Document information

<table>
<thead>
<tr>
<th>Information</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keywords</td>
<td>MC33816, PT2001, injector, fuel pump, automotive</td>
</tr>
<tr>
<td>Abstract</td>
<td>Example of a four cylinder internal combustion engine (ICE) injector drive.</td>
</tr>
</tbody>
</table>
1 Introduction

This typical application covers an example of a four cylinder internal combustion engine (ICE) injector drive. A boost converter managed by the MC33816/PT2001 provides the high voltage.

This application covers both MC33816 and PT2001 devices. Reuse similar code and registers for both devices. HW and SW are 100 % compatible.

2 Overview

This overview presents a typical hardware topology and related software example to drive four injectors managed in two banks, a single low-pressure pump drive, and a variable frequency modulation (VFM) DC-to-DC converter.

A battery voltage between 9.0 V and 16 V supplies the MC33816. A battery voltage between 5 V and 24 V supplies the PT2001. Consider a protection circuitry against overvoltage and reverse battery on the $V_{BAT}$ supply line.

To supply internally the I/O buffers, supply an external 5.0 V to the $V_{CC5}$ pin and the $V_{CCIO}$ pin.

In this example, generate the $V_{CCP}$ voltage internally to enable the drivers.

To manage a VFM, define the boost converter topology. A pi filter prevents circuitry disturbance propagation from the boost regulation area to battery line.

The two banks can manage two injectors each. In this configuration, injection overlaps are not possible inside a bank. The diodes D12, D13, D14, and D15 are required to allow current recirculation when the respective low-side metal-oxide-semiconductor field-effect transistors (MOSFETs) are switched off. The diodes D10_2 and D11_2 provide a current recirculation path to ground when the respective high-side MOSFET is off.
3 Application schematic

Figure 1. Typical four injector two bank application schematic
4 Application instructions

This topology can be used on the evaluation board KIT33816FRDMEVM or KITPT2001FRDMEVM. Register settings and microcode downloads can be achieved by using the KL25Z embedded on the KIT33816FRDMEVM or KITPT2001FRDMEVM.

This topology allows managing two banks of two cylinders. Injector actuations are limited to one injection per bank at the same time. 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.
- The bank #2 is managed by the digital microcore Uc1Ch1.

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

- The VFM is managed by the digital microcore Uc0Ch2.
- The fuel pump is managed by the digital microcore Uc1Ch2.

The following is the start-up sequence:

1. Apply a battery voltage between 9.0 V and 16 V.
2. Download the registers Channel Configuration, then Main Configuration, IO Configuration, and Diagnostic Configuration.
3. Download the dedicated microcode in the logic Channel 1 and logic Channel 2 data RAMs.
4. Set to logic 1 to the pre-flash enable bit and the en_dual_seq bit in the Flash_enable register of channel 1 (0x100) and channel 2 (0x120).

The following sections detail the registers configuration and the microcodes.

Once the DC-to-DC converter output has reached its nominal voltage, the STARTx pins can actuate the injector drivers. Each STARTx pin individually triggers each injector pin rising edge and stops actuation on the falling edge.

- START1 drives INJECTOR 1
- START2 drives INJECTOR 2
- START3 drives INJECTOR 3
- START4 drives INJECTOR 4
- START5 drives FUEL PUMP 1

Boost regulation is stopped during the injection boost phase.

5 Software requirements

5.1 Current profile management for injection

The current profile is managed so to generate an initial high current through the injector. This high current slew rate minimizes the opening delay. This high current is maintained during a given time to ensure injector opening. Then the current is decreased to maintain the injector open, up to the end of injection (EOI).
The code dedicated to the injection is loaded into the code RAM 1. This code is executed independently by the microcores Uc0Ch1 and Uc1Ch1. Each one of the microcores generates a current profile, as described through the injector per the STARTx pin state by the following figure.

![Typical peak and hold current profile](image)

**Figure 2. Typical peak and hold current profile**

When a rising edge is detected on a STARTx pin, the injection starts with the injection boost phase.

This profile can be stopped at any time by detecting a falling edge on the STARTx pin. In this case, the EOI phase is executed.

During the boost phase, the corresponding low-side driver is simultaneously switched on with the high-side switch connected to the $V_{boost}$ voltage. If the boost current target $I_{boost}$ is reached, the high-side driver is switched off and the current recirculates for a fixed time ($t_{peak\_off}$) through the diode connected to ground. Then the peak phase starts.

During the peak phase, the high-side switch connected to $V_{BAT}$ voltage is turned on. If the peak current target $I_{peak}$ is met, the high-side driver is switched off and the current recirculates through the diode connected to ground for the fixed time ($t_{peak\_off}$). The high-side driver is then switched on again. This cycle repeats until that the internal counter reaches its terminal value ($t_{peak\_tot}$), then the bypass phase begins.

During the bypass phase, all the low-side and high-side switches are turned off. The current decays through the injector, the diode connected to ground, and the diode connected to $V_{boost}$ for a fixed time ($t_{bypass}$). The hold phase then starts.

During the hold phase, the low-side driver is simultaneously switched on with the high-side switch connected to the $V_{BAT}$ voltage. If the hold current target $I_{hold}$ is reached, the high-side driver is switched off and the current recirculates through the diode connected to ground for a fixed time ($t_{hold\_off}$). The high-side driver is switched on again, and the cycle repeats until the STARTx pin goes LOW or the internal counter reaches its terminal value ($t_{hold\_tot}$ (timeout)). The EOI is forced if no falling edge is detected on the STARTx pin.
All the current thresholds and timings are accessed in the data RAM. The typical values are in Table 1, but must be defined according to the injector used and the injection profile expected.

