Application Report
SLAA259 – November 2005

Digital Fan Control with Tachometer using MSP430

Randy Wu MSP430

ABSTRACT

This application note describes a fan controller system designed using the MSP430F417 microcontroller (MCU). The associated source code files are provided in a downloadable zip file along with the application report. This application report along with the source code can be used as a production-ready foundation for designs using multiple fan controls including the support for tachometer functionality.

1 Introduction

Cooling fans are commonly used to blow out hot air generated in a system. Typical examples are power supply racks or network servers. A very simple cooling system would be a cooling fan running continuously at full speed. Even though this serves the purpose it consumes a lot of power and generates a lot of noise. Cooling fans can be used more efficiently if their speed is set according to the temperature of the system. This results in a closed loop system where the cooling fan tries to maintain a stable system temperature by constantly adjusting its speed according to the current temperature.

In this application example such a system is demonstrated using MSP430 microcontroller to measure temperature and control the fan speed using its PWM output. A Softbaugh™ ES417 development board is used in this application for convenience. The ES417 has an MSP430F417 with access to its port pins and an LCD display. The LCD is used to display system status, temperature, and fan speed (tachometer) while the cooling fan is running.

The code can be easily adapted to run on any MSP430 device and can be customized for additional functionality. A 3-wire and a 4-wire DC Brushless cooling fans control systems are demonstrated.

The example source code is implemented both in C and Assembly languages. The software incorporates the following functions:

- System monitoring and control services
- Tachometer capture and LCD drivers
- Pulse Width Modulation (PWM) drivers
- Temperature sensor drivers

Figure 1 shows the software flow chart for this application.

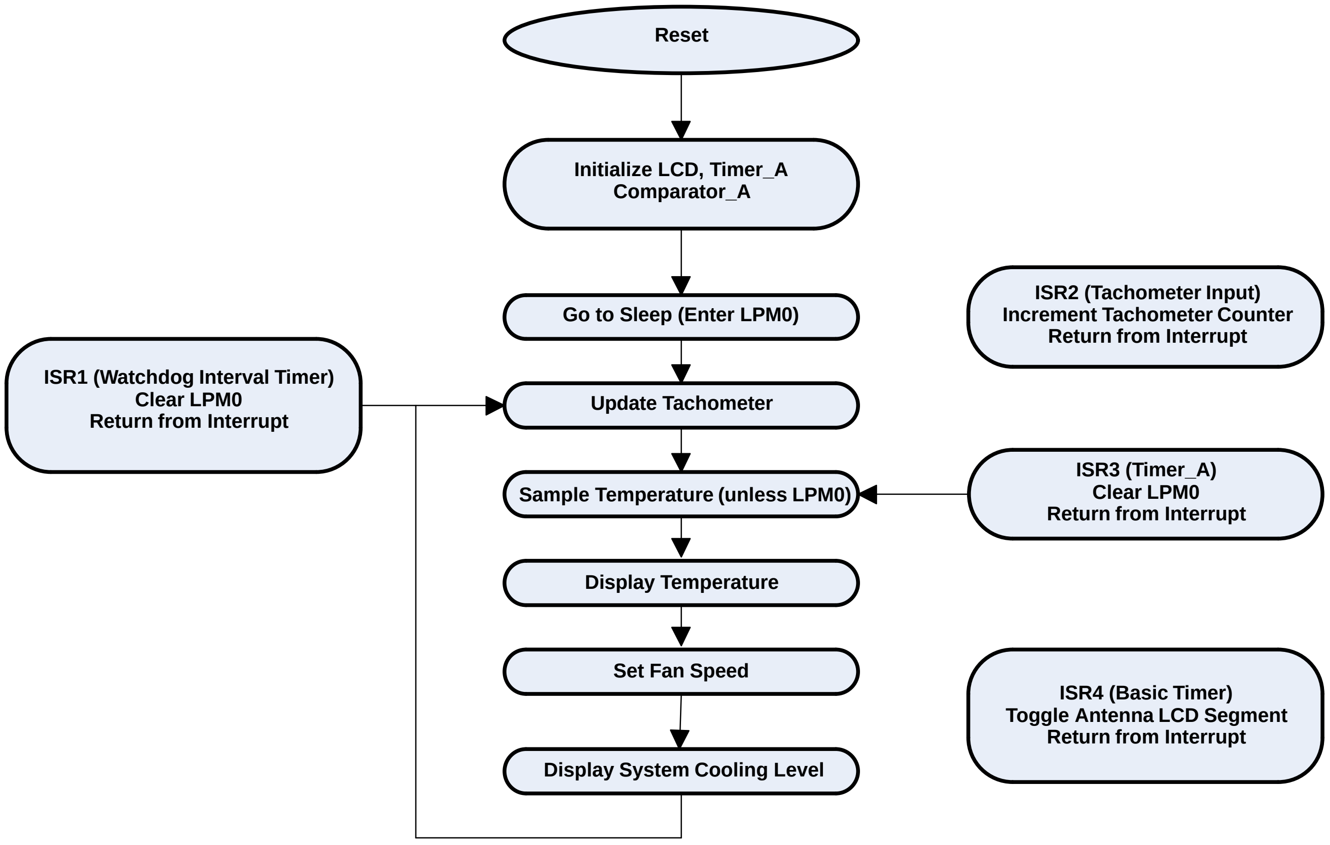
Reset
Initialize LCD, Timer_A Comparator_A
Go to Sleep (Enter LPM0)
Update Tachometer
Sample Temperature (unless LPM0)
Display Temperature
Set Fan Speed
Display System Cooling Level
ISR1 (Watchdog Interval Timer) Clear LPM0 Return from Interrupt
ISR2 (Tachometer Input) Increment Tachometer Counter Return from Interrupt
ISR3 (Timer_A) Clear LPM0 Return from Interrupt
ISR4 (Basic Timer) Toggle Antenna LCD Segment Return from Interrupt

