# MC33816 Diagnostics

## 1    Introduction

This application note explains how to use MC33816 diagnostics in a typical four cylinder internal combustion engine (ICE) application. The field of powertrain is just one example where diagnostics are required at very high speed. The 33816 diagnostics manage this through four independent microcores.

This application note seeks to address different fault cases and describes how to program the microcode to detect them during idle and actuation mode.

NXP analog ICs are manufactured using the SMARTMOS process, a combinational BiCMOS manufacturing flow that integrates precision analog, power functions, and dense CMOS logic together on a single cost-effective die.

## 2    Overview

The 33816 is a 12 channel gate driver IC for automotive engine control applications. The IC consists of five external MOSFET high-side pre-drivers and seven external MOSFET low side pre-drivers. The 33816 provides a flexible solution for the MOSFET's gate drive with a versatile control and optimized latency time. Gate drive, diagnosis, and protection are managed through four independent microcores, two Code RAM, and two Data RAM banks.

## Contents

# 3 Application Schematic

The MC33816 typical application controls two injection banks, one DC-DC and a Pump Bank.
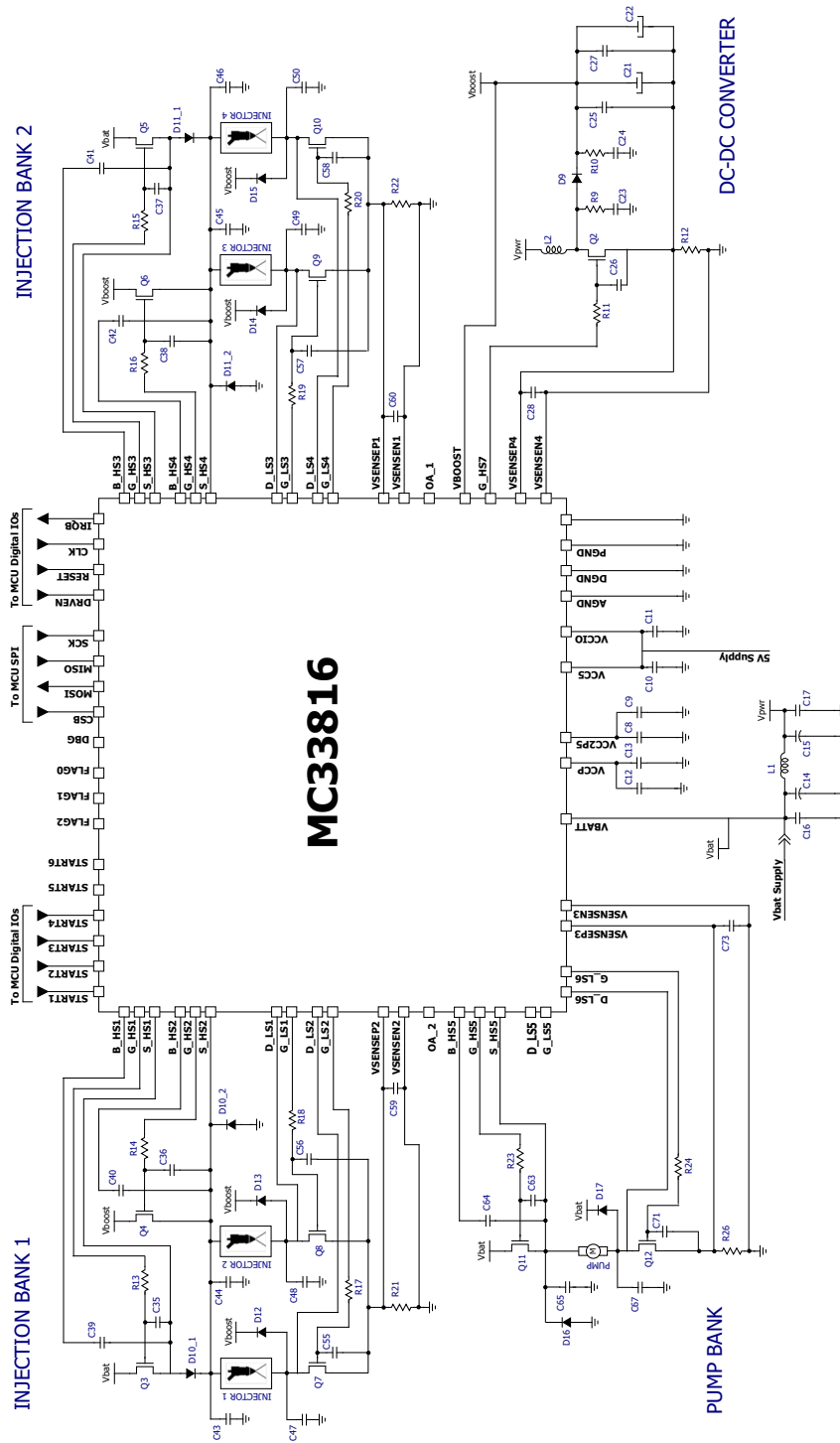


**Figure 1. Typical Four Injector Two Bank Application Schematic**

# 4    Application Instructions

This topology can be used on the evaluation board KIT33816FRDMEVM. Register settings and microcode downloads can be achieved by using the KL25Z embedded on the KIT33816FRDMEVM.

Each bank is individually managed by one microcore of the digital channel 1 as described next:

- The bank # 1 is managed by the digital microcore Uc0Ch1 with diagnostics
- The bank # 2 is managed by the digital microcore Uc1Ch1 without diagnostics

The two microcores of the second channel (Channel 2) drive the DC-DC and the fuel pump as described next:

- The VFM (Variable Frequency Modulation) is managed by the digital microcore Uc0Ch2
- The fuel pump is managed by the digital microcore Uc1Ch2.

   **This application note only focuses on BANK1 diagnostics managed by the digital microcore Uc0Ch1. Refer to AN4849 for register settings and microcode related to injection or DC-DC, unless specified in this document.**

The following is the start-up sequence:

- Apply a battery voltage between 9.0 V and 16 V
- Download the registers Channel Configuration, Main Configuration, IO Configuration, and Diagnostic Configuration
- Download the dedicated microcode in the Logic Channel 1 and Logic Channel 2 Data RAM
- Set '1' in the pre-flash enable bit and en dual seq bit in the Flash_enable register of channel 1 (0x100) and channel 2 (0x120)

The register configurations and the microcodes are detailed in the following chapters.

**Table 1. Example of Injection Current Profile Key Parameters**

| Parameter Name | Description | Value |
|---|---|---|
| $I_{BOOST}$ | Current threshold in Boost Phase | 16.09 A |
| $I_{PEAK}$ | Current threshold in Peak Phase (Depends on injectors type) | 14.89 A |
| $I_{HOLD}$ | Current threshold in Hold Phase | 8.89 A |
| $t_{PEAK\_OFF}$ | Fixed time for high-side switch off in Peak Phase | 10 μs |
| $t_{PEAK\_TOT}$ | Fixed time for end of Peak Phase | 500 μs |
| $t_{BYPASS}$ | Fixed time for Bypass Phase | 20 μs |
| $t_{HOLD\_OFF}$ | Fixed time for high-side switch off in Hold Phase | 10 μs |
| $t_{HOLD\_TOT}$ | Fixed time for end of Hold Phase (timeout) | 10 ms |
| $t_{INJMAXBOOST}$ | Maximum time allowed to reach $I_{BOOST}$ (Depends on injectors type) | 500 μs |

**Diagnostics interrupts description:**

Diagnostics interrupts are handled in two different subroutines: automatic interrupt and software interrupt.

Status_reg_uc0 register (0x105) is used to inform MCU on error detected, and then control register is used to unlock the Bank.

In all cases, the IRQB pin is set low to inform the MCU about the error detected in MC33816.

**Software Interrupts** can be filtered by their req id (stored in Uc0_irq_status Registers 0x10F):

Pre-Diagnostics checks (req id = 1): interrupt occurs if the high-side $V_{BOOST}$/ $V_{BAT}$, $V_{DS}$ or $V_{SRC}$ are low or $V_{DS}$ low-side is low.

If an error occurs, the Status_reg_uc0 register (0x105) bit 7 sets high, Bank1 sets OFF, and MCU needs to write a 1 to the control register (0x101) bit 7 to unlock the Bank1.

Boost Error (Req id = 1): If $I_{BOOST}$ is not reached before $t_{INJMAXBOOST}$ = 500 $\mu$s, this number has to be set according to the injector characteristics.

If an error occurs, Status_reg_uc0 register (0x105) bit 5 sets high, Bank1 sets OFF, and MCU needs to write a 1 to the Ctrl_reg_uc0 register (0x101) bit 5 to unlock the Bank1.

Hold Error (Req id = 2): If Start signal is still high after $t_{HOLD\_OFF}$.

If an error occurs, the Status_reg_uc0 register (0x105) bit 4 sets high, Bank1 sets OFF, and MCU needs to write a 1 to the Ctrl_reg_uc0 register (0x101) bit 4 to unlock the Bank1.

**An automatic Interrupt** occurs during actuation, if comparators feedback is different than the error table (see Diagnostics Configuration Registers).

If an error occurs, the Status_reg_uc0 register (0x105) bit 6 sets high, Bank1 sets OFF, and MCU needs to write a 1 to the Ctrl_reg_uc0 register (0x101) bit 6 to unlock the Bank1.

**Table 2. Status_reg_uc0 Registers (0x105) Configuration**

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | status_register | | | | | | | | | | | | | | | |
| Value | X | X | X | X | X | X | X | X | SW Interrupt Status | Auto Interrupt Status | Boost Error Status | Hold Error Status | X | X | X | X |

Reading this register indicates the type of fault.

**Table 3. Ctrl_reg_uc0 Registers (0x101) Configuration**

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | control_register_shared | | | | | | | | control_register | | | | | | | |
| Value | X | X | X | X | X | X | X | X | SW Interrupt Unlock | Auto Interrupt Unlock | Boost Error Unlock | Hold Error Unlock | X | X | X | X |

Depending on the status register information, one of the bits must be set to 1 to unlock the BANK.

**MC33816 Diagnostics,** Rev. 3.0

# 5 Diagnostic Descriptions

The MC33816 gives the possibility to check faults using two different methods:

- Automatic diagnostics (Actuation phase):
  - Boost Phase (HSBoost ON): automatic diagnostics are used during actuation phase; it performs a coherency check between an output and the related $V_{DS}$ feedback (for all the outputs) and $V_{SRC}$ feedback (for the high-side outputs only).
  - Peak and Hold phase (HSBat ON): automatic diagnostics are used during actuation phase; it performs a coherency check between an output and the related $V_{DS}$ feedback (for all the outputs) and $V_{SRC}$ feedback (for the high-side outputs only).
- Idle Diagnostics (Pre-actuation): Internal voltage biasing $V_{BIAS}$ should be applied to the load to enable diagnostics in this phase.
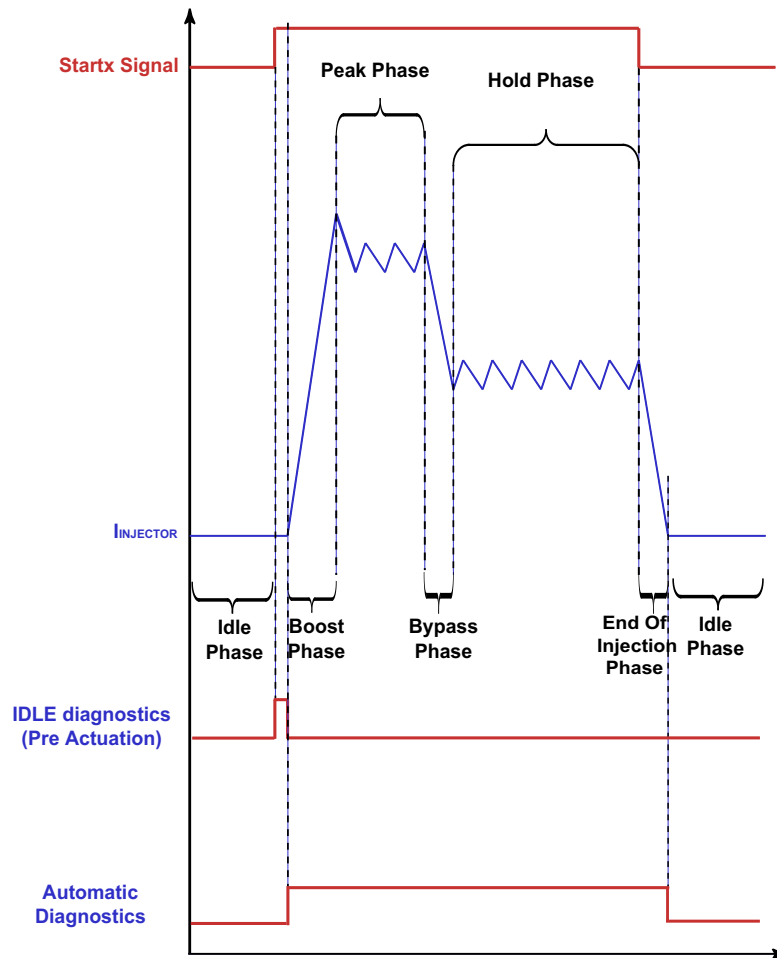


**Figure 2. Typical Peak and Hold Current Profile with Diagnostics**

Several fault cases could occur in the application, this chapter describes most of them, and explains how the MC33816 is able to detect them.

### 5.1        Idle Diagnostics (Pre-actuation)

As described in Figure 2, idle diagnostics start after a rising edge on the start 1 or start 2 (Bank 1). A voltage biasing $V_{BIAS}$ should be applied to the load, to enable electrical diagnosis while the external load is not actuating the power stage.

