Technical Article

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Improve Time to Market Easily by Customizing Your Power Management Integrated Circuit

Franco Contadini, Staff Engineer

Abstract

Power management integrated circuits (PMICs) are attractive to reduce the complexity and size of end applications. However, customization of the power-up setting is needed due to the default start-up sequence and output voltages not being aligned with the application's requirements. Most often a nonvolatile memory (NVM) is not available to store these settings. A low power microcontroller is a good solution with the ability and tools to program the PMIC control registers at power up without requiring firmware development. This article will address the challenge of achieving integration using a toolchain that simplifies PMIC customization and minimizes development time since no firmware development is required.

Introduction

Handheld devices, smart cameras, and other portable devices require highly integrated and powerful devices to reduce size and cost. The MAX77714 is an example of a highly integrated PMIC including several buck converters, low dropout (LDO) regulators, general purpose input outputs (GPIOs), and a real-time clock (RTC) that can be used to generate all supply rails needed from the application.

A flexible sequencer allows customized power-up and power-down sequences for target applications.

This PMIC is available with a couple of V_{OUT}/sequencer options (defined from the full part number) that don't match the application requirement. To increase flexibility, an alternative solution could use a low power microcontroller that wakes up at power on, programs device registers, and then sleeps to save power.

A medium-volume application requiring high integration needs a simple solution to use off-the-shelf devices, customizing its operation for their needs.

A hardware designer's dream is to have a tool that automatically generates firmware code, allowing them to avoid involving software resources in the early development stage, while still optimizing resource usage.

MAX28200

The MAX28200 is a low power, low size (1.7 mm × 1.8 mm) RISC microcontroller with several peripherals. For this application, we use two GPIO pins as a bit bang I^2C main.

The supply voltage is up to 3.6 V, so an always-on LDO regulator is used to power the microcontroller if the input voltage exceeds 3.6 V. The device can be programmed using the I^2C node interface.

MAX77714

The MAX77714 (Figure 1) is a highly integrated PMIC that includes four buck converters, nine LDO regulators, eight GPIOs, RTC, and a flexible power sequencer (FPS). Bucks and LDO regulators power up with default sequence and voltages, following voltage rails, can be modified using an I^2C embedded interface. However, in most cases, it is important that the default sequence and voltage match the requirement of a device connected to the PMIC that could be damaged if voltages are above its maximum rating.

Power-up and power-down sequences are also important since devices such as field programmable gate arrays (FPGAs) require that voltages are enabled and disabled in the correct sequence to operate correctly. Figure 2 shows the default power-up sequence. There are eight time slots, and voltages can be enabled in any of these time slots. The device can be powered with a voltage of up to 5.5 V.

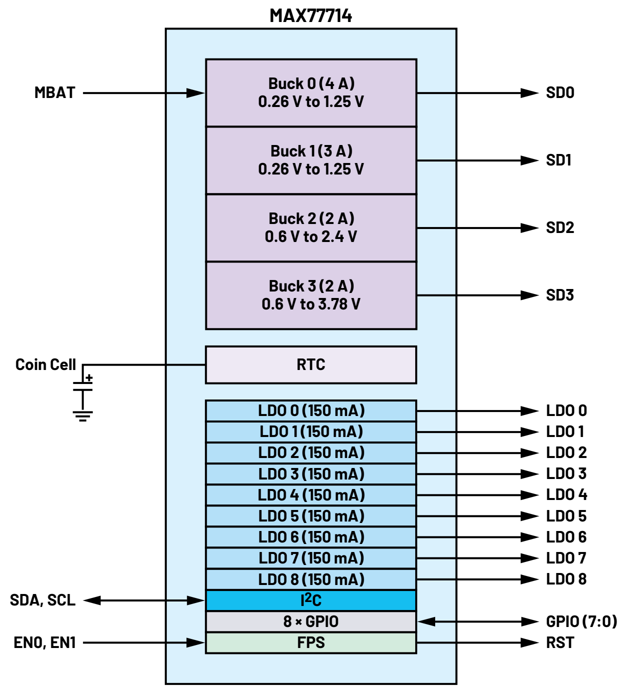


Figure 1. A MAX77714 simplified block diagram.