Figure 1. Software Flow Chart

2 System Architecture

This section describes how to set up the development platform. The following hardware components are required:

- SoftBaugh ES417 Development Board
- MSP430 USB Debugging Interface (MSP-FET430UIF)
- DC Brushless Cooling Fan (3-wire or 4-wire interface) (e.g. Delta part no. FFB1212SHE-F00)
- Fan power supply

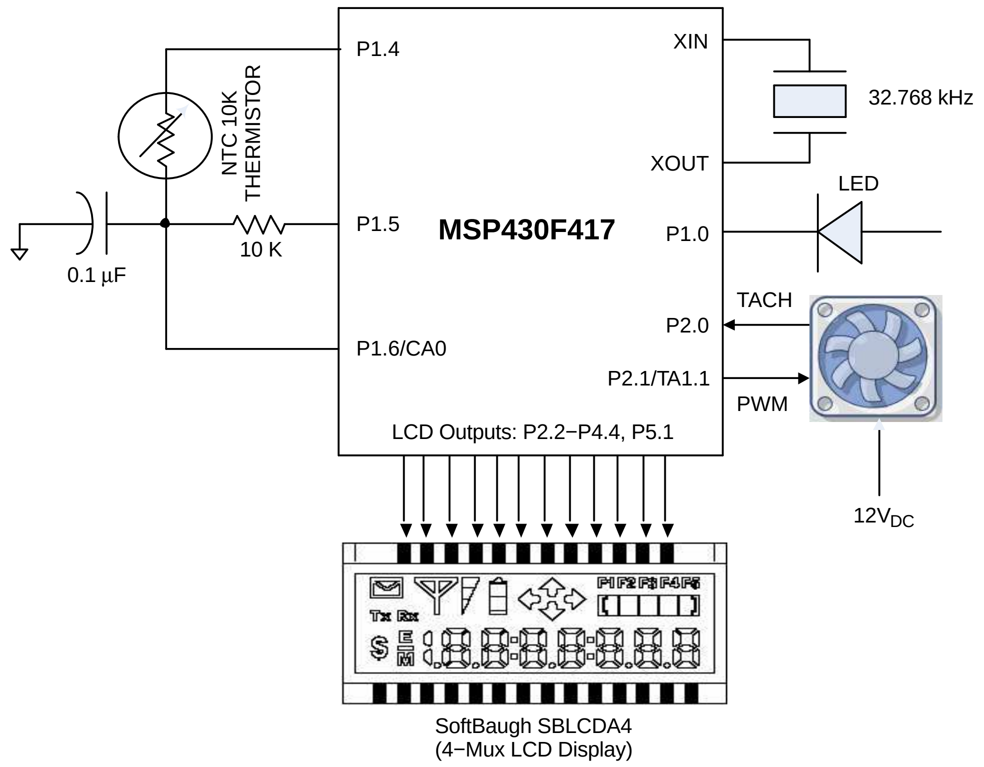

Figure 2. MSP430 Hardware Block Diagram

The provided MSP430 software allows the user to easily connect a cooling fan of choice to a readily-available MSP430 development board such as the SoftBaugh ES417 development board, of which a complete running example has been provided. Since most fans have a built-in frequency generator signal which increases with fan speed, the MSP430 can monitor this signal to implement a tachometer for measuring the speed of the fan.

The following figures show how to connect a typical DC Brushless cooling fan (3 or 4 wire interface) to the MSP430 micro-controller on the SoftBaugh ES417 development board in order to run the provided sample code. Standard 12 VDC cooling fans are used in this example; refer to either Figure 1 or Figure 2 based on the number of wires.

Refer to your specific fan's operational parameters in the datasheet to ensure proper interconnection circuitry. Care needs to be exercised when selecting the right transistors and resistor values to meet the inrush current draw for the type of fan(s) being used. The provided figures serve as typical scenarios for 3-wire and 4-wire 12VDC cooling fans which have a built-in Frequency Generator signal for counting revolutions.

