Keywords: monitoring, sequencing, programming, SMBus, JTAG, power supplies, multivoltage, multi-voltage

APPLICATION NOTE 4715

In-Circuit Programming for the MAX16065–MAX16068 and MAX16070/MAX16071 Flash-Programmable System Managers

By: Eric Schlaepfer, Applications Engineer
Aug 11, 2010

Abstract: The MAX16065–MAX16068 and MAX16070/MAX16071 microprocessor supervisors can be programmed after they are soldered to the application circuit board. This means that only the unprogrammed device needs to be stocked, and that the latest version of the configuration information can be written to the device during manufacturing test. A few simple measures ensure that the application circuit allows the programming hardware to share the SMBus™ or JTAG bus lines and provides power for the device during programming. This application note provides the programming algorithm for both the SMBus and the JTAG buses.

Introduction

The MAX16065/MAX16066, MAX16067, MAX16068 and MAX16070/MAX16071 products sequence and monitor power supplies in complex multivoltage systems. Power supplies are monitored by digital comparators and sequenced with a programmable state machine. **Table 1** summarizes the devices in these product families.

Table 1. Summary of System Managers

Part	Monitoring Inputs	Sequencing Outputs
MAX16065	12	12
MAX16066	10	8
MAX16067	6	6
MAX16070	12	—
MAX16071	8	—
MAX16068	6	—

The devices include an SMBus-compatible interface and a JTAG interface, through which all of the device registers can be accessed and the internal configuration flash memory programmed. The parts are all in-circuit programmable, i.e., they can be programmed after being soldered to the application circuit board as long as a few simple guidelines are followed. In-circuit programming also means that only the unprogrammed device needs to be stocked, and that the latest version of the configuration information

can be written to the device during manufacturing test.

Providing Power

The devices have a supply voltage range of 2.8V to 14V. Typical applications connect V_{CC} to a 12V intermediate bus voltage or a 3.3V auxiliary supply.

For Maxim system managers with sequencing outputs, it is possible to program a device with a partially powered board. For example, the 3.3V auxiliary voltage could be applied without any other supplies, or the 12V intermediate bus voltage could be applied. All downstream power supplies should be off since the device is not programmed at that point. Another option is to use a commonly available dual diode to allow power to be supplied from a programming connector. Because of the voltage drop caused by the diode, this approach works best when the device is powered from a 12V bus.

If diode OR-ing is not used and the system manager is to be programmed with power applied to the board, careful attention must be paid to the state of the sequencing outputs. This is to prevent any downstream power supplies from turning on prematurely.

When unprogrammed, the system manager has high-impedance outputs. Power supplies with active-high enable inputs should have pulldown resistors; power supplies with active-low enable inputs should have pullup resistors. The sequencing outputs can be configured as either push-pull or open-drain. Because the open-drain configuration requires an external pullup resistor, it should be used with active-low enable inputs only.

It is not advisable to connect a system manager in a JTAG chain, as power may not be applied to upstream devices in the chain, effectively cutting off access to the device. Rather, a JTAG multiplexer or a dedicated JTAG port is preferred. A system manager that has no control over power-supply sequencing can safely be included in the JTAG chain.

Sharing the Bus

A potential problem occurs when an IC needs to communicate with the system manager during normal operation. One example is when a system supervisory microprocessor needs to access the ADC readings of a MAX16065. When the board is unpowered or partially powered and the MAX16065 is being programmed, other devices connected to the SMBus or JTAG bus could interfere. The easiest solution is to program the MAX16065 through the JTAG interface and connect the supervisory microprocessor to the SMBus interface. If the microprocessor supports a true open-drain SMBus bus I/O (that is, pins that lack the ESD diode to V_{CC}) and if the pullup resistors are large enough, it is possible to share the SMBus interface for both programming and normal operation. If the microprocessor's SMBus lines are not open-drain, the ESD diodes will clamp the bus lines and interfere with programming.

If the system microprocessor does not have true open-drain SMBus lines, a circuit like the one in **Figure 1** can be used to automatically switch between the microprocessor and the programming SMBus interface.