This $V_{BIAS}$ voltage is generated by:
- the activation of the $SRC_{PUX}$ pull-up voltage source connected to each of the S_HSx pins. Each pull-up voltage source is supplied from VCC5
- the activation of each $SRC_{PDX}$ pull-down current source connected to each of the D_LSx pins. Each pull-down voltage source is referenced to ground

When the battery voltage $V_{BATT}$ is in the nominal range or greater, the external load is biased at a minimum voltage of typically 3.8 V. In a low battery voltage condition ($V_{BATT}$ < 8.0 V), the load is biased at half the $V_{BATT}$ voltage, to guarantee symmetrical voltage margins to high-side and low-side VDS comparators.
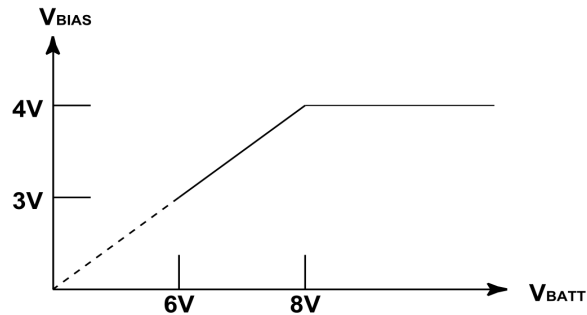


**Figure 3. Biasing Voltage vs. $V_{BATT}$**

The bias generators can be kept ON even during actuation, to control the voltage on the source, even if the MOSFET is OFF. This does not impact the application, because of their low strength. If at least one MOSFET is turned ON, it fixes the voltage on the load and does not affect the bias.

These pre-actuation diagnostics are used to ensure the injectors can be turned ON safely. If an error occurs in any of the following cases, the MC33816 keeps Bank 1 OFF until the MCU writes a 1 through the SPI to the Ctrl_reg_uc0 register (0x101) bit 7.

### 5.1.1 Normal Behavior
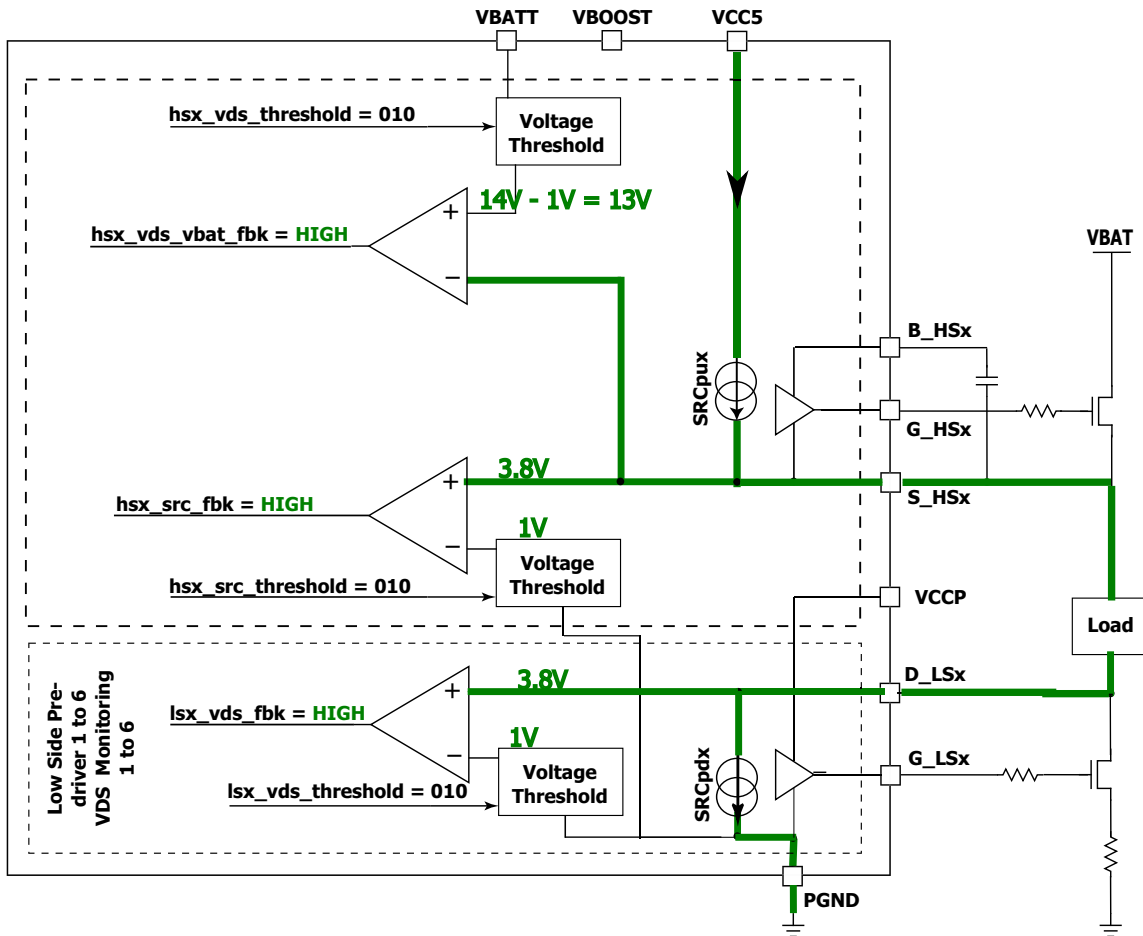


**Figure 4. Idle Normal Mode**

During normal operation, a current limited pull-up voltage source (SRC$_{PUX}$) generates a voltage on S_HSx (min. 3.8 V). Its current goes to the load and to a pull-down current source on D_LSx, generating a 3.8 V min. voltage. Drain source voltage on the high-side is not monitored directly, and since there is no pin for the drain, monitoring is directly done from V$_{BAT}$ or V$_{BOOST}$, and only HS2 and HS4 can use V$_{BOOST}$ as a reference. Voltage thresholds are selected to be lower than the voltage generated.

**Table 4. Normal Mode Truth Table**

| Error Case | LSx_vds_fbk | HSx_src_fbk | Hsx_vds_Vbat_fbk | S_HSx voltage V$_{BAT}$ | D_LS_x voltage |
|---|---|---|---|---|---|
| Normal mode | 1 | 1 | 1 | 3.8 | 3.8 |

### 5.1.2 High-side Source or Low-side Drain Shorted to GND
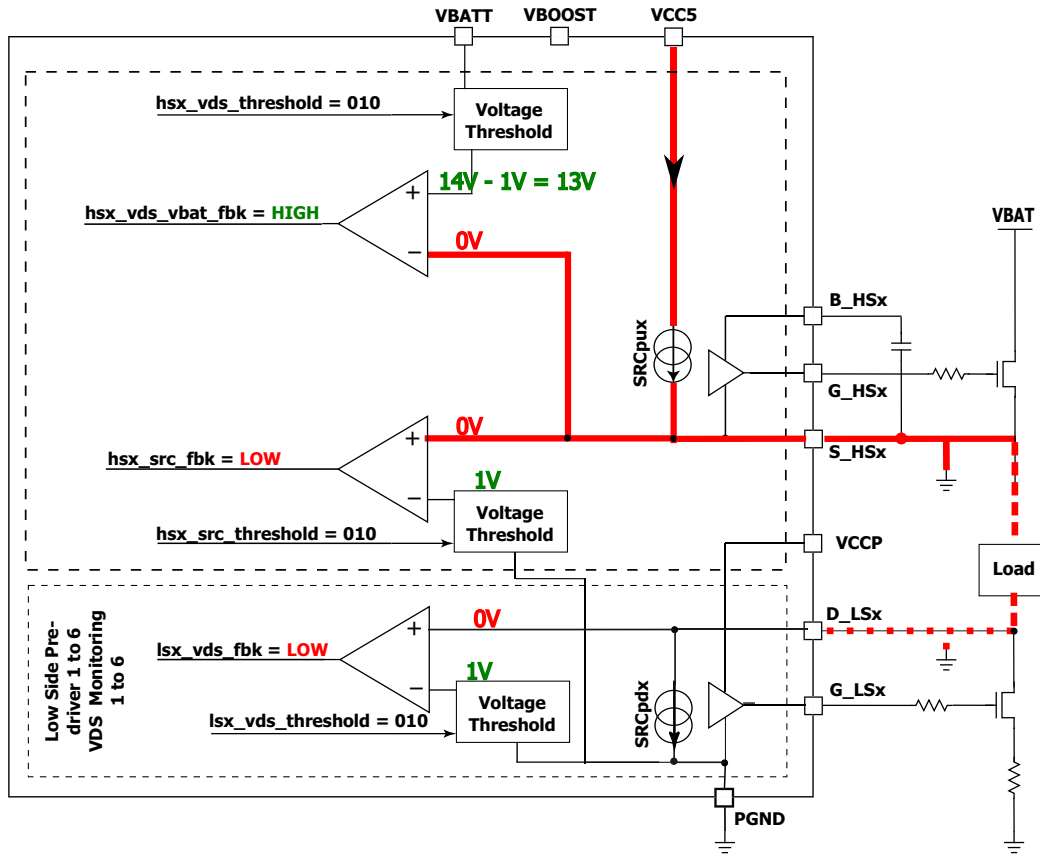


**Figure 5. High-side Source or Low-side Drain Shorted**

In cases where the High Source (S_HSx) shorts to GND or the Low-side Drain (D_LSx) shorts to GND, the current limited voltage source pulls to ground, and the voltage on S_HSx and D_LSx is 0 V. A diagnostic error is detected, and since the high-side $V_{SRC}$ and low-side $V_{DS}$ feedback are low, the bank does not turn ON.

**Table 5. S_HSx or D_LSx Shorted to GND Truth Table**

| Error Case | LSx_vds_fbk | HSx_src_fbk | Hsx_vds_Vbat_fbk | S_HSx voltage $V_{BAT}$ | D_LS_x voltage |
|---|---|---|---|---|---|
| Normal mode | 1 | 1 | 1 | 3.8 | 3.8 |
| D_LS GND short | 0 | 0 | 1 | 0 | 0 |

## 5.1.3 Drain Source Low-side Shorted to GND



**Figure 6. Low-side Drain Source Shorted**

In cases where the Low-side Drain Source shorts, D_LSx pulls to 0 V, the current limited voltage source pulls to ground, and the voltage on S_HSx and D_LSx is 0 V. A diagnostic error is detected, since the high-side $V_{SRC}$ and low-side $V_{DS}$ feedback are low.

**Table 6. Drain Source Low-side Shorted Truth Table**

| Error Case | LSx_vds_fbk | HSx_src_fbk | Hsx_vds_Vbat_fbk | S_HSx voltage $V_{BAT}$ | D_LS_x voltage |
|---|---|---|---|---|---|
| Normal mode | 1 | 1 | 1 | 3.8 | 3.8 |
| Low-side Drain Source Short | 0 | 0 | 1 | 0 | 0 |

### 5.1.4 Drain Source High-side Shorted to VBAT



**Figure 7. High-side Drain Source Shorted**

The diagnostic fails in cases where the high-side drain source shorts. As a consequence, S_HSx and D_LSx pulls up to $V_{BAT}$, the difference between drain and source on the high-side goes negative, resulting in low feedback on the high-side $V_{DS}$.

**Table 7. Drain Source High-side Shorted Truth Table**

| Error Case | LSx_vds_fbk | HSx_src_fbk | Hsx_vds_Vbat_fbk | S_HSx voltage $V_{BAT}$ | D_LS_x voltage |
|---|---|---|---|---|---|
| Normal mode | 1 | 1 | 1 | 3.8 | 3.8 |
| How-side Drain Source Short | 1 | 1 | 0 | 14 | 14 |

## 5.1.5 Open Load



**Figure 8. Open Load**

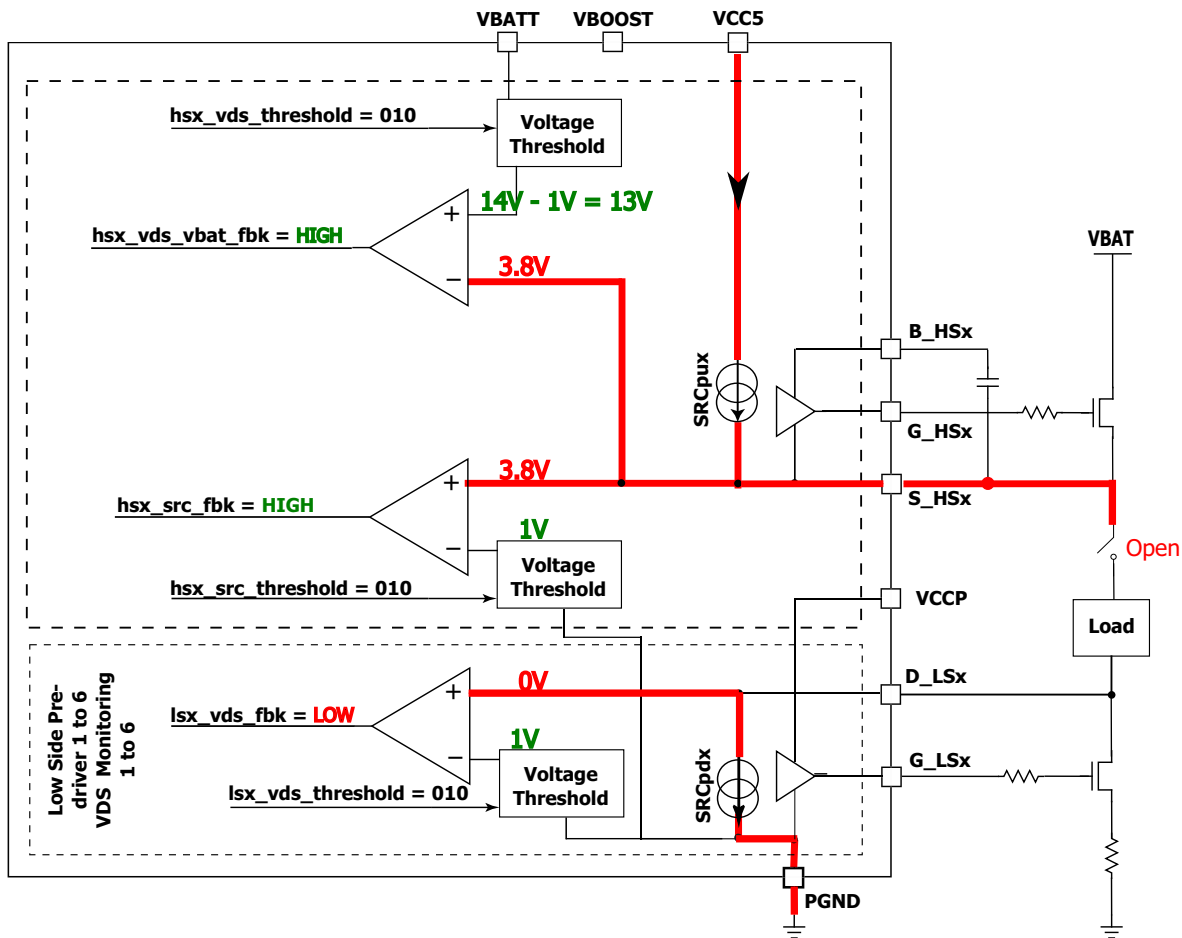If one of the sides of the load is not connected properly, there is no current path between S_HSx and D_LSx. The voltage on D_LSx is forced to ground, because of the $SRC_{PDX}$ current pull-down. The diagnostic fails, because the low-side $V_{DS}$ feedback is low.