### Table 1. Example of injection current profile key parameters ($R_{\text{SENSE}} = 10 \, \text{mΩ}$)

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_{\text{boost}}$</td>
<td>current threshold in boost phase</td>
<td>16.72 A</td>
</tr>
<tr>
<td>$I_{\text{peak}}$</td>
<td>current threshold in peak phase</td>
<td>16.72 A</td>
</tr>
<tr>
<td>$I_{\text{hold}}$</td>
<td>current threshold in hold phase</td>
<td>8.92 A</td>
</tr>
<tr>
<td>$t_{\text{peak_off}}$</td>
<td>fixed time for high-side switch off in peak phase</td>
<td>10 μs</td>
</tr>
<tr>
<td>$t_{\text{peak_tot}}$</td>
<td>fixed time for end of peak phase</td>
<td>200 μs</td>
</tr>
<tr>
<td>$t_{\text{bypass}}$</td>
<td>fixed time for bypass phase</td>
<td>30 μs</td>
</tr>
<tr>
<td>$t_{\text{hold_off}}$</td>
<td>fixed time for high-side switch off in hold phase</td>
<td>10 μs</td>
</tr>
<tr>
<td>$t_{\text{hold_tot}}$</td>
<td>fixed time for end of hold phase (timeout)</td>
<td>10 ms</td>
</tr>
</tbody>
</table>

In the present case, most of the code branches (jump) are managed according to the counters end of count and the current threshold, by the mean of the wait table. The wait table rows are affected, as shown in Table 2 and are changed according to the injection phase.

### Table 2. Example of wait table definition

<table>
<thead>
<tr>
<th>Phase</th>
<th>Boost phase</th>
<th>Peak phase</th>
<th>Bypass phase</th>
<th>Hold phase</th>
<th>EOI phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row 1</td>
<td>if STARTx goes LOW then jump to EOI phase</td>
<td>if STARTx goes LOW then jump to EOI phase</td>
<td>if STARTx goes LOW, then jump to EOI phase</td>
<td>if STARTx goes LOW, then jump to EOI phase</td>
<td>-</td>
</tr>
<tr>
<td>Row 2</td>
<td>if the injection current reaches $I_{\text{boost}}$ then jump to peak phase</td>
<td>if $t_{\text{peak_tot}}$ is reached, then jump to bypass phase</td>
<td>if $t_{\text{bypass}}$ is reached, then jump to hold phase</td>
<td>if $t_{\text{hold_tot}}$ is reached, then jump to EOI phase</td>
<td>-</td>
</tr>
<tr>
<td>Row 3</td>
<td>-</td>
<td>if $t_{\text{peak_off}}$ is reached, jump to peak on phase (sub phase)</td>
<td>-</td>
<td>if $t_{\text{hold_off}}$ is reached, jump to hold on phase (sub phase)</td>
<td>-</td>
</tr>
<tr>
<td>Row 4</td>
<td>-</td>
<td>if $t_{\text{peak}}$ is reached, jump to hold off phase (sub phase)</td>
<td>-</td>
<td>if $t_{\text{hold}}$ is reached, jump to hold off phase (sub phase)</td>
<td>-</td>
</tr>
<tr>
<td>Row 5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

A rising edge issues an injection start. A falling edge triggers the end of injection. In case of an overlap between two STARTx pins on the same bank, it is managed by the smart start function of the device. In this case, the first STARTx rising edge is considered. The second STARTx pin high-state is considered when the first actuation is finished. The action of the injection corresponding to the second STARTx pin is stopped when the second STARTx pin falling edge occurs.
5.1.1 General registers setup

The MC33816/PT2001 registers are set up according to their default states, unless defined by the following.

5.1.1.1 Main configuration register

To run two microcores per channel, the Ck_per register (0x1C0) must be set up with a ck_per value of 3 or more.

Table 3. Ck_per register (0x1C0)

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>reserved</td>
<td>ck_per</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>-</td>
<td>000011</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.1.2 Injection banks management registers setup

The MC33816/PT2001 registers are set up according to their default states, unless defined by the following.

5.1.2.1 IO configuration registers

The microcore Uc0Ch1 must have access to the pre-drivers HS1, HS2, LS1, and LS2.
Table 4. Out_acc_uc0_ch1 register (0x184)

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>reserved</td>
<td>acc_uc0_ch1_ls7</td>
<td>acc_uc0_ch1_ls6</td>
<td>acc_uc0_ch1_ch1</td>
<td>acc_uc0_ch1_ch1</td>
<td>acc_uc0_ch1_ch1</td>
<td>acc_uc0_ch1_ch1</td>
<td>acc_uc0_ch1_ch1</td>
<td>acc_uc0_ch1_ch1</td>
<td>acc_uc0_ch1_ch1</td>
<td>acc_uc0_ch1_ch1</td>
<td>acc_uc0_ch1_ch1</td>
<td>acc_uc0_ch1_ch1</td>
<td>acc_uc0_ch1_ch1</td>
<td>acc_uc0_ch1_ch1</td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>0000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

In the same way, the Uc1Ch1 must have access to the pre-drivers HS3, HS4, LS3, and LS4.

Table 5. Out_acc_uc1_ch1 register (0x185)

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>reserved</td>
<td>acc_uc1_ch1_ls7</td>
<td>acc_uc1_ch1_ls6</td>
<td>acc_uc1_ch1_ch1</td>
<td>acc_uc1_ch1_ch1</td>
<td>acc_uc1_ch1_ch1</td>
<td>acc_uc1_ch1_ch1</td>
<td>acc_uc1_ch1_ch1</td>
<td>acc_uc1_ch1_ch1</td>
<td>acc_uc1_ch1_ch1</td>
<td>acc_uc1_ch1_ch1</td>
<td>acc_uc1_ch1_ch1</td>
<td>acc_uc1_ch1_ch1</td>
<td>acc_uc1_ch1_ch1</td>
<td>acc_uc1_ch1_ch1</td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>0000</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

The microcore Uc0Ch1 has default access to the current sense block # 1, and to microcore Uc1Ch1 to the current sense block # 2. The corresponding registers content does not need to be changed.

Table 6. Cur_block_access_1 register (0x188)

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>reserved</td>
<td>acc_uc1_ch1_curr_4h_4neg</td>
<td>acc_uc1_ch1_curr3</td>
<td>acc_uc1_ch1_curr2</td>
<td>acc_uc0_ch1_curr4l</td>
<td>acc_uc0_ch1_curr4l</td>
<td>acc_uc0_ch1_curr4l</td>
<td>acc_uc0_ch1_curr4l</td>
<td>acc_uc0_ch1_curr4l</td>
<td>acc_uc0_ch1_curr4l</td>
<td>acc_uc0_ch1_curr4l</td>
<td>acc_uc0_ch1_curr4l</td>
<td>acc_uc0_ch1_curr4l</td>
<td>acc_uc0_ch1_curr4l</td>
<td>acc_uc0_ch1_curr4l</td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>000000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

5.1.2.2 Channel 1 configuration registers

The banks 1 and 2 are driven according to the logic level of their respective STARTx pin. A HIGH level on STARTx triggers the activation of the corresponding injector. A LOW level on the STARTx pin automatically stops the actuation, whatever the injection phase.