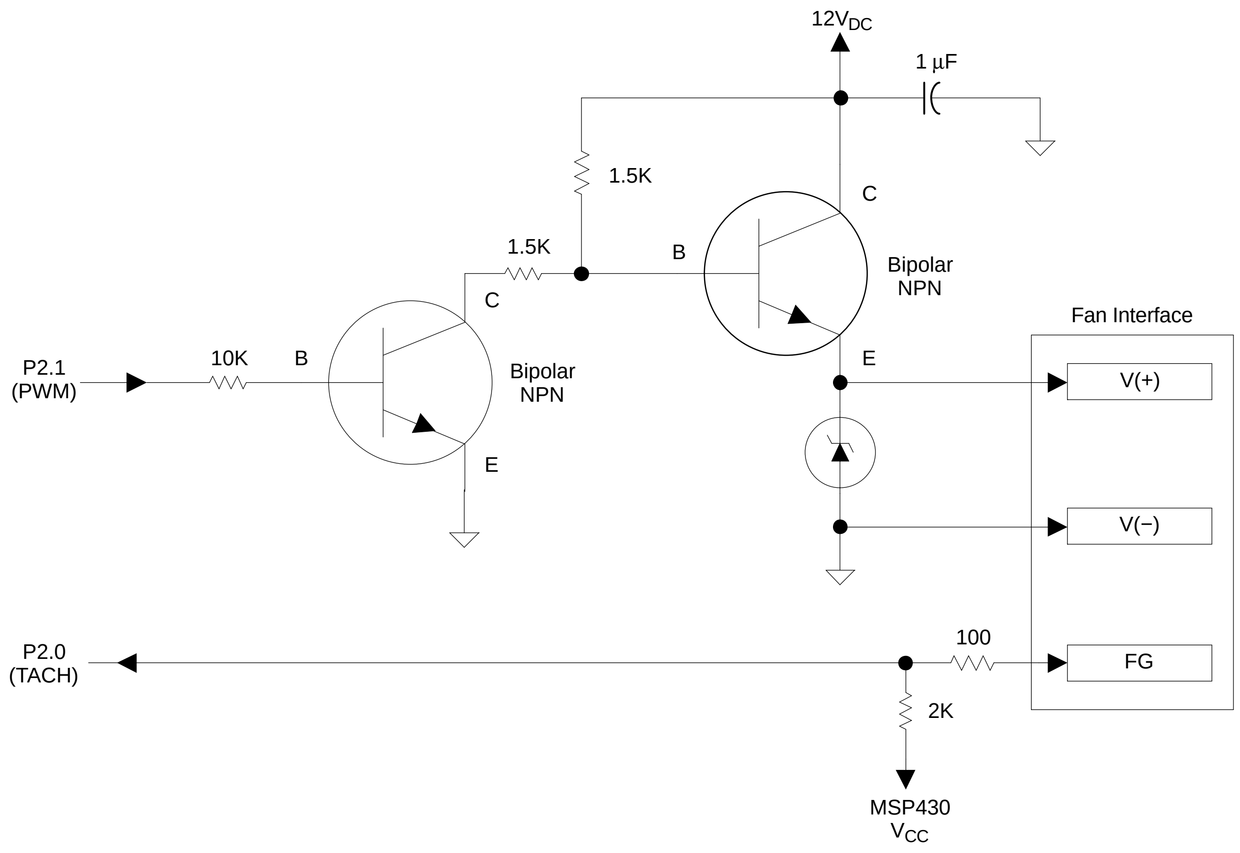

Figure 3. Typical Cooling Fan Interface Circuitry (3 Wires)