### Table 8. OpenLoad Truth Table

| Error Case | LSx_vds_fbk | HSx_src_fbk | Hsx_vds_Vbat_fbk | S_HSx voltage $V_{BAT}$ | D_LS_x voltage |
|---|---|---|---|---|---|
| Normal mode | 1 | 1 | 1 | 3.8 | 3.8 |
| OpenLoad on Low-side | 0 | 1 | 1 | 3.8 | 0 |
| OpenLoad on How-side | 0 | 1 | 1 | 3.8 | 0 |

## 5.1.6 Faults Not Detected in Idle Phase

There are different cases that cannot be detected in the idle phase:

- High-side $V_{BAT}$ or $V_{BOOST}$ open: not possible to be detected since both are OFF in idle phase
- LS open: not possible to be detected since it is OFF in idle phase
- Short between load pins: not possible to detect, because it allows the bias current to pass through

All these faults are detected in the actuation mode only.

## 5.2 Actuation Phase

The bias voltage used for idle diagnostics is kept ON, to predict the voltage on each pin even if the MOSFETs are OFF. In this case, when the MOSFETs are OFF, there is a 3.8 V voltage on the high-side source. In each case, if an error occurs, the MC33816 turns bank 1 OFF, keeps it OFF, and sets the Status_reg_uc0 register (0x105) bit 6 high until the MCU writes a 1 to the control register bit 6.

### 5.2.1 Actuation Diagnostics Peak and Hold phase (HS Boost OFF, HS Bat ON, LS ON)

#### 5.2.1.1 Normal Mode
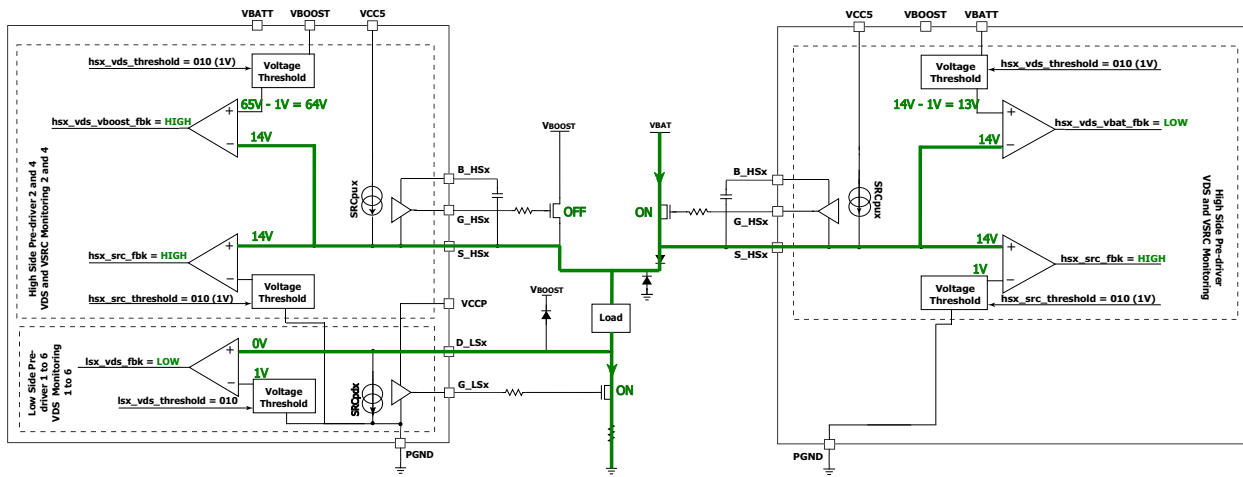


**Figure 9. Normal Mode Peak & Hold**

During peak and hold phase, the low-side is fully ON and $V_{BAT}$ high-side is controlled in PWM to regulate the current inside the injector.

To have a device as flexible as possible, detection error during automatic diagnostics is configurable for each low-side and high-side. To configure which case will lead the device to an error, it is necessary to set the registers "Error_table" for each low-side $V_{DS}$, high-side $V_{DS}$, and high-side $S_{RC}$ where diagnostics are needed (refer to Diagnostics Configuration Registers). In Normal mode, the 33816 comparator outputs should be in the following state:

**Table 9. Actuation in Normal Mode Truth Table**

| Error Case | LSx_vds_fbk | HSx_src_vbat_fbk | Hsx_vds_Vbat_fbk | HSx_vds_Vboost_fbk | HSx_vsrc_Vboost_fbk |
|---|---|---|---|---|---|
| Normal mode | 0 | 1 | 0 | 1 | 1 |

## 5.2.1.2 High-side (Bat or Boost) Source Shorted to GND
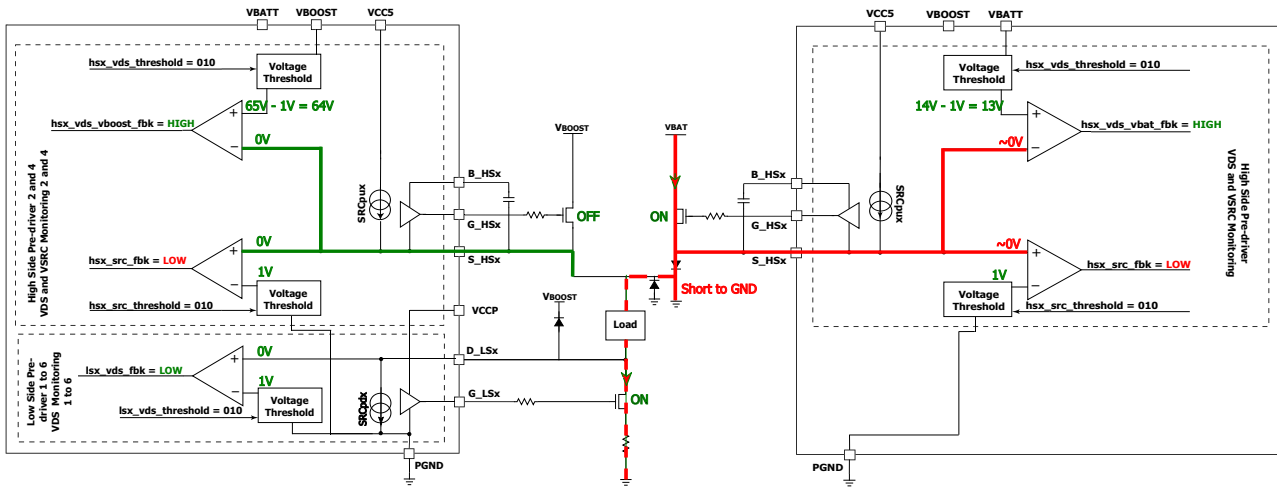


**Figure 10. High-side Source Shorted to GND**

When S_HSx shorts to GND, the 33816 detects an overcurrent, due to the $V_{DS}$ monitoring on the $V_{BAT}$ high-side. The high-side shuts down as soon as the current is substantial enough to generate a higher drop across the MOSFET than the threshold. In this case, it's important to set a threshold (1.0 V, in this case) and a filter time to the lowest value allowed by the application, to quickly detect it (refer to Filter Time). The automatic diagnostic fails, because high-side $V_{DS}$ feedback is high.

### Table 10. High-side Source Shorted to GND Truth Table

| Error Case | LSx_vds_fbk | HSx_src_vbat_fbk | Hsx_vds_Vbat_fbk | HSx_vds_Vboost_fbk | HSx_vsrc_Vboost_fbk |
|---|---|---|---|---|---|
| Normal mode | 0 | 1 | 0 | 1 | 1 |
| S_HSx $V_{BAT}$ or $V_{BOOST}$ GND Short | 0 | 0 | 1 | 1 | 0 |

This case is also applicable when there is a short between the two load pins, substantial current flows inside $V_{BAT}$ HS and LS until the difference between drain and source is higher than the threshold.

## 5.2.1.3 High-side $V_{BOOST}$ Short Drain Source
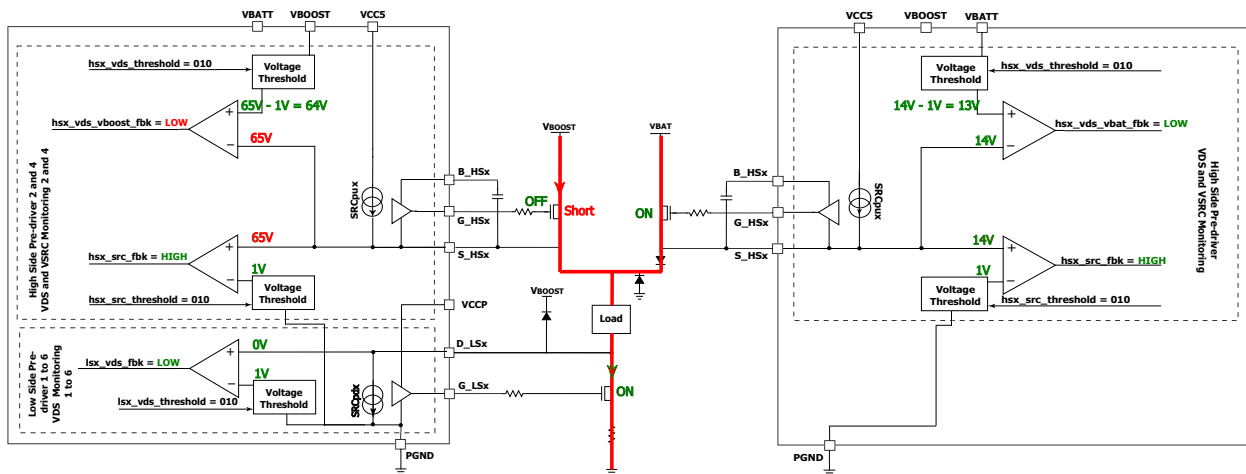


**Figure 11. High-side $V_{BOOST}$ Shorted**

During a peak and hold phase, $V_{BOOST}$ high-side should be OFF, but if there is a short-circuit between the drain and source, the voltage on the $V_{BOOST}$ high-side source rises to $V_{BOOST}$. The automatic diagnostic fails, because $V_{DS}$ on the $V_{BOOST}$ high-side is low.

**Table 11. High-side $V_{BOOST}$ Drain Source Shorted Truth Table**

| Error Case | LSx_vds_fbk | HSx_src_vbat_fbk | Hsx_vds_Vbat_fbk | HSx_vds_Vboost_fbk | HSx_vsrc_Vboost_fbk |
|---|---|---|---|---|---|
| Normal mode | 0 | 1 | 0 | 1 | 1 |
| HSvboot Drain Source short | 0 | 1 | 0 | 0 | 1 |

### 5.2.1.4 High-side $V_{BAT}$ Open
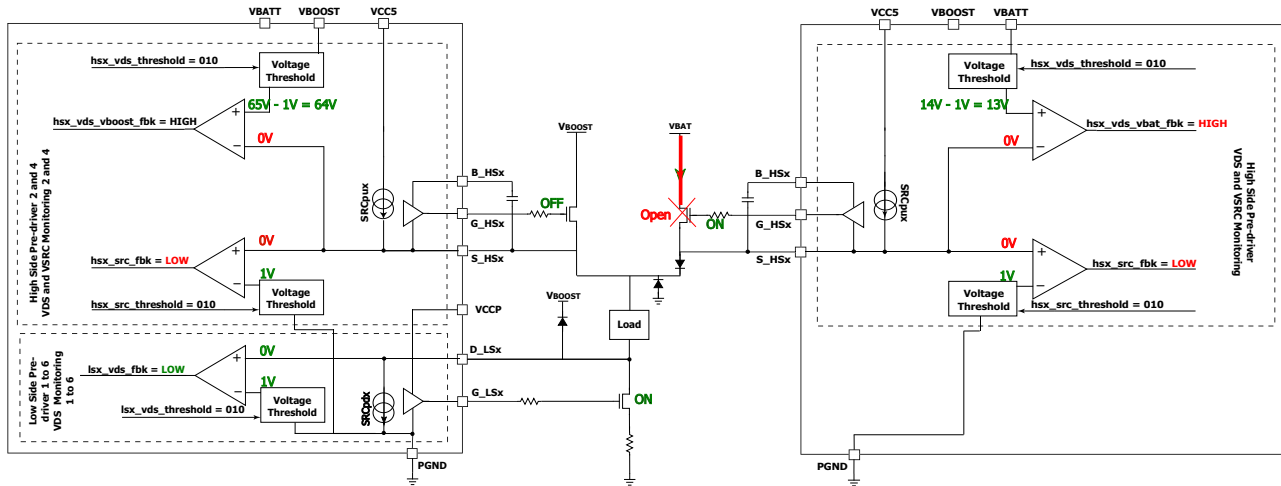


**Figure 12. High-side Open**

During peak and hold phase, high-side $V_{BAT}$ is ON. If it's open or not controlled properly, S_HSx voltage will be lower than expected. The automatic diagnostic fails because on $V_{BAT}$ high-side, the $V_{DS}$ feedback is high and $V_{SRC}$ feedback is low.