The microcore Uc0Ch1 must be enabled by the START1 and START2 pins, while the microcore Uc1Ch1 must be enabled by the START3 and START4 pins.

Consequently, the Start_config_reg register of the channel 1 (0x104) must be set up, as shown in Table 7.
Table 7. Start_config_reg registers (0x124)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>reserved</td>
<td>00</td>
</tr>
<tr>
<td>13</td>
<td>smart_start_uc1</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>smart_start_uc0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>start6_sens_uc1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>start6_sens_uc0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>start5_sens_uc1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>start5_sens_uc0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>start4_sens_uc1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>start4_sens_uc0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>start3_sens_uc1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>start3_sens_uc0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>start2_sens_uc1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>start2_sens_uc0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>start1_sens_uc1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>start1_sens_uc0</td>
<td>1</td>
</tr>
</tbody>
</table>

The Code_width register and the two checksum value registers must be set to verify permanently the code integrity. The checksum is recalculated in the MC33816/PT2100 at runtime each time a microcode line is executed and compared to the checksum register value. If there is a mismatch, an error is reported.

Table 8. Code_width registers (0x107)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>reserved</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>code_width</td>
<td>0001011000</td>
</tr>
</tbody>
</table>

Table 9. Checksum_h registers (0x108)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>checksum_high</td>
<td>1000111001100010</td>
</tr>
</tbody>
</table>

Table 10. Checksum_l registers (0x109)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>checksum_low</td>
<td>0111101100100011</td>
</tr>
</tbody>
</table>

The code entry point of Uc0Ch1 is 0 as the first line executed, is the first code RAM line of the channel 1.

Table 11. Uc0_entry_point registers (0x10A) of channel 1

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>reserved</td>
<td>000000</td>
</tr>
<tr>
<td>13</td>
<td>entry_point_address</td>
<td>0000000000</td>
</tr>
</tbody>
</table>

The code entry point of Uc1Ch1 is the 44th code RAM line of the channel 1.

Table 12. Uc1_entry_point registers (0x10B) of channel 1

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>reserved</td>
<td>000000</td>
</tr>
<tr>
<td>13</td>
<td>entry_point_address</td>
<td>0000101100</td>
</tr>
</tbody>
</table>

The microcore Uc0Ch1 must be set up to run the two microcores of the channel as each microcore drives an injection bank.
Both microcores are enabled by setting the pre_flash enable bit and the en_dual_uc bit to logic 1.

Table 13. Flash_enable register (0x100) of channel 1

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>reserved</td>
<td>checksum_disable</td>
<td>checksum_disable</td>
<td>checksum_disable</td>
<td>checksum_disable</td>
<td>checksum_disable</td>
<td>checksum_disable</td>
<td>checksum_disable</td>
<td>checksum_disable</td>
<td>checksum_disable</td>
<td>checksum_disable</td>
<td>checksum_disable</td>
<td>checksum_disable</td>
<td>checksum_disable</td>
<td>checksum_disable</td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>0000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
<td></td>
</tr>
</tbody>
</table>

5.1.2.3 Diagnosis configuration registers

The high-side and low-side drivers must be directly controlled by the microcores. Consequently the output_routing fields of the high-side and low-side drivers output configuration register must be set to the value 15.

Table 14. Hsx_output_config registers (0x155, 0x158, 0x15B, 0x15E)

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
</tr>
<tr>
<td>Reset</td>
<td>0000000000</td>
<td>0000</td>
<td>1111</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 15. Lsx_output_config registers (0x142, 0x145, 0x148, 0x14B)

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
<td>reserved</td>
</tr>
<tr>
<td>Value</td>
<td>00000000000</td>
<td>1111</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.1.3 Injection banks management algorithm

**Initialization Phase**
- Current sense operational amplifier gain setting
- Load the existing label code RAM address into the register j1
- Load the idle line label Code RAM address into the register j2
- Define wait table entry #1: Jump to End Of Injection Phase if the start signal goes low

**Start 1 signal is high?**
- Yes

**Start 2 signal is high?**
- No

**Shutdown definition**
- Driver shutdown affection
  - SH1 (VBAT)
  - SH2 (VREG2)
  - LS1
  - Jump to Boost Phase

**Boost Phase**
- Load the boost phase current threshold in the current DAC
- Define wait table entry #2: Jump to Peak Phase when current is over threshold
- Set flag to low (set to the VFM in idle Phase while the Injection Boost Phase is ongoing)
- Vboost MOSFET (HS2) and LS MOSFET (LSx) turn on, Vbat MOSFET (HS1) turn off
- Wait for wait table entry 1 or 2 to be satisfied

**Peak Phase**
- Load the total length of the peak phase in counter 1
- Load the peak current threshold in the current DAC
- Define wait table entry #2: Jump to Bypass Phase when t1 reaches end of counter
- Define wait table entry #3: Jump to Peak On Phase when t2 reaches end of counter
- Define wait table entry #4: Jump to Peak Off Phase when current is over threshold
- Set flag to high (VFM can go out of idle Phase)

**Peak On Phase**
- Vbat MOSFET (HS1) and LS MOSFET (LSx) turn on, Vboost MOSFET (HS2) turn off
- Wait for wait table entry 1, 2 or 4 to be satisfied

**Peak Off Phase**
- Load in the counter 2 the length of the peak off phase
- LS MOSFET (LSx) turn on, Vboost MOSFET and Vbat MOSFET (HS2) turn off
- Wait for wait table entry 1, 2 or 3 to be satisfied

**Bypass Phase**
- Load in the counter 3 the length of the bypass phase
- Vboost MOSFET, Vbat MOSFET (HS2) and LS MOSFET (LSx) turn off
- Define wait table entry #4: Jump to hold when t3 reaches end of counter
- Wait for wait table entry 4 or 5 to be satisfied

**Hold Phase**
- Load the total length of the hold phase in counter 2
- Load the hold current threshold in the DAC
- Define wait table entry #3: Jump to End Of Injection Phase when t1 reaches end of counter
- Define wait table entry #3: Jump to Hold On Phase when t1 reaches end of counter
- Define wait table entry #4: Jump to Hold Off Phase when current is over threshold

**Hold On Phase**
- Vbat MOSFET (HS2) and LS MOSFET (LSx) turn on, Vboost MOSFET turn off
- Wait for wait table entry 1, 2 or 3 to be satisfied

**Hold Off Phase**
- Load in the counter 1 the length of the hold off phase
- LS MOSFET (LSx) turn on, Vboost MOSFET and Vbat MOSFET (HS2) turn off
- Wait for wait table entry 1, 2 or 4 to be satisfied

**End Of Injection**
- Vboost MOSFET, Vbat MOSFET (HS2) and LS MOSFET (LSx) turn off
- Jump back to idle Phase

Figure 4. Algorithm example for a bank of two injectors

Refer to **Injection banks management source code**.
5.2 DC-to-DC management

In variable frequency mode, on/off switching is triggered by the sense current rising above an upper current threshold and falling below a lower current threshold. This mode uses a hysteretic current control loop within a hysteretic voltage control loop. Once the current thresholds are programmed, hardware controls the current regulation loop, while software (microcode) controls the voltage regulation loop. Duty cycle and frequency vary with operating conditions.

