Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

AN78692

PSoC® 1 – Intelligent Fan Controller

Author: Sanjeev Kumar
Associated Project: Yes
Associated Part Family: CY8C28xxx
Software Version: PSoC Designer™ 5.4
Related Application Notes: AN66627

AN78692 demonstrates how to quickly and easily develop a four-wire brushless DC fan control system using PSoC® 1. The Fan Controller User Module, available in PSoC Designer™, helps manage the fans in a variety of configurations. This application note also shows how to combine fan control and temperature sensing to create a complete thermal management solution using PSoC 1. This application note assumes familiarity with PSoC 1, the PSoC Designer integrated development environment (IDE), and C programming. To get started with PSoC, click here.

Contents

Introduction

System cooling is a critical component of any high-power electronic system. As circuits become smaller, the demands on system designers to improve the efficiency of system thermal management are increasing. System cooling fans must run at the optimum speed to ensure that the system temperature is always below a defined limit. To achieve this, the electronic system needs a temperature measurement unit and a closed-loop fan speed controller.

Thermal management can be done by running fans at full speed. However, running numerous fans at high speed will result in the following problems:

- Increased audible noise
- Increased power consumption
- Decreased lifetime due to mechanical wear and tear
- Increased clogging (dust collection)

Operating fans at lower than required speeds leads to inefficient cooling and results in the overheating of components, which causes component failures. To overcome this problem, you need to control fan speeds according to environmental conditions (that is, temperature).

You can control fan speeds in the following ways:

- Direct pulse-width modulation (PWM): This method involves switching the power supply on and off at a certain frequency (that is, modifying the duty cycle to control the speed).
- Linear regulation: In this method, the DC voltage across the fan is controlled by a linear regulator, which then controls the fan speed.

- DC-DC regulation: This method is similar to linear regulation but uses a switching regulator instead of a linear regulator.

The direct PWM method is commonly used because it offers reduced power dissipation, low cost, and ease of design.

Most brushless DC fans used in thermal management have four wires. This application note discusses the speed control of four-wire fans using PSoC® 1.

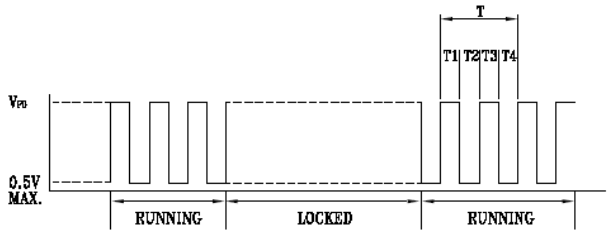
Four-Wire Fan Basics

The four wires of brushless DC fans are power, ground, tachometer output, and PWM input. Figure 1 shows a typical four-wire brushless DC fan.

Figure 1. Typical Four-Wire DC Fan

Four-wire DC fans include Hall-effect sensors, which sense the rotating magnetic fields generated by the rotor as it spins. The output of the Hall-effect sensor is a pulse-train that has a period inversely proportional to the rotational speed of the fan. The number of pulses that are produced per revolution depends on the number of poles in the fan. For the most common four-pole brushless DC fan, the tachometer output from the Hall-effect sensor generates two pulses per fan revolution. If the fan stops rotating due to mechanical failure or other fault, the tachometer output signal remains static at either a logic LOW or logic HIGH level. The speed of this fan is measured in revolutions per minute (rpm). The fan referenced in this application note is AVC DB04028B12UP090. It is available in kit CY8CKIT-036. Figure 2 shows the tachometer output for this fan.

Figure 2. Tachometer Output of Fan



$$N: Speed(RPM) = \frac{60}{(T1 + T2 + T3 + T4)}$$

Fans come in standard sizes: 40 mm, 80 mm, and 120 mm are common. The most important feature when selecting a fan for a cooling application is the amount of air the fan can move. The specification is listed in either cubic feet per minute (CFM) or cubic meters per minute (m^3/min). The size, shape, and pitch of the fan blades all contribute to the fan's capability to move air. To move the same volume of air in a given time frame, smaller fans must run at a higher speed than larger fans.

Applications that need smaller fans due to space limitations generate significantly more acoustic noise. This unavoidable tradeoff must be made in order to meet system-level needs.

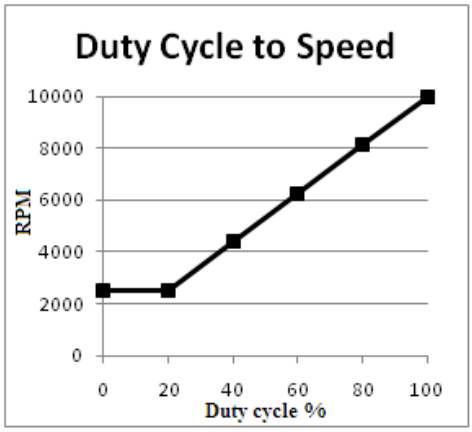
To manage acoustic noise generation, you can configure PSoC 1 to drive fans at the minimum possible speed, simultaneously maintaining safe operating temperature limits. This also extends the operating life of the fan compared to systems that run all of the fans at full speed all the time.

Fan manufacturers specify how the PWM duty cycle relates to nominal fan speed and provide tables of data points or graphs that show the relationship. Figure 3 provides an example, with the PWM control duty cycle shown as a percentage on the horizontal axis and fan speed in rpm on the vertical axis.

Fan manufacturers specify duty-cycle-to-rpm relationships in their datasheets and give tolerances as high as ± 20 percent. To guarantee that a fan will run at the desired speed, system designers would need to run the fans at speeds 20 percent higher than nominal to ensure that any fan from that manufacturer provides sufficient cooling. High speeds result in excessive acoustic noise and greater power consumption.

It is important to note that at low duty cycles, fans do not all behave in the same way. Some fans stop rotating as the duty cycle approaches 0 percent, whereas others continue to rotate at a low rpm. In both cases, the duty-cycle-to-rpm relationship can be nonlinear or not specified. When using duty-cycle-to-rpm information, select two data points from the linear region in which the behavior of the fan is well defined.

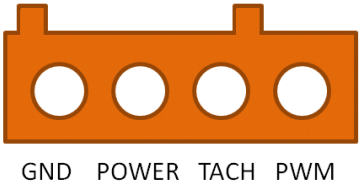
Figure 3. Example Duty-Cycle-to-Speed Chart



Fan Cables and Connectors

At the cabling level, wire color coding is not consistent across manufacturers, but the connector pin assignment is standard. Figure 4 shows the bottom view of the connector. Note that the connectors are keyed to prevent incorrect insertion into the fan controller board. The keying scheme chosen also enables four-wire fans to connect to control boards that were designed to support three-wire fans without modification. (Three-wire fans do not have PWM speed control signals.)

Figure 4. Four-Wire DC Fan Connector Pin Assignment



Fan Speed Control

With the direct PWM method, you can maintain the nominal speed of the fan in the following ways:

- In open-loop speed control, the host or main controller (the CPU in a computerized environment) on the system instructs the fan controller (PSoC 1) to run the fans at a specific duty cycle. In this case, the host already has the desired speed versus duty cycle information.In closed-loop control, the fan controller ensures that the fan is running at the desired speed by measuring the actual speed and adjusting the duty cycle accordingly to reach the desired speed. The tolerance of closed-loop control is defined as follows:

$$\text{Tolerance}\,\% = \frac{\text{ActualSpeed} \pm \text{DesiredSpeed}}{\text{DesiredSpeed}} \times 100$$

The tolerance usually is defined by the host system. The slave fan controller performs closed-loop control only when the measured tolerance is greater than the tolerance specified by the host.

