



Maxim > Design Support > Technical Documents > Application Notes > Microprocessor Supervisor Circuits > APP 4283

Keywords: supervisory, multivoltage, monitor, eight, temperature sensors, current monitor, thresholds, SMBus, I2C, JTAG, EEPROM, configurable, nonvolatile fault memory, ADC, servers, telecom, networking equipment

APPLICATION NOTE 4283

In-Circuit Programming of the MAX16031/MAX16032 EEPROM-Based System Monitors

By: Eric Schlaepfer, Applications Engineer

Nov 07, 2008

Abstract: The MAX16031/MAX16032 system monitors can be programmed after being soldered to the application circuit board. This means that only unprogrammed devices need 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, which are detailed in this application note, ensure that the application circuit allows the programming hardware to share the I²C or JTAG bus line and provides power for the device during programming. The programming algorithm is also provided for both the I²C bus and the JTAG bus.

The MAX16031/MAX16032 EEPROM-based system managers are system supervisory devices that monitor eight power supply voltages, three temperature sensors, and one current. Each parameter is compared to four different thresholds, and several fault outputs can be configured to assert under a variety of conditions.

These monitors include an SMBus™-compatible I²C interface and a JTAG interface, both of which can access all of the device registers and program the internal configuration EEPROM. The MAX16031/MAX16032 are in-circuit programmable, as long as a few simple guidelines are followed.

Providing Power

The MAX16031/MAX16032 have a supply voltage range spanning 3V to 14V. Some applications connect V_{CC} to a 12V intermediate bus voltage, while others connect V_{CC} to a 3.3V auxiliary supply.

It is possible to program these devices 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 while all downstream power supplies are forced off to prevent power from reaching any other circuits. Another option is to use a commonly available dual diode to allow power to be supplied from the programming connector. Due to the voltage drop caused by the diode, this approach works best when the MAX16031/MAX16032 are powered from a 12V bus.

Sharing the Bus

A potential problem occurs when some device, other than the microprocessor (μP), needs to communicate with the MAX16031/MAX16032 during normal operation. One example is when a system supervisory μP needs to access the ADC readings of the MAX16031/MAX16032. When the board is unpowered or partially powered and the MAX16031/MAX16032 is being programmed, other devices connected to the I²C or JTAG bus could interfere. The easiest solution is to program the MAX16031/MAX16032 through the JTAG interface and connect the supervisory μP to the I²C interface. If the μP supports true open-drain I²C bus I/O (that is, pins that lack the ESD diode to V_{CC}) and the pullup resistors are large enough, it can be possible to share the

I²C bus for both programming and normal operation. If the μ P's I²C bus lines are not open-drain, the ESD diodes will clamp the bus lines and interfere with programming. Sharing JTAG may require a JTAG bus multiplexer powered from 3.3V.

If the system μ P does not have true open-drain I²C bus lines, a circuit like the one in **Figure 1** can be used to automatically switch between the μ P and the programming I²C bus.

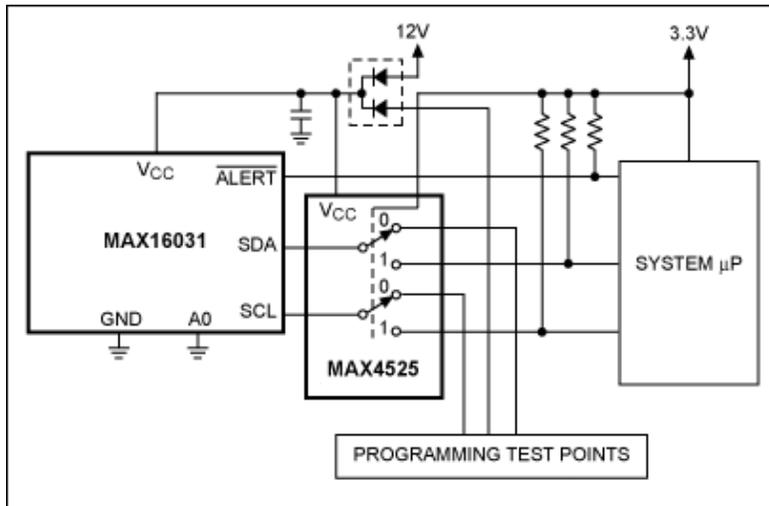


Figure 1. The MAX16031 shares its I²C bus through the MAX4525 multiplexer/switch.

The MAX4525 multiplexer in Figure 1 switches between the I²C connected to the system μ P and the I²C connected to the programming test points. The switch is controlled by the V_{CC} of the system μ P. If V_{CC} is not applied, but 12V is, the switch connects I²C to the programming test points. Once V_{CC} is applied, the switch connects the I²C to the system μ P. Note that, during programming mode, the programming hardware that connects to the test points must provide appropriate I²C pullup resistors.

Application Circuit Examples

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

Powered from a 12V Intermediate Bus and Programmed Through the I²C Bus

In **Figure 2**, the MAX16031 is powered from the 12V intermediate bus. A dual diode allows power to be provided by the programming connector, which also connects to the I²C lines of the MAX16031. These are shared with an onboard system-management μ P with open-drain I²C outputs that do not load the bus, even when the μ P is unpowered. As an alternative to an external programmer, the system-management μ P can also program the MAX16031 on initial power-up. This also makes it easy to update the configuration of the MAX16031 without special hardware.

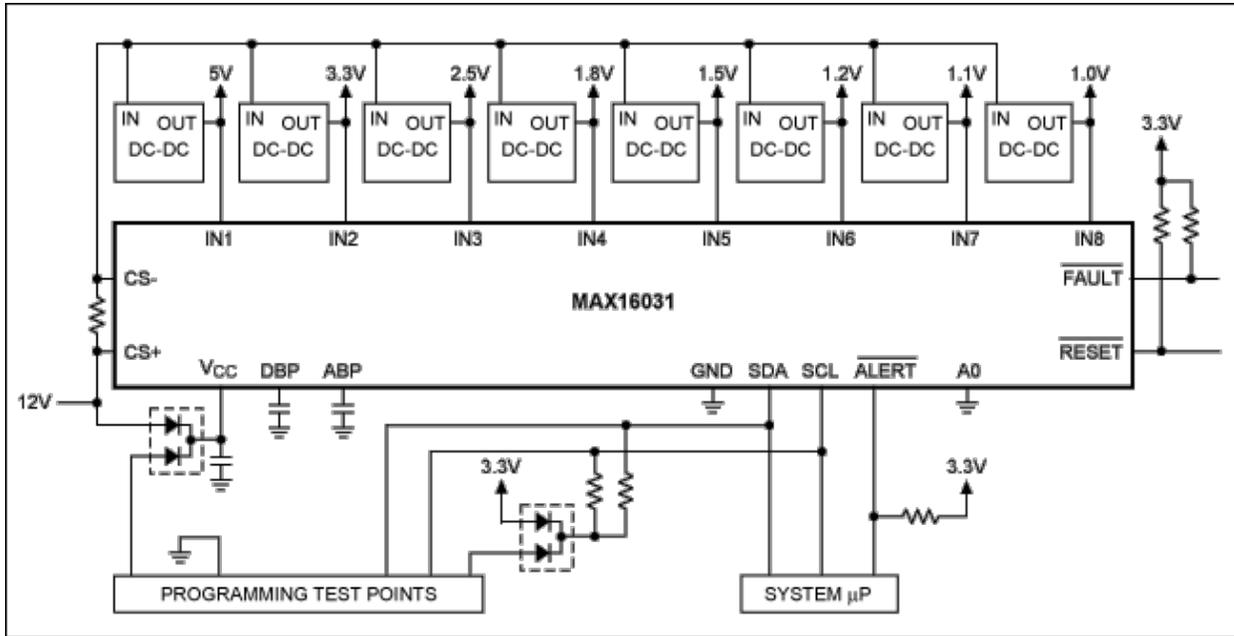


Figure 2. The MAX16031 is powered from a 12V intermediate bus and programmed in-circuit through the I²C bus.

Powered from a 3.3V Auxiliary Bus and Programmed Through the JTAG Port

Figure 3 shows the MAX16031 being powered from a 3.3V auxiliary bus. Programming is accomplished through dedicated JTAG lines brought out to programming test points. For this example, the 3.3V auxiliary bus must be supplied before programming can be accomplished. The I²C interface still connects to a system-management μP.

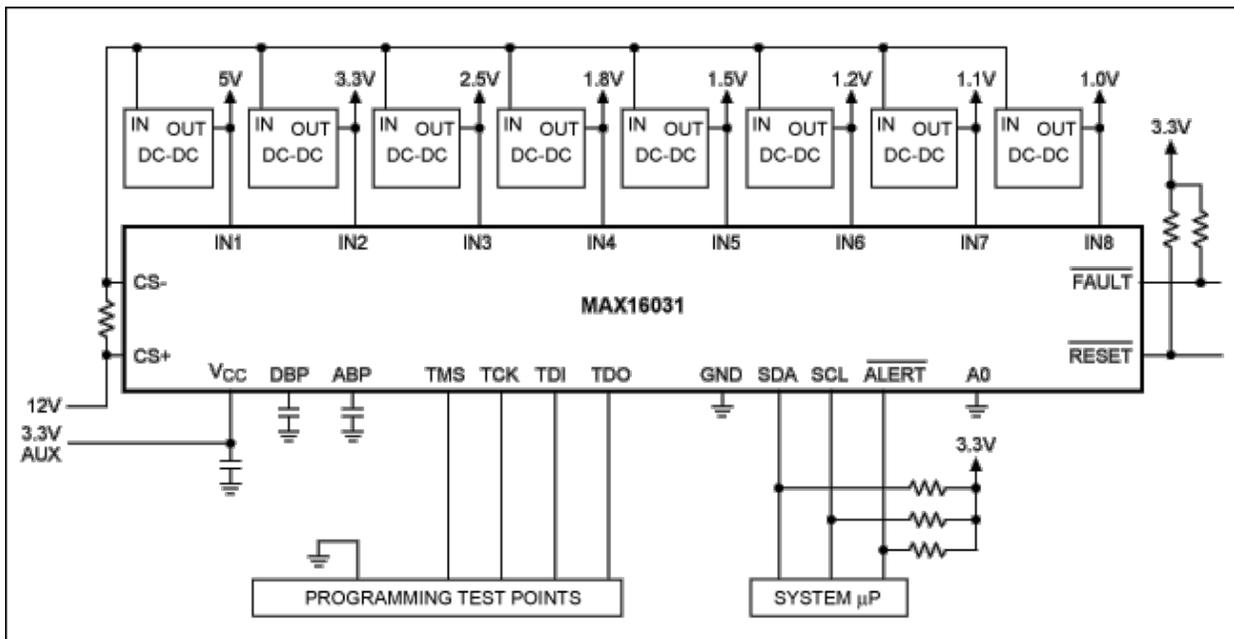


Figure 3. The MAX16031 is powered from a 3.3V auxiliary bus and programmed through the JTAG port.

Programming Algorithm

The MAX16031/MAX16032 have built-in EEPROM that stores the device configuration parameters. When power is applied, the contents of the EEPROM are transferred to the RAM registers. Both RAM and EEPROM are accessible from the JTAG and I²C interfaces. To correctly program the MAX16031/MAX16032, the desired parameters must be programmed to the EEPROM—see the memory map in **Table 1**.

Table 1. MAX16031/MAX16032 memory map

00h	ADC Readings		80h	Nonvolatile Fault EEPROM
16h			8Eh	Reserved
17h	Configuration Registers		8Fh	Reserved
5Ch	Reserved		96h	Configuration EEPROM
5Dh			97h	
5Eh	Configuration Registers		DC	Reserved
5Fh	Reserved		DDh	
60h			Reserved	DEh
7Fh	DFh			Reserved
			E0h	
		7Fh		

Configuration Files

The MAX16031 evaluation kit (EV kit) software provides two types of configuration files. One is a human-readable text file produced by selecting **System → Save Configuration...** This file can be used for I²C programming. The second is produced by selecting **System → Save as SVF...** This file is in the serial vector format (SVF) used by many PLD vendors for JTAG programming.

The text file format is as follows:

```
[Registers]
register number=register value
.
.
.
```

All values are decimal. The registers begin at 23 and go to 98. These addresses correspond to RAM registers, not EEPROM addresses. To obtain the EEPROM address, add 128 to the RAM register address.

The SVF file format is described in more detail in the [Serial Vector Format Specification](#) (PDF, 85.2kB).

I²C Programming Procedure

To program the MAX16031/MAX16032's EEPROM configuration memory, it is necessary to first make sure that the configuration-lock bit in register r5Fh[0] is zero. If it is not zero, write a '1' to that bit to clear it. To write to the EEPROM, load the starting address (97h) and follow it with a series of block-write commands (I²C) or write commands (JTAG). See the [MAX16031/MAX16032](#) data sheet for details on the I²C protocols.

Pseudocode for a typical EEPROM programming process is as follows:

```

SendByte(5Fh) // Check lock bit
If ReadByte() & 1 == 1 Then
  WriteByte(5Fh, 01h) // Clear lock bit if needed
Loop Address from 97h to DFh
  SendByte(Address) // Load address
  WriteBlock(Data, 10h) // Write a block of 16 bytes
  Wait(16 * 11 milliseconds) // Wait for programming
  SendByte(Address)
  ReadBlock(DataRead, 10h) // Read back data block
  If DataRead != Data Then
    RepeatCount = RepeatCount + 1
    If RepeatCount == 3 Then
      Fail
    Else
      RepeatCount = 0
      Address = Address + 10h // Advance to next block
  Success

```

JTAG Programming Procedure

To program the device, use standard third-party JTAG tools, the MAX16031/MAX16032 BSDL file, and an SVF data file generated by the EV kit software together 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 MAX16031 EV kit software:

```

ENDDR IDLE;
ENDIR IDLE;
SIR 5 TDI(00) TDO(01);
SDR 32 TDI(00000000) TDO(00001197) MASK(0FFFFFFF);

```

The TDO(00001197) 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(00001197) MASK(0FFFFFFF);

```

The mask statement ignores the 4-bit revision code field (see **Table 2**).

Table 2. IDCODE Register Bitmap

	Revision (4 Bits)	Part Number (16 Bits)	Manufacturer (11 Bits)
Binary	0010	0000 0000 0000 0001	00011001011 1
Hex	2	0001	197

Related Parts

MAX16031	EEPROM-Based System Monitors with Nonvolatile Fault Memory	Free Samples
MAX16032	EEPROM-Based System Monitors with Nonvolatile Fault Memory	Free Samples

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 4283: <http://www.maximintegrated.com/an4283>

APPLICATION NOTE 4283, AN4283, AN 4283, APP4283, Appnote4283, Appnote 4283

Copyright © by Maxim Integrated Products

Additional Legal Notices: <http://www.maximintegrated.com/legal>