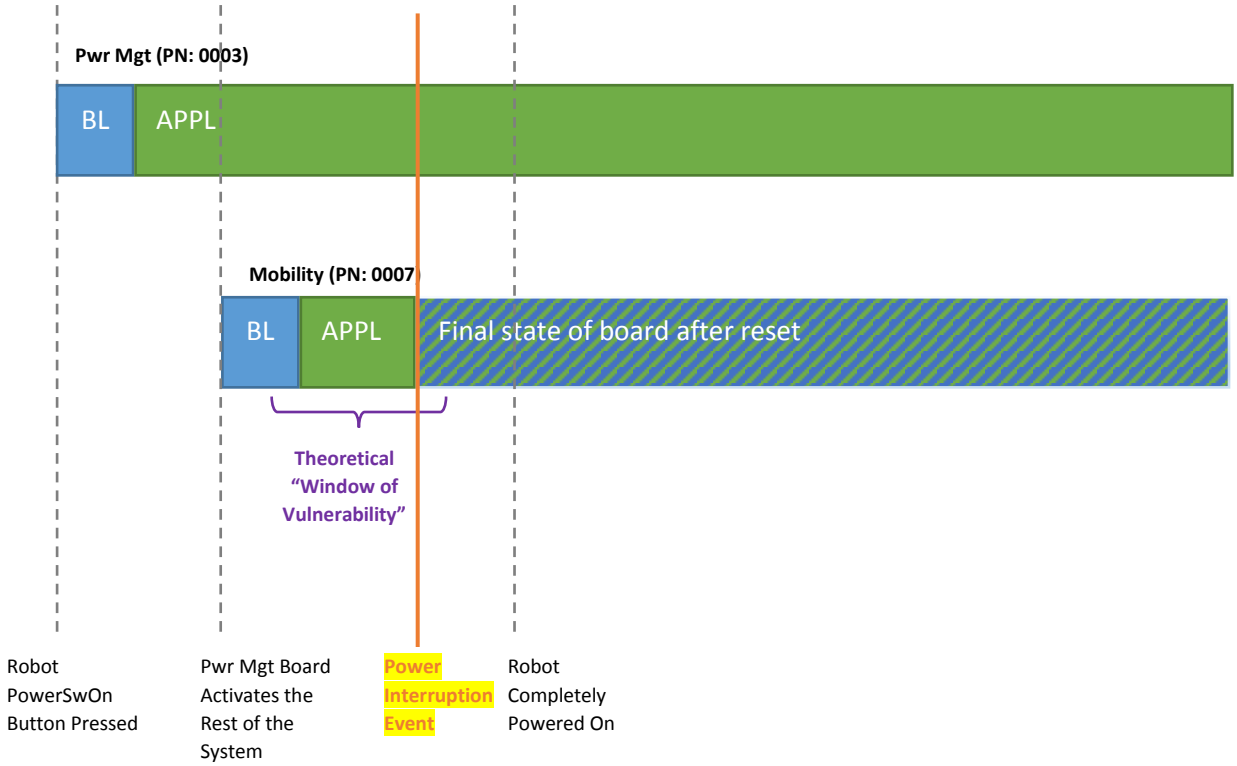

Figure 4 - Theoretical "Window of Vulnerability" During GVR-BOT Power-On Sequence

If a Power Interruption Event occurred within the window, the failure would occur and the robot's Mobility board would get stuck in bootloader mode. If the Power Interruption Event occurred outside this window, the robot would power up normally and transition to application mode.

Testing the "Window of Vulnerability" Theory

Test Configuration

For the test, a GVR-BOT test bench setup was used, powered by a single BB-2590 battery. It is depicted in Figure 5, including the instrumentation.

