Application Note

DA9063-A Power Management for
Renesas R-Car M3 Platform

AN-PM-097

Abstract

The R-Car M3 System-on-Chip (SoC)-based platform from Renesas is part of a family of platforms (R-Car series) for automotive infotainment systems. The M3 is aimed at the mid-level segment, and is optimized for automotive Human Machine Interface (HMI), infotainment and integrated dashboards.

The platform features the Dialog DA9063-A as system PMIC (Power Management IC) and the Dialog DA9224-A multi-phase sub-PMIC step-down buck converter to power and supervise the complete system.

Through a description of the general system configuration, power capabilities and requirements and an overview of the component interconnections, it will be shown that the combination of DA9063-A and DA9224-A are highly suited as the R-Car power management system solution for M3 platforms.
DA9063-A Power Management for Renesas R-Car M3 Platform

Contents

Abstract ................................................................................................................................. 1

Contents ............................................................................................................................ 2

Figures ............................................................................................................................... 2

Tables ................................................................................................................................. 3

1 Introduction ...................................................................................................................... 4

2 Renesas R-Car M3 SoC Description ............................................................................... 4

3 DA9063-A, and DA9224-A Description ......................................................................... 5

4 R-Car M3 SoC Power Requirements ............................................................................ 7

5 R-Car M3 SoC Power Tree System Diagram ................................................................. 8

6 Cold boot Sequence for R-Car M3 ............................................................................... 9

7 Operation ....................................................................................................................... 10

  7.1 DVFS and AVS ............................................................................................................. 10

  7.2 Memory Retention Mode (Sleep mode) ...................................................................... 10

    7.2.1 Warm Boot vs Cold Boot .................................................................................. 11

    7.2.2 Sleep Timer ....................................................................................................... 11

8 Reference Design .......................................................................................................... 13

  8.1 Measurement Results ............................................................................................... 14

Appendix A DA9063-7C-HO2-A Detailed Register Description ........................................ 19

Appendix B Software Implementation ................................................................................ 24

  B.1 Set DA9063 register 0x94......................................................................................... 24

  B.2 Set DA9063 BKUP_TRG bit or DA9063 GP_ID_1 register ...................................... 24

  B.3 Read DA9063 BKUP_TRG bit or DA9063 GP_ID_1 register .................................... 24

  B.4 Set DA9063 PMIC RTC alarm ............................................................................... 25

  B.5 Turn on DA9063 PMIC RTC alarm ......................................................................... 26

  B.6 Linux device driver PMIC RTC ............................................................................. 26

Revision History ............................................................................................................... 29

Figures

Figure 1: R-Car M3 System Block Diagram ....................................................................... 4

Figure 2: DA9063-A System Block Diagram .................................................................... 5

Figure 3: DA9224-A System Block Diagram .................................................................... 6

Figure 4: Start-Up Sequence (Timing Is Not To Scale) ...................................................... 7

Figure 5: R-Car M3 and PMIC Interconnections ............................................................... 8

Figure 6: DA9063-7C-HO2-A Power-Up Sequence .......................................................... 9

Figure 7: SoC Sequence for Memory Retention Entry ..................................................... 12

Figure 8: SoC Sequence for Memory Retention Exit ....................................................... 12

Figure 9: The Dialog R-Car M3 Reference Design ........................................................... 13

Figure 10: DDR_1.1V Efficiency ..................................................................................... 14

Figure 11: DDR_1.8V Efficiency ...................................................................................... 14

Figure 12: D1.8V Efficiency ............................................................................................ 15

Figure 13: D3.3V Efficiency ............................................................................................ 16

Figure 14: VDD_0.8 Efficiency ....................................................................................... 17

Figure 15: DVFS_0.8 Efficiency ...................................................................................... 18
Tables

Table 1: DA9063-7C-HO2-A Register Settings

.. pagebreak:: 19
1 Introduction

This document describes how to interconnect the DA9063-A Power Management IC (PMIC) and the DA9224-A sub-PMIC to the Renesas R-Car M3 System on a Chip (SoC). The DA9063-A is a highly integrated chip that supports Dynamic Voltage Control (DVC) technology, enabling significant power saving: this feature supports the Dynamic Voltage and Frequency Scaling (DVFS) technology that is used by many processors.

As a result of their highly integrated features, the DA9063-A PMIC, and DA9224-A sub-PMIC significantly reduce the overall system cost and size compared to a discrete solution. This application note addresses only the power supply related features: discussion of other features of the optimized PMIC is beyond the scope of this document.

For further information on the DA9063-A, and DA9224-A please refer to the datasheets available via your local Dialog sales office.

For information about Renesas R-Car M3 SoC, please refer to Renesas website:


2 Renesas R-Car M3 SoC Description

Renesas R-Car M3 is a platform for automotive infotainment with an SoC containing eight cores (ARM® Cortex®-A57 Quad Core, ARM Cortex-A53 Quad, ARM Cortex-R7 Dual lock-step) and PowerVR GX6650 GPU.

Figure 1 shows a typical system block diagram of the R-Car M3 SoC application. The embedded cores require suitable power management that is readily achieved using the Dialog DA9063-A, and DA9224-A.

Figure 1: R-Car M3 System Block Diagram
3 DA9063-A, and DA9224-A Description

The DA9063-A (Figure 2) is a high-current system PMIC suitable for dual- and quad-core processors that require up to 5 A core processor supply. The DA9063-A contains:

- Six DC-DC buck converters designed to use small external 1 μH inductors, capable of supplying in total up to 12 A continuous output current (0.3 V to 3.3 V). The buck converters do not require external Schottky diodes; they dynamically optimize their efficiency depending on the load-current using an Automatic Sleep Mode (ASM) and incorporate pin and software controlled Dynamic Voltage Control (DVC) to support processor load adaptive adjustment of the supply voltage. In addition BuckPro includes the facility to implement VTT memory bus termination if required.

- 11 SmartMirror™ programmable low-dropout (LDO) regulators rated up to 300 mA. All support remote capacitor placement and can operate from low 1.5 V/1.8 V input supplies. This allows these LDOs to be cascaded with (in other words: supplied by) a suitable buck supply to improve overall system efficiency.