The code dedicated to the boost converter regulation loop is loaded into the code RAM 2. This code is executed independently by the microcores Uc0Ch2.

At boost startup, the current through the inductor oscillates. This current is maintained between a current lower to the inductor saturation current and a positive current close to zero by turning the low-side switch on/off. When this switch is on, the current grows through the sense resistor and the low-side switch. When the switch is open, the current decays through the diode and loads the output capacitor. It increases the voltage until the $V_{\text{BOOST}}$ voltage reaches the $V_{\text{BOOST, HIGH}}$ threshold. This phase uses the asynchronous mode and the current modulation is managed by an independent circuitry enabled by the microcore.
When the $V_{\text{BOOST\_HIGH}}$ threshold is reached, the synchronous mode is enabled. In this case, the microcore takes the direct control of the low-side switch. The low-side switch is turned off until the boost voltage goes below the $V_{\text{BOOST\_LOW}}$ threshold.

Each time the $V_{\text{BOOST\_HIGH}}$ threshold is reached, the $V_{\text{BOOST\_LOW}}$ threshold is set up. The synchronous mode is activated after a $t_{\text{BOOST\_FILTER}}$ filter time required by the voltage comparator circuitry enablement.

Each time the boost voltage falls below the $V_{\text{BOOST\_LOW}}$ threshold the $V_{\text{BOOST\_HIGH}}$ threshold is set up. The asynchronous mode is activated after a $t_{\text{BOOST\_FILTER}}$ filter time.

Figure 6. VFM startup sequence

Figure 7. VFM voltage and current diagram
Table 16. Example of VFM DC-to-DC converter key parameters

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{BOOST_HIGH}$</td>
<td>$V_{BOOST}$ voltage HIGH threshold</td>
<td>65.31 V</td>
</tr>
<tr>
<td>$V_{BOOST_LOW}$</td>
<td>$V_{BOOST}$ voltage LOW threshold</td>
<td>64.69 V</td>
</tr>
<tr>
<td>$I_{SENSE4_HIGH}$</td>
<td>HIGH current threshold</td>
<td>3.44 A</td>
</tr>
<tr>
<td>$I_{SENSE4_LOW}$</td>
<td>LOW current threshold</td>
<td>0.41 A</td>
</tr>
</tbody>
</table>

In the present case, most of the code branches (jump) are managed according to the $V_{BOOST}$ voltage and the flag0 state with the wait table. The wait table rows are affected as shown in Table 17 and are changed according to the actuation phase.

Table 17. Example of wait table definition for the fuel pump drive

<table>
<thead>
<tr>
<th>Phase</th>
<th>Async phase (dcdc_on)</th>
<th>Sync phase (dcdc_off)</th>
<th>Idle phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row 1</td>
<td>if flag0 is LOW, then jump to idle phase</td>
<td>if flag0 is LOW, then jump to idle phase</td>
<td>-</td>
</tr>
<tr>
<td>Row 2</td>
<td>-</td>
<td>if $V_{BOOST} &lt; V_{BOOST_LOW}$, then jump to async phase</td>
<td>-</td>
</tr>
<tr>
<td>Row 3</td>
<td>if $V_{BOOST} &gt; V_{BOOST_HIGH}$, then jump to sync phase</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Row 4</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Row 5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

To avoid regulation disturbances, the boost voltage regulation is stopped by the mean of the internal flag 0 when an injection phase starts.

The $t_{BOOST\_FILTER}$ time is defined in the Boost_filter register (0x19D). This filter time and type can be adjusted to improve the $V_{BOOST}$ voltage stability.

Table 18. Boost_filter register (0x19D)

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|
| Name          |    |    |    |    |    |    |   |   |   |   |   |   |   |   |   |
| Value         | 000 | 0  |    |    |    |    |   |   |   |   |   |   |   |   |   |

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

5.2.1 DC-to-DC management registers setup

The MC33816/PT2001 registers are set up according to their default states, unless defined by the following.

5.2.1.1 IO configuration registers

The Uc0Ch2 must have access to the pre-driver LS7 only.

Table 19. Out_acc_uc0_ch2 register (0x186)

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>000000000000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
The Uc0Ch2 must have access to the current sense feedback 4L and 4H.

Table 20. Cur_block_access_2 register (0x189)

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>reserved</td>
<td>acc uc1 ch2 curr 4h_4neg</td>
<td>acc uc1 ch2 curr3</td>
<td>acc uc1 ch2 curr2</td>
<td>acc uc0 ch2 curr1</td>
<td>acc uc0 ch2 curr4l</td>
<td>acc uc0 ch2 curr3</td>
<td>acc uc0 ch2 curr2</td>
<td>acc uc0 ch2 curr1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>000000</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.2.1.2 Diagnosis configuration registers

The low-side driver 7 must be directly controlled by the microcore Uc0Ch2. Consequently, the output_routing fields of its output configuration register must be set to the value 15.