**Table 12. High-side $V_{BAT}$ Open Truth Table**

| Error Case | LSx_vds_fbk | HSx_src_vbat_fbk | Hsx_vds_Vbat_fbk | HSx_vds_Vboost_fbk | HSx_vsrc_Vboost_fbk |
|---|---|---|---|---|---|
| Normal mode | 0 | 1 | 0 | 1 | 1 |
| HS vbat open | 0 | 0 | 1 | 1 | 0 |

This case is only detectable in actuation mode.
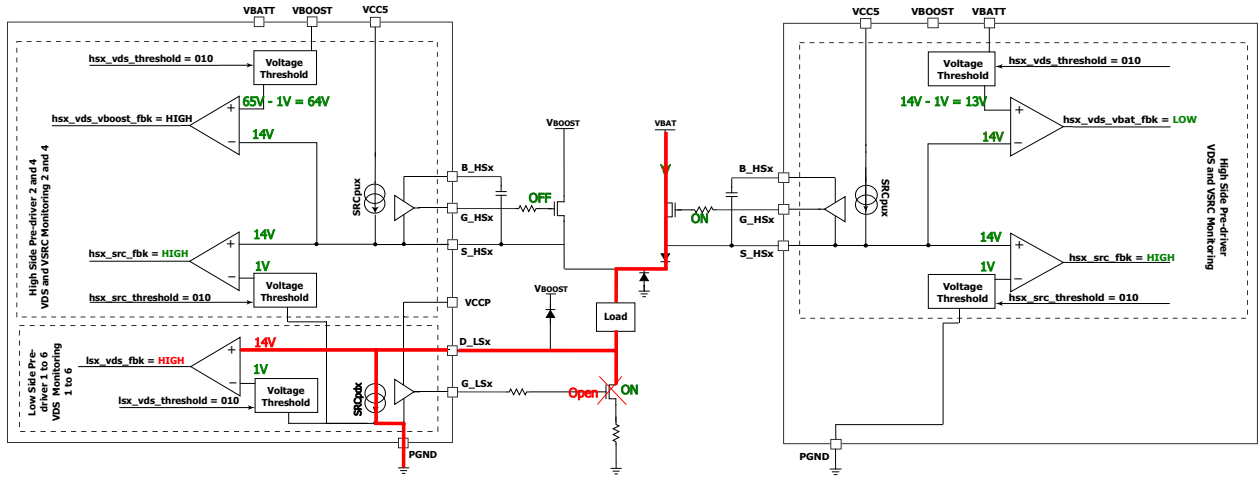
### 5.2.1.5 Low-side Open



**Figure 13. Low-side Open**

With the low-side open, current on the D_LSx pin flows through the load to the internal pull-down (SCR$_{PDX}$) and the voltage rises to V$_{BAT}$. Automatic diagnostics fail, because the low-side V$_{DS}$ feedback is high.

### Table 13. Low-side Open Truth Table

| Error Case | LSx_vds_fbk | HSx_src_vbat_fbk | Hsx_vds_Vbat_fbk | HSx_vds_Vboost_fbk | HSx_vsrc_Vboost_fbk |
|---|---|---|---|---|---|
| Normal mode | 0 | 1 | 0 | 1 | 1 |
| LS Open | 1 | 1 | 0 | 1 | 1 |

This is one case only detectable in actuation mode.

### 5.2.1.6 Drain Low-side Shorted to V$_{BAT}$ or V$_{BOOST}$
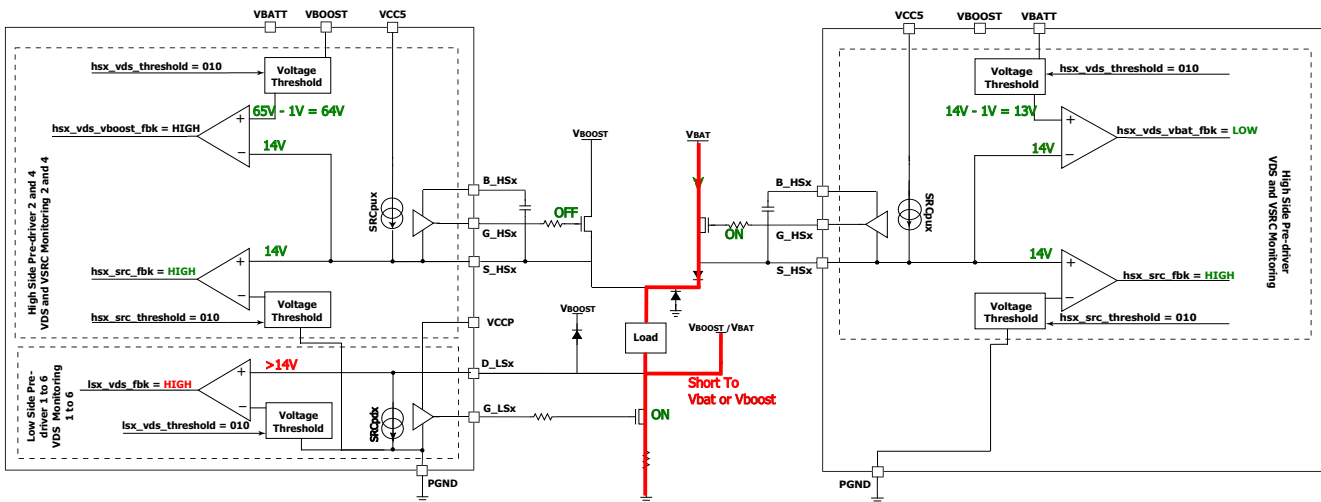


**Figure 14. Drain Low-side Shorted**

When the low-side drain is shorted to V$_{BAT}$ or V$_{BOOST}$ (low probability case), the voltage on D_LSx rises to V$_{BAT}$/V$_{BOOST}$. Voltage thresholds and filter times must be set to the lowest value allowed by the application, to detect the error as fast as possible. The automatic diagnostic fails, because the low-side V$_{DS}$ feedback is high.

**MC33816 Diagnostics,** Rev. 3.0

**Table 14. D_LS Battery Short Truth Table**

| Error Case | LSx_vds_fbk | HSx_src_vbat_fbk | Hsx_vds_Vbat_fbk | HSx_vds_Vboost_fbk | HSx_vsrc_Vboost_fbk |
|---|---|---|---|---|---|
| Normal mode | 0 | 1 | 0 | 1 | 1 |
| D_LS Batt short Depending on external MOS behavior | 1 | 1 | 0 | 1 | 1 |

#### 5.2.1.7 Cases Undetectable During Peak and Hold Phase

There are different cases that can't be detected during peak and hold phase:

- Drain low-side shorted to GND: not detectable since the low-side is ON, in this case (detectable in idle phase)
- High-side $V_{BAT}$ drain source shorted: not detectable since the high-side is ON, in this case (detectable in idle phase)
- High-side $V_{BOOST}$ open: not detectable since the high-side $V_{BOOST}$ is OFF in this mode (detectable during $V_{BOOST}$ phase)

### 5.2.2 Actuation Diagnostics Boost Phase (HS Boost ON, HS Bat ON, LS ON)

During Boost phase, boost voltage is used to turn the injector ON as fast as possible, high-side $V_{BOOST}$ and low-side are ON. The high-side $V_{BAT}$ source needs to be turned ON, to avoid errors during diagnostics, which has no impact on the application. Another option would be to disable automatic diagnostics on the high-side $V_{BAT}$ source during boost phase.

In this example, the MC33816 automatic diagnostics are configured using instruction "endiags" (refer to Application Source Code). During actuation phase, automatic diagnostics monitor HS $V_{BAT}$ $V_{DS}$, HS $V_{BAT}$ $V_{SRC}$, HS $V_{BOOST}$ $V_{DS}$, and LS $V_{DS}$ continuously. To simplify the diagnostics code, HS $V_{BAT}$ is kept ON during Boost phase to avoid unwanted errors on the $V_{BAT}$ source.

#### 5.2.2.1 Normal Mode
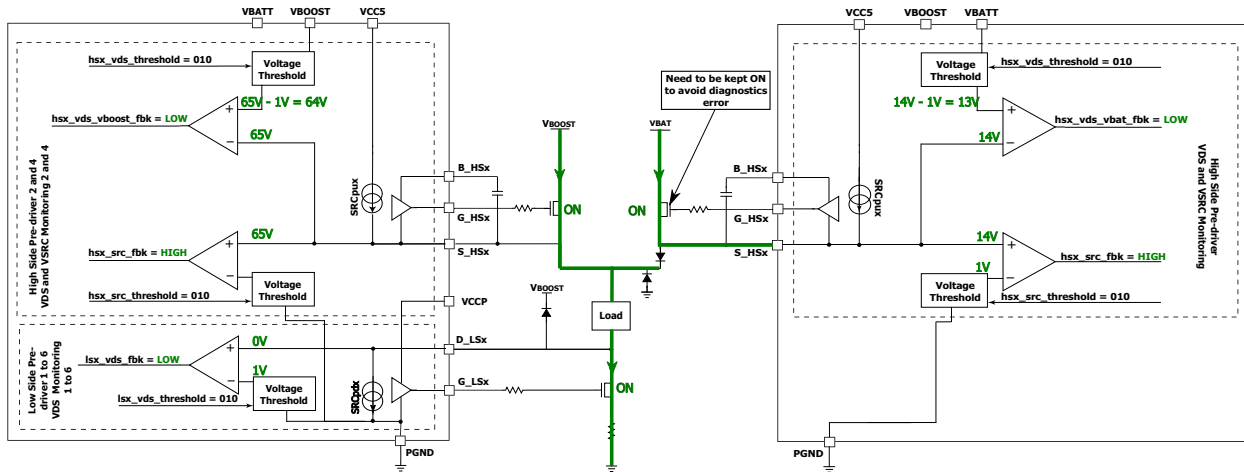


**Figure 15. Boost Phase Normal Mode**

During boost phase, the high-side boost is fully ON to reach boost current as fast as possible, high-side $V_{BAT}$ is ON (for diagnostic purposes), and the low-side is fully ON. As with the peak and hold phase, the high-side $V_{BOOST}$ error table must be set-up accordingly (see Diagnostics Configuration Registers).

**Table 15. Boost Phase Normal Mode Truth Table**

| Error Case | LSx_vds_fbk | HSx_src_vbat_fbk | Hsx_vds_Vbat_fbk | HSx_vds_Vboost_fbk | HSx_vsrc_Vboost_fbk |
|---|---|---|---|---|---|
| Normal mode | 0 | 1 | 0 | 0 | 1 |

## 5.2.2.2 High-side Boost Source Shorted to GND
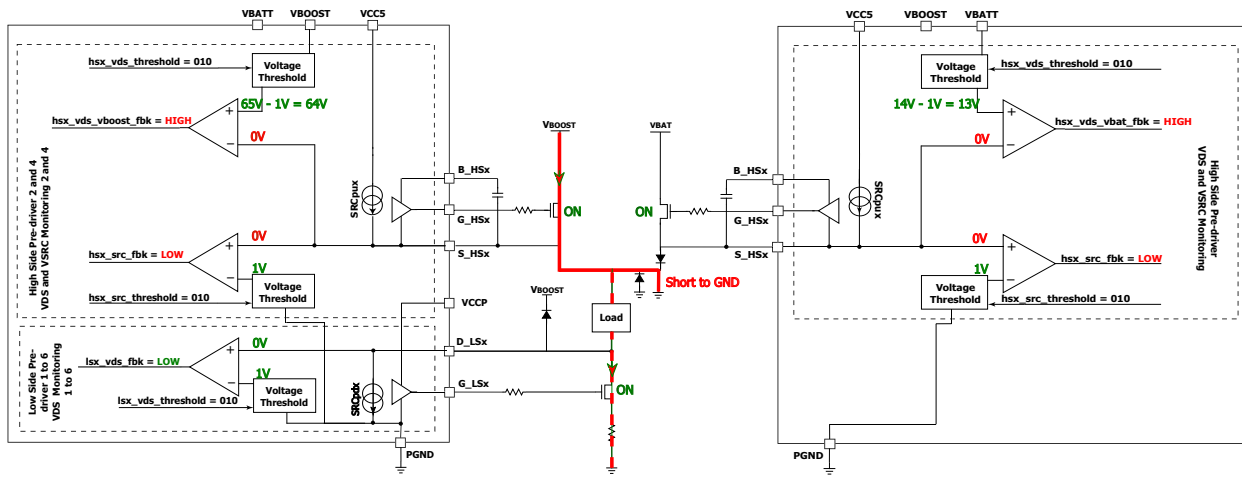


**Figure 16. High-side Source Shorted to GND**

The same behavior as in the Peak and Hold phase except this time the short is from $V_{BOOST}$ to GND. The comparator threshold must be set as low as possible to detect the overcurrent faster and avoid any damage to the MOS. The automatic diagnostic on high-side $V_{BOOST}$ fails because $V_{DS}$ monitoring is high.

**Table 16. High-side Boost Source Shorted to GND Truth Table**

| Error Case | LSx_vds_fbk | HSx_src_vbat_fbk | Hsx_vds_Vbat_fbk | HSx_vds_Vboost_fbk | HSx_vsrc_Vboost_fbk |
|---|---|---|---|---|---|
| Normal mode | 0 | 1 | 0 | 0 | 1 |
| HSvbat Drain Source short | 0 | 0 | 1 | 1 | 0 |

## 5.2.2.3 High-side V_BOOST Open
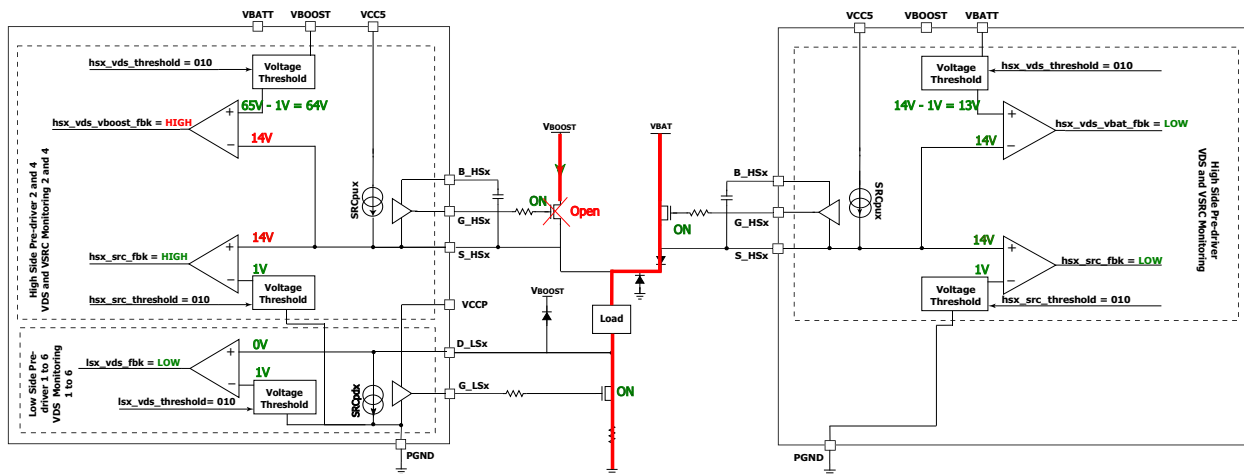


**Figure 17. High-side V_BOOST Open**

When $V_{BOOST}$ high-side is open, the voltage on S_HSx floats and forced to 0 V, due to the parasitic leakage on the S_HSx pin. The automatic diagnostic on high-side $V_{BOOST}$ fails because $V_{DS}$ feedback is high.