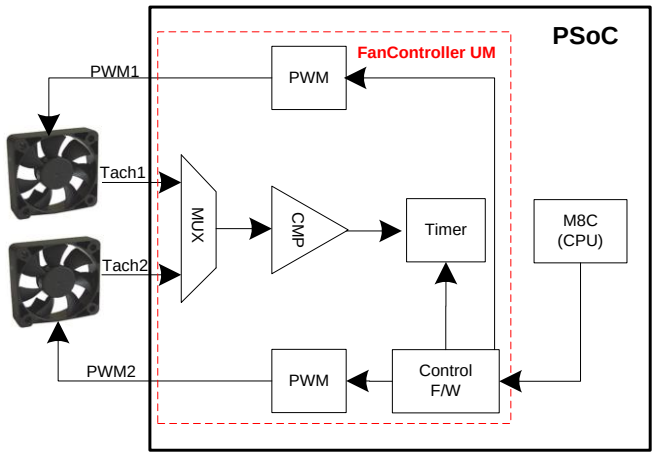
This application note presents the fan controller user module in PSoC 1, which you can use to set up a thermal management system.

Distributed with this application note is a PSoC Designer™ compressed file that contains four example design projects. Save the file to a convenient location on your hard drive and extract the contents to a local folder. The four different fan controller projects are listed as follows:

1. Open-loop fan controller with I^2C for duty cycle control
2. Open-loop fan controller with switches for duty cycle control
3. Closed-loop fan controller with I^2C for rpm control
4. Closed-loop fan controller with switches for rpm control

Figure 5 shows the top-level design schematic for the fan controller using PSoC 1.

Figure 5. Top-Level Design



Fan Controller User Module

The basic blocks of the fan controller user module are the PWM, timer, comparator, and analog multiplexer. Table 1 describes these underlying basic blocks.

Table 1. Hardware Resources

Basic Block	Description
PWM	PWMs are used to generate a pulse-width modulated signal to drive the fan.
Timer	This timer measures speed using tachometer pulses from the fan.
Comparator	A hysteresis comparator is used to debounce tachometer pulses from the fan.
Analog multiplexer	This basic block sends tachometer signals to the comparator.

PWM

The resolution of the PWM drivers can be set to 8 bit or 10 bit, as required for your application. Ten-bit resolution gives finer speed control but uses more digital resources in the application. You can select the frequency of the PWM from 24 to 48 kHz.

Timer

The capture in this 16-bit timer is connected to the output of the comparator so the capture mechanism is triggered by a clean tachometer signal coming from the comparator. The ISR of the timer is set on overflow/terminal count, so the timer will be reloaded during every overflow. If the timer overflows and reloads more times, it will indicate a stuck-fan fault. The corresponding fan fault variable will be updated, and an alert will be generated if the parameter has been selected in the user module.

The fan controller user module indicates a fan fault if the speed drops below 460 rpm. The minimum speed for the hypothetical fan referenced in Figure 3 is 2,800 rpm. This means that even at a 0 percent duty cycle, the fan will run at 2,800 rpm. Some fans, however, have a speed of 0 rpm at a 0 percent duty cycle.

Comparator

The comparator is used to debounce the tachometer signal using hysteresis. The output of the comparator is connected to the capture mechanism of the tachometer timer. The connected tachometer signal will generate a comparator ISR. This ISR acts as a fan controller ISR. On this ISR, the timer will start measuring the frequency of the connected tachometer signal. After the tachometer measurement is completed, the speed measurement will be updated in the tachometer element of the fan parameter array. Then the tachometer signal from the fan will be disconnected and the next fan will be connected for measurement.

Analog Multiplexer

The analog multiplexer sends tachometer signals to the comparator block. In devices that have an analog multiplexer bus (such as CY24x94 or CY28xxx), any port pin (pins for connecting the tachometer signal) can be routed to the comparator. In devices that do not have an analog multiplexer, the number of port pins that can be connected to a comparator block is limited to the number of inputs that the analog column input multiplexer can take (eight Port-0 pins).

Cypress Development Kits for Thermal Management

The following Cypress kits can be used for quick and easy evaluation of thermal management solutions:

- CY8CKIT-001, PSoC Development Kit (DVK)
- CY8CKIT-036, Thermal Management Expansion Board Kit (EBK)
- CY8CKIT-020, PSoC CY8C28 Family Processor module kit (comes with CY8CKIT – 001)
- CY8CKIT-002, PSoC MiniProg3 Program and Debug Kit (comes with CY8CKIT-001) OR CY3217, MiniProg1
- 12 V, 1 A DC power supply adapter (comes with CY8CKIT-001)
- 12 V, 2 A DC power supply adapter (comes with CY8CKIT-036)

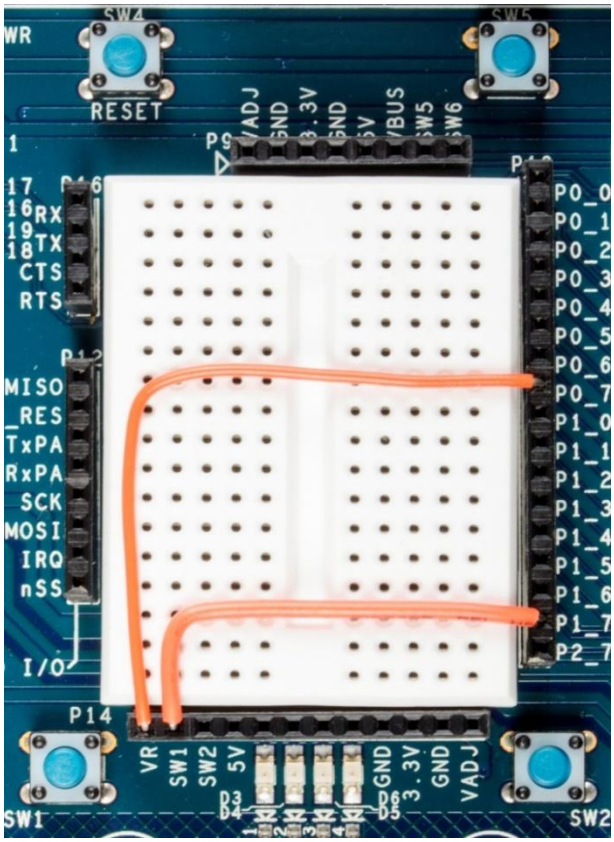
Hardware Connection Steps

This section describes how to set up the hardware to run the example projects.

CY8CKIT-001 PSoC DVK

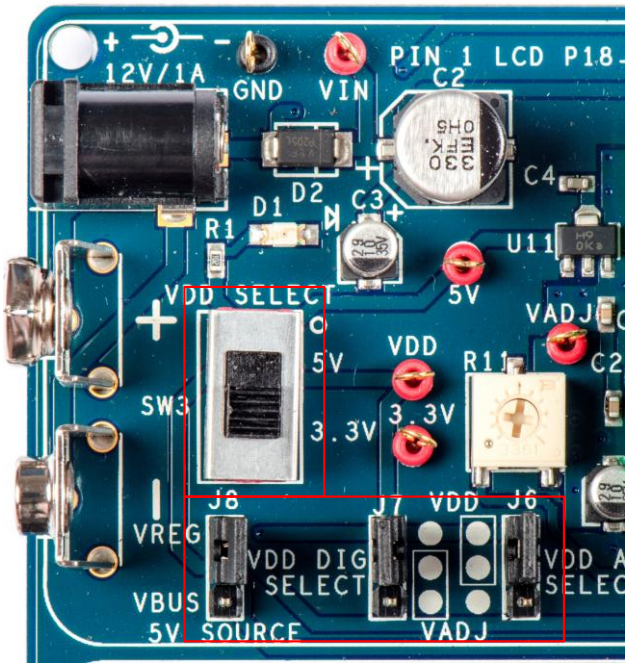
1. For Example 5, use jumper wires to make the following connections in the pin header (breadboard) area of the PSoC DVK baseboard, as Figure 6 shows. For the remaining examples, use the connections indicated in each example description.
   - VR to P0_7
   - SW1 to P1_7