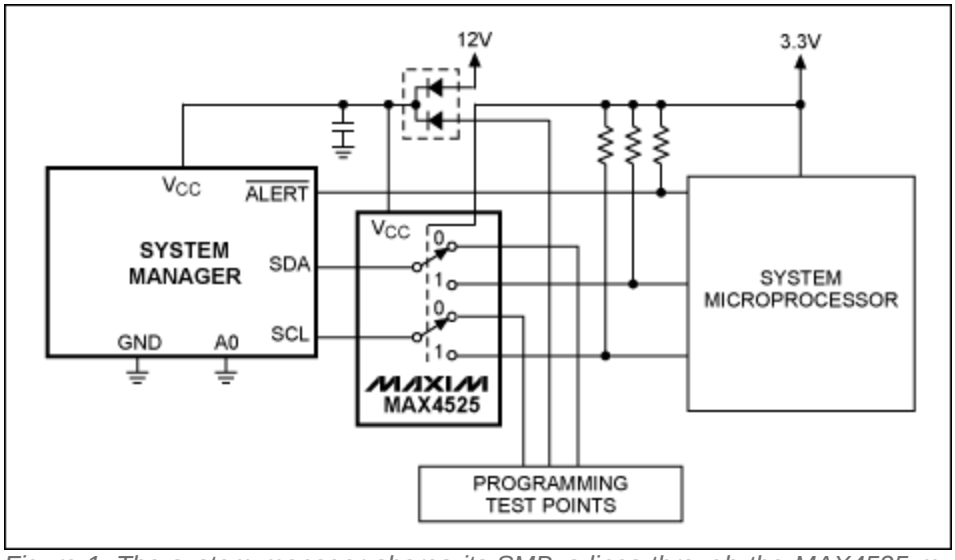

Figure 1. The system manager shares its SMBus lines through the MAX4525 multiplexer/switch.

The MAX4525 multiplexer in Figure 1 switches between the SMBus lines connected to the system microprocessor and the SMBus lines connected to the programming test points. The switch is controlled by the V_{CC} of the system microprocessor. If V_{CC} is not applied but 12V is, the switch connects the SMBus interface to the programming test points. Once V_{CC} is applied, the switch connects the SMBus lines to the system microprocessor.

Application Circuit Examples

The following figures show three different application circuits designed for in-circuit programming.

Powered from a 12V Intermediate Bus and Programmed through the SMBus Interface

The **Figure 2** circuit powers the MAX16065 from the 12V intermediate bus, which is monitored by the analog enable line, EN. When the 12V bus rises above the threshold set by the resistive divider on EN, the MAX16065 attempts sequencing if it has been programmed to do so. An unprogrammed MAX16065 does nothing and the sequencing outputs remain at a high-impedance level.

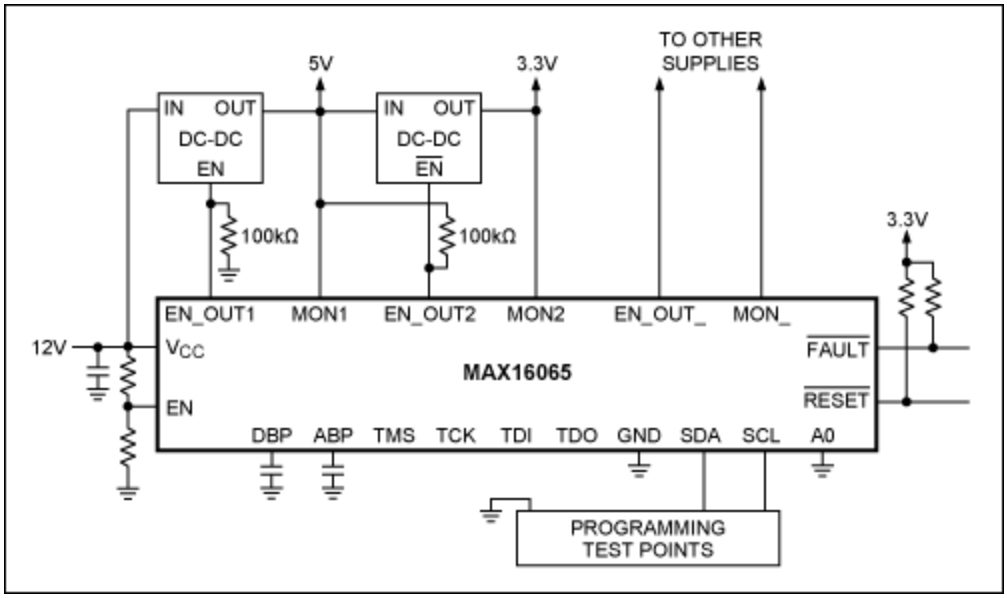

Figure 2. The MAX16065 is powered from the 12V intermediate bus and programmed through the SMBus interface.