Table 21. Ls7_output_config register (0x152)

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>reserved</td>
<td>fast dcdc_en</td>
<td>output_routing</td>
<td>inv</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>0000000000</td>
<td>0</td>
<td>1111</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.2.2 DC-to-DC management algorithm

Initialization Phase
- Current sense operational amplifier gain setting
- Load the min current threshold in DAC 4L
- Load the max current threshold in DAC 4H
- Set the DAC mode to set Vboost voltage

Boost Phase
- Define wait table entry #1: Jump to Idle Phase when flag0 is low (boost injection phase ongoing)
- Define wait table entry #2: Jump to Asynchronous Phase when Vboost voltage is below Vboost_min
- Define wait table entry #3: Jump to Synchronous Phase when Vboost voltage is above Vboost_max

Asynchronous Phase
- Load the Vboost_max threshold in vboost_dac register
- Enable the Asynchronous mode
- Wait for wait table entry 1 or 2 to be satisfied

Synchronous Phase
- Load the Vboost_max threshold in vboost_dac register
- Enable the Synchronous mode (LS7 driver off)
- Wait for wait table entry 1 or 3 to be satisfied

Idle Phase
- Enable the Synchronous mode (LS7 driver off)
- Jump to previous line while flag0 is low
- Unconditional jump to Asynchronous Phase

Figure 8. Algorithm example for VFM

Refer to DC-to-DC management source code.

5.3 Fuel pump drive

The current profile is managed to generate an initial high current peak. The current is maintained while the START5 pin keeps the pump running to the end of actuation (EOA).

The code dedicated to the pump driving is loaded into the code RAM 2. This code is executed by the microcore Uc1Ch2. The microcore generates a current profile, as described by the following, per the START5 pin state.
When a rising edge is detected on a START5 pin, the pump actuation starts with the peak phase.

This profile can be stopped at any time by a falling edge detected on the START5 pin, where the EOA phase is executed.

During the peak phase, the corresponding low-side driver is simultaneously switched on with the high-side switch connected to the $V_{BAT}$ voltage. If the peak current target $I_{\text{peak}}$ is reached, the hold phase begins.

During the hold phase, the low-side driver is simultaneously switched on with the high-side switch connected to the $V_{BAT}$ voltage. If the hold current target $I_{\text{hold}}$ is reached, the high-side driver is switched off and the current recirculates through the diodes connected to $V_{BAT}$ and ground for the fixed time $t_{\text{hold\_off}}$. The high-side driver is then switched on again. This cycle repeats until the SATR5 pin goes LOW or the internal counter reaches its terminal value $t_{\text{hold\_tot}}$ (timeout). The EOA is forced if no falling edge is detected on the START5 pin.

All the current thresholds and timings are accessed in the data RAM. The typical values are the following, but must be defined according to the current profile expected.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_{\text{peak}}$</td>
<td>current threshold in peak phase</td>
<td>4.79 A</td>
</tr>
<tr>
<td>$I_{\text{hold}}$</td>
<td>current threshold in hold phase</td>
<td>3.27 A</td>
</tr>
<tr>
<td>$t_{\text{hold_off}}$</td>
<td>fixed time for high-side switch off in hold phase</td>
<td>10 μs</td>
</tr>
<tr>
<td>$t_{\text{hold_tot}}$</td>
<td>fixed time for end of hold phase (timeout)</td>
<td>10 ms</td>
</tr>
</tbody>
</table>
In this case, most of the code branches (jump) are managed according to the counters end of count and the current threshold by the means of the wait table. The wait table rows are affected, as shown in Table 23, and are changed according to the actuation phase.

### Table 23. Example of wait table definition for the fuel pump drive

<table>
<thead>
<tr>
<th>Phase</th>
<th>Peak phase</th>
<th>Hold phase</th>
<th>EOA phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row 1</td>
<td>if STARTx goes LOW, then jump to EOA phase</td>
<td>if STARTx goes LOW, then jump to EOA phase</td>
<td>-</td>
</tr>
<tr>
<td>Row 2</td>
<td>if the current reaches ( I_{\text{peak}} ), then jump to hold phase</td>
<td>if ( t_{\text{hold_tot}} ) is reached, then jump to EOA phase</td>
<td>-</td>
</tr>
<tr>
<td>Row 3</td>
<td>-</td>
<td>if ( t_{\text{hold_off}} ) is reached, then jump to hold on phase (sub phase)</td>
<td>-</td>
</tr>
<tr>
<td>Row 4</td>
<td>-</td>
<td>if ( t_{\text{hold}} ) is reached, then jump to hold off phase (sub phase)</td>
<td>-</td>
</tr>
<tr>
<td>Row 5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

#### 5.3.1 Fuel pump drive registers setup

The Uc1Ch2 must have access to the pre-drivers HS5 and LS5.

#### 5.3.2 IO configuration registers

### Table 24. Out_acc_uc1_ch2 register (0x187)

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>reserved</td>
<td>acc__uc1__ch2__ls7</td>
<td>acc__uc1__ch2__ls6</td>
<td>acc__ acc__</td>
<td>acc__ acc__</td>
<td>acc__ acc__</td>
<td>acc__ acc__</td>
<td>acc__ acc__</td>
<td>acc__ acc__</td>
<td>acc__ acc__</td>
<td>acc__ acc__</td>
<td>acc__ acc__</td>
<td>acc__ acc__</td>
<td>acc__ acc__</td>
<td>0000</td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>0000</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

The Uc1Ch2 must have access to the current sense feedback # 3.

### Table 25. Cur_block_access_2 register (0x189)

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>reserved</td>
<td>acc__uc1__ch2__curr4h__4neg</td>
<td>acc__uc1__ch2__curr4l</td>
<td>acc__uc1__ch2__curr3</td>
<td>acc__uc1__ch2__curr2</td>
<td>acc__uc1__ch2__curr1</td>
<td>acc__uc0__ch2__curr4h__4neg</td>
<td>acc__uc0__ch2__curr3</td>
<td>acc__uc0__ch2__curr2</td>
<td>acc__uc0__ch2__curr1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>000000</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.3.2.1 Channel 2 configuration registers

The pump is driven according to the logic level of the START5 pin. A HIGH level on START5 drives the pump on, while a LOW level stops the pump. The microcore Uc1Ch2 must be enabled by the START5 pin. Consequently, the Start_config_reg register of channel 2 (0x124) must be set up.

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>reserved</td>
<td>smart_start_uc1</td>
<td>reserved</td>
<td>smart_start_uc6</td>
<td>start5_sens_uc1</td>
<td>start4_start3_start2_sens_uc1</td>
<td>start1_sens_uc1</td>
<td>start6_start5_sens_uc0</td>
<td>start4_start3_start2_sens_uc0</td>
<td>start1_sens_uc0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

The Code_width register and the two checksum value registers must be set to verify permanently the code integrity. The checksum is recalculated in the MC33816/PT2001 at runtime, each time a microcode line is executed. The code width and checksum values are provided for a DC-to-DC converter regulation code and a pump drive code.