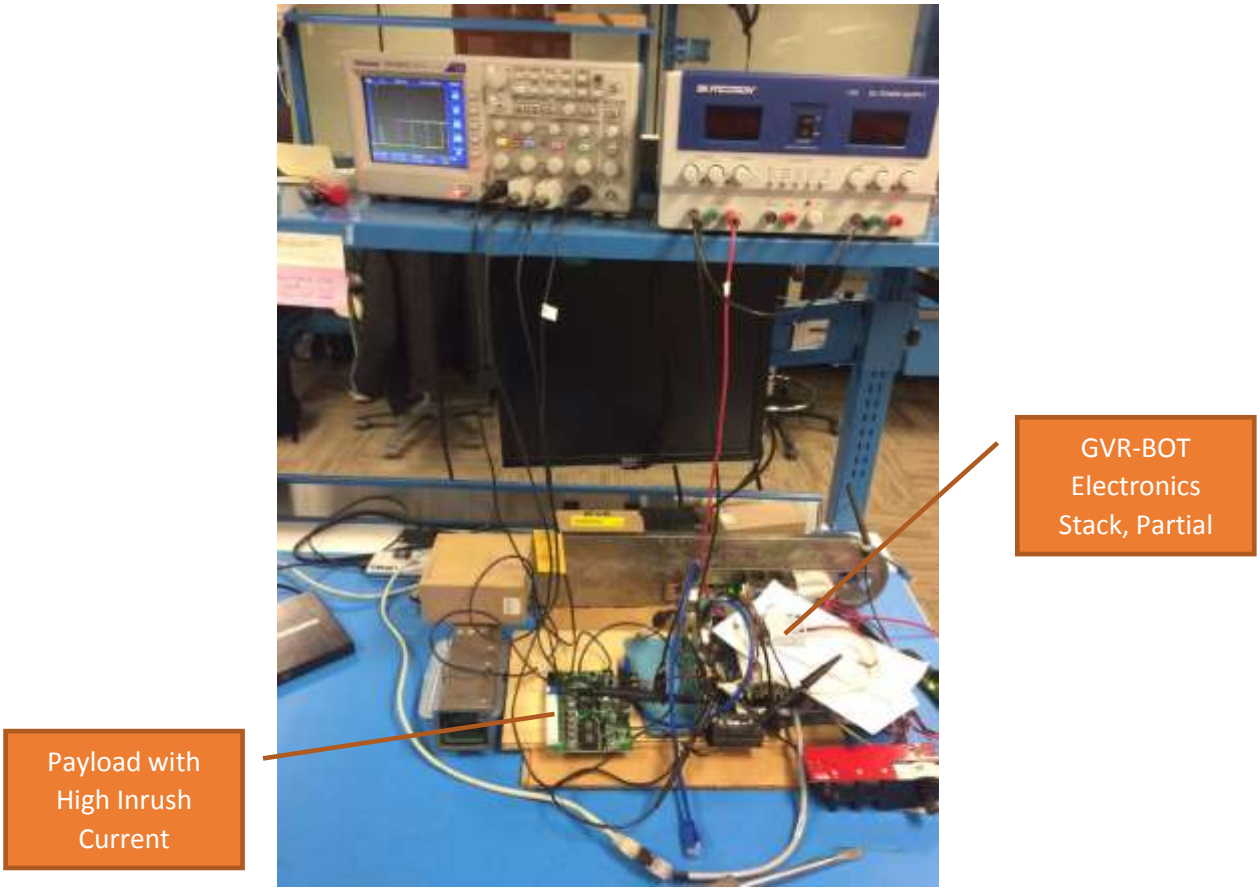

Figure 5 - "Window of Vulnerability" Test Bench Setup

The "High Inrush Payload" was a M4-ATX-HV ATX Power Supply made by Mini-Box, serial number EEC21154712036. Previous testing determined that this power supply will pull more than 100 Amps of inrush current for ~0.4 milliseconds while powering up. This was found to exceed the ability of the robot's battery to supply current and cause a dip in system voltage from 34 VDC down to less than 5 VDC during the inrush event.

The release numbers of the software used during the testing are recorded in Table 1.

Table 1 - Software Release Numbers of Test Assets

Hardware PN	Hardware Name	Bootloader SW Release	Application SW Release
0003	Pwr Mgt Board	001.003.000	Unit Under Test
0007	Mobility Board	001.003.000	000.002.017
0008	Flipper Board	001.003.000	000.001.005

The Unit Under Test (UUT) software was based on Pwr Mgt board SW Release 000.002.004. The only modification to this software was the insertion of a software delay using the _builtin_avr_delay_cycles() function. The delay was placed before the GVR-BOT payload port power MOSFETs were activated in the software.

During the testing, the delay was varied from 0 milliseconds (baseline 000.002.004 Pwr Mgt Software) to 30 milliseconds in 3.3 millisecond increments. An oscilloscope was used to record PowerOnSw, Input Voltage of the Mobility Board, Input Voltage of the High Inrush Payload, and Payload Port P3. It was configured to trigger on the falling edge of the PowerOnSw signal which is the signal that turns the robot on.