Figure 6. CY8CKIT-001 PSoC DVK Breadboard



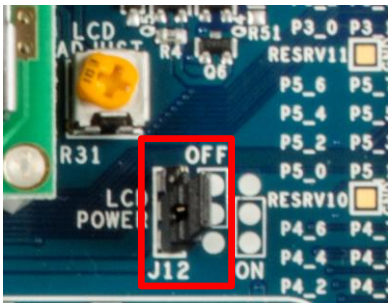
2. Set the jumpers as follows, and as shown in Figure 7:
   - J8 to VREG (1-2)
   - SW3 to 3.3 V
   - J6 to VDD (1-2)
   - J7 to VDD (1-2)

Figure 7. CY8CKIT-001 PSoC DVK Power Jumpers



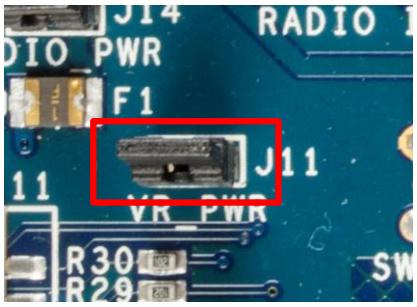
3. Attach the LCD included with the PSoC DVK and set the LCD power jumper (J12) in the ON position, as shown in Figure 8.

Figure 8. CY8CKIT-001 PSoC DVK LCD Power Jumper



4. Make sure that the VR_PWR jumper (J11) is installed as shown in Figure 9.

Figure 9. CY8CKIT-001 PSoC DVK VR_PWR Jumper

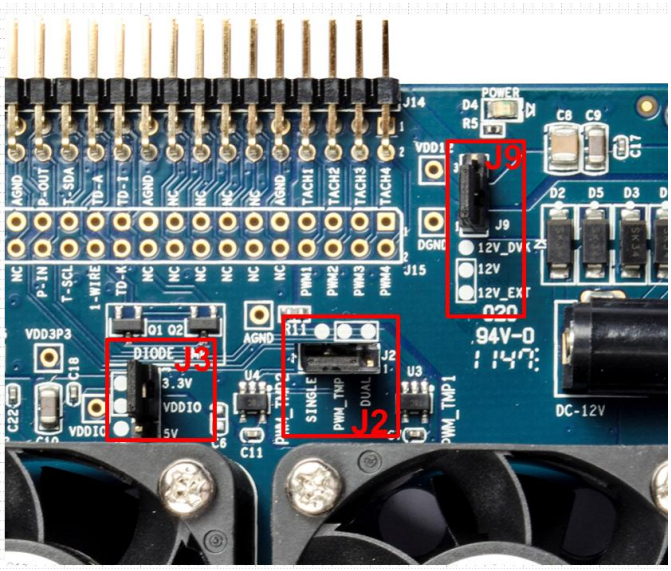


CY8CKIT-036 PSoC Thermal Management EBK Jumper Settings

Set the jumpers as follows, and as shown in Figure 10:

- J2 to SINGLE
- J3 to 3.3 V
- J9 to 12V_EXT
- For Example 5, connect fans to FAN1 (J7), FAN2 (J8).
- For Examples 1–4, connect fans to FAN1(J7), FAN3 (J10).

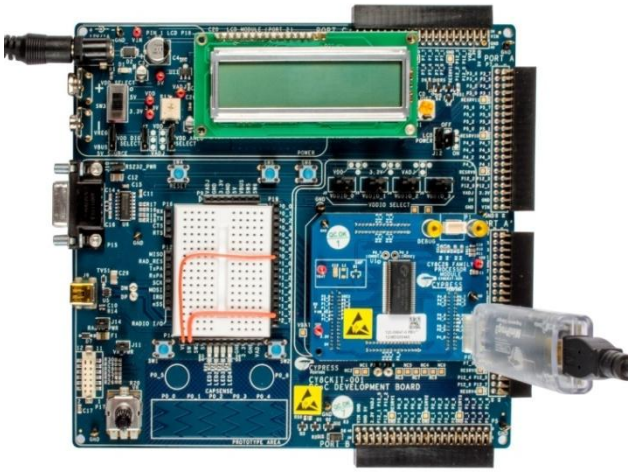
Figure 10. Jumper Settings



Programming

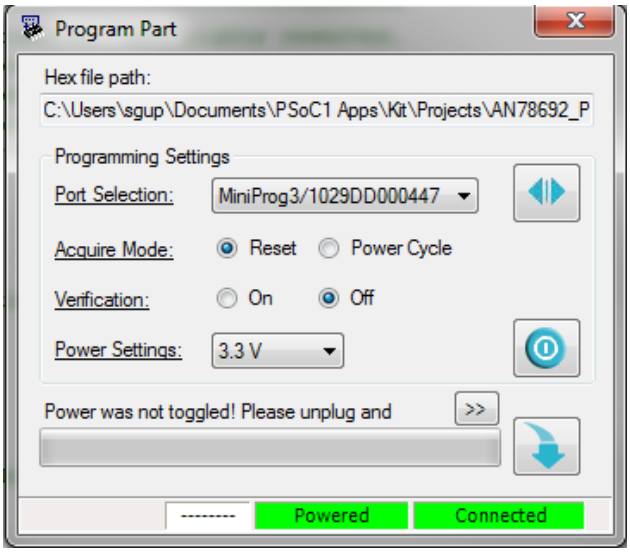
1. If this is the first time that the example project firmware is being programmed into PSoC, make sure that the PSoC Thermal Management EBK is not connected to the CY8CKIT-001 DVK.
2. Make sure that the CY8C28 family processor module (CY8CKIT-020) is connected to CY8CKIT-001.
3. Connect the MiniProg3 first to a USB port on the PC and then to the PROG port on the CY8C28 family processor module, as shown in Figure 11.

Figure 11. MiniProg3 Connected



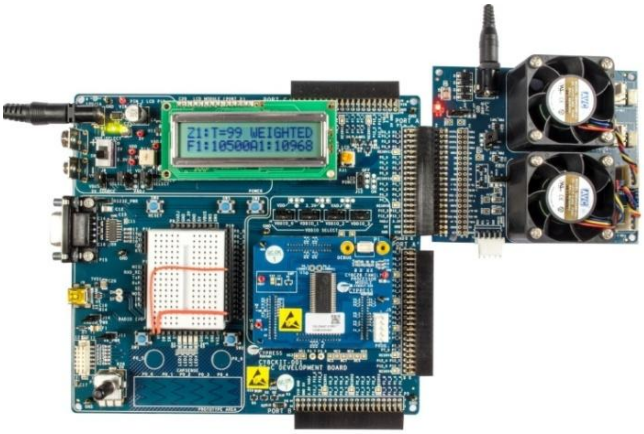
4. Create a new project by cloning the example project provided in the PSoC 1 Thermal Management EBK. Follow these steps:
   a. Open PSoC Designer.
   b. Select **File** > **New Project**. Enter a name for your new project. Click **OK**.
   c. The next window has an option to select an existing project for cloning. Click the Browse button under **Clone Project** and select the PSoC Designer project file in the *.cmx or *.soc format from the location in which the PSoC 1 Thermal Management EBK is installed.
   d. Select **Use Same Target Device**. Then select the target device and press **OK**.
   e. After cloning the project, press F6 or select **Build** > **Generate/Build** *[Project Name]* **Project** to generate and build the project.
5. In PSoC Designer, select **Program** > **Program Part**.
   a. Make sure the program settings correspond to those given in Figure 12. Then click the Program button.
   b. When programming has been successfully completed, remove the MiniProg3.

Figure 12. PSoC Programmer



After the device has been programmed, follow the setup shown in Figure 13.