Figure 2: DA9063-A System Block Diagram
The DA9224-A (Figure 3) provides two, dual-phase synchronous step-down converters suitable for supplying CPUs that require high currents. The converter operates using a small external 0.22 μH inductor on each phase. It produces an output voltage in the range of 0.3 V to 1.57 V. The input voltage range of 2.8 V to 5.5 V makes it suited to a wide variety of low-voltage systems.

To guarantee the highest accuracy and support multiple PCB routing scenarios without loss of performance, a remote sensing capability is implemented on each DA9224-A output.

Each DA9224-A buck operates with two phases and is capable of delivering up to 10 A continuous output current per buck.

Figure 3: DA9224-A System Block Diagram
4 R-Car M3 SoC Power Requirements

Several power domains in the R-Car M3 SoC platform require precise voltage management for reliable system operation. The primary power domains are:

- VDD_DFVS
- VDD_08V
- DDR_1.1V
- DDR_1.8V

Other supplies will be required for peripherals, I/O interfaces, SD cards, and such. Additionally, the system power management must comply with the specific power-up and power-down sequence guidelines for the R-Car SoC (shown in Figure 4).

Figure 4: Start-Up Sequence (Timing Is Not To Scale)

Figure 5 shows the PMIC interconnections that satisfies this requirement.
5  R-Car M3 SoC Power Tree System Diagram

Figure 5: R-Car M3 and PMIC Interconnections
6 Cold boot Sequence for R-Car M3

Figure 6DA9063-7C-HO2-A Power-Up Sequence

Please contact your local Dialog representative for the recommended OTPs.
7 Operation

When 5 V is applied to the $V_{SYS}$ supply the DA9063-A system PMIC starts up automatically. It follows the start-up sequence programmed in the OTP, enabling output power rails in the order specified. GPIO7 and 11 are configured to control the enabling of the two sub-PMIC bucks and are also part of the power sequencer timing. In this way the start-up timing of the sub-PMIC buck outputs are controlled.

Once the sequencer has completed the start-up sequence the nRESET signal from the DA9063-A is released to allow the SoC to start operation.

The outputs of LDO6 and LDO7 are combined to provide the specified 300 mA load current for the SD0 card supply. Similarly, LDO8 and LDO9 are combined to power SD1 card supply. LDO10 is a 300 mA LDO and can supply the SD2 card individually. LDO4 provides power for SD3 card supply. If fewer SD cards are used in a customer end application or lower current SD cards are used then unused LDOs may be reused elsewhere in the end application.

GPIO1, 2, and 13 provide the ability for the SoC to select respectively the SD0, SD3 and SD2 card output voltages by controlling the logic level applied to the GPIO input. A logic low from the SoC produces an SD card voltage of 1.8 V output on the respective output; a logic high produces an output of 3.3 V. The SD1 card voltage is set in OTP to be 1.8 V with the ability to change this to 3.3 V via an I²C write.

LDO5 is used to generate the internal power supplies for the Dialog system and sub-PMICs.

7.1 DVFS and AVS

The DVFS voltage is set in OTP to be the default start-up voltage that is guaranteed to ensure the system powers up and runs. Deviations in the manufacturing process result in some processors that are capable of operating from lower voltages than the nominal. AVS allows for adjustments to the DVFS voltage to cater for these processors.

The SoC can adjust the DVFS set voltage by an I²C write to the DA9224-A, VBUCKA_CTRL_A register (address 0xD7). The output voltage can be adjusted in 10 mV steps.

The DVFS power rail can also be switched to a higher voltage for a short period of time. The higher voltage, BOOST mode is selected when the SoC takes the BOOST pin high.

The BOOST voltage is set in OTP to be the default boost voltage and can be adjusted by an I²C write to the DA9224-A, VBUCKA_CTRL_B register (address 0xD8). The output voltage can be adjusted in 10 mV steps.

7.2 Memory Retention Mode (Sleep mode)

PMIC_RSTBn from the SoC is used to enter and exit memory retention mode. When PMIC_RSTBn is taken to a logic low the system PMIC performs a power-down sequence. During the power-down sequence the bucks supplying the DDR1.1V and DDR1.8V outputs are re-configured to prevent them from switching off.

Under normal ACTIVE mode conditions all bucks are programmed to operate in PWM mode to produce predictable noise performance. Efficiency is reduced at low load currents when operating in this mode so when entering memory retention mode the bucks supplying DDR1.1V and DDR1.8V are automatically changed to operate in Pulse-Frequency Modulation (PFM) mode. In doing this the quiescent current from the 5 V input is reduced to less than 1 mA.

When PMIC_RSTBn is taken to a logic high once more or a wake event occurs the DDR1.1V and DDR1.8V bucks are automatically re-configured to operate in Pulse-Width Modulation (PWM) mode before the system returns fully to ACTIVE mode by following the start-up sequence.
7.2.1 Warm Boot vs Cold Boot

Figure 8 shows the procedure for exiting from memory retention mode. Memory retention mode exit is triggered by a wake-up event. A wake-up event can be triggered from a wake-up enabled GPIO edge, nONKEY going low, the SYS_EN pin rising or an RTC alarm being triggered. The wake-up event causes the PMIC to move up the sequencer and then release nRESET to start the SoC. At this point the SoC will begin the software boot-up procedure.

During system boot-up the SoC checks the state of BKUP_TRG to determine if the start-up requires a cold or warm boot process. BKUP_TRG is programmed in OTP to be low during a system power-up to indicate a cold boot is required. Before entering memory retention mode the SoC sets BKUP_TRG to be high to indicate a warm boot is required when the system is next started. If all power is lost BKUP_TRG reverts to the OTP setting which results in a cold boot.

If the SoC GPIO, GP1-08, is connected to BKUP_TRG output pin on DA9063-A the SoC can directly determine the BKUP_TRG status by reading GP1-08.

If GP1-08 is not connected to BKUP_TRG the SoC can determine if a warm or cold boot is required by either reading BKUP_TRG status via an I2C read of the DA9063-A or an I2C read of GP_ID_1 (address 0x122) register in DA9063-A. This register is programmed to be 0x00 in the OTP but the register value can be changed by the SoC using an I2C write. The register value is sustained through entry and exit of memory retention mode but reset to OTP value during power-up from off. The SoC can set GP_ID_1 to a non-zero value before entering memory retention mode and when booting up can interrogate the register to determine whether the boot-up is warm or cold.