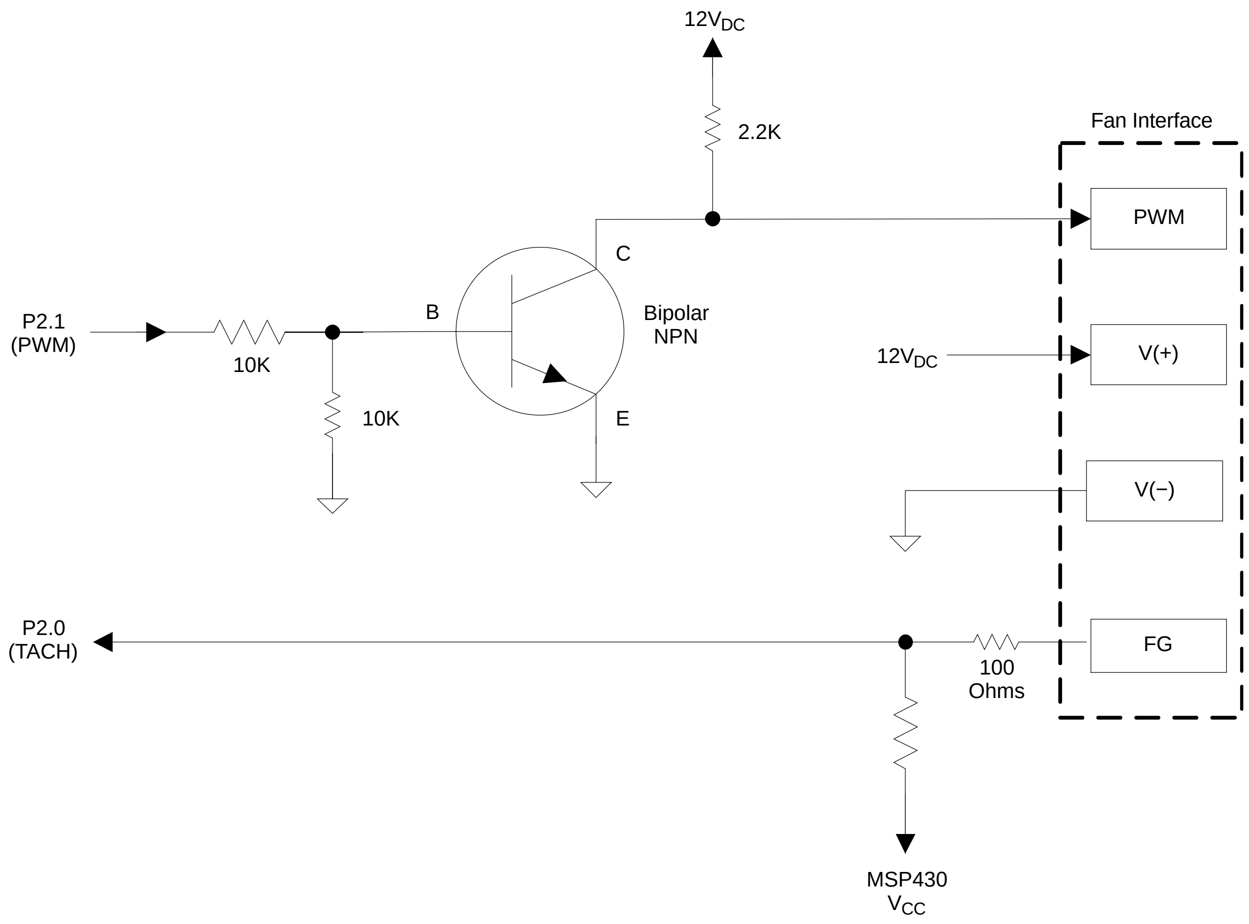

Figure 4. Typical Cooling Fan Interface Circuitry (4 Wires)