Figure 13. Hardware Setup



Getting Started with the Fan Controller User Module

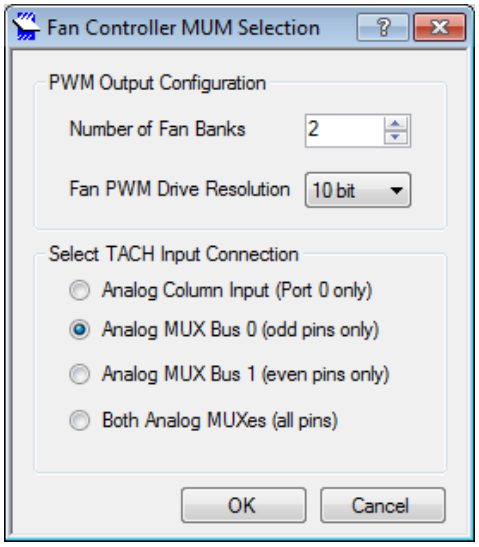
Open any of the four example projects. Right-click the fan controller user module and open the fan controller multi-user module (MUM) selection window, shown in Figure 14.

A fan bank is defined as multiple fans that are driven by the same PWM speed control signal. For the example projects, you will drive two fans with two different PWM signals representing two banks.

The resolution of the PWM drivers can be set to 8 bit or 10 bit, as required for your application. Ten-bit resolution gives finer speed control but uses more digital resources in the application. All projects use 10-bit PWM resolution.

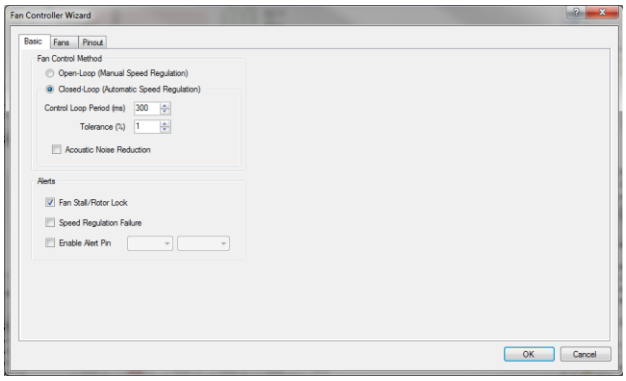
The tachometer connection can be routed from any GPIO using the analog MUX bus options. These options vary depending on the device selected.

Figure 14. Fan Controller MUM Selection



Right-click the fan controller user module and click **Fan Controller Wizard** to open the wizard, as shown in Figure 15.

Figure 15. Fan Controller Wizard Basics Tab



In the Basic tab, select the method of speed regulation.

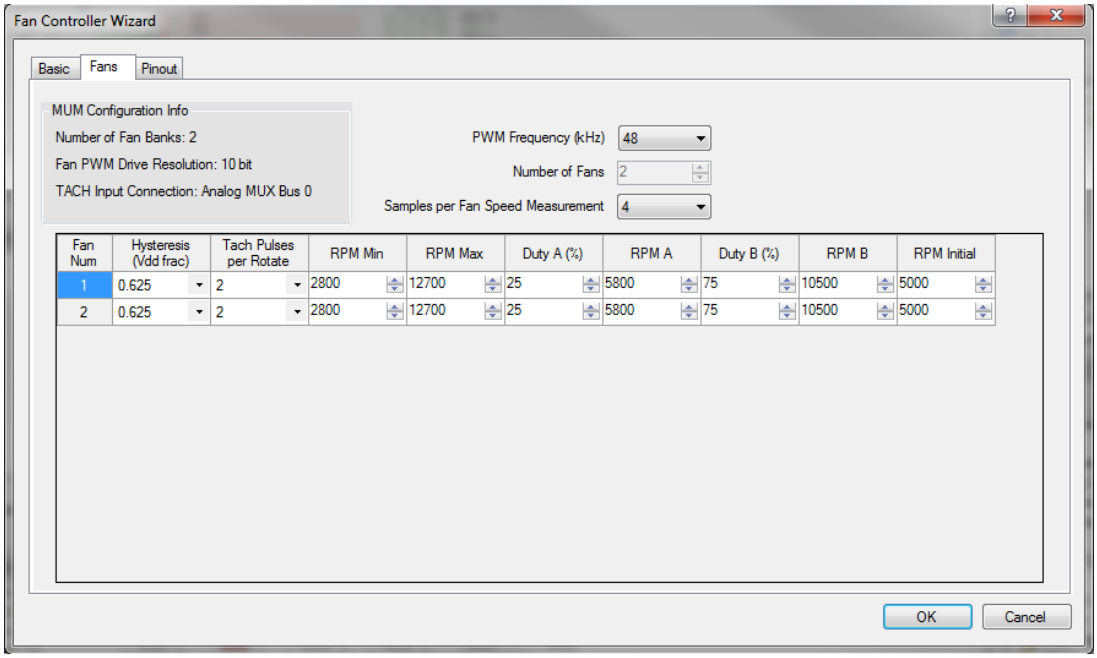
Parameters for control loop period, tolerance, acoustic noise reduction, and speed regulation apply only to projects with closed-loop speed regulation (projects 3 and 4). The parameter for control loop period controls how frequently the PWM duty cycle is adjusted for each fan.

The tolerance parameter can be adjusted from 1 to 10 percent. The lower the tolerance is, the lower the error will be between desired and actual rpm of the fans.

Alerts can be generated for fan stall/rotor lock for both open- and closed-loop methods of speed regulation. A speed regulation failure alert can be generated only in case of closed-loop speed regulation. You can route the Generated alert to a GPIO by selecting the Enable alert pin.

The Fans tab enables you to configure all parameters related to the PWM fan drivers (see Figure 16). You can set the PWM frequency to 24 kHz or 48 kHz. The industry standard for PWM drive signals is 24 kHz, but this controller will support fans that allow higher PWM frequencies.

Figure 16. Fan Controller Wizard Fans Tab

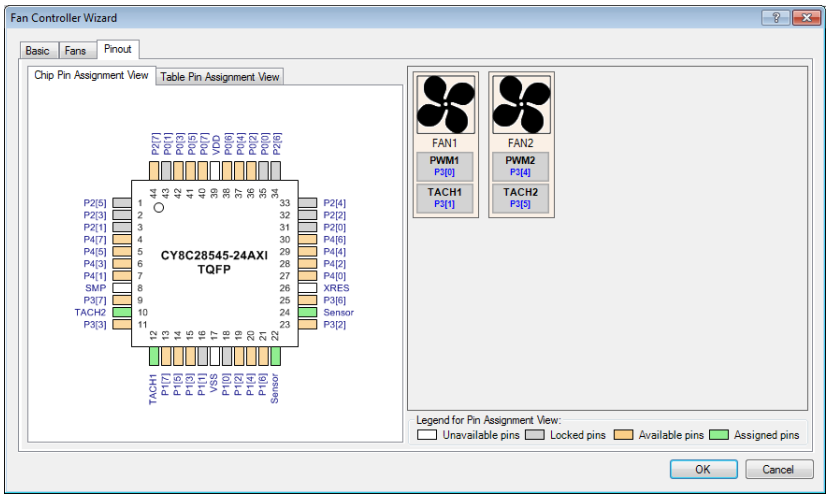


In the final section of this tab, you can enter the electromechanical parameters of the fans you are working with. You can find that information in the fan manufacturer's datasheet. The parameters for duty A (%), rpm A, duty B (%), and rpm B represent two data points from the fan's duty-cycle-to-speed chart. The tach pulses per rotation set the number of pulses that are produced per revolution, which depends on the number of poles in the fan. The default fan parameter indicated may be slightly different than the one shown in your fan datasheet.

The parameters in the Fans tab are the same for all example projects.

The Pinout tab (see Figure 17) enables you to map the signals to physical PSoC 1 port pins. The pinout is based on the CY8CKIT-036 connection to CY8CKIT-001. To assign the pin information, drag and drop the PWM/Tach connection to available pins in the Chip Pin Assignment view or Table Pin Assignment view.

