Application Note

Z8 Encore! XP® Temperature Sensed Speed Control of 12 V BLDC Fan Reference Design

AN023403-0708

Z8 Encore! XP® Flash Microcontrollers

Abstract

This application note demonstrates the method for interfacing and controlling the 12 V brushless DC (BLDC) fan's speed with Zilog's Z8 Encore! XP®. The fan speed is controlled based on the objects temperature. When the temperature around an object exceeds the higher limit, the fan starts rotating and the speed is directly proportional to the rise in temperature. The fan stops rotating once the temperature is brought under a lower limit. This application note also demonstrates the method for selecting and using the Z8 Encore! XP temperature sensor.

Four fans are controlled by the Z8 Encore! XP and provides a mechanism to avoid the system inrush current and to detect the fan fault condition. The fan fault condition is indicated by visual indicators. Also for demonstration purpose, the fans are set to run at different speeds by changing the position of the switches. Appendix B—Schematics on page 10 displays the schematics.

Note: *The source code associated with this Application Note* `AN0234-SC01.zip` *is available for download at* www.zilog.com

Developing the Fan Speed Control Application

The system development calls are used for interfacing an external temperature sensor to measure the temperature around an object. In this method, the temperature sensor is not interfaced; instead the temperature sensor output is simulated by providing a scaled input to the analog to digital converter (ADC) terminal for testing purpose. Figure 1 on page 2 displays the block diagram of temperature sensed fan speed controller.

The design of the fan speed control application depends on the temperature input, based on Z8 Encore! XP microcontroller unit (MCU). Figure 1 on page 2 highlights the different peripherals of Z8 Encore! XP MCU used in this design. The input from the temperature sensor is fed to the on-chip analog-to-digital converter (ADC). The output of the ADC is used to calculate the duty cycle of the Pulse Width Modulator (PWM), which varies proportionally to the input temperature and accordingly the fan speed varies. The feedback from all the four fans is taken as the tachometer inputs. The presence of tachometer inputs are constantly monitored to detect the fan fault condition. The fan fault condition and also the over temperature condition is displayed by an LED. An AND gate is used to control the initial switching ON of the fan and the inrush current.

The hardware is built and tested with the following components. The main component of the hardware is Z8 Encore! XP MCU. This does not require any external crystal or RC network as an internal oscillator is provided. By using 3.3 V power supply the power consumption and heat dissipation of the Z8 Encore! XP MCU is minimized.

Figure 6 on page 10 displays the schematic diagram of general interface to the Z8 Encore! XP MCU. The output of the temperature sensor is connected to pin-27(PB-0). The reset circuit combination is connected to pin-21. Different port pins are used as displayed in the Appendix B—Schematics on page 10.