One power supply uses an active-high, push-pull enable signal, and the other uses an active-low, open-drain enable signal. Appropriate pullup and pulldown resistors prevent these power supplies from turning on while the unprogrammed outputs are at a high-impedance level.

It is important to note that push-pull outputs cannot be pulled up above V_{DBP}; open-drain outputs cannot be pulled up above 6V. The SMBus connections are brought out to programming test points; appropriate pullup resistors must be provided by the programming hardware. The circuit is simple because no other device on the circuit board needs to connect to the SMBus lines.

Powered from a 3.3V Auxiliary Supply and Programmed through a JTAG Multiplexer

In **Figure 3** the MAX16067 derives power from a 3.3V auxiliary supply. The JTAG connections are shared with other devices using a DS26900 JTAG multiplexer that is also powered from the 3.3V auxiliary supply. Programming can be accomplished by providing 3.3V without needing to power up the 12V intermediate bus.

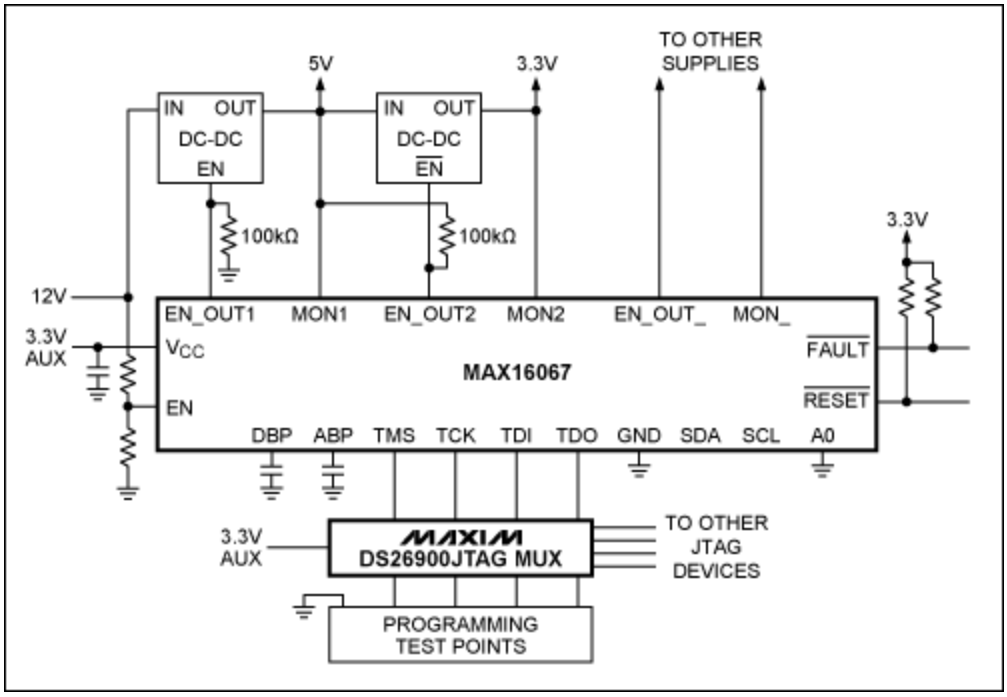

Figure 3. The MAX16067 is powered from a 3.3V auxiliary supply and programmed through the DS26900 JTAG multiplexer.

Powered from a 12V Intermediate Bus and Programmed through JTAG

Figure 4 shows the MAX16066 powered with a diode OR-ed connection to the 12V intermediate bus, so that power can safely be applied without powering on any downstream power supplies. The JTAG and power connections are brought out to programming test points.