Please refer to Appendix B for examples of the software implementation.

7.2.2 Sleep Timer

Taking PMIC_RSTBn high will result in a system wake-up event, with the PMIC following the power-up sequence.

An alternative approach is to use a sleep timer to wake the system after a predetermined time in memory retention mode.

This can be implemented on the Dialog power management solution by use of the RTC clock and RTC alarm function.

As before the SoC sets BKUP_TRG to be a logic high to indicate a warm boot when exiting sleep. Before entering sleep mode the SoC reads the RTC time within the DA9063-A. The SoC adds the required sleep interval to the RTC time and sets the DA9063-A alarm time to this new value. The SoC then sets the alarm to be active. Finally the SoC clears all interrupts before entering memory retention mode.

After the set time has elapsed the RTC alarm is triggered causing a wake-up event to be triggered within the DA9063-A. This causes the PMIC to move from memory retention mode to active waking up the SoC in the process.

The SoC checks BKUP_TRG state and determines the wake-up event is a warm boot.

Figure 7 shows the process by which the SoC enters memory retention mode if required. Firstly the SoC determines whether the system is shutting down to off or memory retention mode. If memory retention mode is required the following sequence is followed:

- The SoC performs an I2C write of 0x99 to PMIC register address 0x94.
- The SoC performs an I2C write to set PMIC GPIO3 = 1
  - An alternative option is the SoC performs an I2C write to set GP_ID_1 = 0x01
- If a wake from memory retention mode after a predetermined time is required the SoC performs an I2C read of the RTC time, adds on the required sleep time and writes back an RTC ALARM time into the PMIC before setting the RTC ALARM_ON bit.
- The SoC sets the DDR into self-refresh mode before performing an I2C write of 0x02 to register address 0x0E, thus clearing the SYSTEM_EN bit.
Figure 7: SoC Sequence for Memory Retention Entry

Figure 8: SoC Sequence for Memory Retention Exit

Please refer to Appendix B for examples of the software implementation.
8 Reference Design

Figure 9 shows a photograph of a Dialog R-Car M3 reference design evaluation board PCB. This PCB allows the PMIC solution to be evaluated. The 42 mm x 30 mm white lined box in the center of the PCB shows the core, active area used for components and tracking of the solution. The remainder of the PCB is purely for easy access connections.

![Figure 9: The Dialog R-Car M3 Reference Design](image-url)
8.1 Measurement Results

**Figure 10: DDR_1.1V Efficiency**

![Graph showing DDR_1.1V Efficiency](image1)

**Figure 11: DDR_1.8V Efficiency**

![Graph showing DDR_1.8V Efficiency](image2)
Figure 12: D1.8V Efficiency
DA9063 BuckMem & IO: Efficiency at Vout = 3.3 V

Figure 13: D3.3V Efficiency
Figure 14: VDD_08 Efficiency
Figure 15: DVFS_08 Efficiency
Appendix A DA9063-7C-HO2-A Detailed Register Description

Key Settings
- Normal start-up
- Voltage monitor
- Buck Core1 and Buck Core2 dual phase mode.
- Buck Mem and Buck IO merged mode
- 2-wire control interface, standard speed
- RTC enabled
- LDO 3, 4, 6, 7 and 10 are GPIO controlled by host for 1.8 V or 3.3 V SD card supply select