The example software has 6 speed levels defined for the system (L0 through L5). Additional temperature band levels can be added to the software easily if required. Each level is defined by low and high temperature limits and defines the speed of the fan that must be maintained while the temperature reading falls in a specific level.

When the temperature reading increases and crosses the boundary into the next level, the fan speed increases accordingly (conversely, the fan speed decreases when the temperature reading drops to a lower level). The provided software example keeps track of the current system level setting which is automatically set by the MSP430 based on the temperature reading of the external thermistor.

The LCD on the ES417 board will constantly update the RPM of the cooling fan as well as the temperature as measured at the externally-connected thermistor. As the temperature of the thermistor increases, the system level will increase and speed up the operation of the cooling fan.

The system cooling level is also displayed on the LCD. For example, if an F2 is displayed in the upper-right hand corner of the LCD screen, the system is currently running at cooling level 2. A total of 5 levels (denoted by F1 through F5) are defined in the example application; each are denoted by the corresponding F<x> segments on the LCD screen where <x> = current system level setting.

The horizontal power bar located right under the F<x> row is also used to display the system level as a quick visual indicator of the fan’s “cooling strength”. The UP and DOWN arrows are used to indicate if the fan speed is increasing or decreasing from the previous reading.

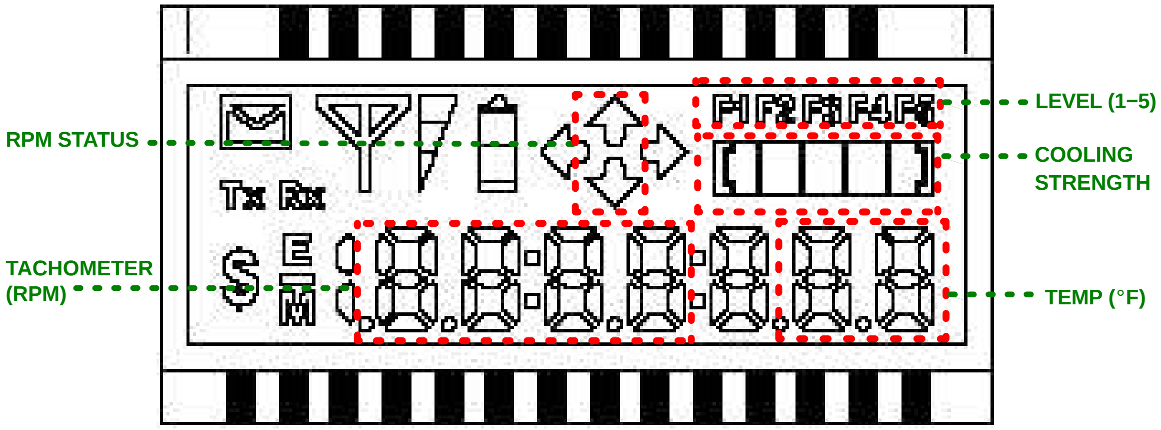

Figure 5. Fan Control System User Interface using LCD Display (SBLCDA4)

2.1 Connecting an External Temperature Sensor to the MSP430

There are a few ways that the MSP430 micro-controller can take periodic temperature readings during run-time of the system. If the designer wishes to use the MSP430's own temperature as the criteria for changing the speed of the fan(s), most MSP430 devices have their own built-in temperature sensor which can be treated as an additional Analog-to-Digital Converter channel. Otherwise, if the temperature readings need to be taken at some point outside and somewhat far away from the micro-controller board, then a thermistor can be strategically placed at the desired point in the system and routed to the fan controller board with 2 wires as part of an overall cable harness.

A typical 10-kΩ negative temperature coefficient (NTC) thermistor (variable resistor) can be easily connected to the Comparator_A input of the MSP430F417. The resistance changes based on the ambient temperature and thus will change the DC voltage being monitored by the ADC input channel. The MSP430 can then determine the temperature based on the measured voltage and adjust the speed of the fan(s) in the system accordingly.