Boot Counter and Reset Counter

Another feature of the test setup was the monitoring of the Boot Counter and Reset Counter. The Boot Counter is an EEPROM location that is incremented by the bootloader software after its initialization is completed and before it reads the Signature byte.

The Reset Counter is incremented by the application software immediately after it writes 0x55 to the Signature byte.

Under normal power-on conditions, the Boot Counter will indicate the number of the times the circuit board has been powered on since built. The Reset counter will record the number of power-on events the circuit board has experienced, except those times where the circuit board has been placed in bootloader mode, waiting to be reflashed.

Test Procedure

The test procedure for each test trial was:

1. Flash Pwr Mgt Board with UUT Software
2. Power up system without High Inrush Payload attached
3. Verify correct system power-on and record values of Boot Counter and Reset Counter for the Mobility Board and Flipper Board
4. Power down system
5. Attach High Inrush Payload to Payload Port P2
6. Power on system and record voltage on oscilloscope
7. Remove the High Inrush Payload from Payload Port P2
8. Record the Boot Counter and Reset Counter values of the Flipper Board
9. If Mobility board is in bootloader mode, set to application mode and reboot system
10. Record the values of the Mobility Boot Counter and Reset Counter – subtract one from these values if the Mobility Board was in bootloader mode in the previous step

In Step 10, the Boot Counter and Reset Counters are decremented so that the values of the counters are recorded for the *first* power-on event of the test, not a reset event that may have occurred.

Test Results and Discussion

Twenty test trials were performed for this testing, including two trials using the Baseline software with no delay. A summary of the results are displayed in Table 2.

Table 2 - "Window of Vulnerability" Test Results

Trial #	Calculated Delay (milliseconds)	Measured Delay (milliseconds)	Any Bootloader Modes?	Mobility Boot Counter	Mobility Reset Counter	Flipper Boot Counter	Flipper Reset Counter
1	0.0	0.0	Mobility	2	0	1	1
2	0.0	0.0	Mobility	2	0	1	1
3	8.3	8.0	Mobility	2	0	1	1
4	8.3	8.0	Mobility	2	0	1	1
5	12.5	12.0	Mobility	2	0	1	1
6	12.5	12.0	Mobility	2	0	1	1
7	16.7	16.0	Mobility	2	0	1	1
8	16.7	16.0	Mobility	2	0	1	1
9	20.8	20.0	None	2	1	1	1
10	20.8	20.0	None	2	1	1	1
11	25.0	24.0	None	2	201	1	1
12	25.0	24.0	None	2	256	1	1
13	25.0	24.0	None	2	256	1	1
14	29.2	28.0	None	2	2	1	1
15	29.2	28.0	None	2	2	1	1
16	33.3	32.0	None	2	63746	1	1
17	33.3	32.0	None	2	2	1	1
18	33.3	32.0	None	2	2	1	1
19	37.5	36.0	None	2	2	1	1
20	37.5	36.0	None	2	2	1	1

The Calculated Delay column holds the number of milliseconds that were intended to be added to the Pwr Mgt software. This number is calculated by knowing the CPU speed of 24 MHz, where each clock cycle is ~41.67 nanoseconds. The function _builtin_avr_delay_cycles(<DELAY CONSTANT>) was used for the delay, where <DELAY CONSTANT> is the number of clock cycles of delay hard-coded in the UUT software.

For example, Trial 5 was performed with the highlighted line added to the mc_init_power() function in the Main.c file, as seen in Figure 6.

```
void mc_init_power() {
   // Enable Left and Right Rails
   // Set direction as output.
   CONTROL_SIGNAL_PORT.DIRSET = LEFT_RAIL_OUTPUT_PIN_bm | RIGHT_RAIL_OUTPUT_PIN_bm;
   // Set output high.
   CONTROL_SIGNAL_PORT.OUTSET = LEFT_RAIL_OUTPUT_PIN_bm | RIGHT_RAIL_OUTPUT_PIN_bm;

   // Enable the payloads.
   // Set direction as output, but don't enable the 4 payload ports yet.
   CONTROL_SIGNAL_PORT.DIRSET = PAYLOAD_OUTPUT_PINS_gm;
    _builtin_avr_delay_cycles(300000); // 300000 = 12.5 msec delay
   // Set output high.
   CONTROL_SIGNAL_PORT.OUTSET = PAYLOAD_OUTPUT_PINS_gm;

   // Turn on the green LED.
   CONTROL_SIGNAL_PORT.DIRSET = GREEN_LED_OUTPUT_PIN_bm | RED_LED_OUTPUT_PIN_bm;
   CONTROL_SIGNAL_PORT.OUTSET = GREEN_LED_OUTPUT_PIN_bm | RED_LED_OUTPUT_PIN_bm;

   // Set direction for kill signal (active low).
   POWER_SIGNAL_PORT.DIRSET = POWER_KILL_OUTPUT_PIN_bm;
   // Take high so we don't kill power.
   POWER_SIGNAL_PORT.OUTSET = POWER_KILL_OUTPUT_PIN_bm;
}
```