**MC33816 Diagnostics,** Rev. 3.0

**Table 17. High-side Boost Open Truth Table**

| Error Case | LSx_vds_fbk | HSx_src_vbat_fbk | Hsx_vds_Vbat_fbk | HSx_vds_Vboost_fbk | HSx_vsrc_Vboost_fbk |
|---|---|---|---|---|---|
| Normal mode | 0 | 1 | 0 | 0 | 1 |
| HS vboot open | 0 | 1 | 0 | 1 | 1 |

This case is undetectable in idle phase.

### 5.2.2.4 Low-side Open
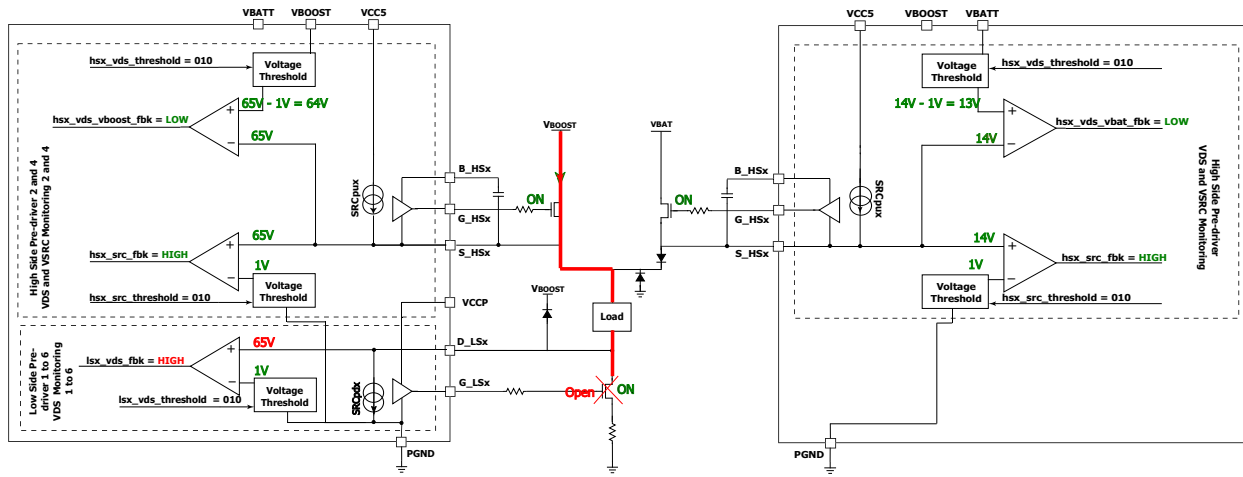


**Figure 18. Low-side Open**

When the low-side is not connected properly, the voltage on D_LSx is around a $V_{BOOST}$ of 65 V. The automatic diagnostic fails due to $V_{DS}$ feedback on the low-side.

**Table 18. Low-side Open Truth Table**

| Error Case | LSx_vds_fbk | HSx_src_vbat_fbk | Hsx_vds_Vbat_fbk | HSx_vds_Vboost_fbk | HSx_vsrc_Vboost_fbk |
|---|---|---|---|---|---|
| Normal mode | 0 | 1 | 0 | 0 | 1 |
| LS open | 1 | 1 | 0 | 0 | 1 |

This case is undetectable in idle phase.

### 5.2.2.5 Drain Low-side Shorted to V$_{BAT}$ or V$_{BOOST}$
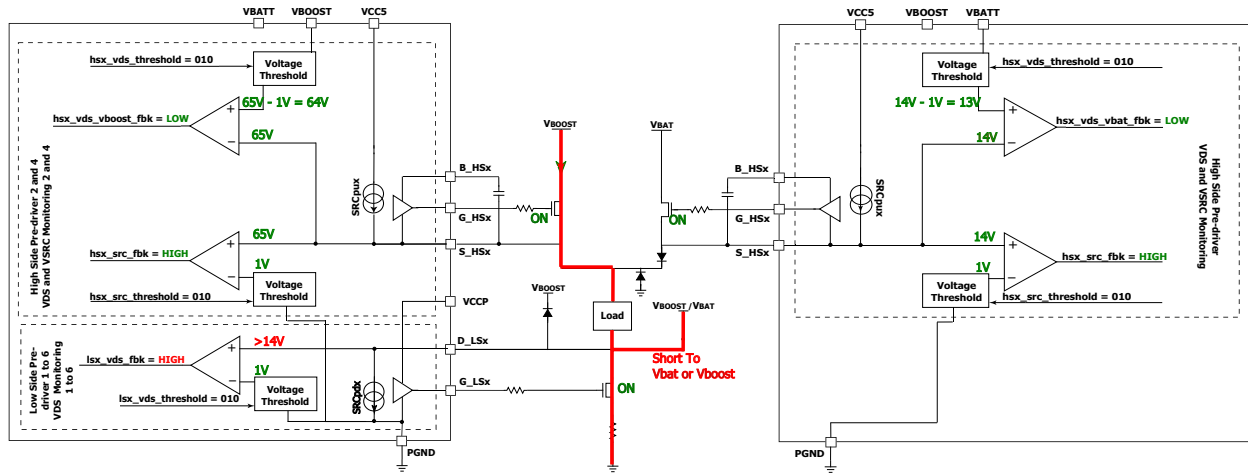


**Figure 19. Drain Low-side Shorted to V$_{BAT}$ or V$_{BOOST}$**

This is the same behavior as in the Peak and Hold phase, when the drain low-side is shorted to V$_{BAT}$ or V$_{BOOST}$ (Low probability), with a short to GND on V$_{BOOST}$ or V$_{BAT}$. The automatic diagnostic fails, because the voltage on D_LSx is higher than the V$_{DS}$ threshold.

**Table 19. Drain Low-side Shorted to V$_{BAT}$ or V$_{BOOST}$ Truth Table**

| Error Case | LSx_vds_fbk | HSx_src_vbat_fbk | Hsx_vds_Vbat_fbk | HSx_vds_Vboost_fbk | HSx_vsrc_Vboost_fbk |
|---|---|---|---|---|---|
| Normal mode | 0 | 1 | 0 | 0 | 1 |
| D_LS Boost/Bat short | 1 | 1 | 0 | 0 | 1 |

### 5.2.2.6 Cases Undetectable During Boost Mode

There are different cases undetectable in the Boost phase:

- Drain low-side shorted to GND: not detectable, since the low-side is ON in this case (detectable in the idle phase)
- High-side V$_{BAT}$ or V$_{BOOST}$ drain source shorted: not detectable, since in this case, high-side is ON (detectable in the idle phase)
- High-side V$_{BOOST}$ open: not detectable, since the high-side V$_{BOOST}$ is OFF in this mode (detectable during the V$_{BOOST}$ phase)
- A short on the high-side V$_{BAT}$ source S_HSx (before diode): the high-side V$_{BAT}$ is OFF in this mode, but detectable during the idle phase

# 6    Software

## 6.1        Interrupt State Machine

The following state diagrams describe how the MCU knows which interrupt occurred and which fault has been detected during both automatic and software interrupts.

For injectors actuation and DCDC state diagram, refer to AN4849.

**Initialization Phase**
   Set IRQB pin High
   Current sense operational amplifier gain setting
   Load the eoinj line label Code RAM address into the register jr1
   Load the idle line label Code RAM address into the register jr2
   Define wait table entry # 1: Jump to End Of Injection Phase if the start signal goes low

Start 1 signal is high?

Yes

Automatic Diagnostics are also enabled during those 2 phases

**Pre Actuation**
**( Idle diagnostics)**

If     - High Side Vbat Vds = Low
or     - Low Side Vds = Low
or     - High Side Vboost Vds = Low
or     - High Side Vbat Vsrc = Low
or     - High Side Vboost Vsrc = Low

**Boost Phase**

IBoost reached before injMaxTBoost ?

**Hold Phase**

Start pin still high after Thold_tot ?

No          No

Yes (Software Interrupt 1)          No (Software Interrupt 2)          Yes (Software Interrupt 3)

**Software Interrupt :**
   Turn OFF Bank 1
   Disable automatic diagnostics
   Set IRQB pin Low
   Turn ON DCDC

IRQ Status = 0          No          IRQ Status = 1          No          IRQ Status = 2

Yes (Boost error)          Yes (Idle Diagnsotics error)          Yes (Hold error)

**IBoost  never reached:**
   Set Status bit 5 High
   Wait here until Control bit b5 write to 1

**IBoost  never reached:**
   Set Status bit 7 High
   Wait here until Control bit b7 write to 1

**IBoost  never reached:**
   Set Status bit 4 High
   Wait here until Control bit b4 write to 1

Control bit b5 = 1          No          Control bit b7 = 1          No          Control bit b4 = 1          No

**Restore:**
   Set IRQB High
   Reset all control, status, err-seq registers
   and reenable irq  generation from auto diag

**Restore:**
   Set IRQB High
   Reset all control, status, err-seq registers
   and reenable irq  generation from auto diag

**Restore:**
   Set IRQB High
   Reset all control, status, err-seq registers
   and reenable irq  generation from auto diag

**Figure 20. Software Interrupt State Machine**

**Initialization Phase**
  Set IRQB pin High
  Current sense operational amplifier gain setting
  Load the eoinj line label Code RAM address into the register jr1
  Load the idle line label Code RAM address into the register jr2
  Define wait table entry # 1: Jump to End Of Injection Phase if the start signal goes low

Start 1 signal is high?

Yes

**Pre Actuation**
**( Idle diagnostics)**

If    - High Side Vbat Vds = Low
or    - Low Side Vds = Low
or    - High Side Vboost Vds = Low
or    - High Side Vbat Vsrc = Low
or    - High Side Vboost Vsrc = Low

No

**Actuation Phase = Boost + Peak + Hold**

Error occured ?
(according to Error table settings)

No -> normal operation

Yes (Software Interrupt 1)

Yes (Automatic Interrupt)

**Software interrupt (see previous state diagram)**

**Automatic Interrupt :**
  Turn OFF Bank 1
  Disable automatic diagnostics
  Set IRQB pin Low
  Turn ON DCDC
  Set Status Regsiter b6 high

MCU writes Control bit b6 = 1 ?

No

Yes

**Restore:**
  Set IRQB High
  Reset all control, status, err-seq registers
  and reenable irq  generation from auto diag

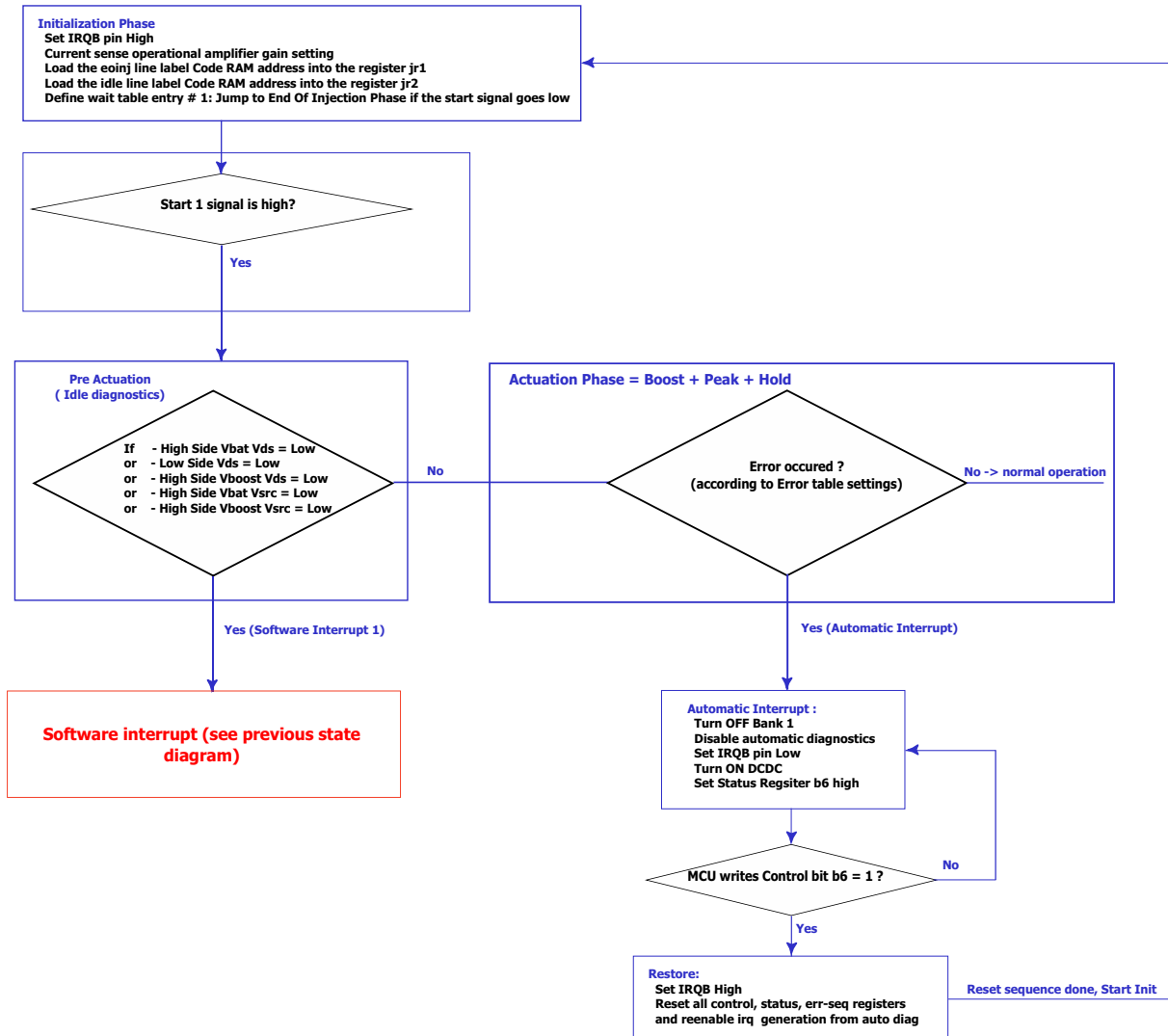Reset sequence done, Start Init

**Figure 21. Automatic Interrupt State Machine**

## 6.2 General Registers Setup

Unless specified, use the register settings described in AN4849. Only registers related to diagnostics and interrupts are described in the following chapter.