Since temperature is an analog measurement, it must be converted to a digital format before it can be analyzed by the micro-controller. To reduce cost, complexity, and board size, the Slope A/D technique can be used instead of using a device which has a built-in ADC module. Having to rely on a discrete ADC module adds extra cost to the overall system.

The on-chip Comparator_A module in the MSP430 is used to perform the analog-to-digital conversion. The technique is based on the charging/discharging of a capacitor with a known capacitance. The number of clock cycles it takes to charge/discharge the capacitor is then counted by using the interrupt-driven capture functionality of the Timer_A module. Refer to the application note listed in the References section for a more detailed description of the Slope A/D conversion technique.

In the provided system software example, the resistance values have been conveniently provided for every temperature between -50°C and +99°C in 1°C increments. The supplied Slope A/D temperature driver will translate the variable resistance of thermistor into the corresponding temperature. The MSP430 can then take the appropriate action based on the latest temperature reading. The provided sample code is based on an NTC thermistor with a characteristic of Beta = 3895 (ratio = 9.05) from 0°C to 50°C. Software modifications will be required if using a thermistor with drastically different characteristics to obtain accurate temperature readings.

2.2 System Monitor and Fan Control Logic

The MSP430 must be responsible for monitoring alarm conditions as well as controlling the operation of the cooling fans of the system. A typical fan requires a Pulse Width Modulation (PWM) signal to dictate how fast the fan should be turning. To generate a PWM signal, the PWM frequency and duty cycle of the signal must be pre-determined for each speed level required (normally measured in RPMs, or Revolutions Per Minute).

All MSP430 devices have at least 1 hardware timer with at least 3 associated Capture and Control Registers (CCRs). Thus, all MSP430 devices can generate at least 2 PWM signals of the same PWM frequency and different duty cycles. By updating the value of a CCR, the MSP430 can effectively change the speed(s) of the fan(s) by increasing or decreasing the duty cycle of the generated PWM signal which is fed into the PWM input of the cooling fan, without producing any audible PWM noise.

Some cooling fans are equipped with a tachometer output (Frequency Generator signal) which can be fed as an input to the micro-controller. For example, a particular cooling fan may generate 2 pulses per complete revolution of the fan. The MSP430 can monitor these inputs and count the number of pulses and then translate that number into Revolutions Per Minute (RPM) of the cooling fan. This value could be reported back to the system master and possibly cause an alarm condition if the RPM value is out of the desired tolerance. An appropriate action can be taken by having the MSP430 turn the color of a bi-colored LED from green to red and/or sending a message to an external device such as its associated system master module.

In the provided system software example, the Watch Dog Timer is used as an interval timer to periodically check the temperature of the thermistor and then set the speeds of the fans according to the temperature reading of the slope A/D converter (which uses the comparator functionality plus a reference resistor)). The interrupt service routine (ISR) can be easily modified to incorporate the speed selection criteria of the specific fan controller application design. The provided source code contains the code to set up the watch dog timer to act as the system monitor to take periodic temperature readings, check for alarm conditions, and update the speed of the fan(s). The default interval time is 1 second based on a 32.768 kHz auxiliary clock (ACLK) crystal input.

2.3 Displaying the Tachometer on the LCD Screen

There are 4 numeric digits on the LCD screen of the ES417 development board (out of a possible 7) which are used to show the tachometer reading of the fan in RPM (revolutions per minute). The remaining 3 digits are used to display the temperature in degrees Fahrenheit as measured at the external thermistor using the Slope A/D conversion method. As the speed of the fan changes, the FG (frequency generator) output of the cooling fan will produce a signal which can be translated to RPM by the MSP430.

For example, a specific cooling fan with a built-in FG signal may generate 2 rising and falling edges for every complete revolution of the fan wheel. The information on the characteristics of the FG signal will be characterized in the cooling fan’s data specification. The MSP430 is programmed to count the number of rising or falling edges in a 1-second period, divide that number of pulses by 2 to get the number of revolutions during that 1-second period, and then multiply by 60 seconds to get RPM. The provided sample source code follows this translation method and displays a 4-digit result on the LCD screen which is updated every second.