Figure 6 - Modified Source Code for mc_init_power() Function used in Trial 5

The Measured Delay values are the delays determined using the oscilloscope measurements during that Trial. The resolution of the oscilloscope was accurate to +/- 1 millisecond.

The values shown in the Boot Counter and Reset Counter columns do not include the “before” and “after” values - only the incremented values that were found by subtracting one from the other. This was done to make the results easier to interpret: the expected values are 1 for Boot Counter and 1 for Reset Counter. This result happened consistently with the Flipper Board.

Based on the results, it appears that the failures are only occurring when the delay is less than ~20 milliseconds. Values of delay greater than that do not cause spurious bootloader failures.

To further understand the exact mechanism of the results that were observed, a more in-depth analysis of the relationship between Boot Counter, Reset Counter, and Delay time must be performed.

Boot Counter, Reset Counter, and Failure Modes

With the exception of Trial 16 – which will be discussed later – the test results in Table 2 fall into 4 general categories. These categories are enumerated in Table 3.

Table 3 – “Window of Vulnerability” Test Result Categories

Test Result Category	Final Condition of Mobility Software	Ultimate Test Result	Mobility Boot Counter	Mobility Reset Counter	Example
1	Stuck in bootloader mode	Failure	2	0	Trial 7
2	Application mode	Pass	2	1	Trial 9
3	Application mode	Pass	2	Corrupted	Trial 12
4	Application mode	Pass	2	2	Trial 17

In all cases, the Mobility software is booted two times, as indicated by the Boot Counter. The exact point at which the Power Interruption Event happens determines the values that are seen for the Reset Counter.

To understand the analysis of these four test result categories, a closer look at the exact sequence of instructions must be provided. In Figure 7, the instructions germane to this discussion have been labelled.

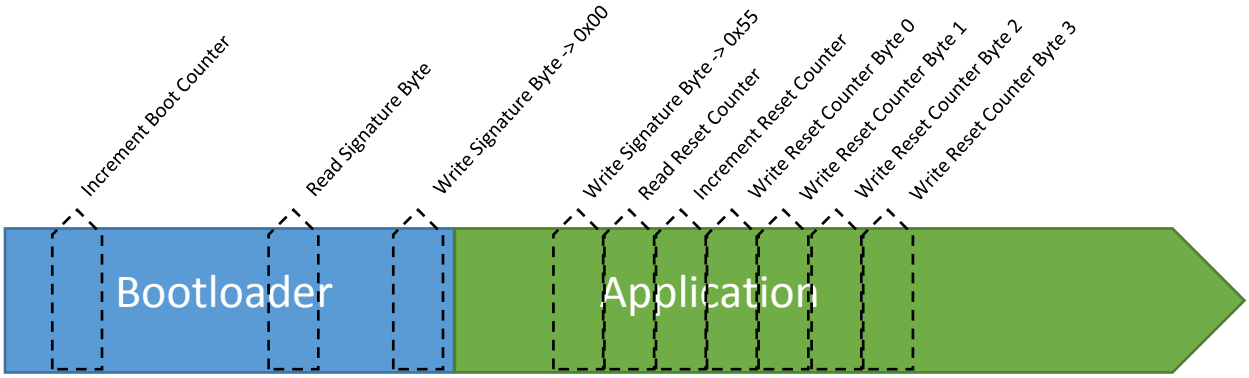

Figure 7 - Detailed Sequence of Embedded Board Power-On Sequence

Processor execution starts on the left in this diagram and moves to the right as the instructions are executed. Instructions not relevant to this discussion are not labeled and are represented as "blank space" in the execution flow. The labelled instructions are the source code instructions written in the C programming language. Many of these instructions represent tens – sometimes hundreds – of assembly code instructions to the processor.

The Reset Counter consists of 4 bytes of information which are written sequentially, with the least significant byte (Byte 0) written first.