Table 1: DA9063-7C-HO2-A Register Settings

<table>
<thead>
<tr>
<th>Register Address</th>
<th>Function</th>
<th>Register Value</th>
<th>Register Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00A</td>
<td>IRQ_MASK_A</td>
<td>0x00</td>
<td>nONKEY, RTC, and some status IRQ masks</td>
</tr>
<tr>
<td>0x00B</td>
<td>IRQ_MASK_B</td>
<td>0x10</td>
<td>Charger wakeup and temperature, current, or voltage IRQ masks</td>
</tr>
<tr>
<td>0x00C</td>
<td>IRQ_MASK_C</td>
<td>0x00</td>
<td>GPI7 to 0 and ADCIN1-3 IRQ masks</td>
</tr>
<tr>
<td>0x00D</td>
<td>IRQ_MASK_D</td>
<td>0x00</td>
<td>GPI15 to 8 and external control signal IRQ masks</td>
</tr>
<tr>
<td>0x00E</td>
<td>CONTROL_A</td>
<td>0x03</td>
<td>PSM target status, companion charger control</td>
</tr>
<tr>
<td>0x00F</td>
<td>CONTROL_B</td>
<td>0x09</td>
<td>Power-down / -up signaling</td>
</tr>
<tr>
<td>0x010</td>
<td>CONTROL_C</td>
<td>0x5B</td>
<td>Debounce, boot, DVC, and DEF_SUPPLY control</td>
</tr>
<tr>
<td>0x011</td>
<td>CONTROL_D</td>
<td>0x68</td>
<td>Watchdog and LED blink control</td>
</tr>
<tr>
<td>0x012</td>
<td>CONTROL_E</td>
<td>0x04</td>
<td>RTC, ecomode, feedback pins, V_LOCK</td>
</tr>
<tr>
<td>0x013</td>
<td>CONTROL_F</td>
<td>0x00</td>
<td>Watchdog reset, shutdown, and wakeup</td>
</tr>
<tr>
<td>0x014</td>
<td>PD_DIS</td>
<td>0x40</td>
<td>Disable / pause blocks when below the PSS sequencer PD_DIS slot</td>
</tr>
<tr>
<td>0x015</td>
<td>GPIO_0_1</td>
<td>0xDE</td>
<td>GPIO0 and 1 control</td>
</tr>
<tr>
<td>0x016</td>
<td>GPIO_2_3</td>
<td>0xED</td>
<td>GPIO2 and 3 control</td>
</tr>
<tr>
<td>0x017</td>
<td>GPIO_4_5</td>
<td>0xEE</td>
<td>GPIO4 and 5 control</td>
</tr>
<tr>
<td>0x018</td>
<td>GPIO_6_7</td>
<td>0xE9</td>
<td>GPIO6 and 7 control</td>
</tr>
<tr>
<td>0x019</td>
<td>GPIO_8_9</td>
<td>0xF4</td>
<td>GPIO8 and 9 control</td>
</tr>
<tr>
<td>0x01A</td>
<td>GPIO_10_11</td>
<td>0xEE</td>
<td>GPIO10 and 11 control</td>
</tr>
<tr>
<td>0x01B</td>
<td>GPIO_12_13</td>
<td>0xDF</td>
<td>GPIO12 and 13 control</td>
</tr>
<tr>
<td>0x01C</td>
<td>GPIO_14_15</td>
<td>0xEE</td>
<td>GPIO14 and 15 control</td>
</tr>
<tr>
<td>0x01D</td>
<td>GPIO_MODE0_7</td>
<td>0x80</td>
<td>GPIO0 to 7 mode control</td>
</tr>
<tr>
<td>0x01E</td>
<td>GPIO_MODE8_15</td>
<td>0x08</td>
<td>GPIO8 to 15 mode control</td>
</tr>
<tr>
<td>0x01F</td>
<td>SWITCH_CONT</td>
<td>0x30</td>
<td>Rail switches</td>
</tr>
<tr>
<td>0x020</td>
<td>BCORE2_CONT</td>
<td>0x00</td>
<td>BUCKCORE2 control</td>
</tr>
<tr>
<td>0x021</td>
<td>BCORE1_CONT</td>
<td>0x00</td>
<td>BUCKCORE1 control</td>
</tr>
<tr>
<td>0x022</td>
<td>BPRO_CONT</td>
<td>0x00</td>
<td>BUCKPRO control</td>
</tr>
<tr>
<td>0x023</td>
<td>BMEM_CONT</td>
<td>0x00</td>
<td>BUCKMEM control</td>
</tr>
<tr>
<td>0x024</td>
<td>BIO_CONT</td>
<td>0x00</td>
<td>BUCKIO control</td>
</tr>
<tr>
<td>Register Address</td>
<td>Function</td>
<td>Register Value</td>
<td>Register Description</td>
</tr>
<tr>
<td>------------------</td>
<td>------------</td>
<td>----------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>0x025</td>
<td>BPERI_CONT</td>
<td>0x00</td>
<td>BUCKPERI control</td>
</tr>
<tr>
<td>0x026</td>
<td>LDO1_CONT</td>
<td>0x00</td>
<td>LDO1 control</td>
</tr>
<tr>
<td>0x027</td>
<td>LDO2_CONT</td>
<td>0x00</td>
<td>LDO2 control</td>
</tr>
<tr>
<td>0x028</td>
<td>LDO3_CONT</td>
<td>0x40</td>
<td>LDO3 control</td>
</tr>
<tr>
<td>0x029</td>
<td>LDO4_CONT</td>
<td>0x40</td>
<td>LDO4 control</td>
</tr>
<tr>
<td>0x02A</td>
<td>LDO5_CONT</td>
<td>0x00</td>
<td>LDO5 control</td>
</tr>
<tr>
<td>0x02B</td>
<td>LDO6_CONT</td>
<td>0x20</td>
<td>LDO6 control</td>
</tr>
<tr>
<td>0x02C</td>
<td>LDO7_CONT</td>
<td>0x20</td>
<td>LDO7 control</td>
</tr>
<tr>
<td>0x02D</td>
<td>LDO8_CONT</td>
<td>0x00</td>
<td>LDO8 control</td>
</tr>
<tr>
<td>0x02E</td>
<td>LDO9_CONT</td>
<td>0x00</td>
<td>LDO9 control</td>
</tr>
<tr>
<td>0x02F</td>
<td>LDO10_CONT</td>
<td>0x60</td>
<td>LDO10 control</td>
</tr>
<tr>
<td>0x030</td>
<td>LDO11_CONT</td>
<td>0x00</td>
<td>LDO11 control</td>
</tr>
<tr>
<td>0x031</td>
<td>SUPPLIES</td>
<td>0x00</td>
<td>Vibrator output level</td>
</tr>
<tr>
<td>0x032</td>
<td>DVC_1</td>
<td>0x00</td>
<td>Dynamic voltage control</td>
</tr>
<tr>
<td>0x033</td>
<td>DVC_2</td>
<td>0x00</td>
<td>Dynamic voltage control</td>
</tr>
<tr>
<td>0x034</td>
<td>ADC_MAN</td>
<td>0x20</td>
<td>ADC manual and automatic measurement control</td>
</tr>
<tr>