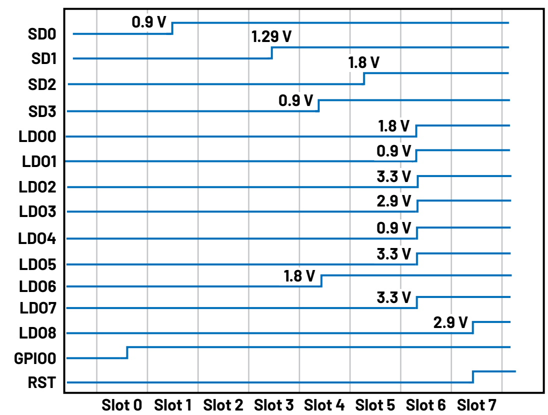


Figure 2. A MAX77714EWC+ default power-up sequence.

Device Interfacing

P0.0 and P0.1 are used to generate I²C transactions toward PMIC, P0.2 monitors the status of GPIO4, and P0.3 selects, using an SPDT switch, which is the main of the I²C (the MAX28200 or host micro). The analog switch is optional, needed only if the host micro needs to control the PMIC following power-up (see Figure 3).

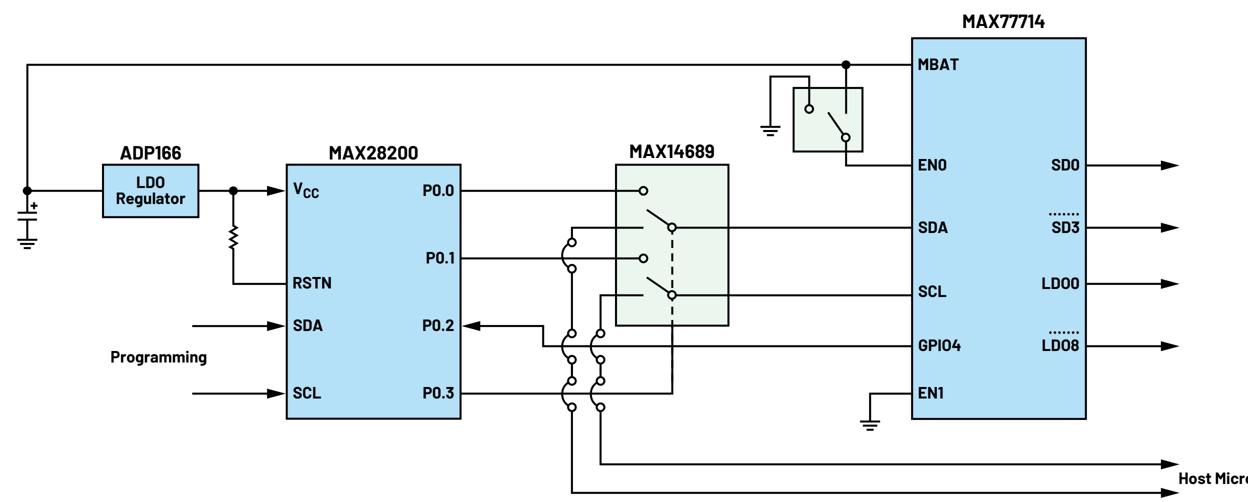


Figure 3. The MAX28200 and MAX77714 interfacing with the MAX14689 and the ADP166.

At power-on reset (POR) as shown in Figure 4:

1. P0.3 is asserted low to select the device as the I²C main for the PMIC.

2. The following desired data registers are written.

3. P0.3 is asserted high to select the host micro as the I²C main and the device enters sleep mode.

4. P0.2, which is connected to GPIO4, is used to wake up the device and reprogram registers in case of power down.

Note that GPIO4 must be asserted high at the end of the configuration.

Once the PMIC is enabled using EN0 and EN1 signals, it will power up with a new sequence/voltage configuration.

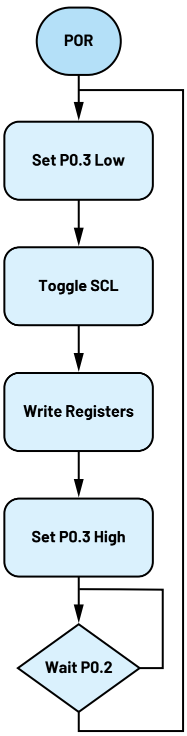


Figure 4. A MAX28200 program flow.

The MAX77714 power-up and power-down sequence has been defined using the MAX77714 GUI available here.

Register values can be exported in an Excel file using the GUI Import/Export tab.