Test Result Category 1

In Category 1 results, the Mobility board enters the bootloader software and increments the Boot Counter, then sets the Signature byte to 0x00. Power is interrupted before the Reset Counter is incremented and before the Signature byte is overwritten by 0x55 by the application software. This means that when the second processor power-on happens, the Boot Counter is incremented a second time but the software is stuck in bootloader mode so it never will increment the Reset Counter. The location of the Power Interruption Event within the execution sequence is shown in Figure 8.

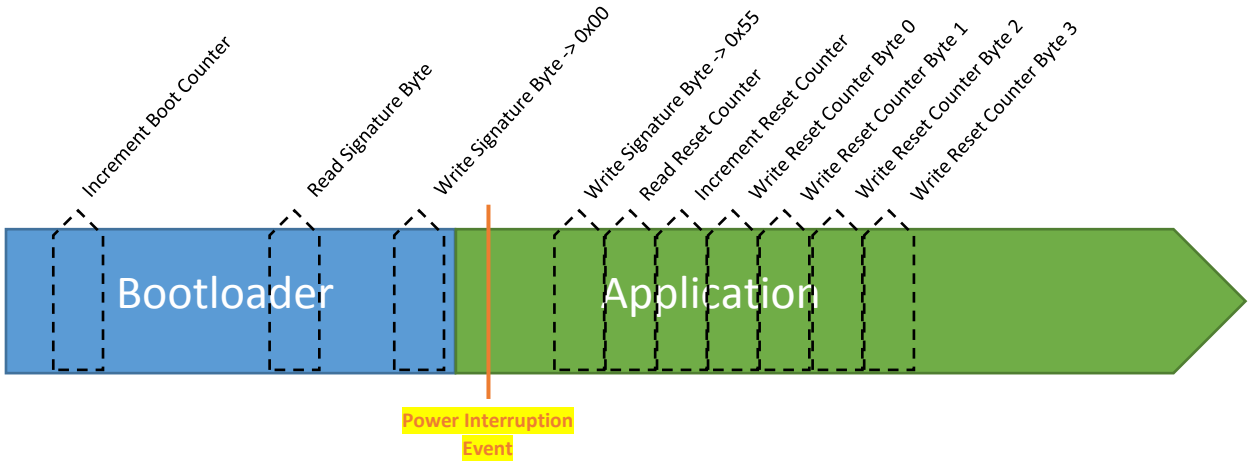

Figure 8 - Category 1 Power Interruption Event

Test Result Category 2

Category 2 is not a system failure – the Mobility board powers up correctly to application mode. The Reset Counter value is inconsistent with a correct power-on. This happens because the Power Interruption Event occurs after the bootloader increments the Boot Counter and after the application software starts and gets a chance to overwrite the Signature byte with 0x55. The processor is powered down almost immediately after the 0x55 is written, because the application software never gets a chance to write the incremented Reset Counter value to EEPROM. After the processor reboots, both the Boot Counter and Reset Counter do get incremented correctly. The approximate location of the Power Interruption Event is provided in Figure 9.