<td>0x035</td>
<td>ADC_CONT</td>
<td>0x01</td>
<td>ADC automatic measurement control</td>
</tr>
<tr>
<td>0x036</td>
<td>VSYS_MON</td>
<td>0xAA</td>
<td></td>
</tr>
<tr>
<td>0x083</td>
<td>ID_2_1</td>
<td>0x00</td>
<td>PSS sequence control</td>
</tr>
<tr>
<td>0x084</td>
<td>ID_4_3</td>
<td>0x99</td>
<td>PSS sequence control</td>
</tr>
<tr>
<td>0x085</td>
<td>ID_6_5</td>
<td>0x91</td>
<td>PSS sequence control</td>
</tr>
<tr>
<td>0x086</td>
<td>ID_8_7</td>
<td>0x99</td>
<td>PSS sequence control</td>
</tr>
<tr>
<td>0x087</td>
<td>ID_10_9</td>
<td>0x99</td>
<td>PSS sequence control</td>
</tr>
<tr>
<td>0x088</td>
<td>ID_12_11</td>
<td>0x07</td>
<td>PSS sequence control</td>
</tr>
<tr>
<td>0x089</td>
<td>ID_14_13</td>
<td>0x88</td>
<td>PSS sequence control</td>
</tr>
<tr>
<td>0x08A</td>
<td>ID_16_15</td>
<td>0x76</td>
<td>PSS sequence control</td>
</tr>
<tr>
<td>0x08B</td>
<td>ID_18_17</td>
<td>0x37</td>
<td>PSS sequence control</td>
</tr>
<tr>
<td>0x08C</td>
<td>ID_20_19</td>
<td>0x00</td>
<td>PSS sequence control</td>
</tr>
<tr>
<td>0x08D</td>
<td>ID_22_21</td>
<td>0x00</td>
<td>PSS sequence control</td>
</tr>
<tr>
<td>0x08E</td>
<td>ID_24_23</td>
<td>0x05</td>
<td>PSS sequence control</td>
</tr>
<tr>
<td>0x08F</td>
<td>ID_26_25</td>
<td>0x00</td>
<td>PSS sequence control</td>
</tr>
<tr>
<td>0x090</td>
<td>ID_28_27</td>
<td>0x00</td>
<td>PSS sequence control</td>
</tr>
<tr>
<td>0x091</td>
<td>ID_30_29</td>
<td>0x04</td>
<td>PSS sequence control</td>
</tr>
<tr>
<td>0x092</td>
<td>ID_32_31</td>
<td>0x10</td>
<td>PSS sequence control</td>
</tr>
<tr>
<td>0x095</td>
<td>SEQ_A</td>
<td>0xBA</td>
<td>PSS sequencer slot end points</td>
</tr>
<tr>
<td>0x096</td>
<td>SEQ_B</td>
<td>0x4D</td>
<td>PSS sequencer slot end points</td>
</tr>
<tr>
<td>0x097</td>
<td>WAIT</td>
<td>0x10</td>
<td>Power sequencer wait cycle</td>
</tr>
<tr>
<td>0x098</td>
<td>EN_32K</td>
<td>0xEA</td>
<td>RTC clocking control</td>
</tr>
</tbody>
</table>
## Register Address | Function | Register Value | Register Description
--- | --- | --- | ---
0x099 | RESET | 0x48 | Reset timer control
0x09A | BUCK_ILIM_A | 0xFF | Buck current limit
0x09B | BUCK_ILIM_B | 0xFF | Buck current limit
0x09C | BUCK_ILIM_C | 0xFF | Buck current limit
0x09D | BCORE2_CFG | 0x81 | BUCKCORE2 control
0x09E | BCORE1_CFG | 0x81 | BUCKCORE1 control
0x09F | BPRO_CFG | 0x81 | BUCKPRO control
0x0A0 | BIO_CFG | 0x81 | BUCKPRO control
0x0A1 | BMEM_CFG | 0x81 | BUCKMEM control
0x0A2 | BPERI_CFG | 0x81 | BUCKPERI control
0x0A3 | VBCORE2_A | 0x50 | BUCKCORE2 voltage A
0x0A4 | VBCORE1_A | 0x50 | BUCKCORE1 voltage A
0x0A5 | VBPRO_A | 0x7F | BUCKPRO voltage A
0x0A6 | VMEM_A | 0x7D | BUCKMEM voltage A
0x0A7 | VBIO_A | 0x7D | BUCKIO voltage A
0x0A8 | VBPERI_A | 0x32 | BUCKPERI voltage A
0x0A9 | VLDO1_A | 0x3C | LDO1 voltage A
0x0AA | VLDO2_A | 0x3C | LDO2 voltage A
0x0AB | VLDO3_A | 0x78 | LDO3 voltage A
0x0AC | VLDO4_A | 0x78 | LDO4 voltage A
0x0AD | VLDO5_A | 0x32 | LDO5 voltage A
0x0AE | VLDO6_A | 0x32 | LDO6 voltage A
0x0AF | VLDO7_A | 0x32 | LDO7 voltage A
0x0B0 | VLDO8_A | 0x14 | LDO8 voltage A
0x0B1 | VLDO9_A | 0x14 | LDO9 voltage A
0x0B2 | VLDO10_A | 0x32 | LDO10 voltage A
0x0B3 | VLDO11_A | 0x22 | LDO11 voltage A
0x0B4 | VBCORE2_B | 0x50 | BUCKCORE2 voltage B
0x0B5 | VBCORE1_B | 0x50 | BUCKCORE1 voltage B
0x0B6 | VBPRO_B | 0x7F | BUCKPRO voltage B
0x0B7 | VMEM_B | 0x7D | BUCKMEM voltage B
0x0B8 | VBIO_B | 0x7D | BUCKIO voltage B
0x0B9 | VBPERI_B | 0x32 | BUCKPERI voltage B
0x0BA | VLDO1_B | 0x3C | LDO1 voltage B
0x0BB | VLDO2_B | 0x3C | LDO2 voltage B
0x0BC | VLDO3_B | 0x2D | LDO3 voltage B
0x0BD | VLDO4_B | 0x2D | LDO4 voltage B
0x0BE | VLDO5_B | 0x32 | LDO5 voltage B
<table>
<thead>
<tr>
<th>Register Address</th>
<th>Function</th>
<th>Register Value</th>
<th>Register Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0BF</td>
<td>VLDO6_B</td>
<td>0x14</td>
<td>LDO6 voltage B</td>
</tr>
<tr>
<td>0x0C0</td>
<td>VLDO7_B</td>
<td>0x14</td>
<td>LDO7 voltage B</td>
</tr>
<tr>
<td>0x0C1</td>
<td>VLDO8_B</td>
<td>0x32</td>
<td>LDO8 voltage B</td>
</tr>
<tr>
<td>0x0C2</td>
<td>VLDO9_B</td>
<td>0x32</td>
<td>LDO9 voltage B</td>
</tr>
<tr>
<td>0x0C3</td>
<td>VLDO10_B</td>
<td>0x14</td>
<td>LDO10 voltage B</td>
</tr>
<tr>
<td>0x0C4</td>
<td>VLDO11_B</td>
<td>0x22</td>
<td>LDO11 voltage B</td>
</tr>
<tr>
<td>0x0C5</td>
<td>BBAT_CONT</td>
<td>0x00</td>
<td>Backup battery charger</td>
</tr>
<tr>
<td>0x0C6</td>