Figure 17. Fan Controller Wizard Pinout Tab for Closed-Loop Fan Control



If the fans are arranged in banks, then one PWM will drive all of the fans in a bank and each fan will have its own tachometer signal (see Figure 18). In that case, different fans in a bank will share the same PWM pin and each fan will have a tachometer signal routed to individual pins. The Pinout tab in Figure 19 shows an allocation of individual fans to fan banks, with fans 1 and 2 allocated to Fan Bank 1, and fans 3 and 4 allocated to Fan Bank 2.

Figure 18. Fan Controller Wizard Pinout Tab for Open-Loop Fan Control with Fan Banks Equal to Fan Number

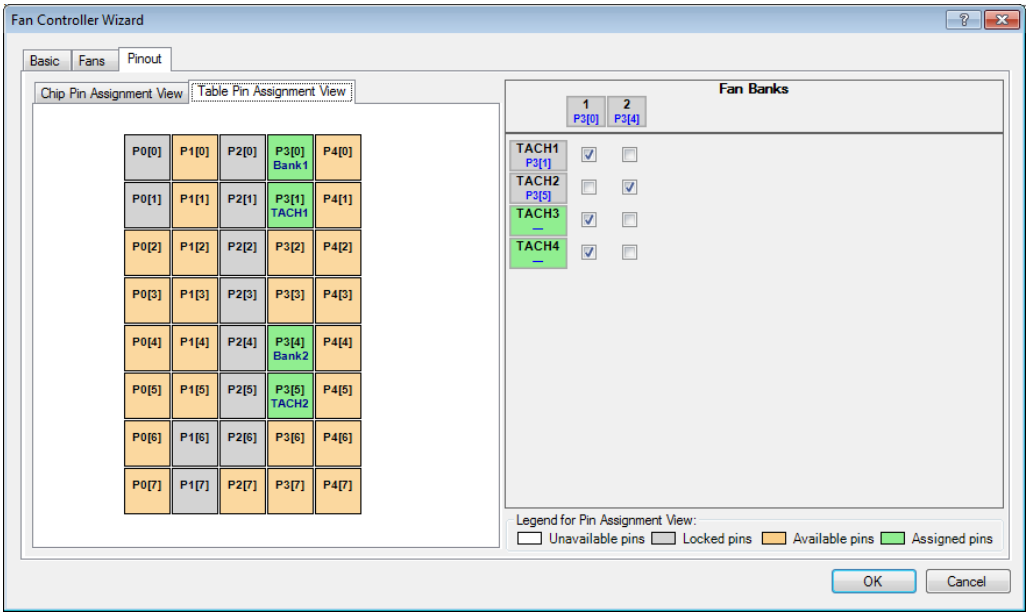
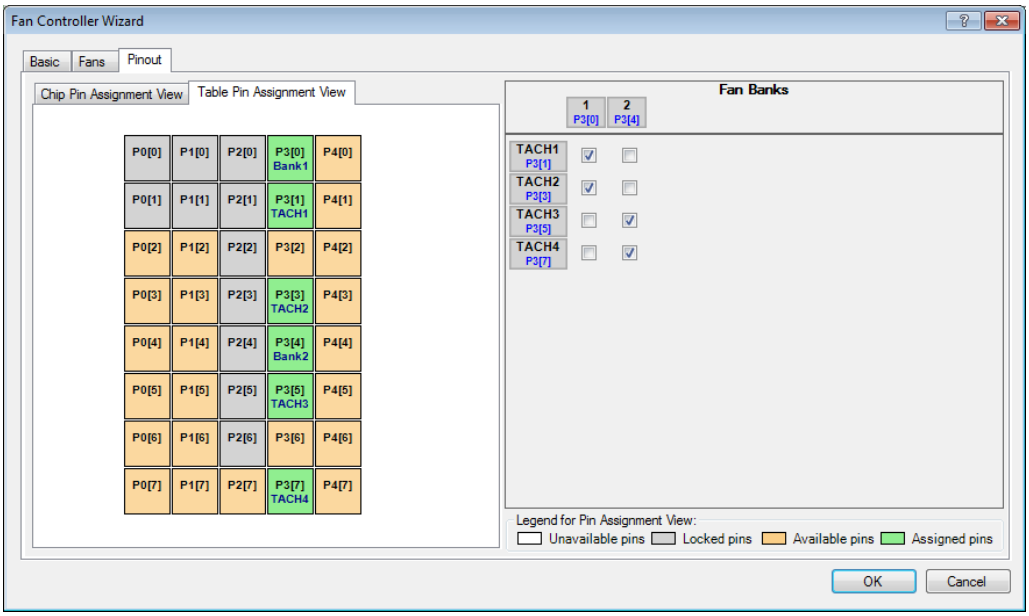


Figure 19. Fan Controller Wizard Pinout Tab for Open-Loop Fan Control with Four Fans in Two Fan Banks



Example 1: Open-Loop Fan Control with I²C

Example 1 uses a slave I²C interface on top of the fan controller user module. The I²C interface is used to:

- Send fan parameters to the host controller
- Receive duty cycle values from the host

In this example, the fan controller receives duty cycle information from the host and sets the fan PWM duty cycle accordingly. The host therefore controls the speed through the fan controller. Figure 20 represents the flow of Example 1.

The MiniProg3 programmer is also capable of implementing a USB-to-I²C bridge adapter. This feature has the advantage of sending and receiving commands to PSoC 1 over I²C. Numerous files intended for use with the bridge control panel software have been provided and are located in the folder called Bridge Control Panel Files.

To launch the bridge control software GUI, double-click the relevant *.iic* file (Fan_Open_Loop.iic or Fan_Closed_Loop.iic). If your computer does not recognize the *.iic* file extension, manually launch the bridge control panel from the Start menu under the Cypress folder. Open the relevant *.iic* file manually. Then select **Chart** > **Variable Settings** from the pull-down menu and click **Load**. Open the*.ini* (Open_Loop.ini or Closed_Loop.ini) file provided.

Figure 21 illustrates how the bridge control panel GUI looks when launched and properly configured.

Leave the DVK powered on and connect the white five-pin connector on the MiniProg3 to the five-pin header J5 (the same port used for programming) on the CY8CKIT-001. Make sure that the green Connected and Powered status boxes are visible at the bottom of the bridge control panel.