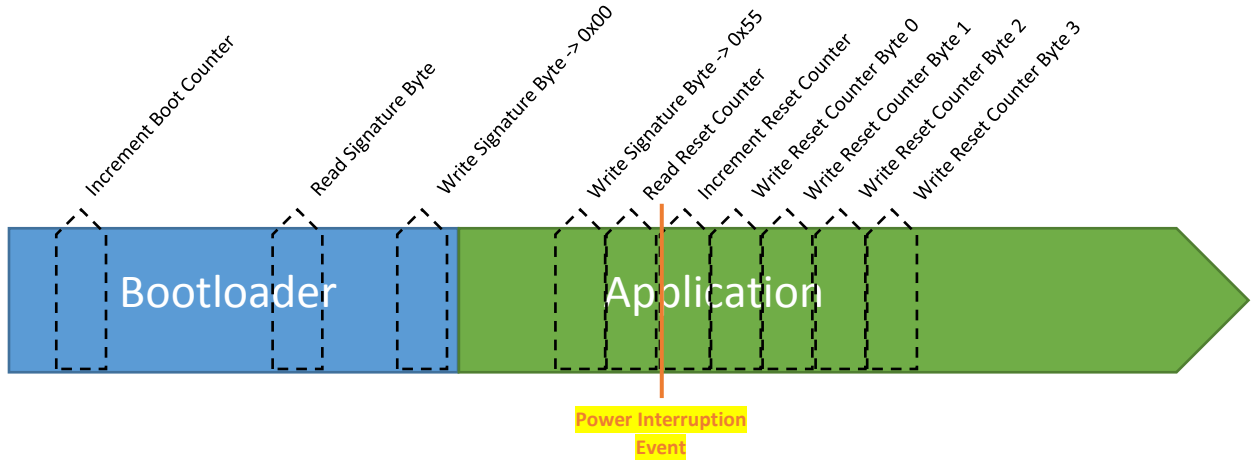

Figure 9 - Category 2 Power Interruption Event

Test Result Category 3

Category 3 does not result in a system failure but does corrupt the Reset Counter. In this test result, the Power Interruption Event occurs after the bootloader increments the Boot Counter and after the application software overwrites the Signature byte with 0x55. The interruption occurs during the writing of the Reset Counter which corrupts the value. As mentioned previously, the Reset Counter is four bytes and each byte is written individually as separate instructions in the source code. Careful examination of the values of the Reset Counter reveal that the Power Interruption Event occurs during the writing of the least significant byte of the number, resulting in a 0xFF instead of the correct, incremented value. Figure 10 shows where the Power Interruption Event is happening for this category of test result.

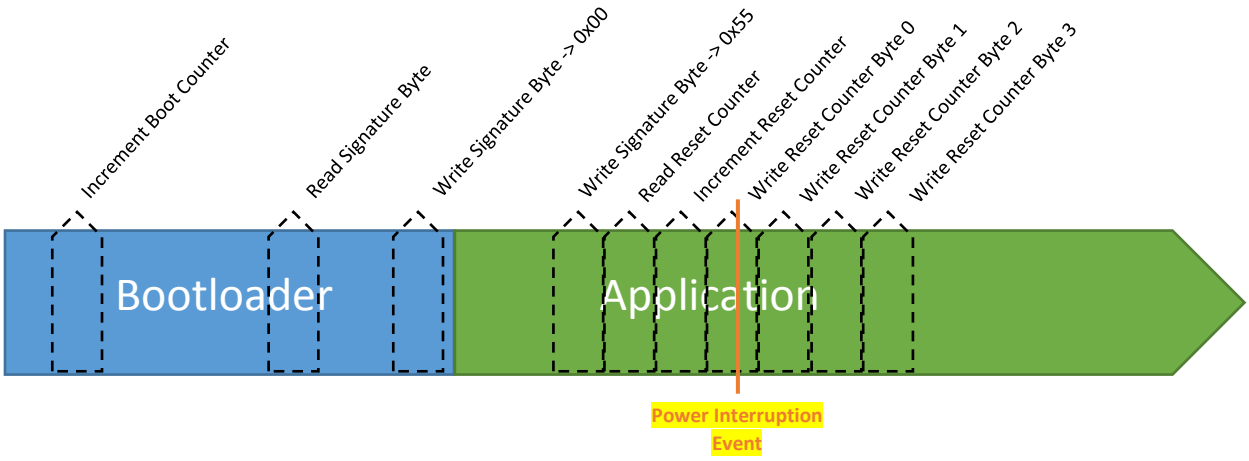

Figure 10 - Category 3 Power Interruption Event

Test Result Category 4

Category 4 is the correct operation of the robot – the Power Interruption Event happens after the portions of the software involving writing the Reset Counter, Boot Counter, and Signature Byte have had a chance to execute. Both Boot Counter and Reset Counter values are incremented during the first power-on, then again during the second reboot. In Figure 11, a successful reboot event is shown.