<td>GPO11_LED</td>
<td>0x00</td>
<td>High power GPO PWM</td>
</tr>
<tr>
<td>0x0C7</td>
<td>GPO14_LED</td>
<td>0x00</td>
<td>High power GPO PWM</td>
</tr>
<tr>
<td>0x0C8</td>
<td>GPO15_LED</td>
<td>0x00</td>
<td>High power GPO PWM</td>
</tr>
<tr>
<td>0x0C9</td>
<td>ADC_CFG</td>
<td>0xE0</td>
<td>ADC automatic measurement control</td>
</tr>
<tr>
<td>0x0CA</td>
<td>AUTO1_HIGH</td>
<td>0x00</td>
<td>ADC measurement thresholds</td>
</tr>
<tr>
<td>0x0CB</td>
<td>AUTO1_LOW</td>
<td>0x00</td>
<td>ADC measurement thresholds</td>
</tr>
<tr>
<td>0x0CC</td>
<td>AUTO2_HIGH</td>
<td>0x00</td>
<td>ADC measurement thresholds</td>
</tr>
<tr>
<td>0x0CD</td>
<td>AUTO2_LOW</td>
<td>0x00</td>
<td>ADC measurement thresholds</td>
</tr>
<tr>
<td>0x0CE</td>
<td>AUTO3_HIGH</td>
<td>0x00</td>
<td>ADC measurement thresholds</td>
</tr>
<tr>
<td>0x0CF</td>
<td>AUTO3_LOW</td>
<td>0x00</td>
<td>ADC measurement thresholds</td>
</tr>
<tr>
<td>0x105</td>
<td>INTERFACE</td>
<td>0xB9</td>
<td>Host interfaces</td>
</tr>
<tr>
<td>0x106</td>
<td>CONFIG_A</td>
<td>0x86</td>
<td>Host interfaces and other IOs</td>
</tr>
<tr>
<td>0x107</td>
<td>CONFIG_B</td>
<td>0x7F</td>
<td>VDD_FAULT comparator</td>
</tr>
<tr>
<td>0x108</td>
<td>CONFIG_C</td>
<td>0x50</td>
<td>Buck duty cycle and clock polarity</td>
</tr>
<tr>
<td>0x109</td>
<td>CONFIG_D</td>
<td>0x01</td>
<td></td>
</tr>
<tr>
<td>0x10A</td>
<td>CONFIG_E</td>
<td>0xFF</td>
<td>BUCK and rail switch default settings</td>
</tr>
<tr>
<td>0x10B</td>
<td>CONFIG_F</td>
<td>0x07</td>
<td>LDO default and bypass mode settings</td>
</tr>
<tr>
<td>0x10C</td>
<td>CONFIG_G</td>
<td>0xFF</td>
<td>LDO default settings</td>
</tr>
<tr>
<td>0x10D</td>
<td>CONFIG_H</td>
<td>0xF0</td>
<td></td>
</tr>
<tr>
<td>0x10E</td>
<td>CONFIG_I</td>
<td>0x04</td>
<td></td>
</tr>
<tr>
<td>0x10F</td>
<td>CONFIG_J</td>
<td>0xE3</td>
<td></td>
</tr>
<tr>
<td>0x110</td>
<td>CONFIG_K</td>
<td>0x00</td>
<td>GPIO pull resistors</td>
</tr>
<tr>
<td>0x111</td>
<td>CONFIG_L</td>
<td>0x00</td>
<td>GPIO pull resistors</td>
</tr>
<tr>
<td>0x112</td>
<td>CONFIG_M</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x113</td>
<td>CONFIG_N</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x114</td>
<td>MON_REG_1</td>
<td>0x8A</td>
<td></td>
</tr>
<tr>
<td>0x115</td>
<td>MON_REG_2</td>
<td>0x36</td>
<td></td>
</tr>
<tr>
<td>0x116</td>
<td>MON_REG_3</td>
<td>0x04</td>
<td></td>
</tr>
<tr>
<td>0x117</td>
<td>MON_REG_4</td>
<td>0xCC</td>
<td></td>
</tr>
<tr>
<td>0x121</td>
<td>GP_ID_0</td>
<td>0x02</td>
<td></td>
</tr>
<tr>
<td>0x122</td>
<td>GP_ID_1</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>Register Address</td>
<td>Function</td>
<td>Register Value</td>
<td>Register Description</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------</td>
<td>----------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>0x123</td>
<td>GP_ID_2</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x124</td>
<td>GP_ID_3</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x125</td>
<td>GP_ID_4</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x126</td>
<td>GP_ID_5</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x127</td>
<td>GP_ID_6</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x128</td>
<td>GP_ID_7</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x129</td>
<td>GP_ID_8</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x12A</td>
<td>GP_ID_9</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x12B</td>
<td>GP_ID_10</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x12C</td>
<td>GP_ID_11</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x12D</td>
<td>GP_ID_12</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x12E</td>
<td>GP_ID_13</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x12F</td>
<td>GP_ID_14</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x130</td>
<td>GP_ID_15</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x131</td>
<td>GP_ID_16</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x132</td>
<td>GP_ID_17</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x133</td>
<td>GP_ID_18</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x134</td>
<td>GP_ID_19</td>
<td>0x00</td>
<td></td>
</tr>
<tr>
<td>0x183</td>
<td>CUSTOMER_ID</td>
<td>0x02</td>
<td>Chip ID</td>
</tr>
<tr>
<td>0x184</td>
<td>CONFIG_ID</td>
<td>0x7C</td>
<td>Customer ID</td>
</tr>
</tbody>
</table>
Appendix B Software Implementation