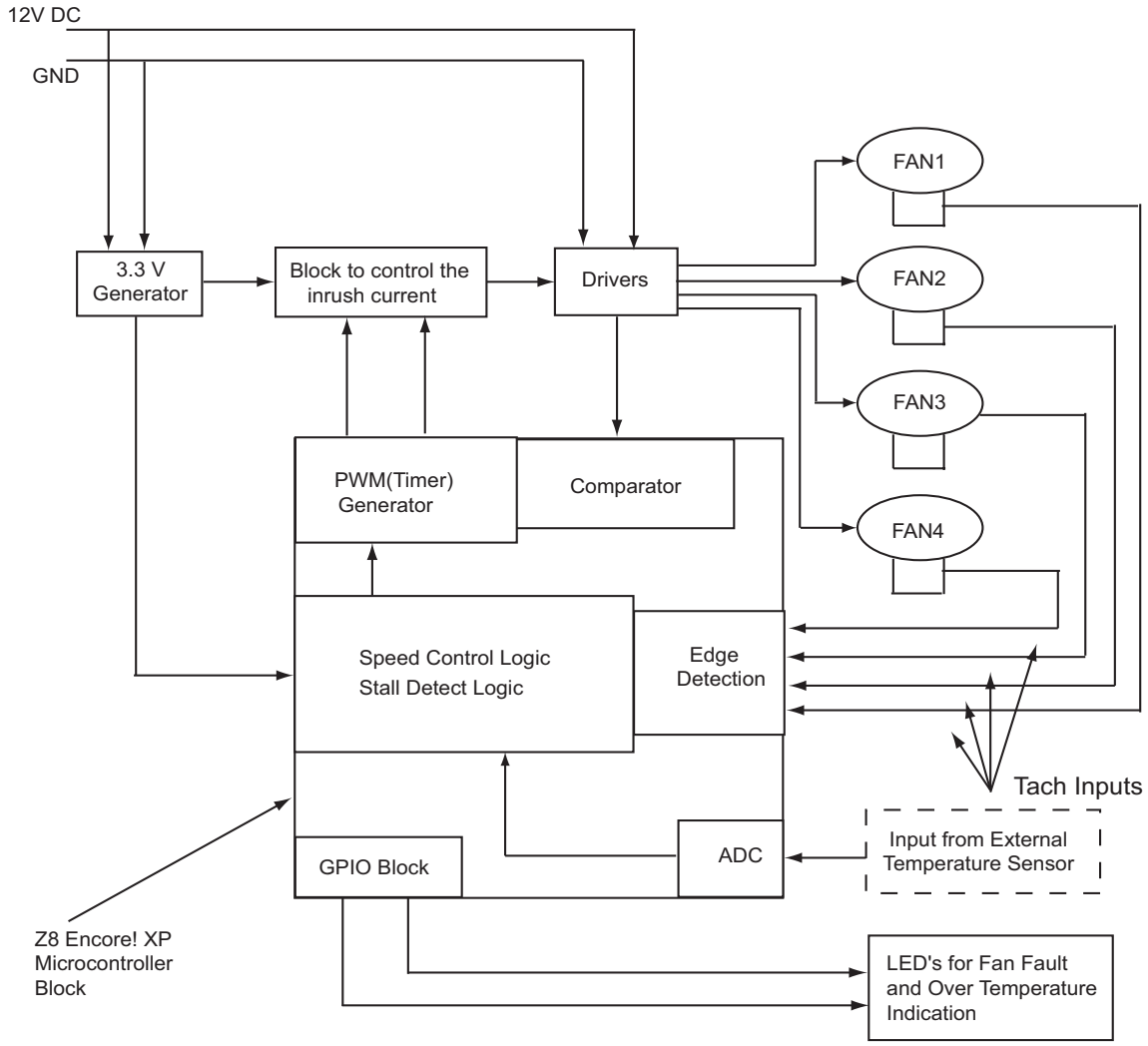

Figure 1. Block Diagram of Temperature Sensed Fan Speed Controller

The description of interface is provided below: For the schematic diagrams, see Appendix B—Schematics on page 10.

- The circuit includes: FET-1, FET-2, FET-3, and FET-4. These FETs are switched on sequentially with some delay between them. This is done to avoid all the four fans getting switched on at once, there by reducing the system inrush current.
- The choice of N-channel MOSFETs depends on the voltage and current rating of the fan. In this example, IRLR024N is chosen based on the 12 V DC BLDC fan consuming maximum of 250 mA of current at full speed.
- Diode D7 to D10 removes any surges in the fan motor winding.
- M1 to M4 are 12 V DC, three wire BLDC fans, being switched by FETs. The source of the FETs

are grounded. LED1 to LED 4 indicates the fan fault condition. LED5 indicates the over temperature condition and LED6 displays the power on condition.

Hardware Details

The on-chip timer of Z8 Encore! XP is configured to generate the PWM output. The PWM duty cycle varies proportionally to the temperature. A single PWM output controls all the four fans simultaneously. Based on the application requirement, the lower and the higher temperature limit is set through the software.

The PWM generation starts once the input temperature exceeds the lower limit. To control the inrush current, the fans are turned ON sequentially by providing some delay between them. A software is used to set the fan switch on delays.

As displayed in the Appendix B—Schematics on page 10, the PWM output control all the four general-purpose output pins, that are used either to enable or disable the AND gate.

Initially, the PWM output is at High state (before generating the PWM output) and all the *PWM output control* pins are held Low. This ensures that the output of the AND gate is Low and FET's are turned OFF. After the input temperature exceeds the lower set limit, the PWM generation starts. After the delay time set to switch on FAN1 (M1), the *PWM_OC1* (PWM output control-1) is set High and allows the PWM output to pass through the gate and control the FET switching.

Similarly, after the delay time set for FAN2 (M2), the *PWM_OC2* (PWM-output control –2) is set high, to switch on FAN2. Similarly, for the FAN 3 and FAN 4. Each of these FANs run at varying duty cycle proportional to the input temperature till the higher temperature limit is reached.

Once the higher temperature limit is crossed, the FAN runs at 100% duty cycle till the temperature is brought below the higher limit. As the temperature reduces, the FAN speed reduces proportionately and stops once the temperature falls below the lower set limit. LED5 provides a visual indication when the temperature exceeds the higher set limit.