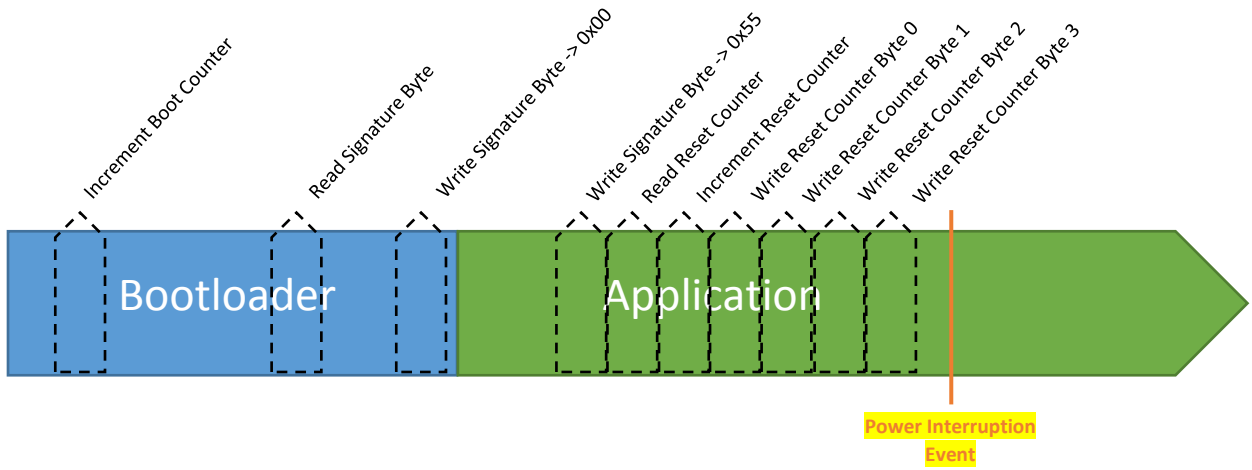

Figure 11 - Category 4 Power Interruption Event

Categories Summarized

All four categories are displayed together in Figure 12 for comparison.

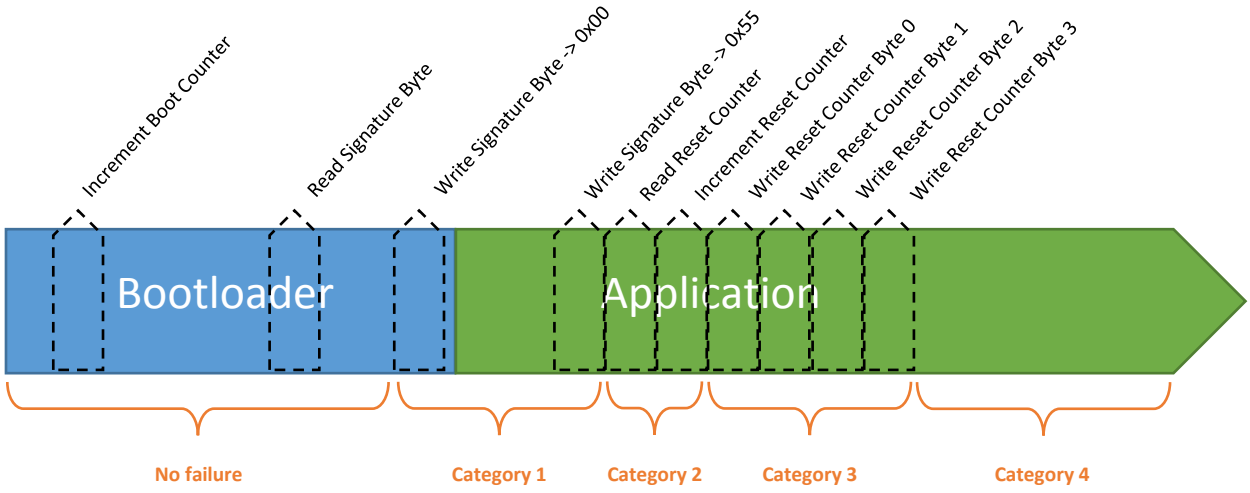

Figure 12 - All Four Categories Together in Sequence

The window of vulnerability was found to exist in the Category 1 area and results in a robot system failure – the Mobility board remains in bootloader.

Testing Anomalies

Trial 16

During the test runs, Trial 16 appeared to have a corruption of the Reset Counter. The other two Trials at the 33.3 millisecond delay timing - Trial 18 and 19 – both operated normally without failure or corruption. The previous Trials at 29.2 millisecond delay timing were also consistent and showed no issues. Analysis of the value of the Reset Counter indicated that Byte 1 was corrupted with a 0xFF. This is similar to what was consistently found with the Trials that used 25.0 millisecond delays, except those corruptions appeared to be in Byte 0 of the Reset Counter.

It is not known why this one anomalous result occurred. Possible causes are:

- Clock speed drift
- Temporary changes to execution speed or timing due to spurious interrupts or some other source
- The wrong software was installed for this test – the software was reflashed before each Trial and it is possible that a different software set than recorded was actually used

Flipper Board

Another unexpected result from testing was the lack of Flipper board failures. The customer reported the Flipper board getting stuck in bootloader mode, but this result has not been replicated in the laboratory.

The software for the bootloader is the same for all three embedded boards, but there are compile time "switches" that add, remove, and change sections of the software during compilation. This results in different executable software for the three embedded boards, even though they are all built from the same source code. Because there are differences in the bootloader executables, the timing of events within the bootloaders will also be different. It is possible that the Flipper board bootloader executes faster than the Mobility board so the "window of vulnerability" has passed by before the Power Interruption Event occurs. This was ruled out, however, because the Boot Counter and the Reset Counter of the Flipper board were incremented by one, indicating no reset event had been experienced by the flipper board.