Before showing how to use the GUI EVKIT it is necessary to know which registers are important for customized startup.

MAX77714 Register Details

Bucks

Each buck instance has three associated registers (SDx_CNFG1, SDx_CNFG2, and SDx_CNFG3) that define start-up voltage, ramp rate, and operation mode.

CNFG1 set output voltage, 7 bits to 8 bits field as follows:

VSD0 = 0.26 V + ((SD0VOUT[6:0] - 1) × 10 mV) →range from 0.26 V to 1.56 V

VSD1 = 0.26 V + ((SD1VOUT[6:0] - 1) × 10 mV) →range from 0.26 V to 1.56 V

VSD2 = 0.6 V + (SD2VOUT[7:0] × 6.25 mV) →range from 0.6 V to 2.194 V

VSD3 = 0.6 V + (SD3VOUT[7:0] × 12.5 mV) →range from 0.6 V to 3.7875 V

CNFG2 and CNFG3 registers are used to select ramp rate (2.5 mV/µs or 10 mV/µs), enable/disable active discharge, enable/disable skip mode, select response time, and overvoltage threshold.

LDO Regulators

Each LDO regulator instance has two associated registers (LDO_CNFG1_Lx, and LDO_CNFG2_Lx).

CNFG1 set output voltage:

0.8 V to 2.375 V for LDO0 and LDO1.

VLDO = 0.8 V + (VOUT_LDO[5:0] × 25 mV)

0.8 V to 3.95 V for LDO2, LDO3, LDO5, LDO6, LDO7, LDO8

VLDO = 0.8 V + (VOUT_LDO[5:0] × 50 mV)

0.4 V to 1.275 V for LDO4

VLDO = 0.4 V + (VOUT_LDO[5:0] × 12.5 mV)

CNFG2 enable/disable overvoltage clamp, enable/disable auto low power mode, enable/disable active discharge, select fast/slow startup.

GPIOs

Each GPIO instance has one associated register CNFG_GPIOx, which selects the direction GPO/GPI, output type (open drain/push-pull), and output drive level in case of GPO, and input drive level, interrupt behavior, and debounce configuration in case of GPI.

FPS

A sequence is defined from xxxFPS registers, LDO0FPS to LDO8FPS for linear regulators, SD0FPS to SD3FPS for buck regulators, GPIO0FPS, GPIO1FPS, GPIO2FPS, GPIO7FPS for GPIOs, and RSTIOFPS for reset.

FPS[2:0] set power-down slot (SLOT0 to SLOT7)

FPS[5:3] set power-up slot (SLOT0 to SLOT7)

FPS[7:6] set assigned sequencer (FPS0, FPS1)

FPS0 is enabled from EN0 and FPS1 is enabled from EN1

The size of the power-up and power-down slot can be customized using the MSTR_PU_PD register:

MSTR_PU_PD[2:0] for power-down slot and MSTR_PU_PD[6:4] for power-up slot

Possible step sizes are: 31 µs, 63 µs, 127 µs, 253 µs, 508 µs, 984 µs, 1936 µs, 3904 µs, default value for MAX77714EWC+ is 3904 µs.

GUI in Action

In Figure 5, using the GUI output voltage of SD0 has been modified to 0.7 V.

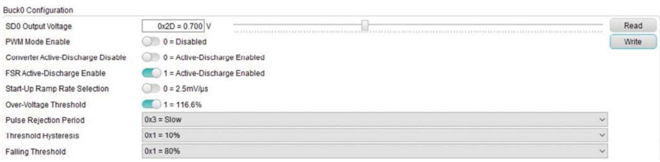


Figure 5. GUI used to modify SDO voltage.

SDO power-up and power-down sequence has been modified to Slot 2 (Figure 6).

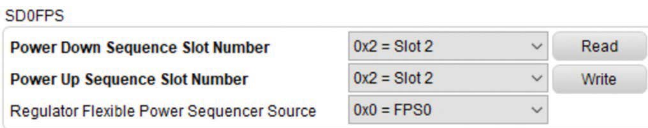


Figure 6. GUI used to modify SDO slot number.

GPIO4 has been configured as a push-pull output, with a drive level high (Figure 7).

Figure 7. GUI used to configure GPIO4.

Finally, modified register values have been exported to Excel files using the Import/Export Tab (Figure 8).