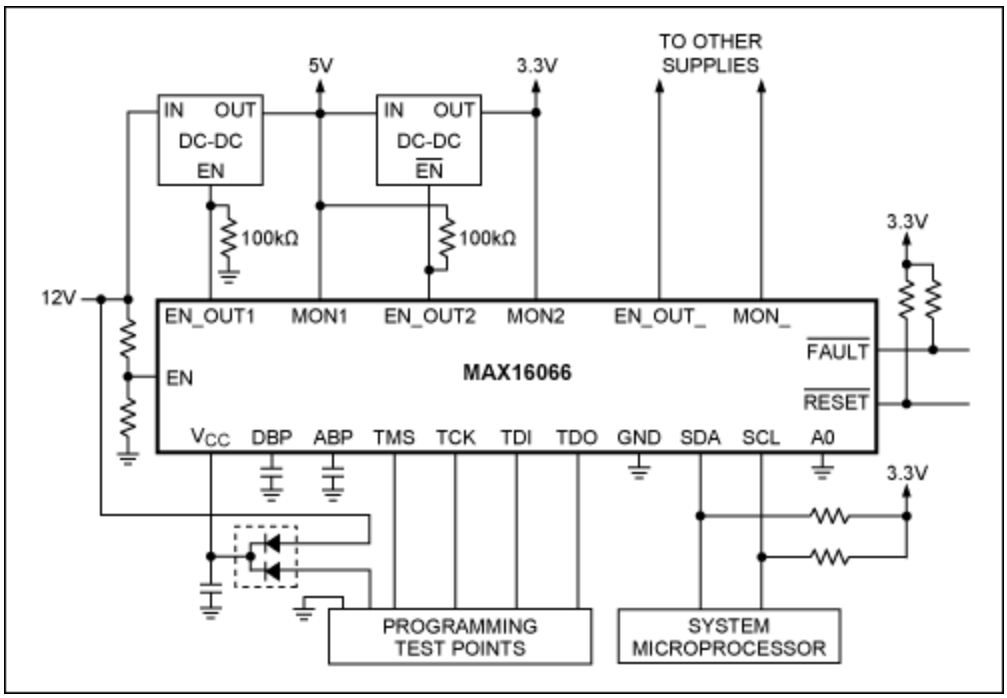

Figure 4. The MAX16066 is powered from a 12V intermediate bus and programmed through JTAG.

Programming Algorithm

The Maxim power-management devices have built-in flash memory that stores the device configuration parameters. When power is applied, the contents of the flash are transferred to the RAM registers. Both RAM and flash memory are accessible from the JTAG and SMBus interfaces. To correctly program a device, the desired parameters must be programmed to the flash memory. See the memory map in **Figure 5**.

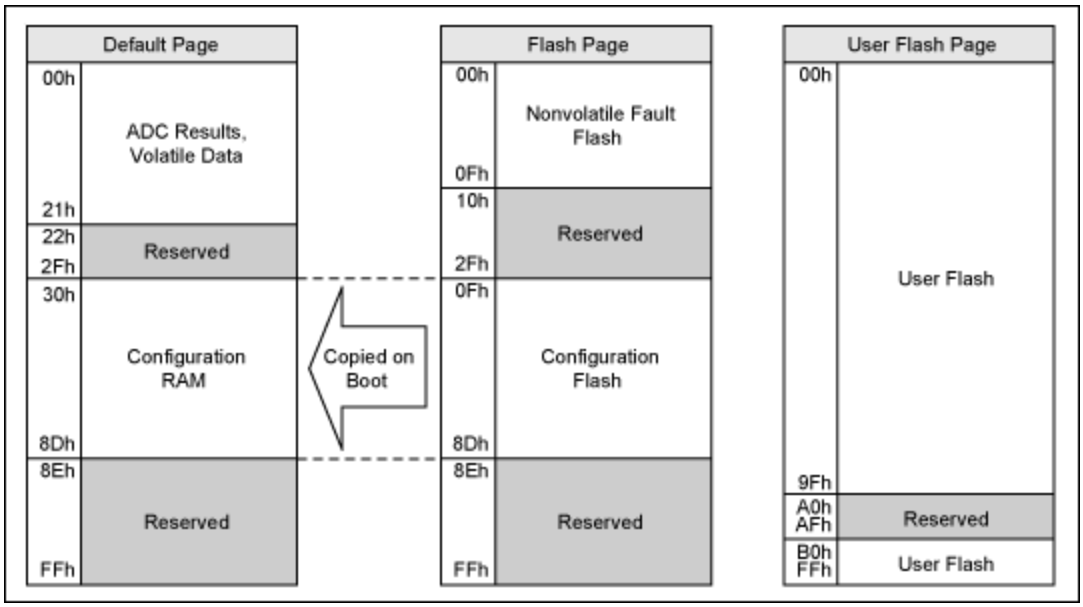

Figure 5. System manager memory map.