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>reserved</td>
<td>code_width</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>-</td>
<td>000100111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>checksum_high</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset</td>
<td>0100000100010100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>checksum_low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset</td>
<td>1010101111010001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The code entry point of Uc0Ch2 is logic 0 the first line executed, is the first code RAM line of the channel 2.

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>reserved</td>
<td>entry_point_address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset</td>
<td>000000</td>
<td>0000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The code entry point of Uc1Ch2 is at the 16th code RAM line of the channel 2.
### Table 31. Uc1_entry_point register (0x12B) of channel 2

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>reserved</td>
<td>entry_point_address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset</td>
<td>000000</td>
<td>0000010000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The registers must be set up to run the two microcores of the channel 2 as the first microcode drives the DC-to-DC converter, and the second microcore drives the fuel pump.

Both microcores are enabled by setting the pre_flash_enable bit and the en_dual_uc bit to logic 1.

### Table 32. Flash_enable register (0x120) of channel 2

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>reserved</td>
<td>checksum_disable</td>
<td>flash_enable</td>
<td>pre_flash_enable</td>
<td>en_dual_uc</td>
<td>dual_uc_failure</td>
<td>checksum_irq_en</td>
<td>checksum_failure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>0000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 5.3.2.2 Diagnosis configuration registers

The high-side and low-side drivers must be directly controlled by the microcore Uc1Ch2. Consequently, the output_routing fields of the high-side and low-side drivers output configuration register must be set to the value 15.

### Table 33. Hs5_output_config register (0x161)

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>reserved</td>
<td>dead_time</td>
<td>output_routing</td>
<td>inv</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset</td>
<td>000000000</td>
<td>0000</td>
<td>1111</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 34. Lsx_output_config registers (0x14E, 0x151)

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>reserved</td>
<td>output_routing</td>
<td>inv</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>00000000000</td>
<td>1111</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.3.3 Fuel pump drive algorithm

![Fuel pump drive algorithm diagram]

Refer to Fuel pump drive source code.

6 PCB layout recommendations

6.1 Ground connections

The MC33816/PT2001 exposed pad must be connected to the printed-circuit board (PCB) ground. All the grounds (AGND, DGND, and PGND) must be ‘in star’ or considering a unique ground layer, such as to minimize the introduction of offset and noise mainly in the signal return lines.

6.2 Sense resistors connection

The sense resistors layout must be considered with special care, to sense the voltage as close as possible to the resistor terminations.

Balanced series resistance, induced by the layout, between the sense resistor positive termination to the VSENSEPx and the sense resistor negative termination to the VSENSENx pin is recommended. The balance can be achieved by implementing similar line lengths.
Figure 11. Example of force and sense connection layout

It is highly recommended to place the sense resistor as close as possible to its corresponding low-side MOSFET transistor.

6.3 Drain and source signal connection

The high-side source signals (S_HSx) must be connected as close as possible to its corresponding high-side MOSFET transistor source pin.

The low-side drain signals (G_LSx) must be connected as close as possible to its corresponding high-side MOSFET transistor source pin.

7 Application source code

7.1 Injection banks management source code

* Copyright 2014 NXP. NXP Confidential. This software is owned or controlled by NXP and may only be used strictly in accordance with the applicable license terms found at https://www.nxp.com/LA_OPT_NXP_SW. The “production use license” in Section 2.3 in the NXP SOFTWARE LICENSE AGREEMENT is expressly granted for this software.

* ### Channel 1 - uCore0 controls the injectors 1 and 2 ###

* ### Variables declaration ###

* Note: The data are stored into the dataRAM of the channel 1.
* Note: The Thold_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 ###

init0:      stgn gain8.68.sssc;                     * Set the gain of the opamp of the current measure block 1
            ldjr1 eoinj0;                           * Load the eoinj line label Code RAM address into the register jr1
            ldjr2 idle0;                            * 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 ###

idle0:      cwer CheckStart start row2;             * Define entry table for high start pin
            stoc on sssc;                                * Turn ON off comp
            WaitLoop:   wait row2;                              * uPC is stuck here for almost the whole idle time
            CheckStart: joslr inj1_start start1;                * Jump to inj1 if start 1 is high
            joslr inj2_start start2;                * Jump to inj2 if start 2 is high
            jmpr WaitLoop;

* ### Shortcuts definition per the injector to be actuated ###

inj1_start: dfsct hs1 hs2 ls1;                      * Set the 3 shortcuts: VBAT, VBOOST, LS
            jmpr boost0;                            * Jump to launch phase

inj2_start: dfsct hs1 hs2 ls2;                      * Set the 3 shortcuts: VBAT, VBOOST, LS
            jmpr boost0;                            * Jump to launch phase

* ### Launch phase enable boost ###

boost0:     stoc off sssc;                          * Turn OFF offset compensation
            “bias all on;”/* Enable all biasing structures, kept ON even during actuation
            load lboost dac_easc _ofs;                * Load the boost phase current threshold in the current DAC
            cwer peak0 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
            stos off on on;                         * Turn VRAT off, BOOST on, LS on
            wait row12;                             * Wait for one of the previously defined conditions

AN4849

All information provided in this document is subject to legal disclaimers. © NXP B.V. 2019. All rights reserved.
### Variables declaration ###

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

#### Initialization phase ####

- **init1:**
  - `stgn gain8.68 assc;`
  - `ldj1 eoinj1;`
  - `ldj2 idle1;`
  - `ldj1 _start rowl;`
- **init2:**
  - `ldj3 eoinj3;`
  - `ldj4 idle3;`
  - `ldj3 _start rowl;`

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

- `id1el1: cwer CheckStart1 start row12;`
- `stos on assc;`
- `WaitLoop1: wait row32; CheckStart1 josr1 in31 _start testCase; josr1 _start testCase; jmpr WaitLoop1;`
- `inj1start: dfct h3 h4 le4; jmpr boost1;`
- `inj2start: dfct h3 h4 le4; jmpr boost1;`

#### Shortcuts definition per the injector to be actuated ####

- `inj3start: dfct h3 h4 le4; jmpr boost1;`
- `inj4start: dfct h3 h4 le4; jmpr boost1;`

#### Launch phase enable boost ####

- `boost1: stos assc; load Iboost dac assc ofs; cwer peakl ocor row2; stf low b0; stos on on; wait row21;`
- `boost2: stos assc; load Iboost dac assc ofs; cwer peakl ocor row2; stf low b0; stos on on; wait row21;`

#### Peak phase continue on Vbat ####

- `peakl: ldcd rst ofs keep Tpeak_tot c1;` (Load the length of the total peak phase in counter 1)
- `peak2: ldcd rst ofs keep Tpeak_tot c2;` (Load the peak current threshold in the current DAC)
- `peak3: ldcd rst ofs keep Tpeak_tot c3;` (Load the peak current threshold in the current DAC)
- `peak4: ldcd rst ofs keep Tpeak_tot c4;` (Load the peak current threshold in the current DAC)
- `peak5: ldcd rst ofs keep Tpeak_tot c5;` (Load the peak current threshold in the current DAC)
- `peak6: ldcd rst ofs keep Tpeak_tot c6;` (Load the peak current threshold in the current DAC)
- `peak7: ldcd rst ofs keep Tpeak_tot c7;` (Load the peak current threshold in the current DAC)
- `peak8: ldcd rst ofs keep Tpeak_tot c8;` (Load the peak current threshold in the current DAC)
- `peak9: ldcd rst ofs keep Tpeak_tot c9;` (Load the peak current threshold in the current DAC)
- `peak10: ldcd rst ofs keep Tpeak_tot c10;` (Load the peak current threshold in the current DAC)
- `peak11: ldcd rst ofs keep Tpeak_tot c11;` (Load the peak current threshold in the current DAC)
- `peak12: ldcd rst ofs keep Tpeak_tot c12;` (Load the peak current threshold in the current DAC)
- `peak13: ldcd rst ofs keep Tpeak_tot c13;` (Load the peak current threshold in the current DAC)
- `peak14: ldcd rst ofs keep Tpeak_tot c14;` (Load the peak current threshold in the current DAC)
- `peak15: ldcd rst ofs keep Tpeak_tot c15;` (Load the peak current threshold in the current DAC)
- `peak16: ldcd rst ofs keep Tpeak_tot c16;` (Load the peak current threshold in the current DAC)
- `peak17: ldcd rst ofs keep Tpeak_tot c17;` (Load the peak current threshold in the current DAC)
- `peak18: ldcd rst ofs keep Tpeak_tot c18;` (Load the peak current threshold in the current DAC)
- `peak19: ldcd rst ofs keep Tpeak_tot c19;` (Load the peak current threshold in the current DAC)
- `peak20: ldcd rst ofs keep Tpeak_tot c20;` (Load the peak current threshold in the current DAC)
### Bypass phase ###

\[ \text{bypass1: ldcd rst ofs keep keep Tbypass c3; } \]
\[ \text{stos ofs off off; } \]
\[ \text{load Vboost off in counter } \]
\[ \text{### Hold phase on Vbat ###} \]

\[ \text{hold1: ldcd rst ofs keep keep Thold_tot c1; } \]
\[ \text{load hold dac ssac ofs; } \]
\[ \text{set DAC access mode to Vboost } \]
\[ \text{### End of injection phase ###} \]

\[ \text{eoinj1: stos off off off; } \]
\[ \text{stf high b0; } \]
\[ \text{### End of Channel 1 - uCore1 code ###} \]

#### 7.2 DC-to-DC management source code ####

* Copyright 2014 NXP. NXP Confidential. This software is owned or controlled by NXP and may only be
  used strictly in accordance with the applicable license terms found at
  https://www.nxp.com/LA_OPT_NXP_SW. The "production use license" in Section 2.3 in the NXP SOFTWARE
  LICENSE AGREEMENT is expressly granted for this software.

* ### Channel 2 - uCore0 controls dc-dc ###

* ### Initialization phase ###

\[ \text{init0: stgn gain5.8 ossc; } \]
\[ \text{load Iboost_L dac_ossc ofs; } \]
\[ \text{load Iboost_H dac4h4n ofs; } \]
\[ \text{stdm null; } \]
\[ \text{### End of Channel 2 - uCore0 code ###} \]
7.3 Fuel pump drive source code

* Copyright 2014 NXP. NXP Confidential. This software is owned or controlled by NXP and may only be used strictly in accordance with the applicable license terms found at
* https://www.nxp.com/LA_OPT_NXP_SW. The "production use license" in Section 2.3 in the NXP SOFTWARE LICENSE AGREEMENT is expressly granted for this software.