### 6.2.1 Main Configuration Registers

### Table 20. Driver_config Register (0x1C5)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | hs5_ ls36_ ovr | vccp_ ext_en | ls7_ ovr | vboost _mon_ en | vboost _disabl e_en | over_ temp_ irq_en | drv_en _irq_ en | vboost _irq_ en | vcc5_ irq_en | vccp_ irq_en | iret_en | irq_ uc1 ch2 en | irq_ uc0_ ch2_ en | irq_ uc1_ ch1_ en | irq_ uc0_ ch1_ en | irq_ mcu_ en |
| Value | X | X | X | X | X | X | X | X | 1 | X | X | X | X | X | X | 1 |

This register doesn't need to be set for the diagnostic on the external MOSFET, since it is handled in the microcode directly. If an error is detected, it forces IRQB low using the microcode. The return address (iret) is also determined in the microcode.

As an example, set vcc5_irq_en to '1', to force IRQB low in cases of undervoltage on Vcc5. When the undervoltage is gone, the IRQB pin is kept low until the user writes a '1' in the uv_vcc5 bit (Driver_status register (0x1D2)).

### 6.2.2 6.1.2 IO Configuration Registers

This register (one for each microcore) selects the feedback by which each microcore is enabled. Setting the bit to '1' generates an interrupt towards UcXChY, in case an error is detected on the HSx or LSx feedback.

### Table 21. Fbk_sens_uc0ch1 Register (0x180)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | ls6_ vds_ sens | ls5_ vds_ sens | ls4_ vds_ sens | ls3_ vds_ sens | ls2_ vds_ sens | ls1_ vds_ sens | hs5_ vsrc_ sens | hs5_ vds_ sens | hs4_ vsrc_ sens | hs4_ vds_ sens | hs3_ vsrc_ sens | hs3_ vds_ sens | hs2_ vsrc_ sens | hs2_ vds_ sens | hs1_ vsrc_ sens | hs1_ vds_ sens |
| Value | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

In this particular application (see the schematics in KIT33816AEEVM), microcore 0 Channel 1 controls HS1 as high-side $V_{BAT}$ and HS2 as high-side $V_{BOOST}$, LS1 and LS2. As show in table Table 21, an interrupt is generated if an error occurs on LS2 $V_{DS}$, LS1 $V_{DS}$, HS2 $V_{DS}$, HS1 $V_{DS}$, HS2 $V_{SRC}$, and HS1 $V_{SCR}$.

### 6.2.2.1    Threshold settings

Each comparator threshold is set on three bits. The $V_{DS}$ and $V_{SRC}$ thresholds are defined by registers 0x18A and 0x18B, for the high-side pre-drivers, and by the Vds_threshlod_ls_1 (0x18C) for the low-side pre-drivers.

As described during fault description, these threshold must be set according to the external MOSFET and maximum current level used in the application. As with KIT33816AEEVM, $R_{DS(ON)}$ MOSFET ~40 mΩ (worst case condition). The maximum current used in this application is 16.09 A, and overcurrent detection (using $V_{DS}$ monitoring) must be set at around 30% higher than max current allowed (20 A).

**Table 22. $V_{DS}$ and $V_{SRC}$ Monitoring Typical Threshold Selection**

| Threshold(2:0) | VDS (V) (LS and HS) | VSRC (V) (only HS) |
|:---:|:---:|:---:|
| 000 | 0.00 | 0.0 |
| 001 | 0.5 | 0.5 |
| 010 | 1 | 1 |
| 011 | 1.5 | 1.5 |
| 100 | 2.0 | 2.0 |
| 101 | 2.45 | 2.5 |
| 110 | 2.95 | 3.0 |
| 111 | 3.45 | 3.5 |

**High-side $V_{DS}$ Threshold Calculation**

$V_{DS}$ Threshold (HS) = Overcurrent x $R_{DS(ON)}$ = 20 A x 0.040 Ω = 0.8 V → **1.0 V** threshold selected. In this case, overcurrent = 1.0 V/0.04 Ω = 25 A.

**Table 23. Vds_threshold_hs Register (0x18A)**

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | Reserved | Vds thr Hs5 | | | Vds thr Hs4 | | | Vds thr Hs3 | | | Vds thr Hs2 | | | Vds thr Hs1 | | |
| Value | 0 | xxx | | | xxx | | | xxx | | | 010 | | | 010 | | |

**Low-side $V_{DS}$ Threshold Calculation**

Low-side the $V_{DS}$ monitoring is done between D_LSx and GND, sense resistance must be included in the calculation.

$V_{DS}$ threshold (LS) = Overcurrent x ($R_{DS(ON)}$ + $R_{SENSE}$) = 20 A x (0.040 Ω + 0.015 Ω) = 1.1 V → **1.0 V** threshold selected, in this case overcurrent = 1.0 V / 0.055 Ω = 18 A.

**Table 24. Vds_threshold_ls_1 Register (0x18C)**

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | Reserved | Vds thr Ls5 | | | Vds thr Ls4 | | | Vds thr Ls3 | | | Vds thr Ls2 | | | Vds thr Ls1 | | |
| Value | 0 | xxx | | | xxx | | | xxx | | | 010 | | | 010 | | |

**Software**

### High-side SRC threshold

$V_{SRC}$ is used mostly during idle phases to understand the type of fault present. It is better to keep the "detection threshold" far from the polarization condition. During actuation in this application, recirculation is done through a diode, keeping the voltage of the HS source below ground. In this case, any $V_{SRC}$ voltage is ok, to prevent false diagnostics. In order to avoid detecting noise and to be far from the 3.8 V threshold, the MC33816 $V_{SRC}$ threshold is set to 1.0 V

### Table 25. Vsrc_threshold_hs Register (0x18B)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | Reserved | Vsrc thr Hs5 | | | Vsrc thr Hs4 | | | Vsrc thr Hs3 | | | Vsrc thr Hs2 | | | Vsrc thr Hs1 | | |
| Value | 0 | xxx | | | xxx | | | xxx | | | 001 | | | 001 | | |

## 6.2.3 Channel 1 Configuration Registers

Unless specified, use the same settings specified in the AN4849.

### Table 26. Ctrl_reg_uc0 Control Registers for the Microcore 0 (0x101, 0x121)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | control_register_shared | | | | | | | | control_register | | | | | | | |
| Value | xxxxxxxx | | | | | | | | 00000000 | | | | | | | |

control_register: Control bits 4,5,6,and 7 are used to control the turn ON of the bank after a fault occurs
- B4: if START pin is still high after $t_{HOLD\_TOT}$ is reached (see Application Source Code)
- B5: if $I_{BOOST}$ is not reached before the specified time
- B6: if errors are detected during actuation (automatic diagnostics)
- B7: if errors are detected during pre-actuation phase (idle diagnostics)

Entry point for each microcode as specified, corresponds to the location in the CRAM where each uC starts.

### Table 27. Uc0_entry_point Registers (0x10A, 0x12A)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | | | | | entry_point_address | | | | | | | | | |
| Value | 000000 | | | | | | 100110000 | | | | | | | | | |

With the code provided, uc0 channel 1 starts line152 label "init0", interrupt code should not be taken in account in the entry code.

### Table 28. Uc1_entry_point Registers (0x10B, 0x12B)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | | | | | entry_point_address | | | | | | | | | |
| Value | 000000 | | | | | | 000001000 | | | | | | | | | |

With code provided, uc1 channel starts line 091 label "init1".

**Software**

It is required to specify the location in the CRAM, because the automatic interrupt is handled here.

### Table 29. Diag_routine_addr Registers (0x10C, 0x12C)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | | | diagnosis_routine_address_uc1 | | | | | | diagnosis_routine_address_uc0 | | | | | |
| Value | 0000 | | | | Xxxxxx | | | | | | 000000 | | | | | |

- diagnosis_routine_address_uc0: Automatic diagnostics are located at line 0 (label irq_auto)
- diagnosis_routine_address_uc1: Not used in this example

The same settings on software interrupt are needed to specify the location in the CRAM where SW interrupts are handled.

### Table 30. Sw_interrupt_routine_addr Registers (0x10E, 0x12E)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | sw_irq _falling _edge _start_ uc1 | sw_irq _rising _edge _start_ uc1 | sw_irq _falling _edge start_ uc0 | sw_irq _rising _edge _start_ uc0 | software_interrupt_routine_address_uc1 | | | | | | software_interrupt_routine_address_uc0 | | | | | |
| Value | x | x | x | x | xxxxxx | | | | | | 000111 | | | | | |

- software_interrupt_routine_address_uc0: Line 7 in the CRAM
- software_interrupt_routine_address_uc1: Not used in this example
- sw_irq_rising_edge_start_uc0: Not used in this example
- sw_irq_falling_edge start_uc0: Not used in this example.

## 6.3 Diagnostics Configuration Registers

### 6.3.1 LS1 & LS2 Output Register

#### 6.3.1.1 Filter Time

These registers define the automatic diagnostics filtering. Values depend on noise in the application and MOSFET switching time to get stable for reliable feedback when diagnostics start.

**Table 31. Lsx_diag_config1 Registers (0x140, 0x143, 0x146, 0x149, 0x14C, 0x14F)**

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | filter_type | filter_length | | | | | | disable_window | | | | | | |
| Reset | 00 | | 0 | 111011 | | | | | | 1011010 | | | | | | |

- filter_type: Set to 0, in this case, means any different sample resets the filter counter
- filter_length: The filtering time is: $t_{FTN} = t_{CK}$ x (5 + 1) = 1/6 MHz x 6 = 1 $\mu$s
- disable_window: this 7-bit parameter configures a time period during which any check on the LSx_Vds_feed signal is disabled after any change on the output_command signal. $t_{DTL} = t_{CK}$ x (14 + 4) = 1/6 MHz x 18 = 3.0 $\mu$s
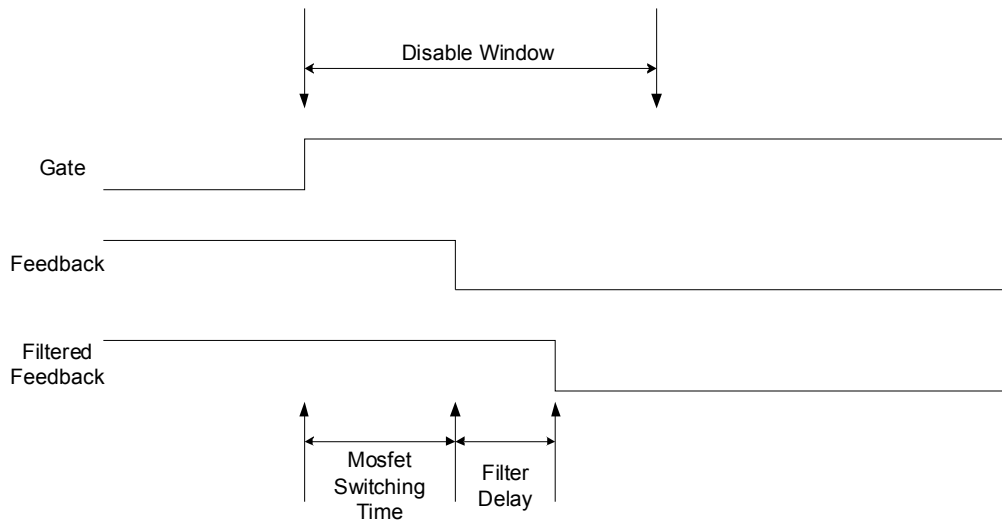


**Figure 22. FIlter Time and Disable Windows Diagram**

#### 6.3.1.2 Error Table

Using the Diagnostic Descriptions section, error tables can be easily generated,

**Table 32. Lsx_diag_config2 Registers (0x141, 0x144, 0x147, 0x14A, 0x14D, 0x150)**

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | | | | | | | | | | | error_table | | | |
| Reset | 000000000000 | | | | | | | | | | | | 1001 | | | |

- error_table: this 4-bit parameter defines the logical value of an error signal, issued from the output and the related $V_{DS}$ feedback signal. This table defines the output of the coherency check between the driven output and the acquired feedback; a logic one value means there is no coherency in the check, and then an error signal towards the microcore should be generated

## Table 33. Error Table for Both Low-sides

| | output_command = 0 (Pre-driver switched OFF) | output_command = 1 (Pre-driver switched ON) |
|---|---|---|
| lsx_vds_fbk = 0 ($V_{DS}$ below threshold) | error_table (0) = 1 | error_table (2) = 0 (OK) |
| lsx_vds_fbk = 1 ($V_{DS}$ above threshold) | error_table (1) = 0 (OK) | error_table (3) = 1 |

Normal mode in this application:
- Low-side is ON and the $V_{DS}$ comparator should be low
- Low-side is OFF and the $V_{DS}$ comparator should be high

### 6.3.2 HS1 / HS2 Output Register

#### 6.3.2.1 Filter Time
Use the same filtering as the low-side, since the same MOSFET and slew rates are used for both.

## Table 34. Hsx_diag_config_1 Registers (0x153, 0x156, 0x159, 0x15C, 0x15F)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | Filter_type | filter_length | | | | | | disable_window | | | | | | |
| Reset | 00 | | 0 | 111011 | | | | | | 1011010 | | | | | | |

- filter_type: Set to 0, in this case, means any different sample resets the filter counter
- filter_length: The filtering time is: $t_{FTN} = t_{CK}$ x (5 + 1) = 1/6 MHz x 6 = 1 μs
- disable_window: this 7-bit parameter configures a time period during which any check on the LSx_Vds_feed signal is disabled after any change on the output_command signal. $t_{DTL} = t_{CK}$ x (14 + 4) = 1/6 MHz x 18 = 3.0 μs