B.1 Set DA9063 register 0x94
For setting PMIC register 0x94, perform a write operation on register 0x94 and overwrite this register with a new value of 0x99. A typical pseudocode call would be:

```
int write(unsigned int reg, unsigned int val);

int err = 0;
if (err = write(0x94, 0x99)) {
    error("Unable to write register 0x94\n");
    return err;
}
```

B.2 Set DA9063 BKUP_TRG bit or DA9063 GP_ID_1 register
Depending upon which method is used (BKUP_TRG or GP_ID_1).
For setting BKUP_TRG, reference to the DA9063 Datasheet describes the register 0x1D (GPIO_MODE0_7). Bit 3 in this register is GPIO3_MODE and has the following settings:
- 0: GPI: debouncing off GPO: Sets output to low level (active low for sequencer control)
- 1: GPI: debouncing on GPO: Sets output to high level (active high for sequencer control)
Performing a read-modify-write operation on GPIO_MODE0_7 and enabling this GPIO3_MODE bit as 1 will set BKUP_TRG=1. A typical pseudocode call would be:

```
int update_bits(unsigned int reg, unsigned int mask, unsigned int val);

int err = 0;
if (err = update_bits(0x1D, 0x08, 0x08)) {
    error("Unable to update bit3 in register 0x1D\n");
    return err;
}
```
For setting GP_ID_1, reference to the DA9063 Datasheet describes the register 0x122 (GP_ID_1). Data from fuse array (OTP). Overwriting this register with a value of 0x01 will set GP_ID_1=1. Typical pseudocode would be:

```
int write(unsigned int reg, unsigned int val);

if (err = write(0x122, 0x01)) {
    error("Unable to write register 0x122\n");
    return err;
}
```

B.3 Read DA9063 BKUP_TRG bit or DA9063 GP_ID_1 register
Depending upon which method is used (BKUP_TRG or GP_ID_1).
For reading BKUP_TRG, reference to the DA9063 Datasheet describes the register 0x1D (GPIO_MODE0_7). Bit 3 in this register is GPIO3_MODE. Performing a read operation on GPIO_MODE0_7 and testing bit 3 will define the setting of BKUP_TRG. A typical pseudocode call would be:

```
int read(unsigned int reg, unsigned int *val);

int val = 0x00;
```
if (err = read(0x1D, &val)) {
    error("Unable to read register 0x1D\n");
    return err;
} else {
    if (val & 0x08)
        print("WARM boot\n");
    else
        print("COLD boot\n");
}

Alternatively, if the SoC GPIO, GP1-08, is connected to BKUP_TRG output pin on DA9063 the SoC can directly determine the BKUP_TRG status by reading GP1-08.

For reading GP_ID_1, reference to the DA9063 Datasheet describes the register 0x122 (GP_ID_1). Data from fuse array (OTP). Reading this register with a non-zero value will define GP_ID_1=1.

Typical pseudocode would be:

```c
int read(unsigned int reg, unsigned int *val);
int val = 0x00;
if (err = read(0x122, &val)) {
    error("Unable to read register 0x122\n");
    return err;
} else {
    if (val)
        print("WARM boot\n");
    else
        print("COLD boot\n");
}
```

### B.4 Set DA9063 PMIC RTC alarm

The operations for setting an RTC alarm in the DA9063 device are as follows.

Convert the contents of the general Linux RTC alarm structure held as values for seconds, minutes, hours, days, months and years into a simple data buffer which can be directly written to the alarm registers of the DA9063.

Reference to the DA9063 Datasheet provides a description of the RTC alarm registers 0x40 (COUNT_S) to 0x45 (COUNT_Y) inclusive – these registers form the length of the data block that should be written with the new alarm time. Typical pseudocode showing a conversion between the usual Linux RTC time structure and the writable DA9063 data buffer would be:

```c
#define MONTHS_TO_DA9063(month) ((month) + 1)
#define YEARS_TO_DA9063(year) ((year) - 100)

struct rtc_time *tm;
/* seconds */
data[0] &= ~0x3F;
data[0] |= tm->tm_sec & 0x3F;
/* minutes */
data[1] &= ~0x3F;
data[1] |= tm->tm_min & 0x3F;
/* hours */
data[2] &= ~0x1F;
data[2] |= tm->tm_hour & 0x1F;
/* days */
data[3] &= ~0x1F;
data[3] |= tm->tm_mday & 0x1F;
```
DA9063-A Power Management for Renesas R-Car M3 Platform

/* months */
data[4] &= ~0x0F;
data[4] |= MONTHS_TO_DA9063(tm->tm_mon) & 0x0F;
/* years */
data[5] &= ~0x3F;
data[5] |= YEARS_TO_DA9063(tm->tm_year) & 0x3F;

Any pending alarms are stopped before the data buffer is written to the DA9063 RTC registers. Typical pseudocode would be:

```c
int update_bits(unsigned int reg, unsigned int mask, unsigned int val);
/* stop alarm */
update_bits(0x4B, 0x40, 0x00);
```

A single bulk write function should be called. The data should be block written to the DA9063 device in a single I²C transfer. In this case the registers 0x40 (COUNT_S) to 0x45 (COUNT_Y) inclusive are written with the alarm data. Writing to the COUNT_Y register in the DA9063 will latch the all registers from COUNT_S and COUNT_Y into the current DA9063 RTC calendar counters. Typical pseudocode would be:

```c
int bulk_write(unsigned int reg, const void *val, size_t val_count);
/* write alarm */
bulk_write(0x46, &data, 6);
```

B.5 Turn on DA9063 PMIC RTC alarm

Reference to the DA9063_datasheet describes the register 0x4B (ALARM_Y). Bit 6 in this register is described as ALARM_ON and has the following settings:

- 0: Alarm function is disabled
- 1: Alarm enabled

The Linux kernel device driver will set this bit using a read-modify-write operation to enable the RTC alarm in the DA9063. At the time of writing, the device driver uses the standard Linux regmap framework function for updating individual bits in a single register. Typical pseudocode would be:

```c
int update_bits(unsigned int reg, unsigned int mask, unsigned int val);
/* start alarm */
update_bits(0x4B, 0x40, 0x40);
```