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

```assembly
### Initialization phase ###
initil:   stgn gain19.4 osc;                      * Set the gain of the opamp of the current measure
  block 1
  ldjr1 eoincl;                                     * Load the eoinj line label Code RAM address into the
  register jr1
  ldjr2 idel1;                                    * Load the idle line label Code RAM address into the
  register jr2
  cwer jr1 _start row1;                           * If the start signal goes low, go to eoinj phase

### Idle phase - the uPC loops here until start signal is present ###
idel1:    joslr act5_start start5;                * Perform an actuation on act5 if start 5 (only) is active
  joslr act6_start start6;                        * Perform an actuation on act5 if start 5 (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;
act6_start: dfsct hs5 ls6 undef;                  * Set the 2 shortcuts: VBAT, LS

### Launch peak phase on bat ###
peak1:    load Ipeak dac_ossc _ofs;               * Load the boost phase current threshold in the current DAC
  cwer hold1 c3 row2;                             * Jump to peak phase when current is over threshold
  stos on on keep;                               * Turn VBAT off, BOOST on, LS on
  wait row12;

### Hold phase on Vbat ###
hold1:    ldcd rst _ofs keep keep Thold_tot c1;    * Load the length of the total hold phase in counter
  load 1hold dac_ossc ofs;                       * Load the hold current threshold in the DAC
  cwer eoincl tcl row2;                         * Jump to eoinj phase when tcl reaches end of count
  cwer hold_on1 to3 row2;                      * Jump to hold_on when t22 reaches end of counter
  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 for one of the previously defined conditions
  wait row123;

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

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

8 References

### 9 Revision history

<table>
<thead>
<tr>
<th>Rev</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
</table>
| 5   | 20190715   | • Figure 1: updated VBATT pin circuitry  
• Moved revision history to Section 9. |
| 4   | 20190329   | • The format of this document has been redesigned to comply with the new identity guidelines of NXP Semiconductors.  
• Legal texts have been adapted to the new company name where appropriate.  
• added product type PT2001 |
| 3   | 11/2014    | • Replaced KIT33816AEEVM by KIT33816FRDMEVM  
• Add a reference link for KT33816FRDMUG |
| 2   | 9/2014     | • Updated timeout value (T_HOLD_TOT) in the example of injection current profile key parameters table  
• Updated bit names in the Out_acc_uc1_ch2 register (0x187) table |
| 1   | 1/2013     | initial version                                                                                                                                   |
10 Legal information

10.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

10.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors. In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory. Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors’ aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification. Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer’s sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer’s applications and products planned, as well as for the planned application and use of customer’s third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products. NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer’s applications or products, or the application or use by customer’s third party customer(s). Customer is responsible for doing all necessary testing for the customer’s applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer’s third party customer(s). NXP does not accept any liability in this respect.

Suitability for use in automotive applications — This NXP Semiconductors product has been qualified for use in automotive applications. Unless otherwise agreed in writing, the product is not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer’s own risk.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — While NXP Semiconductors has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer’s applications and products, and NXP Semiconductors accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

10.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

NXP — is a trademark of NXP B.V.

SMARTMOS — is a trademark of NXP B.V.
Tables

Tab. 1. Example of injection current profile key parameters (RSENSE = 10 mΩ) .......................... 6
Tab. 2. Example of wait table definition ................................................................. 6
Tab. 3. Ck_per register (0x1C0) ................................................................. 7
Tab. 4. Out_acc_uc0_ch1 register (0x184) ......................................................... 8
Tab. 5. Out_acc_uc1_ch1 register (0x185) ......................................................... 8
Tab. 6. Cur_block_access_1 register (0x188) .............................................. 8
Tab. 7. Start_config_reg registers (0x124) ............................................... 9
Tab. 8. Code_width registers (0x107) ......................................................... 9
Tab. 9. Checksum_h registers (0x108) ....................................................... 9
Tab. 10. Checksum_l registers (0x109) ....................................................... 9
Tab. 11. Uc0_entry_point registers (0x10A) of channel 1 ................................ 9
Tab. 12. Uc1_entry_point registers (0x10B) of channel 1 ................................ 9
Tab. 13. Flash_enable register (0x100) of channel 1 ..................................... 10
Tab. 14. Hsx_output_config registers (0x155, 0x158, 0x15B, 0x15E) ................. 10
Tab. 15. Lsx_output_config registers (0x142, 0x145, 0x148, 0x14B) ................. 10
Tab. 16. Example of VFM DC-to-DC converter key parameters .................. 14
Tab. 17. Example of wait table definition for the fuel pump drive .............. 14
Tab. 18. Boost_filter register (0x19D) .................................................... 14
Tab. 19. Out_acc_uc0_ch2 register (0x186) ............................................. 14
Tab. 20. Cur_block_access_2 register (0x189) ..................................... 15
Tab. 21. Ls7_output_config register (0x152) ........................................... 15
Tab. 22. Example of fuel pump drive key parameters .................................. 17
Tab. 23. Example of wait table definition for the fuel pump drive .............. 18
Tab. 24. Out_acc_uc1_ch2 register (0x187) ............................................. 18
Tab. 25. Cur_block_access_2 register (0x189) ..................................... 18
Tab. 26. Start_config_reg register (0x124) ........................................... 19
Tab. 27. Code_width register (0x127) .................................................... 19
Tab. 28. Checksum_h register (0x128) ................................................... 19
Tab. 29. Checksum_l register (0x129) ................................................... 19
Tab. 30. Uc0_entry_point register (0x12A) of channel 2 ......................... 19
Tab. 31. Uc1_entry_point register (0x12B) of channel 2 ......................... 19
Tab. 32. Flash_enable register (0x120) of channel 2 ................................ 20
Tab. 33. Hsx_output_config register (0x161) ........................................ 20
Tab. 34. Lsx_output_config registers (0x14E, 0x151) ................................ 20
Tab. 35. Revision history .............................................................. 26

Figures

Fig. 1. Typical four injector two bank application schematic ...................... 3
Fig. 2. Typical peak and hold current profile ........................................... 5
Fig. 3. Actuation driven by STARTx pins without overlap ............................. 7
Fig. 4. Algorithm example for a bank of two injectors ................................ 11
Fig. 5. Simplified DC-to-DC converter topology for VFM .......................... 12
Fig. 6. VFM startup sequence ............................................................. 13
Fig. 7. VFM voltage and current diagram ............................................. 13
Fig. 8. Algorithm example for VFM ..................................................... 16
Fig. 9. Typical fuel pump current profile .............................................. 17
Fig. 10. Algorithm example for fuel pump drive ..................................... 21
Fig. 11. Example of force and sense connection layout ............................... 22
# Contents

1 Introduction ......................................................... 2
2 Overview .............................................................. 2
3 Application schematic ........................................ 3
4 Application instructions ..................................... 4
5 Software requirements ....................................... 4
  5.1 Current profile management for injection ...........4
  5.1.1 General registers setup ............................. 7
  5.1.1.1 Main configuration register ................... 7
  5.1.2 Injection banks management registers setup ....7
  5.1.2.1 IO configuration registers ....................... 7
  5.1.2.2 Channel 1 configuration registers .......... 7
  5.1.2.3 Diagnosis configuration registers .............10
  5.1.3 Injection banks management algorithm ..........11
  5.2 DC-to-DC management ................................... 12
  5.2.1 DC-to-DC management registers setup ...........14
  5.2.1.1 IO configuration registers ....................... 14
  5.2.1.2 Diagnosis configuration registers .......... 15
  5.2.2 DC-to-DC management algorithm ................... 16
5.3 Fuel pump drive ...............................................16
  5.3.1 Fuel pump drive registers setup .................. 18
  5.3.2 IO configuration registers ........................ 18
  5.3.2.1 Channel 2 configuration registers .......... 19
  5.3.2.2 Diagnosis configuration registers .......... 20
  5.3.3 Fuel pump drive algorithm ...........................21
6 PCB layout recommendations ......................... 21
  6.1 Ground connections .................................. 21
  6.2 Sense resistors connection .......................... 21
  6.3 Drain and source signal connection .............. 22
7 Application source code .................................. 22
  7.1 Injection banks management source code ........22
  7.2 DC-to-DC management source code ............ 24
  7.3 Fuel pump drive source code ....................... 25
8 References .........................................................25
9 Revision history ................................................ 26
10 Legal information .............................................27