Figure 20. Open-Loop Fan Control Through I²C

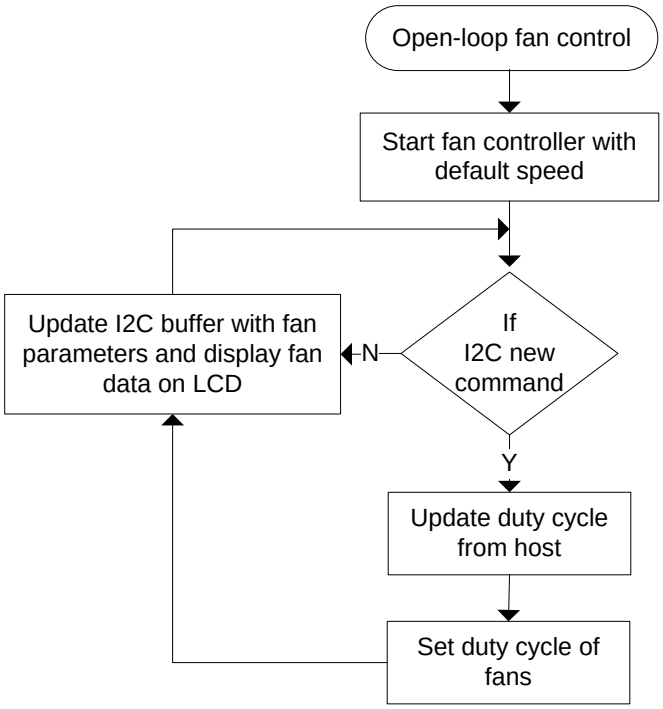
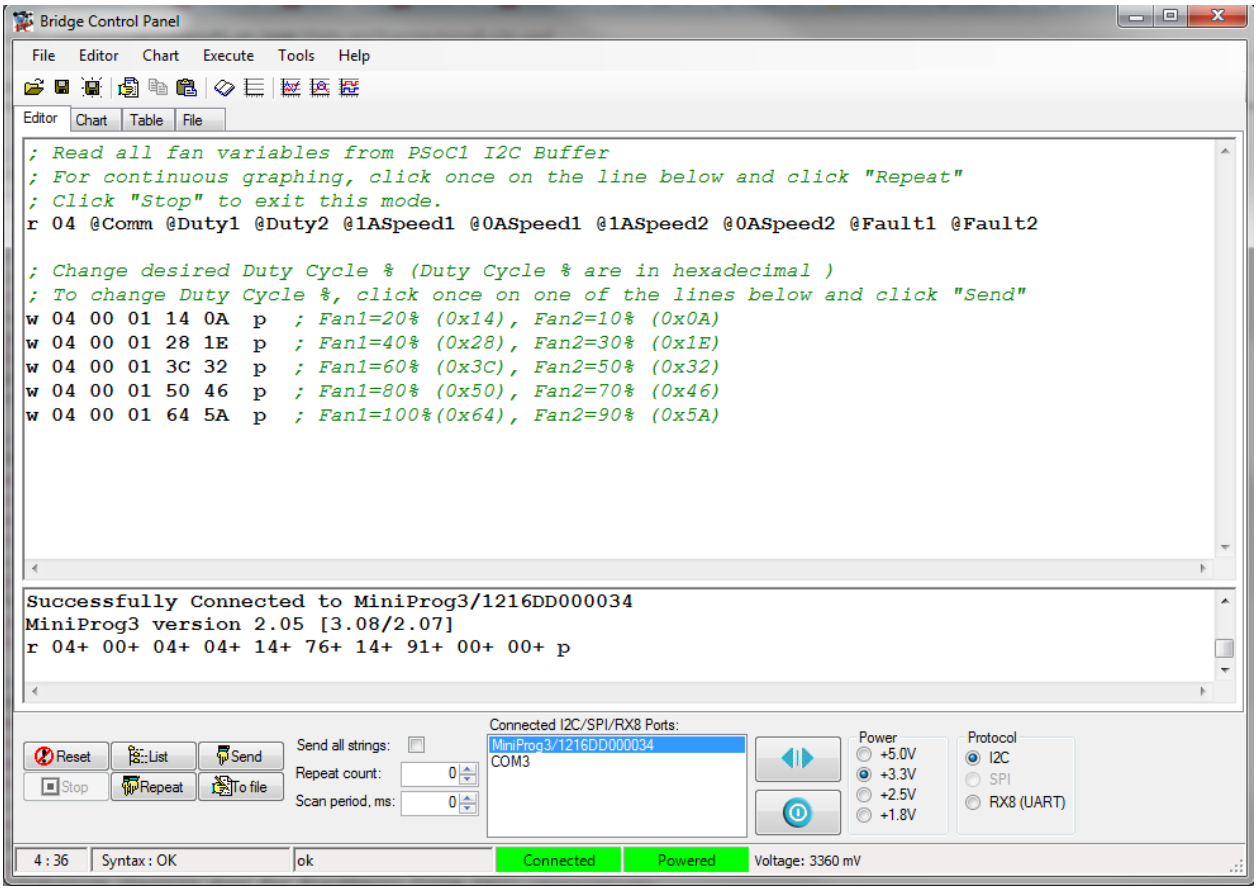


Figure 21. Bridge Control Panel GUI



The Editor tab is exposed by default whenever the bridge control panel software is opened. The editor provides a simple text scripting capability to send and receive I²C data. For full details on the scripting language, refer to the Help provided in the USB-I²C bridge control panel software. The script provided for working with the Example 1 project is shown in Code 1.

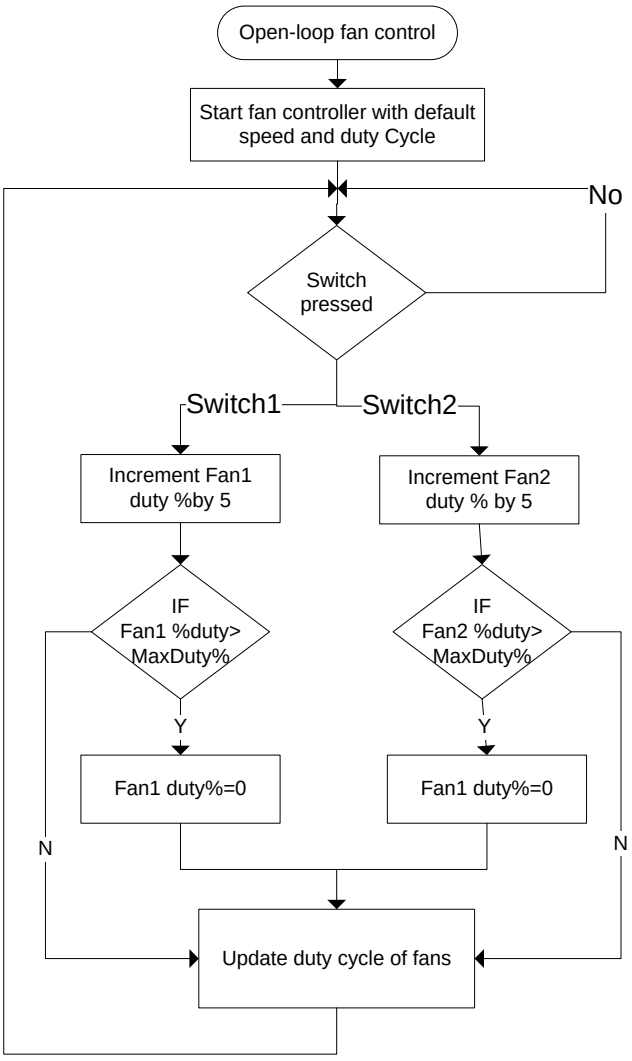
Code 1. USB-I²C Bridge Control Panel Script for Example 1

```
; Read all fan variables from PSoC1 I2C Buffer
; For continuous graphing, click once on the line below and click "Repeat"
; Click "Stop" to exit this mode.
r 04 @Comm @Duty1 @Duty2 @1ASpeed1 @0ASpeed1 @1ASpeed2 @0ASpeed2 @Fault1 @Fault2

; Change desired Duty Cycle % (Duty Cycle % are in hexadecimal )
; To change Duty Cycle %, click once on one of the lines below and click "Send"
w 04 00 01 14 0A  p  ; Fan1=20% (0x14), Fan2=10% (0x0A)
w 04 00 01 28 1E  p  ; Fan1=40% (0x28), Fan2=30% (0x1E)
w 04 00 01 3C 32  p  ; Fan1=60% (0x3C), Fan2=50% (0x32)
w 04 00 01 50 46  p  ; Fan1=80% (0x50), Fan2=70% (0x46)
w 04 00 01 64 5A  p  ; Fan1=100%(0x64), Fan2=90% (0x5A)
```

Example 2: Open-Loop Fan Control with Switches

Example 2 uses switches to control duty cycle on the fan controller user module. Switches are connected to Port 1.6 (SW1) and Port 1.7 (SW2) with ports in pull-up mode; the fans are started in default speed and duty cycles. Each switch controls the corresponding fan duty cycle. At each switch press, the duty cycle is increased by 5 percent. When fan duty cycle goes beyond maximum duty cycle (MaxDuty = 100 percent), duty cycle is set to 0 percent. Figure 22 represents the flow of this project.