#### 6.3.2.2 Error Table

**HS1 (VBAT) error table**

## Table 35. Hsx_diag_config_2 Registers (0x154, 0x157, 0x115A, 0x15D, 0x160)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | | | | | | | error_table_src | | | | error_table_vds | | | |
| Reset | 00000000 | | | | | | | | 0110 | | | | 1001 | | | |

## Table 36. Error Table for High-side $V_{DS}$

| | output_command = 0 (Pre-driver switched OFF) | output_command = 1 (Pre-driver switched ON) |
|---|---|---|
| hsx_vds_fbk = 0 ($V_{DS}$ below threshold) | error_table_vds (0) = 1 | error_table_vds (2) = 0 |
| hsx_vds_fbk= 1 ($V_{DS}$ above threshold) | error_table_vds (1) = 0 | error_table_vds (3) = 1 |

Normal mode in this application:
- High-side is ON and the $V_{DS}$ comparator should be low
- High-side is OFF and the $V_{DS}$ comparator should be high

### Table 37. Error Table for High-side V$_{SRC}$

| | output_command = 0 (Pre-driver switched OFF) | output_command = 1 (Pre-driver switched ON) |
|---|---|---|
| hsx_src_fbk = 0 (V$_{SRC}$ below threshold) | error_table_src (0) = 0 | error_table_src (2) = 1 |
| hsx_src_fbk = 1 (V$_{SRC}$ above threshold) | error_table_src (1) = 1 | error_table_src (3) = 0 |

Normal mode in this application:
- High-side is ON and the V$_{SRC}$ comparator should be high
- High-side is OFF and the V$_{SRC}$ comparator should be low

**HS2 (VBOOST) error table**

### Table 38. Hsx_diag_config_2 Registers (0x154, 0x157, 0x115A, 0x15D, 0x160)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | | | | | | | error_table_src | | | | error_table_vds | | | |
| Reset | 00000000 | | | | | | | | 0100 | | | | 1001 | | | |

| | output_command = 0 (Pre-driver switched OFF) | output_command = 1 (Pre-driver switched ON) |
|---|---|---|
| hsx_vds_fbk = 0 (V$_{DS}$ below threshold) | error_table_vds (0) = 1 | error_table_vds (2) = 0 |
| hsx_vds_fbk= 1 (V$_{DS}$ above threshold) | error_table_vds (1) = 0 | error_table_vds (3) = 1 |

| | output_command = 0 (Pre-driver switched OFF) | output_command = 1 (Pre-driver switched ON) |
|---|---|---|
| hsx_src_fbk = 0 (V$_{SRC}$ below threshold) | error_table_src (0) = 0 | error_table_src (2) = 1 |
| hsx_src_fbk = 1 (V$_{SRC}$ above threshold) | error_table_src (1) = 0 | error_table_src (3) = 0 |

V$_{BOOST}$ high-side source detection is different in this application from V$_{BAT}$ since S_HSVbat and S_HS_VBoost are shorted together through a diode. If V$_{BAT}$ high-side is ON, voltage on the V$_{BOOST}$ source high-side is equal to V$_{PWR}$ - diode. Consequently, the 33816 should not detect an error on high-side V$_{BOOST}$ if command = 0 and source feedback = 1.

# 7      Application Source Code

The following microcode can be directly downloaded from the NXP web site (see KIT33816AEEVM). Using the IDE and SPIGEN, the microcode can be downloaded to the MC33816.

## 7.1      Injection Banks Management Source Code

```
#include "AN_Diag_ch1.def";
*******************************************************************************
*                                          Copyright (c) NXP 2016 *
* File Name: Pierre_test_4inj.dfi                                            *
* Current Revision: 1.0                                                      *
* Purpose: MC33816 example - 1 Bank diagnostic                              *
* Description:  MC33816 Channel 1 main function provide peak and hold current *
*               profile for Uc0Ch1 and use idle and automatic diagnostics   *
*                    Uc1CH1 provide peak and hold without diagnostics*
*                                                                           *
* REV  AUTHOR    DATE       DESCRIPTION OF CHANGE                           *
* ---  ------    --------    --------------------                           *
* 1.0  b16868   2014/03/25    - initial coding                             *
*                                                                           *
*******************************************************************************


*******************************************************************************
* NXP reserves  the right to make  changes without  further notice to any *
* product herein to improve  reliability, function,  or design.  NXP does *
* not  assume  any  liability  arising  out of  the application  or use  of any *
* product, circuit,  or software described herein;  neither does  it convey any *
* license under its patent rights nor the rights of others.  NXP products *
* are not designed, intended,  or authorized for use as  components  in systems *
* intended for  surgical implant into the body,  or other applications intended *
* to support life,  or for any other  application  in which the failure  of the *
* NXP product could create a situation where  personal injury or death may*
* occur.  Should Buyer purchase or use NXP products for any such intended *
* or unauthorized  application,  Buyer shall indemnify and  hold  NXP and *
* its officers, employees, subsidiaries, affiliates,  and distributors harmless *
* against all claims costs, damages, and expenses, and reasonable attorney  fees*
* arising  out of,  directly or indirectly,  any claim of personal  injury  or *
* death  associated  with  such unintended  or unauthorized use,  even if  such *
* claim  alleges  that  NXP  was  negligent   regarding   the  design  or*
* manufacture  of the part.  NXP and the  NXP logo  are registered  *
* trademarks of NXP B.V.                                              *
*******************************************************************************


* ### Channel 1 - uCore0 controls the injectors 1 ###



* Constant definition
#define HSBoost_B1 hs2;
#define HSBAT_B1 hs1;
#define LS1_B1 ls1;
#define LS2_B1 ls2;


*################## STATUS REGISTER ##############
* This bit must be set to 1 if the Iboost current is never reached during the boost phase
#define BoostErrorBit b5;
* This bit must be set to 1 the sequencer is currently executing the Automatic interrupt routine
#define AutoIrqBit b6;
* This bit must be set to 1 the sequencer is currently executing the Idle Diag interrupt routine
#define IdleIrqBit b7;
* This bit must be set to 1 if start pin stays high longer than 10ms
#define HoldErrorBit b4;


*################## FLAGS ##############
* This flag is sent to the DCDC sequencer. It must be active for the whole period the boost voltage is used
* When the boost voltage is used, the DCDC must be deactivated
* flag = 0 => boost voltage is used, DCDC must be deactivated
* flag = 1 => boost voltage not used, DCDC can be active
```

## Application Source Code

```
#define BstFlag b0;


*################## CONTROL REGISTER ##############
* During the interrupt routine, the sequencer wait for this bit to be set to '1' before resuming execution of application
code
#define AutoDiagResetBit b6;
* During the interrupt routine, the sequencer wait for this bit to be set to '1' before resuming execution of application
code
#define IdleDiagResetBit b7;
* During the interrupt routine, the sequencer wait for this bit to be set to '1' before resuming execution of application
code
#define BoostResetBit b5;
* During the interrupt routine, the sequencer wait for this bit to be set to '1' before resuming execution of application
code
#define HoldResetBit b4;



*################## ALU registers  ###############
#define IRQ_stat_Reg r0;




****************************************************************************************************
*                               AUTOMATIC INTERUPT                                                *
****************************************************************************************************
irq_auto:       stos off off off;                       * Disable drivers
                endiaga diagoff;                        * Disable automatic diagnostic
                stirq low;                              * Set the low IRQB pin
                stf high BstFlag;                       * Set flag0 high DCDC active
                stsrb high AutoIrqBit;                  * Set status register bit 5 when automatic diagnosis interrupt trig

auto_waitEnable:jcrr auto_waitEnable AutoDiagResetBit low;  * the sequencer is stuck here until the bit of the control
register is set to '1' b6
                ldjr1 restore;                          * Load restore to jr1 to do a jump far
                jmpf jr1;                               * Jump to restore



****************************************************************************************************
*                               SOFTWARE INTERUPT                                                 *
****************************************************************************************************
irq1_sw:        stos off off off;                       * Disable drivers
                endiaga diagoff;                        * Disable automatic diagnostic
                stirq low;                              * Set low the IRQB pin
                stf high BstFlag;                       * Set flag0 high to release the DC-DC converter idle mode

                * Check which Sw interrupt occured BoostErr 0 or Idle Diag Fail 1
                cp irq IRQ_stat_Reg;                    * copy the irq status registers to a temp ALU reg
                                                        * This register contains also the sw irq ID
                ldirh 0Ch rst;                          * load MSB in ir reg: 0x0C00 in immediate register, to use as mask
for irq status
                and IRQ_stat_Reg;                       * extract the sw id from irq status register (bits 11-10)
                jarr Boost_waitEN all0;                 * if the sw id is 0 => Iboost never reached => go to Boost_waitEN
                                                        * Else => error detected in idle diag=> go to next line => seq
stuck until micro write 1 in control register b8


                ldirh 08h rst;                          * load MSB in ir reg: 0x0800 in immediate register, to use as mask
for irq status
                and IRQ_stat_Reg;                       * extract the sw id from irq status register (bits 11-10)
                jarr sw_waitEnable all0;                 * if the sw id is 0 => Means IRQ = b01 => Idle diagnostics fails
                              * Else => Hold error => go to next line => idle diag fail


Hold_waitEN:    stsrb high HoldErrorBit;                * Start pin stays higher longer than 10ms
                jcrr Hold_waitEN HoldResetBit low;      * Wait here until control bit register is write to 1
                jmpr restore;

sw_waitEnable:  stsrb high IdleIrqBit;                  * IDle diag fail we set status b8 high to let user know which error
```

```
occured
                jcrr sw_waitEnable IdleDiagResetBit low;* Wait here until control bit register is write to 1
                jmpr restore;


Boost_waitEN:   stsrb high BoostErrorBit;               * Iboost never reached, let user know by setting status register
bit b5
                jcrr Boost_waitEN BoostResetBit low;    * Wait here until control bit register is write to 1



restore:        stirq high;                             * Set high IRQB pin
                rstreg all;                             * Reset a) control registers
                                                        *       b) status regsiter
                                                        *       c) err_seq register (status of automatic diagnosis
                                                        *       d) re-enables irq generation from automatic diagnosis
                iret restart rst;                       * Clear interrupt queue and restart from init phase



*********************************************************************************************************
*                                         INIT PHASE
*********************************************************************************************************
init0:      stirq high;                         * Set high IRQB pin
            stgn gain8.68 sssc;                    * Set gain amplifier for current feedback 1
            ldjr1 eoinj0;                    * Load end of injector in jr1 to use jump far
            ldjr2 idle0;                             * Load idle0 in jr1 to use jump far
            cwef jr1 _start row1;                    * If any start goes low go to eoi


*********************************************************************************************************
*                                         IDLE PHASE                                                   *
*********************************************************************************************************
idle0:      joslr inje1 start1;                      * Start injector 1 if start1 goes high
            joslr inje2 start2;                      * Start injector 2 if start2 goes high
            jmpf jr1;                            * Jump to end of injection



*********************************************************************************************************
*                                         SHORTCUT DEFINITION                                          *
*********************************************************************************************************
inje1:      dfsct HSBAT_B1 LS1_B1 HSBoost_B1;     * Shortcut1 = HSVBAT, Shortcut2= LS2_B1, Shortcut3= HSBOOST
            dfcsct dac1;                          * use current feedback1
            jmpr idle_diag0;                      * Jump to idle_diag0


inje2:      dfsct HSBAT_B1 LS2_B1 HSBoost_B1;     * Shortcut1 = HSVBAT, Shortcut2= LS2_B1, Shortcut3= HSBOOST
            dfcsct dac1;                          * use current feedback1
            jmpr idle_diag0;                      * Jump to idle_diag0 (useless here)



*********************************************************************************************************
*                                         PRE-ACTUATION DIAG PHASE                                     *
*********************************************************************************************************
idle_diag0:    bias all on;                         * Enable all biasing structures, kept ON even during actuation
               jocr idle_diag_fail0 _sc1v;          * Error detected if Vds of shortcut1 (HS) is low
               jocr idle_diag_fail0 _sc2v;          * Error detected if Vds of shortcut2 (LS) is low
               jocr idle_diag_fail0 _sc3v;          * Error detected if Vds of shortcut3 (Boost) is low
               jocr idle_diag_fail0 _sc1s;          * Error detected if Vsrc of shortcut1 (HS) is low
               jocr idle_diag_fail0 _sc3s;          * Error detected if Vsrc of shortcut3 (Boost) is low
               jmpr boost0;                         * Jump to actuation phase if no failure detected in idle phase


idle_diag_fail0:reqi 1;                             * Go to software subroutine is fault detected in idle phase HSBat
error
```

```
*************************************************************************************************
*                                        BOOST PHASE                                          *
*************************************************************************************************


boost0:     ldcd rst _ofs keep keep injMaxTBoost c3;      * Start Boost Counter in case Iboost never reached
            load Iboost dac_sssc _ofs;                    * Load Boost current threshold

            cwer peak0 cur1 row2;                         * Define Wait Table Iboost is reached and jump to peak phase
            cwer boost_err0 tc3 row5;                     * Define Wait Table if actuation longer than injMaxGuard go to
eoinj (added from AN4849)

            stf low BstFlag;                              * Turn OFF the boost during this phase
            stos on on on;                                * Vbat high-side On, Vboost HS On et LS1/2 ON, need to turn ON
HS1 also to avoid diag failure
            endiags on on on on;                           * Enable auto diag

            wait row125;                                  * Wait start goes low or Iboost reached or InjMaxTBoost reached


boost_err0: reqi 0;                                       * Go to software subroutine is fault detected in Boost phase,
did not reach Iboost on time (added from AN4849)


*************************************************************************************************
*                                        PEAK PHASE                                           *
*************************************************************************************************


peak0:      ldcd rst _ofs keep keep Tpeak_tot c1;         * Start Tpeak tot counter
            stf high BstFlag;                             * Turn Boost back on
            load Ipeak dac_sssc _ofs;                     * Load the peak current threshold in the current DAC
          cwer bypass0 tc1 row2;                     * Define Wait: Jump to bypass phase when tc1 reaches end of count
           cwer peak_on0 tc2 row3;                        * Define Wait: Jump to peak_on when tc2 reaches end of count
            cwer peak_off0 ocur row4;                       * Define Wait: Jump to peak_off when current is over threshold

peak_on0:   stos on on off;                               * Vbat On LS On, if needed Boost HS can stay ON during this phase
            wait row124;

peak_off0:  ldcd rst ofs keep keep Tpeak_off c2;          * Load in the counter 2 the length of the peak_off phase
            stos off on off;                              * turn OFF HSvbat keep LS ON
            wait row123;


*************************************************************************************************
*                                        BYPASS PHASE                                         *
*************************************************************************************************


bypass0:    ldcd rst ofs keep keep Tbypass c3;            * Load in the counter 3 the length of the off_phase phase
            stos off off off;                             * turn OFF all HS LS1/2
            cwer hold0 tc3 row4;                          * Define Wait: Jump to hold when tc3 reaches end of count
            wait row14;
```