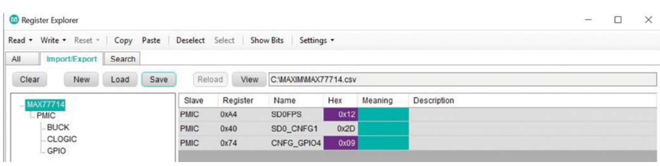


Figure 8. GUI used to export register values.

Below resulting .csv file (Figure 9).

Slave	Address	Name	Hex
0x38	0xA4	SD0FPS	0x12
0x38	0x40	SD0_CNFG1	0x2D
0x38	0x74	CNFG_GPIO4	0x09

Figure 9. The format of the MAX77714 .csv file.

Note that if additional devices connected to the I^2C have to be programmed during power-on you can append additional commands to the .csv file using the same format.

Creating I^2C Main Firmware from a .csv File

The MAX28200 GUI is available here and can be used to program the device with the following steps:

1. Build a .hex file from a .csv file (Figure 10).

2. Programming device (Figure 11 and Figure 12).

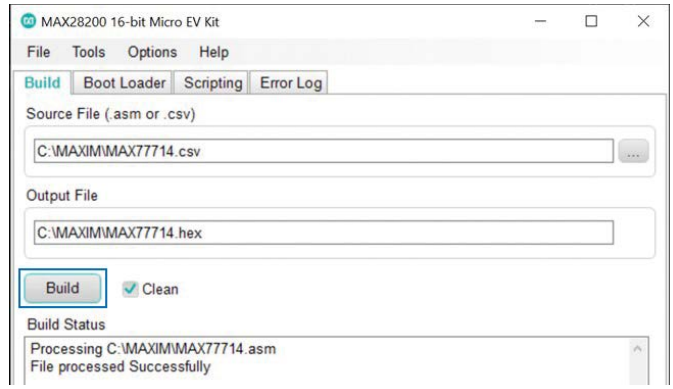


Figure 10. Build a .hex file from the original .csv file.

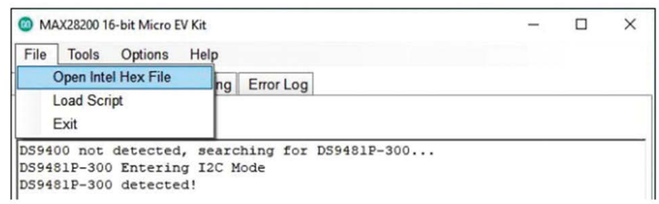


Figure 11. A programming device with the selected script.

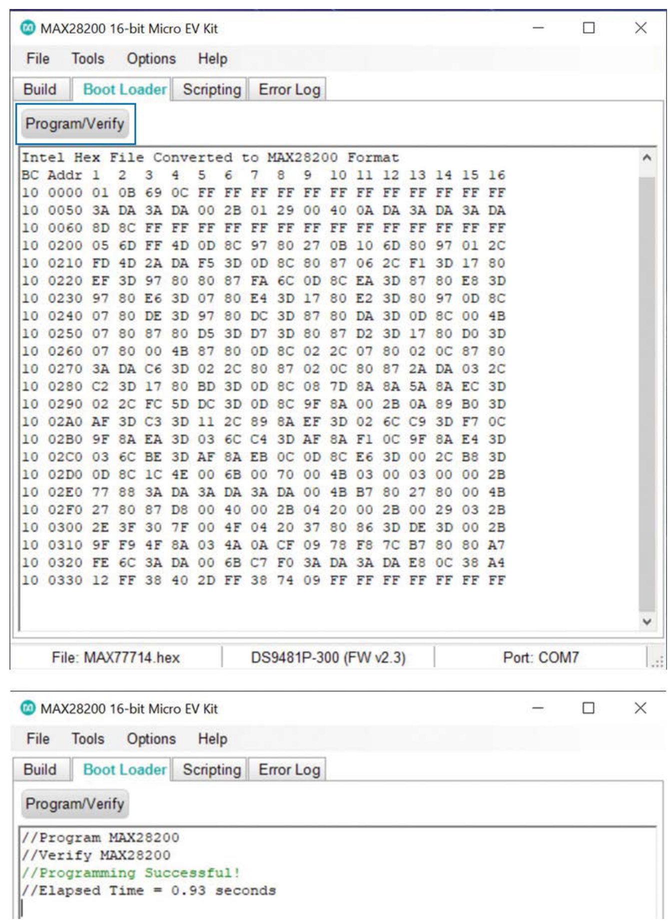
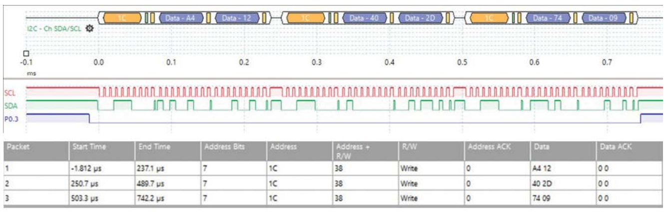