Another difference lies in the hardware design between the two boards. It is possible the dip in voltage that causes the Mobility Board to reset does not cause the same problem in the Flipper Board. The architectures of the two power systems are similar but not exactly the same. The differences can be seen by comparing Figure 13 and Figure 14, which represent the Mobility Board and Flipper Board respectively.

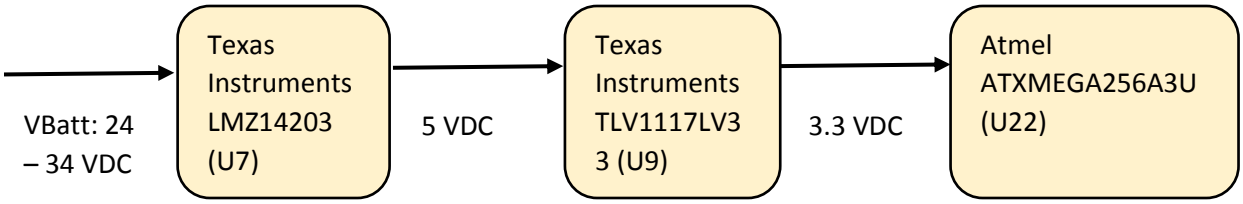

Figure 13 - Mobility Board Power Hardware Architecture

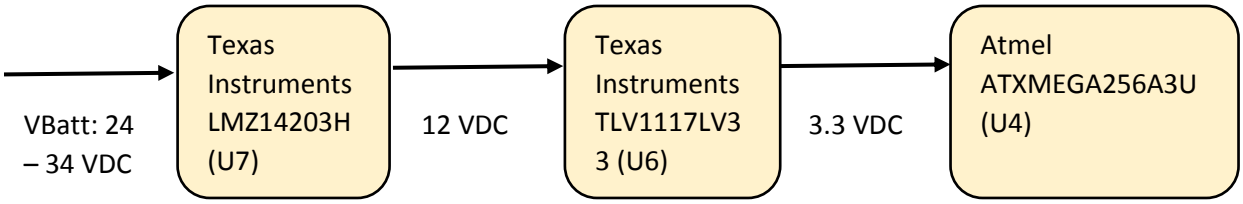

Figure 14 - Flipper Board Power Hardware Architecture

In addition to the one DC-DC converter chip that is different – LMZ14203 vs LMZ14203H – the supporting hardware of resistors and capacitors is also different since the voltage conversion levels are different between the two designs. It is suspected that these electrical hardware differences result in the Flipper Board being more robust to momentary dips in voltage – either the timing, duration, or the "depth" of the voltage dip. This was not confirmed.

It is possible that the customer hardware configurations that are causing this issue are different in one or more aspects from the method being used in the laboratory to recreate it.

Conclusions and Recommendations

Customers are experiencing issues with GVR-BOT robots that result in one or more internal circuit boards being stuck in the bootloader portion of software, incorrectly waiting to be reflashed.

To find the root cause of the issue a method was discovered to reproduce the failure in the laboratory using a high inrush payload. This allowed the creation of a test to find the root cause by varying the timing of when the high inrush payload was activated.

The root cause analysis was based on the theory that a loss of power to the circuit board during the transition from bootloader to application software was corrupting the EEPROM location used to determine whether the user was requesting a reflash event on the next power-on of the system. The EEPROM location is called the Signature Byte. The time window where power loss would cause the problem was called the "window of vulnerability" and the length of this time was found through testing to be 20 milliseconds.

Further investigation of other failure modes during the boot phase of the robot was also performed and four categories of test results were determined.

To reduce the likelihood of this occurring, it is recommended that the activation of the payload ports be moved until the robot's internal circuit boards have correctly transitioned from bootloader to application software.

Appendices

Acronyms and Abbreviations

APPL = Application

BL = Bootloader

DC = Direct Current

EEPROM = Electrically Erasable Programmable Read-Only Memory

HW = Hardware

JTAG = Joint Test Action Group

LED = Light Emitting Diode

MOSFET = Metal Oxide Semiconductor Field Effect Transistor

PN = Part Number

SOC = State of Charge

SW = Software

UUT = Unit Under Test

VDC = Voltage, Direct Current