Configuration Files

The MAX16065EVKIT software provides two types of configuration files. One is a standard Intel® HEX file produced by selecting **File → Save As**. This file can be used for SMBus programming. The second file is produced by selecting **File → Export to SVF File**. This file is in the serial vector format (SVF) used by third-party JTAG tools and in-circuit PCB testers for JTAG programming.

Describing the Intel HEX format is beyond the scope of this document, but details can be found at: http://en.wikipedia.org/wiki/Intel_HEX.

The SVF file format is described in more detail at: www.asset-intertech.com/support/svf.pdf.

SMBus Programming Procedure

To program the flash configuration memory, first make sure that the memory lock bits in register r8Ch (configuration memory page, not flash page) are zero. Write 00 to the register to clear all the memory lock bits. To write to the flash memory, send the appropriate command to enter the flash memory page, load the starting address (which must be aligned to 8-byte boundaries), and send a series of block-write commands. The flash memory is programmed in blocks of 8 bytes. See the MAX16065 data sheet for details on the SMBus protocols.

Pseudocode for a typical flash memory programming process follows.

```
FlashPageOff()
```

```
UserFlashPageOff()
If ReadRegister(8Ch) != 0 Then WriteRegister(8Ch, 00h)
FlashPageOn()

Loop Address from 30h to 8Dh
  SetAddress(Address)           // Load address
  WriteBlock(Data, 08h)         // Write a block of 8 bytes
  Wait(150 milliseconds)        // Wait for programming
SetAddress(Address)
ReadBlock(DataRead, 08h)        // Read back data block
If DataRead != Data Then
  Fail
Else
  Address = Address + 08h       // Advance to next block
End Loop
FlashPageOff()                  // Return to default page
Success
```

To write to the user flash memory, the same process can be used, but with an address range of 00h to FFh (excluding one reserved block at A0h–AFh).

For the flash write operation to succeed, it is important to write blocks of 8 bytes aligned at 8-byte boundaries. (The three LSBs of the address must be zero.)

JTAG Programming Procedure

Use standard third-party JTAG tools, the MAX16065–MAX16068 and MAX16070/MAX16071 BSDL file, and an SVF data file generated by the EV kit software to program the device with either a JTAG programming cable or an in-circuit PCB tester. The BSDL files are available for download.

Note that the SVF files generated by the EV kit software test the IDCODE register. The following code snippet is from a SVF file generated by the MAX16065 EV kit software:

```
ENDDR IDLE;
ENDIR IDLE;
SIR 5 TDI(00) TDO(01);
SDR 32 TDI(00000000) TDO(18001197);
```

The TDO(18001197) statement verifies the complete IDCODE statement, including the device revision code. This statement will fail if a device is used with a different revision code. To make the SVF file ignore the revision code field of the IDCODE register, use the following line instead:

```
SDR 32 TDI(00000000) TDO(18001197) MASK(0FFFFFFF);
```

The mask statement ignores the 4-bit revision code field (see **Table 2**). For the IDCODE corresponding to each system manager part number, refer to **Table 3**.

Table 2. IDCODE Register Bitmap

	Revision (4 Bits)	Part Number (16 Bits)	Manufacturer (11 Bits)	
Binary	0001	1000 0000 0000 0001	00011001011	1
Hex	1	8001	197	

Table 3. Part IDCODE Registers

Part	IDCODE
MAX16065	_8001197
MAX16066	_8002197
MAX16067	_8003197

MAX16070	_8005197
MAX16071	_8006197
MAX16068	_8004197

Note: The underscore represents 4 bits of revision code. This is subject to change—refer to the device data sheet.

More Information
For Technical Support: http://www.maximintegrated.com/support
For Samples: http://www.maximintegrated.com/samples
Other Questions and Comments: http://www.maximintegrated.com/contact

---

Application Note 4715: http://www.maximintegrated.com/an4715
APPLICATION NOTE 4715, AN4715, AN 4715, APP4715, Appnote4715, Appnote 4715