Figure 22. Open-Loop Fan Control with Switches



Example 3: Closed-Loop Fan Control with I²C

Example 3 uses a slave I²C interface on top of the fan controller user module. The interface is used to:

- Send fan parameters to the host controller
- Receive the speed and tolerance values from the host

The project calls FanController_Service in FanController.c for closed-loop control with default speed and tolerance values. The fan receives the desired speed and speed tolerance from the I²C master, updates the fan operating parameters, and displays fan data on the LCD. Figure 23 represents the flow of this example project.

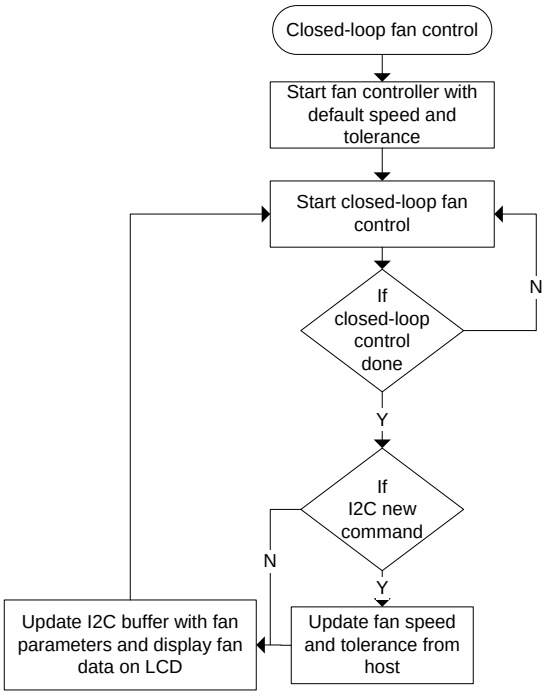
The files referenced here are Fan_Closed_Loop.iic and Closed_Loop.ini. The script provided for working with the Example 3 project is shown in Code 2.

The I²C write commands in the script enable you to change the desired speed independently for the two fans. These commands can be used to see how the fan controller responds to speed changes and to verify the dynamic behavior of the control loop. To send a command, click the script line once and click the Send button or simply press Enter.

Disturbing the fan's airflow by either blocking the air intake or blowing air against the flow causes noticeable changes in fan speed in the open-loop control mode.

When the closed-loop control is enabled, the fan controller will respond quickly and return the fans to their desired speed within the tolerance.

Figure 23. Closed-Loop Fan Control with I²C



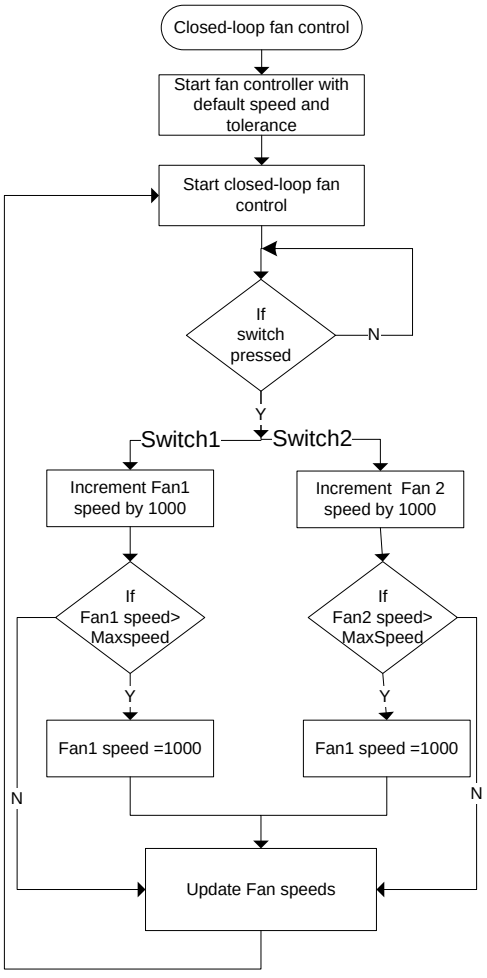
Code 2. USB-I²C Bridge Control Panel Script for Example 3

```
; Read all fan variables from PSoC 1 I2C Buffer
; For continuous graphing, click once on the line below and click "Repeat"
; Click "Stop" to exit this mode.
; r Slave_address [ Buffer_Pointer Write_command  MSB_DesiredSpeed_Fan1 LSB_DesiredSpeed_Fan1
; MSB_DesiredSpeed_Fan2 LSB_DesiredSpeed_Fan2
; MSB_ActualSpeed_Fan1 LSB_ActualSpeed_Fan1 MSB_ActualSpeed_Fan2
LSB_ActualSpeed_Fan2          ; Fan1Fault Fan2Fault
r 04 @Comm @1DSpeed1 @0DSpeed1 @1DSpeed2 @0DSpeed2 @1ASpeed1 @0ASpeed1 @1ASpeed2 @0ASpeed2 @Fault1 @Fault2 p

; Change desired speeds and tolerance %(speeds and tolerance % are in hexadecimal - MSB first)
; To change speeds and tolerance %, click once on one of the lines below and click "Send"
; Write all fan variables from PSoC 1 I2C Buffer

; w Slave_address Buffer_Pointer Write_command MSB_DesiredSpeed_Fan1
; LSB_DesiredSpeed_Fan1 MSB_DesiredSpeed_Fan2 LSB_DesiredSpeed_Fan2

w 04 00  01 2E E0 2E E0  p  ; Fan1=(12000rpm)(0x2EE0), Fan2=(12000rpm)(0x2EE0)
w 04 00  01 1B 58 1B 58  p  ; Fan1=(7000rpm) (0x1B58), Fan2=(7000rpm) (0x1B58)
w 04 00  01 23 28 23 28  p  ; Fan1=(9000rpm) (0x2328), Fan2=(9000rpm) (0x2328)
w 04 00  01 1B 58 23 28  p  ; Fan1=(7000rpm) (0x1B58), Fan2=(9000rpm) (0x2328)
w 04 00  01 23 28 1B 58  p  ; Fan1=(9000rpm) (0x2328), Fan2=(7000rpm) (0x1B58)
w 04 00  01 0F A0 0F A0  p  ; Fan1=(4000rpm) (0x0FA0), Fan2=(4000rpm) (0x0FA0)
```

Example 4: Closed-Loop Fan Control with Switches

Example 4 uses switches to control the speed on the fan controller user module. Switches are connected to Port 1.6 (SW1) and Port 1.7 (SW2) with ports in pull-up mode. The fans are started in default speed and the closed-loop FanController_Service function from FanController.c is called to perform the closed-loop fan control. The switches in this case increase corresponding fan speeds by 1,000 rpm. Figure 24 represents the flow of this project.

Figure 24. Closed-Loop Fan Control with Switches



Example 5: PSoC 1 Thermal Management

This example project demonstrates how the temperature sensors, together with the fans on the PSoC Thermal Management EBK, create a complete thermal management system. It shows how to combine readings from numerous temperature sensors and use the composite temperature to set desired fan speeds according to a customized transfer function.

The thermal management example project uses the concept of a thermal zone, which describes:

- How to combine multiple temperature sensor readings to form a composite zone temperature
- How to map the zone temperature to a fan speed

By definition, each fan is controlled according to its own independent thermal zone. This example has two thermal zones, because only two fans are installed on the PSoC Thermal Management EBK.

Algorithms currently implemented to combine multiple temperature sensors into a composite zone temperature include straight average, weighted average, and maximum. The straight average algorithm returns the average value of all sensors used in that particular zone. The weighted average algorithm first applies the weight to each sensor temperature as defined in the zone configuration and then takes an average of the result. The maximum algorithm simply returns the maximum temperature from sensors in that particular zone. You can select an algorithm for a particular zone by changing the zone configuration.