Figure 12. A programming device with a data pattern.

Test of the Proposed Solution

Once programmed with the MAX77714.hex at power-up, P0.3 is asserted low to connect the MAX77714 I²C lines to the MAX28200, and the SDO and GPIO4 registers are programmed with new values (Figure 13).



Packet	Start Time	End Time	Address Bits	Address	Address + R/W	R/W	Address ACK	Data	Data ACK
1	-1.812 µs	237.1 µs	7	1C	38	Write	0	A4 12	0 0
2	250.7 µs	489.7 µs	7	1C	38	Write	0	40 2D	0 0
3	503.3 µs	742.2 µs	7	1C	38	Write	0	74 09	0 0

Figure 13. The MAX77714 SDO and GPIO4 registers are programmed with new values.

Now when the MAX77714 is enabled, asserting ENO high, SDO powers up on Slot 2 at 0.7 V as programmed. Figure 14 shows power-up before register writing (a) and after register writing (b).

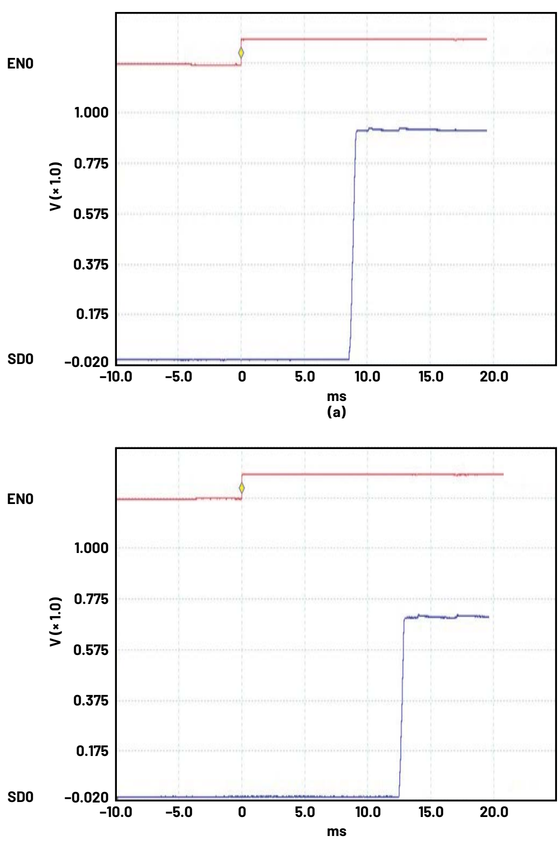


Figure 14. SDO power-up before and after registering writing. (a) Power-up before register writing. (b) Power-up after register writing.

Conclusion

The MAX28200 is an attractive microcontroller solution to provide power-up settings for a PMIC. The tools are available to automate this process without a designer writing a single line of code. This is a very attractive solution for hardware designers not familiar with software that can use a microcontroller as a black box without involving a firmware designer in this process. A designer can match their integration target using the off-the-shelf device customized for a specific application.

About the Author

Franco Contadini has over 35 years' experience in the electronic industry. After 10 years as a board and ASIC designer, he became a field applications engineer supporting industrial, telecom, and medical customers and focusing on power and battery management, signal chains, cryptographic systems, and microcontrollers. Franco has authored several application notes and articles on signal chains and power. He studied electronics at ITIS of Genova.

Engage with the ADI technology experts in our online support community. Ask your tough design questions, browse FAQs, or join a conversation.

ADI EngineerZone™
SUPPORT COMMUNITY

Visit ez.analog.com

©2024 Analog Devices, Inc. All rights reserved. Trademarks and registered trademarks are the property of their respective owners.