3 Software Modifications for a Specific Fan

The following steps are recommended to modify the given example fan controller source code for any type of cooling fan.

3.1 Fan Operating Parameters

Most cooling fans will have a 4-wire interface consisting of Power, Ground, PWM, and FG signals. The datasheet specification for the fan should provide some type of formula and/or table which describes the PWM frequency required and the duty cycles of the PWM signal which will result in specific speeds of the fan (most likely in RPM). A Look Up Table can be implemented using a data array consisting of compare register values which result in a specific fan speed. The location of each value in the Look Up Table can correspond to the desired fan speed.

For example, a fan controller may need to set the speed of the fan(s) anywhere from 1000 RPM to 4000 RPM, in increments of 100 RPM resolution. As a result, an array of 31 values can be created so that the first value contains the compare register value which will result in a fan speed of 1000 RPM. The second array element corresponds to 1100 RPM and so on. The compare register values are pre-calculated and then stored as part of the MSP430’s non-volatile Flash memory so that the values do not need to be calculated on the fly during run-time of the system. The MSP430 can just read the required compare register value from the correct location within the array to change the speed of the fan(s) at any time by simply updating the compare register.

The provided source code contains an example of a Look Up Table which contains pre-determined compare register values for a fan which needs to be run in various increments of duty cycles. The size of this data array and the values in each location can easily be modified in source code for the specific requirements of the design.

The MSP430's internal frequency locked loop module (FLL) is capable of running at speeds above 1 MHz. The FLL's oscillator output can be used to source the timer which is used to derive the PWM signal. For example, most cooling fans require a PWM frequency of 20 kHz. Thus, one can program, in software, the FLL to generate a 1 MHz clock to source the 20 kHz PWM frequency signal used to drive the cooling fan(s). The provided source code contains a routine which can be modified to set the FLL to a specific frequency easily.

The MSP430's Timer_A is configured in UP-mode with SMCLK (FLL) as timer clock source and output unit Out1 set for output mode 3 (OUTMOD_3) to produce a PWM square wave output on the P2.1/TA1.1 output pin. The CCR0 value is set to the period corresponding to the PWM frequency to define Timer_A to count up to the end of each period. Changing the values in CCR1 varies the duty cycle of the PWM signal produced by Timer_A on the peripheral pin P2.1/TA1.1. In other words, CCR0 holds the Period value while CCR1 holds the Compare value to generate a PWM signal without any CPU intervention.

The PWM signal can be used to switch a transistor on/off which in turn supplies power to the DC motor of the fan. As the PWM duty cycle varies, the average power to the motor changes accordingly. This change in average power is what controls the speed of the fan's motor.

3.2 Temperature Operating Parameters

Most fan controller systems will define pre-determined temperature bands which correspond to a specific speed of which the fan(s) must be running. The provided source code defines 4 bands, or adjustment levels, so that when the temperature falls in a specific band, the micro-controller will adjust (or maintain) the fan speed accordingly.

At each level transition point is a hysteresis band where the speed of the fan(s) should not be adjusted since the current temperature is so close to the transition point between each level. For example, one could define a hysteresis gap of ±1°C at each transition point – meaning if the temperature falls within 1 degree of any transition point, the speed of the fan(s) should not change until the temperature reading goes beyond the hysteresis gap. This is normally done to prevent the speed of the fan(s) to switch back and forth sporadically while the system's temperature is fluctuating slightly between two adjustment levels.

The provided source code contains the logic to read the temperature of a 10-kΩ NTC thermistor (variable resistor) anywhere from -50°C and +99°C in 1°C increments. The corresponding temperature in degrees F is also provided if the cooling system is using Fahrenheit for the units of temperature. The different temperature level transitions are also implemented as well as the hysteresis gap for each transition point. The temperature band and hysteresis operating parameters can be modified in the provided source code header file.

4 References

1. *MSP430x4xx Family Users Guide* (SLAU056)
2. *MSP430x41x Mixed Signal Microcontroller* (SLAS340)
3. *Implementing an Ultra-low Power Thermostat with Slope A/D Conversion* (SLAA129)
4. SoftBaugh ES417 Development Board (www.softbaugh.com)

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless