



APPLICATION NOTE 4450

Getting Started with the MAX6651 Fan Controller

By: Mark Underwood

Abstract: The MAX6651 fan controller is configured for closed-loop operation in this application note. The stand-alone MAXQ2000-based firmware is loaded into a MINIQUSB board connected to a MAX6651 EV (evaluation) kit, and the target fan speed is selected by momentarily connecting a pin to ground.

Required Hardware

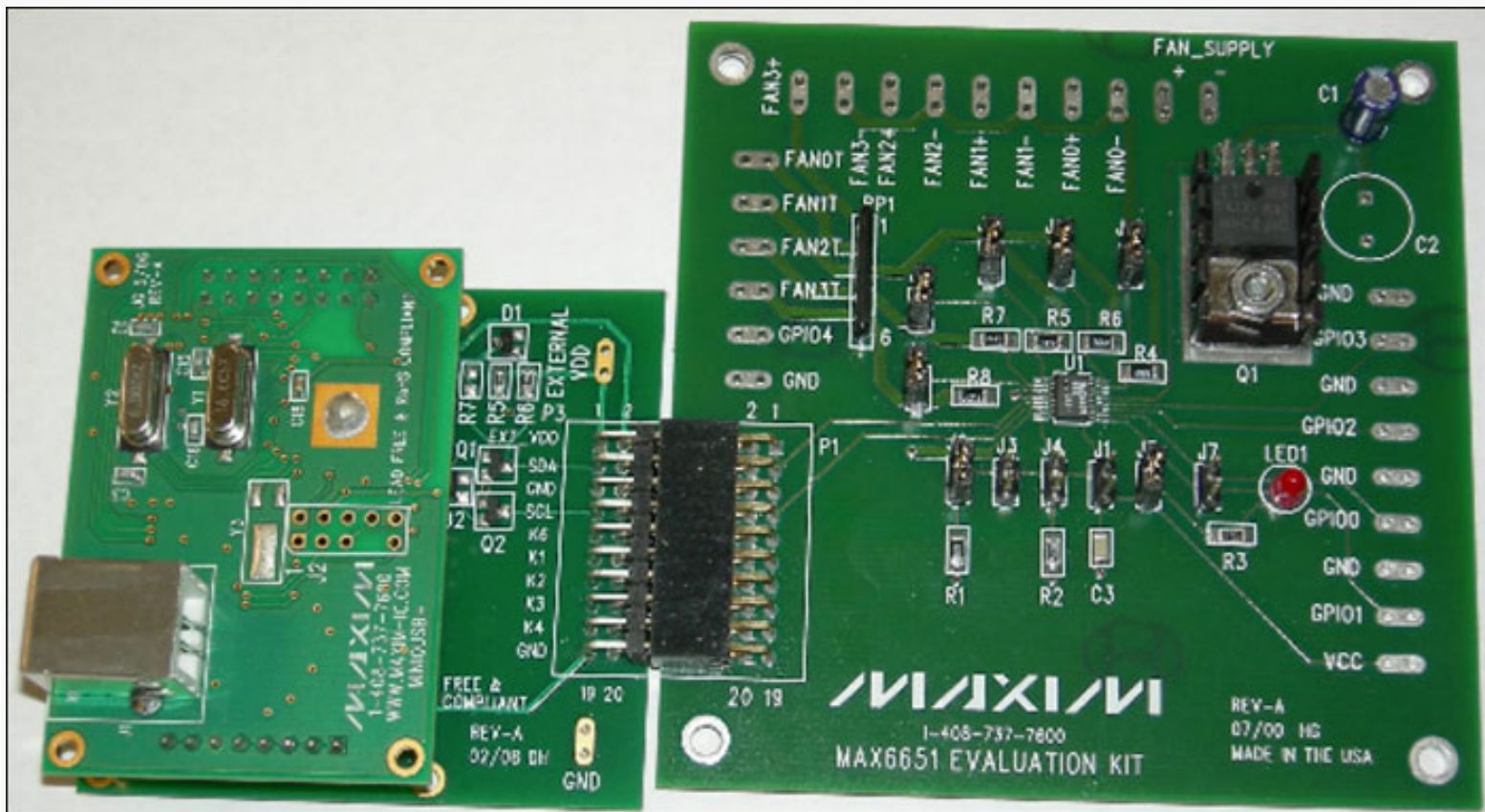
- Maxim MAX6651 EV kit ([MAX6651EVKIT](#))
- Maxim [MINIQUSB+](#) (includes a USB A-B cable and MINIQUSB-XHV expander board)
- Computer with USB, running Windows® 2000, Windows XP®, or Windows Vista™
- 12V DC power supply
- 12V DC fan with tachometer output
- Optional: up to three additional fans of the same type
- Optional: oscilloscope to observe tachometer pulses

Note that up to three additional 12V DC fans can be connected, however, the system only regulates the speed of the first fan. All fans should be of the same type.

This example assumes a nominal fan speed of 6000RPM. If the fan's maximum speed is less than the target, the fan will not reach the target speed and a fault will be detected. If using a slower fan, load the KSCALE=2 or KSCALE=1 firmware in step 2 below. If using a faster fan, load the KSCALE=8 or KSCALE=16 firmware in step 2.

Setup

- [Download](#) and unzip the application note files.
- Assemble the hardware as shown in **Figure 1**.



1. Connect the MINIUSB+ to the PC's USB port. If this is the first time that a MINIUSB+ has been attached to the PC, the plug-and-play wizard will appear. Guide Windows to the installed location of the device driver (which is included in the accompanying zip file).
2. Launch the FW_GUI firmware updater program. At the top left of the window, you should see a preview picture of a MINIUSB+ board. Drop down the combo box and ensure that "Maxim MAX6651 Appnote KSCALE=4" is selected. Click the program's Start button to begin the upgrade. The software will prompt you to connect the USB cable, then after about a minute prompt you to disconnect USB. (The same software can be used to restore the Maxim MINIUSB+ Module firmware.)
3. Connect the MAX6651EVKIT+ board to the MINIUSB+ board using the MINIUSB-XHV+ expander board.
4. With power off, connect a 12V DC power supply to the MAX6651EVKIT FAN_SUPPLY pads.
5. Connect the fan power (typically a red wire) to the MAX6651EVKIT FAN0+ pad.
6. Connect the fan ground return (typically a black wire) to the MAX6651EVKIT FAN0- pad.
7. Connect the fan tachometer (typically a yellow wire) to the MAX6651EVKIT FAN0T pad.
8. Connect the USB cable, so that the MINIUSB+ board is powered by the PC.
9. Enable the 12V DC power supply. The fan will start at full speed then slowly drop to 6000RPM.
10. Change the target fan speed to preset 4 (2000RPM) by momentarily connecting MINIUSB-XHV pin K4 to GND on the P3 connector.
11. Change the target fan speed to preset 3 (3000RPM) by momentarily connecting MINIUSB-XHV pin K3 to GND on the P3 connector.
12. Change the target fan speed to preset 2 (4000RPM) by momentarily connecting MINIUSB-XHV pin K2 to GND on the P3 connector.
13. Turn on the fan in "full-on" mode by momentarily connecting MAX6651EVKIT pad GPIO1 to GND.
14. Turn off the fan by momentarily connecting MINIUSB-XHV pin K6 to GND on the P3 connector.
15. Change the target fan speed to preset 3 (3000RPM) by momentarily connecting MINIUSB-XHV pin K3 to GND on the P3 connector. (This should be about half the nominal fan speed).
16. Decrease the DC power supply to 7V. The fan will slow down, and then gradually return to the correct speed.
17. Increase the DC power supply to 12V. The fan will speed up, and then gradually return to the correct speed.
18. Turn off the fan by momentarily connecting MINIUSB-XHV pin K6 to GND on the P3 connector.
19. Simulate a fan seizure fault by physically restraining the fan armature.
20. Change the target fan speed to preset 2 (4000RPM) by momentarily connecting MINIUSB-XHV pin K2 to GND on the P3 connector. After about one minute, LED1 will light, indicating that the MAX6651 was unable to regulate the fan speed. (More detailed status information would be available to user-customized software by reading MAX6651 Alarm Register, 0x0A. The example software in this application note does not, however, have a way to report these details.)
21. Free the fan armature. The fan will immediately speed up, then gradually slow to the correct speed.

Optional Setup—Multiple Fans

Optionally, up to three additional 12V DC fans can be connected, however, the system only regulates the speed of the first fan. All fans should be of the same type.

Connect the second, third, and fourth fans as follows:

Optional: fan 2 (feed-forward regulation based on fan 1)

```
Fan +12V = MAX6651EVKIT FAN1+
Fan GND = MAX6651EVKIT FAN1- (ground return)
Fan Tach = MAX6651EVKIT FAN1T (MAX6651 TACH1)
```

Optional: fan 3 (feed-forward regulation based on fan 1)

```
Fan +12V = MAX6651EVKIT FAN2+
Fan GND = MAX6651EVKIT FAN2- (ground return)
Fan Tach = MAX6651EVKIT FAN2T (MAX6651 TACH2)
```

Optional: fan 4 (feed-forward regulation based on fan 1)

```
Fan +12V = MAX6651EVKIT FAN3+
Fan GND = MAX6651EVKIT FAN3- (ground return)
Fan Tach = MAX6651EVKIT FAN3T (MAX6651 TACH3)
```

These extra fans can be connected in any order. All fans are driven with the same voltage. Only the fan connected to MAX6651 TACH0 will be regulated; any fan connected to TACH1, TACH2, or TACH3 can be monitored but not regulated.

Optional Setup—MAXQ2000EVKIT instead of MINIUSB

If using the MAXQ2000 EV kit instead of MINIUSB+, follow the normal setup, except in step 1 connect as follows:

```
MAXQ2000 P6.0 = MAX6651 SCL
MAXQ2000 P6.1 = MAX6651 SDA
MAXQ2000 VDDIO = MAX6651 VCC
MAXQ2000 GND = MAX6651 GND
```

The MAXQ2000 EV kit does not use USB to load firmware. Instead, use the MAXQ2000 JTAG 001 loader board that comes with the MAXQ2000 EV kit.

Recompiling the Firmware

The stand-alone firmware was compiled using the IAR Embedded Workbench®, version 3.2 for MAXQ® (available as part of the MAXQ2000 EV kit).

Source file **main.c** contains the main loop and the MAX6651's specific functions. For convenience, there are five prebuilt versions of the demo project, one for each of the prescaler values KSCALE=1, 2, 4, 8, and 16. Each of these projects has its own customized version of **main.c**, for example, **main_Kscale_4.c**. These files all have the same structure. Function **debug_puts()** is defined as a placeholder, to handle optional diagnostic messages. Source file **max6651.h** contains symbolic definitions for the MAX6651 registers. These symbolic definitions can be combined using the C/C++ bitwise Boolean OR operator|as demonstrated by this statement:

```
MAX6651_R02_Config = 0
| MAX6651_Config_ClosedLoop
| MAX6651_Config_12VTach
| MAX6651_Config_ClosedLoop_PreScale_4
;
```

Placing each bit field's symbolic constant on its own line makes the source code more readable and easier to maintain.

Function **MAX6651_Init()** initializes the MAX6651 device by copying the initial register values from global variables: **MAX6651_R00_Speed**, **MAX6651_R02_Config**, **MAX6651_R04_GPIODefinition**, **MAX6651_R08_AlarmMask**, and **MAX6651_R16_Count**. (Global variables are used instead of C function arguments, because stack memory is more limited than data memory.) If the device address is not specified, then all four of the possible device addresses are searched prior to writing the register data. This simplified example code does not check for errors.

The closed-loop fan speed is determined by the value in register 0. This register value is calculated by function **MAX6651_calculate_R00_Speed_from_RPM()**. This value depends on the prescaler setting of global variable **MAX6651_R02_Config**.

The default fan speed and the four software preset speeds are defined by the preprocessor constants: **FAN_SPEED_INITIAL_RPM**, **FAN_SPEED_PRESET_GPIO_K1_IN_RPM**, **FAN_SPEED_PRESET_GPIO_K2_IN_RPM**,

FAN_SPEED_PRESET_GPIO_K3_IN_RPM, and FAN_SPEED_PRESET_GPIO_K4_IN_RPM.

When an alarm condition happens (such as the output reaching its minimum or maximum value without achieving fan-speed regulation), the ALERT output is asserted and the EV kit LED1 lights. Once asserted, this alarm condition remains in effect until the status register is read. The main loop calls **MAX6651_Status_Read()** to read the alarm status register, thereby clearing any prior fault condition. This function also contains optional code to read the MAX6651's GPIO pins and the tachometer count of up to four fans.

Source files **CmodComm.c** and **CmodComm.h** provide the register read/write functions **CmodSMBusWriteByte()** and **CmodSMBusReadByte()**. These functions are implemented using the **maxqi2c.c** library.

Conclusion

This example has briefly shown the MAX6651 regulating fan speed in closed-loop mode, automatically responding to power-supply fluctuation and fan-hardware faults. Complete source code for a project compiled with IAR Embedded Workbench has been given and discussed. Consult the [MAX6651](#) data sheet for further details about device operation.

IAR Embedded Workbench is a registered trademark of IAR Systems AB.

MAXQ is a registered trademark of Maxim Integrated Products, Inc.

Windows is a registered trademark of Microsoft Corp.

Windows Vista is a registered trademark of Microsoft Corp.

Application note 4450: www.maxim-ic.com/an4450

More Information

For technical support: www.maxim-ic.com/support

For samples: www.maxim-ic.com/samples

Other questions and comments: www.maxim-ic.com/contact

Automatic Updates

Would you like to be automatically notified when new application notes are published in your areas of interest? [Sign up for EE-Mail™](#).

Related Parts

MAX6650: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

MAX6651: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

MAXQ2000: [QuickView](#) -- [Full \(PDF\) Data Sheet](#) -- [Free Samples](#)

MINIUSB: [QuickView](#) -- [Full \(PDF\) Data Sheet](#)

AN4450, AN 4450, APP4450, Appnote4450, Appnote 4450

Copyright © by Maxim Integrated Products

Additional legal notices: www.maxim-ic.com/legal