Changing the switch2 (SW2) position, all the four FANs are made to run at different duty cycle from 10 to 100%. These fans run from Low speed to High speed gradually and repeat the cycle till the switch position is changed. When this switch is in ON position, the voltage input at the ADC input port is discarded.

The 12 V, 2.3 W SUNON fans are used with this application note. At full speed, these fans require 200 mA of current. Total current requirement of the system is 900 mA.

Software Details

The software is developed and tested on the hardware ar displayed in Appendix B—Schematics on page 10. This section discusses the function of different software blocks that runs on Z8 Encore! XP MCUs.

The software provides the following functionalities:

- Selecting the system clock
- Initializing the GPIO
- Selecting the Z8 Encore! XP on-chip temperature
- Initializing the TIMER and PWM generator
- Configuring the ADC
- Detecting and displaying the fan fault condition

Selecting the System Clock

The function `init_systemclock()`, defined in the `main.c` file, selects the internal clock, which works at 5.5 MHz.

The oscillator control register must be written in the following sequence:

```
OSCCTL = 0xE7; //Unlock sequence
               //for OSCTL write
OSCCTL = 0x18;
OSCCTL = 0x80;
```

Crystal oscillator, internal precision oscillator, and WDT oscillator are enabled and the internal oscillator functions as the system clock.

Initializing the GPIO

The function `init_led_gpio()`, defined in `main.c` file, initializes the GPIO port to control the LEDs.

For initializing the GPIO, the following configuration are made:

- Port pin PB-0 is configured for analog input (ANA0).
- PC-3, PC-6, and PC-7 are configured for driving the LED.
- PA-1 is configured for PWM output pin.
- PC-0, PC-1, PC-2, and PC-4 are configured as output pins to drive the AND gates.
- PB-4 and PB-5 are configured for driving the fan fault condition LEDs.
- PA-2, PA-3, PA- 4, and PA-5 are configured as input port to read tach windings.
- PC-5 is configured as input pin to read the switch settings.

Selecting the Z8 Encore! XP On-chip Temperature

By default the simulated external temperature sensor input (which is scaled for 0 to 2 V DC) is selected. A provision to select the Z8 Encore! XP on-chip temperature sensor is provided. To select the on-chip temperature sensor, define the `ON_CHIP_TEMP_SENSOR` in `adc.h` file. The on-chip temperature of Z8 Encore! XP is scaled for operating from 20 ^{o}C to 70 ^{o}C.

Configuring the ADC

The `init_ADC ()` function initializes the ADC on-chip peripheral to the following configuration.

- Single ended mode buffered with unity gain
- Continuous conversion
- Selects the internal reference of 2 V
- Configures the ADC interrupt vector
- Configures port pinB-0 to alternate function mode

The `start_adc_conversion()` function initiates the ADC conversion. The ADC output is compared with `LOWER_TEMP_VALUE` to start the PWM generation. The comparison is done in the ADC interrupt service routine

If the ADC value is within the `LOWER_TEMP_VALUE`, the PWM generation is not initiated.

The temperature limits can be set in percentage of full scale input voltage. This is done through the macro definition as `LOWER_TEMP_SET_VALUE` and `HIGHER_TEMP_SET_VALUE` in `adc.h` file

Initializing the TIMER and PWM Generator

Follow the steps below to initialize the timer and PWM generator:

1. The timer0 is initialized to function in PWM single output mode, where the timer outputs a PWM signal through Port A pin-1. The timer's input is the system clock. The `SYSTEM_CLOCK` (in `timer.h` file) is a macro, which defines the system's clock rate.
2. The PWM frequency is at 50 Hz, defined by a macro: `PWM_FREQUENCY` in `timer.h` file.

3. When the temperature exceeds the LOWER_TEMP_VALUE value, the fan starts rotating with a minimum speed. As the temperature increases, the speed of the fan increases proportionately and when the HIGHER_TEMP_VALUE value is reached, the fan rotates with the 100% duty cycle.
4. In order to get proper PWM outputs, the following PWM registers have to be programmed with proper values.
   a) Timer reload high registers
   b) Timer reload low registers

   These registers hold the value to generate the required PWM frequency, as defined in the program. In this application the PWM frequency is set to 50 Hz. The appropriate values for the reload registers are calculated using the equation:

   PWM_RELOAD_VALUE = (SYSTEM_CLOCK) (PWM period in secs) / PRESCALE

   SYSTEM_CLOCK = 5.5 MHz

   PWM period = 50 Hz

   PRESCALE is set to 2

   Then the calculated value of PWM_FREQ_RELOAD_VALUE is 0xD6D8

   This *D6* value is stored in timer reload High register and *D8* is stored in timer reload Low register.