**Application Source Code**

```
*******************************************************************************************************
*                                          HOLD PHASE                                                 *
*******************************************************************************************************

hold0:      ldcd rst _ofs keep keep Thold_tot c1;       * load thold tot inside c1
            load Ihold dac_sssc _ofs;                   * load hold current inside DAC
            cwer hold_error0 tc1 row2;                  * Define Wait: Jump to hold error if start still high after
thold tot
            cwer hold_on0 tc2 row3;                     * Define Wait: Jump to hold on after thold off
            cwer hold_off0 cur1 row4;                   * Define Wait: Jump to hold off when current Ihold reached

hold_on0:   stos on on off;                             * HSvbat ON, LS ON
            wait row124;

hold_off0:  ldcd rst _ofs keep keep Thold_off c2;       * load thold off inside c2
            stos off on off;                            * LS ON
            wait row123;



hold_error0: reqi 2;                                    * If Start high is longer than Thold_tot go to sw interrupt

*******************************************************************************************************
*                                     END OF INJECTION PHASE                                          *
*******************************************************************************************************

eoinj0:     stos off off off;
            endiags off off off off;                    * disable auto diag
            stf high BstFlag;                           * turn ON DCDC
            jmpf jr2;                                   * jump to idle


*############################################################################
*############################################################################

*************************************************************************************

* ### Channel 1 - uCore1 controls injectors 3 and 4 without diagnostics ###

* ### Variables declaration  ###

* Note: The data that defines the profiles are shared between the two microcores.

* ### Initialization phase ###
init1:      stgn gain8.68 sssc;         * Set the gain of the opamp of the current measure block 2
            ldjr1 eoinj1;               * Load the eoinj line label Code RAM address into the register jr1
            ldjr2 idle1;                * Load the idle line label Code RAM address into the register jr2
            cwef jr1 _start row1;       * If the start signal goes low, go to eoinj phase


* ### Idle phase- the uPC loops here until start signal is present ###
idle1:      joslr inj3_start start3;    * Perform an actuation on inj3 if start 3 (only) is active
            joslr inj4_start start4;    * Perform an actuation on inj4 if start 4 (only) is active
            jmpf jr1;                   * If more than 1 start active at the same time(or none), no actuation

* ### Shortcuts definition per the injector to be actuated ###
inj3_start:dfsct hs3 hs4 ls3;           * Set the 3 shorcuts : VBAT, VBOOST, LS
            jmpr boost1;                * Jump to launch phase

inj4_start:dfsct hs3 hs4 ls4;           * Set the 3 shorcuts : VBAT, VBOOST, LS
            jmpr boost1;                * Jump to launch phase

* ### Launch phase enable boost ###
boost1:     load Iboost dac_sssc _ofs;  * Load the boost phase current threshold in the current DAC
            cwer peak1 ocur row2;       * Jump to peak phase when current is over threshold
            stf low b0;                 * set flag0 low to force the DC-DC converter in idle mode
            stos off on on;             * Turn VBAT off, BOOST on, LS on
            wait row12;                 * Wait for one of the previously defined conditions
```

**MC33816 Diagnostics,** Rev. 3.0

NXP Semiconductors

```
                                          * ### Peak phase continue on Vbat ###
peak1:      ldcd rst _ofs keep keep Tpeak_tot c1;* Load the length of the total peak phase in counter 1
            load Ipeak dac_sssc _ofs;    * Load the peak current threshold in the current DAC
            cwer bypass1 tc1 row2;       * Jump to bypass phase when tc1 reaches end of count
            cwer peak_on1 tc2 row3;      * Jump to peak_on when tc2 reaches end of count
            cwer peak_off1 ocur row4;    * Jump to peak_off when current is over threshold
            stf high b0;                 * set flag0 high to release the DC-DC converter idle mode

peak_on1:   stos on off on;             * Turn VBAT on, BOOST off, LS on
            wait row124;                 * Wait for one of the previously defined conditions

peak_off1:  ldcd rst ofs keep keep Tpeak_off c2;* Load in the counter 2 the length of the peak_off phase
            stos off off on;             * Turn VBAT off, BOOST off, LS on
            wait row123;                 * Wait for one of the previously defined conditions

* ### Bypass phase ###
bypass1:    ldcd rst ofs keep keep Tbypass c3;* Load in the counter 3 the length of the off_phase phase
            stos off off off;            * Turn VBAT off, BOOST off, LS off
            cwer hold1 tc3 row4;         * Jump to hold when tc3 reaches end of count
            wait row14;                  * Wait for one of the previously defined conditions

* ### Hold phase on Vbat ###
hold1:      ldcd rst _ofs keep keep Thold_tot c1;* Load the length of the total hold phase in counter 2
            load Ihold dac_sssc _ofs;    * Load the hold current threshold in the DAC
            cwer eoinj1 tc1 row2;        * Jump to eoinj phase when tc1 reaches end of count
            cwer hold_on1 tc2 row3;      * Jump to hold_on when tc2 reaches end of count
            cwer hold_off1 ocur row4;    * Jump to hold_off when current is over threshold

hold_on1:   stos on off on;             * Turn VBAT on, BOOST off, LS on
            wait row124;                 * Wait for one of the previously defined conditions

hold_off1:  ldcd rst _ofs keep keep Thold_off c2;* Load the length of the hold_off phase in counter 1
            stos off off on;             * Turn VBAT off, BOOST off, LS on
            wait row123;                 * Wait for one of the previously defined conditions

* ### End of injection phase ###
eoinj1:     stos off off off;            * Turn VBAT off, BOOST off, LS off
            stf high b0;                 * set flag0 to high to release the DC-DC converter idle mode
            jmpf jr2;                    * Jump back to idle phase

* ### End of Channel 1 - uCore1 code ###
```

## 7.2        DC-DC and Fuel Pump Source Code

```
#include "AN_Diag_ch2.def";
*******************************************************************************
*                                       Copyright (c) NXP 2014    *
* File Name: Pierre_test_4inj.dfi                                           *
* Current Revision: 1.0                                                      *
* Purpose: MC33816 example - 4 injectors control with Peak & Hold           *
* Description:  MC33816 Channel 1 main function provide peak and hold current *
*               profile for Uc0Ch1 and Uc1Ch1                               *
* File created by:  NXP Analog, Tempe                                       *
*                                                                            *
* UPDATE HISTORY                                                             *
* REV  AUTHOR   DATE       DESCRIPTION OF CHANGE                             *
* ---  ------   --------   --------------------                             *
* 1.0  b16868   2014/03/25   - initial coding                              *
*                                                                            *
*******************************************************************************


*******************************************************************************
* NXP reserves  the right to make  changes without  further notice to any *
* product herein to improve  reliability, function,  or design.  NXP does *
* not  assume  any  liability  arising  out of  the application  or use  of any *
* product, circuit,  or software described herein;  neither does  it convey any *
* license under its patent rights nor the rights of others.  NXP products *
* are not designed, intended,  or authorized for use as  components  in systems *
* intended for  surgical implant into the body,  or other applications intended *
* to support life,  or for any other  application  in which the failure  of the *
* NXP product could create a situation where  personal injury or death may*
* occur.  Should Buyer purchase or use NXP products for any such intended *
* or unauthorized  application,  Buyer shall indemnify and  hold  NXP and *
* its officers, employees, subsidiaries, affiliates,  and distributors harmless *
* against all claims costs, damages, and expenses, and reasonable attorney  fees*
* arising  out of,  directly or indirectly,  any claim  of personal  injury  or *
* death  associated  with  such unintended  or unauthorized use,  even if  such *
* claim  alleges  that  NXP   was  negligent   regarding   the  design  or*
* manufacture  of the part.  NXP and the  NXP logo  are registered  *
* trademarks of NXP B.V. *
*******************************************************************************


* ### Channel 2 - uCore0 controls the DCDC ###



*#################### INIT ##############
init0:      stgn gain5.8 ossc;
            load Iboost_L dac_ossc _ofs; *load DAC 4 L with low Vboost current
            load Iboost_H dac4h4n _ofs;  *load DAC 4H with high Vbost current
            stdm null; *Set DAC access mode to Vboost
            cwer idle0 _f0 row1;  *if flag 0 is low turn Off the boost
            cwer dcdcon _vb row2;   *if vboost lower than vboost low then turn on boost
            cwer dcdcoff vb row3;   *if vboost higher than vboost high then turn off boost


dcdcon:     load Vboost_H dac4h4n _ofs;  *set Vboost high
            stdcctl async; *set dcdc to async
            wait row13;

dcdcoff:    load Vboost_L dac4h4n _ofs;  *set Vboost low
            stdcctl sync; *set dcdc to sync
            wait row12;

idle0:      stdcctl sync;
            jocr idle0 _f0;
            jmpr dcdcoff;
```

## Application Source Code

```
 * ### End of Channel 2 - uCore0 code
###********************************************************************************

* ### Channel 2 - uCore1 drives fuel pump ###

 * ### Variables declaration  ###
 * Note: The data are stored into the dataRAM of the channel 1.
 *#define Ipeak 5;        ** The peak current value is stored in the Data RAM address 5
 *#define Ihold 6;        * * The hold current value is stored in the Data RAM address 6
 *#define Thold_off 7;  *  * The hold off time is stored in the Data RAM address 7
 *#define Thold_tot 8; ** The hold phase duration is stored in the Data RAM address 8


 * Note: The Tpeak_tot variable defines the current profile time out. The active STARTx pin is expected to toggle in is
low state before this time out.


 * ### Initialization phase ###

 init1: stgn gain19.4 ossc;                  * Set the gain of the opamp of the current measure block 1
        ldjr1 eoact1;                        * Load the eoinj line label Code RAM address into the register jr1
        ldjr2 idle1;                         * Load the idle line label Code RAM address into the register jr2
        cwef jr1 _start row1;                * If the start signal goes low, go to eoinj phase

 * ### Idle phase- the uPC loops here until start signal is present ###

 idle1: joslr act5_start start5;        * Perform an actuation on act5 if start 5 (only) is active
        joslr act6_start start6;             * Perform an actuation on act6 if start 6 (only) is active
        jmpf jr1;                            * If more than 1 start active at the same time(or none), no actuation


 * ### Shortcuts definition per the injector to be actuated ###
act5_start: dfsct hs5 ls5 undef;                 * Set the 2 shortcuts: VBAT, LS
        jmpr peak1;                          * Jump to launch phase
act6_start: dfsct hs5 ls6 undef;                 * Set the 2 shortcuts: VBAT, LS
        jmpr peak1;                          * Jump to launch phase

 * ### Launch peak phase on bat ###
 peak1: load Ipeak dac_ossc _ofs;            * Load the boost phase current threshold in the current DAC
        cwer hold1 cur3 row2;                * Jump to peak phase when current is over threshold
         stos on on keep;                     * Turn VBAT off, BOOST on, LS on
        wait row12;                          * Wait for one of the previously defined conditions

 * ### Hold phase on Vbat ###
 hold1: ldcd rst _ofs keep keep Thold_tot c1;  * Load the length of the total hold phase in counter 2
        load Ihold dac_ossc _ofs;            * Load the hold current threshold in the DAC
        cwer eoact1 tc1 row2;                * Jump to eoinj phase when tc1 reaches end of count
        cwer hold_on1 tc2 row3;              * Jump to hold_on when tc2 reaches end of count
        cwer hold_off1 cur3 row4;            * Jump to hold_off when current is over threshold

hold_on1:stos on on keep;                     * Turn VBAT on, LS on
        wait row124;                         * Wait for one of the previously defined conditions

hold_off1:ldcd rst _ofs off on Thold_off c2;     * Load the length of the hold_off phase in counter 1 and turn VBAT off,
LS on
        wait row123;                         * Wait for one of the previously defined conditions


 * ### End of injection phase ###
 eoact1:stos off off keep;                   * Turn VBAT off, LS off
        jmpf jr2;                            * Jump back to idle phase

 * ### End of Channel 2 - uCore1 code ###
```

# 8 References

Following are URLs where you can obtain information on NXP products and application solutions:

| Document Number and Description | | URL |
|---|---|---|
| MC33816 | Data Sheet | www.nxp.com/files/analog/doc/data_sheet/MC33816.pdf |
| KIT33816FRDMUG | Kit User Guide | www.nxp.com/files/analog/doc/user_guide/KT33816UG.pdf |
| KITUSBSPIGDGLEVME | Evaluation Kit | www.nxp.com/products/power-management/linear-regulator/evaluation-kit-usb-spi-interface-mc68hc908jw32-dongle:KITUSBSPIDGLEVME |
| AN4849 | Application Note | www.nxp.com/files/analog/doc/app_note/AN4849.pdf |
| AN4954SW | SPI config file | www.nxp.com/files/analog/doc/app_note/AN4954SW.zip |
| **nxp.com Support Pages** | | **URL** |
| nxp.com | | www.nxp.com |
| MC33816 Product Summary Page | | www.nxp.com/products/power-management/engine-and-dc-motor-control/powertrain-engine-control/sd6-programmable-driver-for-solenoid-control:MC33816 |
| Analog Home Page | | www.nxp.com/analog |

# 9 Revision History

| Revision | Date | Description |
|---|---|---|
| 1.0 | 6/2014 | • Initial release |
| 2.0 | 11/2014 | • Replaced KIT33816AEEVM by KIT33816FRDMEVM<br>• Updated References Table |
| 3.0 | 2/2016 | • Updated Filter Time values in Section 6.3.1.1, Filter Time, page 26<br>• Updated Filter Time values in Section 6.3.2.1, Filter Time, page 27 |

√RoHS

Document Number:  AN4954
Rev. 3.0
2/2016