This example project uses the weighted method on both fans. A zone-temperature-to-fan-speed transfer function is then defined for each zone. In this project, the transfer function is table driven on both fans; that is, a look-up table maps the composite zone temperature to fan speed. The configuration (see Table 2) is as follows:

- Two temperature zones: Zone 1 and Zone 2
- Two four-wire brushless DC fans: Fan 1 and fan 2 are installed in Zone 1 and Zone 2, respectively
- Four temperature sensors

Table 2. Zone Configuration

Label	Temperature Sensor	Installed In	Weight
U1 in CY8CKIT-036	I²C output temperature sensor – TMP175	Zone 1	10%
R20 in CY8CKIT-001	Diode temperature sensor		90%
U2 in CY8CKIT-036	One-wire temperature sensor – DS1820	Zone 2	90%
U3 in CY8CKIT-036	PWM temperature sensor – TMP05		10%

Example 5 simulates a thermal management system. Zone 1 combines temperature measurements from two temperature sensors (one analog and one digital). The analog sensor is simulated using a variable potentiometer to allow easy demonstration of fan control over a wide simulated temperature range without the need for an environmental chamber to cycle through temperatures. In Zone 1, the temperature sensors are combined using a weighted average, where the potentiometer is given a 90 percent weight and the I²C digital temperature sensor (U1 on the PSoC Thermal Management EBK) is given a 10 percent weight. You can adjust the potentiometer (R20 on the CY8CKIT-001 DVK) to vary the simulated temperature value. The Zone 1 speed transfer function is table driven. As the temperature increases, fan speed increases, as Table 3 shows. However, when the temperature decreases, the speed will decrease only when the temperature change is more than hysteresis. For example, the speed will change at 30 degrees if the current temperature is at least 35 degrees, because hysteresis is set to 4 degrees for Zone 1.

Table 3. Zone 1 Thermal Profile

Temperature (°C)	Fan Speed
0–14	5,000 rpm
15–34	5,500 rpm
35–54	6,500 rpm
55–74	8,500 rpm
75 and above	10,500 rpm

Zone 2 consists of two temperature sensors and a single fan. The Zone 2 speed transfer function is table driven and is shown in Table 4. Note that the temperature range is narrow and is close to room temperature, which allows for simple testing. (You can just touch a temperature sensor with a warm finger to cause the fan speed to change.) In Zone 2, the temperature sensors are combined using a weighted average: The one-wire temperature sensor (U2 on the PSoC Thermal Management EBK) contributes 90 percent of the weight; the PWM temperature sensor (U3 on PSoC Thermal Management EBK) contributes 10 percent. In this example, the U2 temperature reading will dominate the overall zone temperature calculation.

As the temperature increases, fan speed increases, as Table 4 shows. However, when the temperature decreases, the speed will decrease only when the temperature change is more than hysteresis. For example, the speed will change at 25 degrees if the current temperature is at least 27 degrees, because hysteresis is set to 1 degree for Zone 2.

Table 4. Zone 2 Thermal Profile

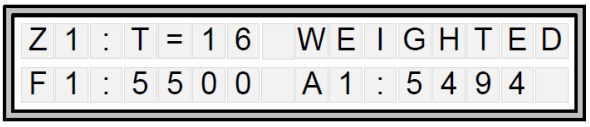
Temperature (°C)	Fan Speed
0–22	5,000 rpm
23–24	6,000 rpm
25–26	7,000 rpm
27–28	9,000 rpm
29 and above	10,000 rpm

The LCD screen displays status information for the thermal management system across three screens. You can cycle through the status screens by pressing SW1 on the CY8CKIT-001 DVK.

Screen 1: Zone 1 Summary

Screen 1 shows the current status of Zone 1, as Figure 25 illustrates. Line 1 of the screen displays the zone number, the current composite zone temperature, and the type of zone temperature calculation algorithm. Line 2 displays the desired fan speed and the actual fan speed for Zone 1. Note that the actual fan speed may not be the same as the desired fan speed. The maximum tolerance is 5 percent.

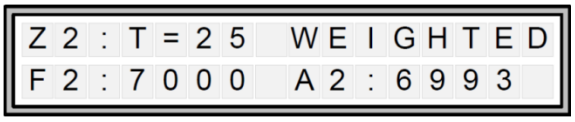
Figure 25. Zone 1 Summary



Screen 2: Zone 2 Summary

Screen 2 shows the current status of Zone 2, as Figure 26 illustrates. Line 1 of the screen displays the zone number, the current composite zone temperature, and the type of zone temperature calculation algorithm. Line 2 displays the desired fan speed and the actual fan speed for Zone 2. Note that the actual fan speed may not be the same as the desired fan speed. The maximum tolerance is 5 percent.

Figure 26. Zone 2 Summary



Screen 3: Temperature Sensor Summary

Screen 3 shows the current temperature sensor readings for all sensors in the system, as Figure 27 illustrates. Line 1 of the screen displays the Zone 1 temperature sensor values. The leftmost temperature reading is the zone's composite temperature, followed by the temperature of each contributing sensor. Line 2 displays the same information for Zone 2.

Figure 27. Temperature Sensor Summary

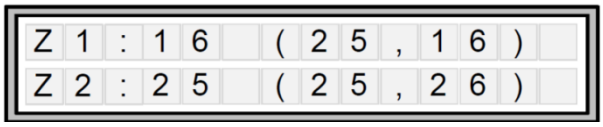
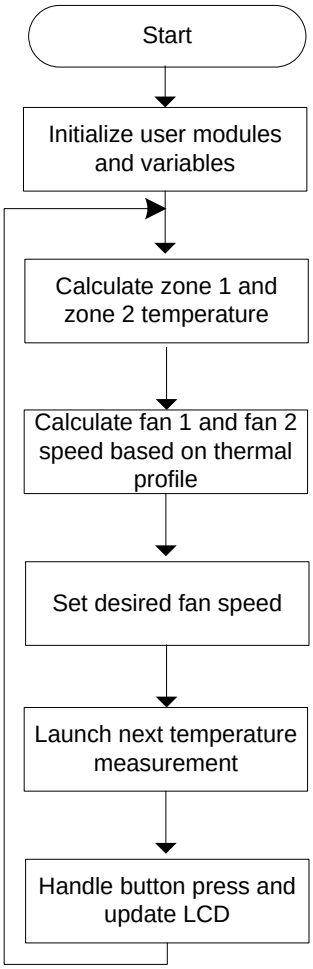


Figure 28 illustrates the basic function of the thermal manager in *thermalmanager.c*, which implements the main service loop.

Figure 28. Thermal Manager Flowchart



Summary

The Fan Controller User Module enables you to quickly and easily create thermal management solutions with minimal firmware development.

The unique ability of the PSoC architecture to combine custom digital logic, analog functions, and an MCU in a single device lets you integrate many external fixed-function ASSPs. This powerful integration capability not only reduces BOM cost, but also results in PCB board layouts that are less congested and more reliable.

About the Author

Name:	Sanjeev Kumar K.
Title:	Senior Applications Engineer
Background:	Sanjeev has a bachelor's degree in Electronics and Communication from the College of Engineering, Guindy, Chennai. He currently works on PSoC 1 based applications at Cypress.

Document History

Document Title: PSoC® 1 – Intelligent Fan Controller – AN78692

Document Number: 001-78692

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	3598263	KUK	05/02/2012	New Application Note.
*A	3821271	KUK	11/29/2012	Updated to new template.
*B	4210991	KUK/SGUP	1/24/2014	Fan Controller IP has been replaced with Fan Controller User Module. Content updated throughout the document. Added Thermal Management project. Edited kit sections and setup details. Updated copyright date.
*C	4736944	ROWA	4/23/2015	Updated template. No technical updates. Completing sunset review.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc cypress.com/go/plc
Memory	cypress.com/go/memory
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/usb
Wireless/RF	cypress.com/go/wireless

PSoC® Solutions

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/go/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com