B.6 Linux device driver PMIC RTC

Reference to the Linux kernel function for setting an RTC alarm in DA9063 is defined in the Opensource Linux device driver, inside the file drivers/rtc/rtc-da9063.c. The prototype for the DA9063 set alarm functions is given as da9063_rtc_set_alarm(). This DA9063 function has existed in the Opensource Linux kernel since Linux mainline v3.16-rc1. The Linux kernel mainline v4.16-rc1 has this function:

```c
static int da9063_rtc_set_alarm(struct device *dev, struct rtc_wkalrm *alrm)
{
    struct da9063_compatible_RTC *rtc = dev_get_drvdata(dev);
    const struct da9063_compatible_RTC_regmap *config = rtc->config;
    u8 data[RTC_DATA_LEN];
    int ret;

da9063_tm_to_data(&alrm->time, data, rtc);
    ret = da9063_rtc_stop_alarm(dev);
    if (ret < 0) {
        dev_err(dev, "Failed to stop alarm: \%d\n", ret);
        return ret;
    }
    update_bits(0x4B, 0x40, 0x40);
    bulk_write(0x46, &data, 6);
    return 0;
}
```
The Linux kernel function for enabling the RTC alarm in the DA9063 is defined in the Opensource Linux device driver, inside the file `drivers/rtc/rtc-da9063.c`. The prototype for this function is given as `da9063_rtc_start_alarm()`. This DA9063 function has existed in the Opensource Linux kernel since Linux mainline v3.16-rc1. The Linux kernel mainline v4.16-rc1 has this function:

```c
static int da9063_rtc_start_alarm(struct device *dev)
{
    struct da9063_compatible_rtc *rtc = dev_get_drvdata(dev);
    const struct da9063_compatible_rtc_regmap *config = rtc->config;

    return regmap_update_bits(rtc->regmap,
        config->rtc_alarm_year_reg,
        config->rtc_alarm_on_mask,
        config->rtc_alarm_on_mask);
}
```

The Linux kernel function for reading the RTC time from the DA9063 is defined in the Opensource Linux device driver, inside the file `drivers/rtc/rtc-da9063.c`. The prototype for this function is given as `da9063_rtc_read_time()`. This DA9063 function has existed in the Opensource Linux kernel since Linux mainline v3.16-rc1. The Linux kernel mainline v4.16-rc1 has this function:

```c
static int da9063_rtc_read_time(struct device *dev, struct rtc_time *tm)
{
    struct da9063_compatible_rtc *rtc = dev_get_drvdata(dev);
    const struct da9063_compatible_rtc_regmap *config = rtc->config;

    unsigned long tm_secs;
    unsigned long al_secs;
    u8 data[RTC_DATA_LEN];
    int ret;

    ret = regmap_bulk_read(rtc->regmap,
        config->rtc_count_secs_reg,
        data, RTC_DATA_LEN);
    if (ret < 0) {
        dev_err(dev, "Failed to read RTC time data: %d\n", ret);
        return ret;
    }

    da9063_data_to_tm(data, &rtc->alarm_time, rtc);
    if (alrm->enabled) {
        ret = da9063_rtc_start_alarm(dev);
        if (ret < 0) {
            dev_err(dev, "Failed to start alarm: %d\n", ret);
            return ret;
        }
    }

    return ret;
}
```
if (!((data[RTC_SEC] & config->rtc_ready_to_read_mask))) {
    dev_dbg(dev, "RTC not yet ready to be read by the host\n");
    return -EINVAL;
}

da9063_data_to_tm(data, tm, rtc);

rtc_tm_to_time(tm, &tm_secs);
rtc_tm_to_time(&rtc->alarm_time, &al_secs);

/* handle the rtc synchronisation delay */
if (rtc->rtc_sync == true && al_secs - tm_secs == 1)
    memcpy(tm, &rtc->alarm_time, sizeof(struct rtc_time));
else
    rtc->rtc_sync = false;

return rtc_valid_tm(tm);
# Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>19-Jan-2018</td>
<td>Update to memory retention mode</td>
</tr>
<tr>
<td>3.0</td>
<td>12-Feb-2018</td>
<td>Software implementation of memory retention mode</td>
</tr>
</tbody>
</table>
DA9063-A Power Management for Renesas R-Car M3 Platform

Status Definitions

<table>
<thead>
<tr>
<th>Status</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAFT</td>
<td>The content of this document is under review and subject to formal approval, which may result in modifications or additions.</td>
</tr>
<tr>
<td>APPROVED or unmarked</td>
<td>The content of this document has been approved for publication.</td>
</tr>
</tbody>
</table>

Disclaimer

Information in this document is believed to be accurate and reliable. However, Dialog Semiconductor does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information. Dialog Semiconductor furthermore takes no responsibility whatsoever for the content in this document if provided by any information source outside of Dialog Semiconductor.

Dialog Semiconductor reserves the right to change without notice the information published in this document, including without limitation the specification and the design of the related semiconductor products, software and applications.

Applications, software, and semiconductor products described in this document are for illustrative purposes only. Dialog Semiconductor makes no representation or warranty that such applications, software and semiconductor products will be suitable for the specified use without further testing or modification. Unless otherwise agreed in writing, such testing or modification is the sole responsibility of the customer and Dialog Semiconductor excludes all liability in this respect.

Customer notes that nothing in this document may be construed as a license for customer to use the Dialog Semiconductor products, software and applications referred to in this document. Such license must be separately sought by customer with Dialog Semiconductor.

All use of Dialog Semiconductor products, software and applications referred to in this document are subject to Dialog Semiconductor’s Standard Terms and Conditions of Sale, available on the company website (www.dialog-semiconductor.com) unless otherwise stated.

Dialog and the Dialog logo are trademarks of Dialog Semiconductor plc or its subsidiaries. All other product or service names are the property of their respective owners.

© 2018 Dialog Semiconductor. All rights reserved.

Contacting Dialog Semiconductor

United Kingdom (Headquarters)
Dialog Semiconductor (UK) LTD
Phone: +44 1793 757700

Germany
Dialog Semiconductor GmbH
Phone: +49 7021 805-0

The Netherlands
Dialog Semiconductor B.V.
Phone: +31 73 640 8822
Email: enquiry@diasemi.com
Web site: www.dialog-semiconductor.com

North America
Dialog Semiconductor Inc.
Phone: +1 408 845 8500

Japan
Dialog Semiconductor K. K.
Phone: +81 3 5769 5100

Taiwan
Dialog Semiconductor Taiwan
Phone: +886 281 786 222

Web site: www.dialog-semiconductor.com

Hong Kong
Dialog Semiconductor Hong Kong
Phone: +852 2607 4271

Korea
Dialog Semiconductor Korea
Phone: +82 2 3469 8200

China (Shenzhen)
Dialog Semiconductor China
Phone: +86 755 2981 3669

China (Shanghai)
Dialog Semiconductor China
Phone: +86 21 5424 9058

© 2018 Dialog Semiconductor. All rights reserved.