5. Set the TPOL bit in the timer control register to one, the PWM output signal begins as a High (1) and transitions to Low (0) when the timer value matches the PWM value. The timer output signal returns to a High (1) after the timer reaches the reload value and is reset to 0001H.
6. The function init_timer0(), initializes the Timer0 for PWM mode and writes the appropriate values to timer reload high and low registers.
7. To get the required duty cycle, appropriate values are loaded to PWM high and PWM low register. Based on the ADC output (which is actually proportional to the external temperature) the appropriate values are loaded after every second to PWM registers to alter the duty cycle. The ADC values are scaled as indicated below and then loaded to the PWM high and PWM low registers.
8. In this application the 8-bit ADC is used and we have 256 different values as the output of the ADC. This value when gets loaded to PWM registers needs to be scaled with respect to PWM_FREQ_RELOAD_VALUE. That is, the actual value to be loaded to PWM registers are calculated as follows:
   PWM output high ratio = (X/PWM_FREQ_RELOAD_VALUE) * 100;
   Where X = (ADC value)
   (PWM_FREQ_RELOAD_VALUE) / 255
   Table 1 lists the duty ratio of the PWM output signal for the different values of the ADC output.

Table 1. Duty Ratio

ADC Value	X=ADC value(0xD6D8) /255	Duty Ratio = (X/PWM_FREQ_RELOAD_VALUE)*100
FF	0xD6D8	100%
7F	0x6B00	49.8%
33	0x2AF8	20%

   The above calculation is applicable till the higher temperature limit is reached. Beyond this point the fan rotates at 100% duty cycle.

9. The Timer0 generates an interrupt once the PWM_FREQ_RELOAD_VALUE is reached. In this case the interrupt is generated at every 20 ms. A software counter is used to count up to one second. At every second the PWM high

and PWM low register value is updated to change the PWM duty cycle. The reloading of PWM register is done in `timer0_isr` routine.

10. A global variable is used to count the number of interrupts, which occurs at every 20 ms. This is used as a base timer and all delays are calculated based on this delay. To have a different PWM frequency other than 50 Hz, then all the required delays have to be redefined based on the application.

Detecting and Displaying the Fan Fault Condition

The tachometer feedback, input from the individual fans, detects the fan fault condition. The tachometer input is monitored constantly. The tachometer output changes from low to high state as long as the fan rotates. If there are no changes in the tachometer output for about the set time delay (This can be set using a macro `FAN_FAULT_LIMIT` in `timer.h`) then that particular fan is identified as faulty fan and the corresponding fan fault LED is switched ON. Once the fan fault condition is detected, all fans are switched OFF. After clearing the fan fault condition (replacing/repairing the fan) the system has to be reset by pressing switch SW1 for normal operation.

Testing

Equipment Used

The following equipment are used for testing:

- Fan speed control reference design board
- Zilog Developer Studio II —Z8 Encore! XP® (ZDS II—IDE)
- A digital tachometer to measure the RPM of the fan

Test Procedure

This section discusses the functional test procedure. The reference design kit is supplied with the required code flashed already. Hence the Step1 and Step2 can be bypassed.

1. Connect 12 V DC power jack to the reference design board. Observe that the power supply indicator LED6 is ON.
2. Download the source code to the board using the Zilog's smart cable or target interface module (TIM). Once the download completes, press rest to watch the functioning of the reference design board.
3. Vary the potentiometer R3 and observe that the fan speed varies with the increase in the voltage at the ADC input pin (This is analogues to a condition of temperature of an object increasing). Observe that the fan speed reduces when the potentiometer is turned in the reverse direction.
4. Observe that the over temperature LED glows when the ADC input voltage exceeds the over temperature set limit. Also observe that all four fans run at 100% duty cycle.
5. Simulate the fan fault condition by forcing the fan to stop physically. Observe that the Corresponding fan fault LED glows and the remaining fans are switched OFF. You can restart the system by pressing the reset switch.
6. Change the switch position SW2 to ON position. Now all the fans start running from a duty cycle of about 10% to 100% in steps and then from 100% to 10%. This sequence continues till the switch changes to OFF position.

Results

You can observe that by changing the position of the potentiometer settings, the voltage at the ADC input pin changes and accordingly the speed of the fan is varies. Figures 2 and 3 displays the following graphs.

- Percentage duty cycle verses the RPM of the fan.
- Temperature verses the RPM of the fan.

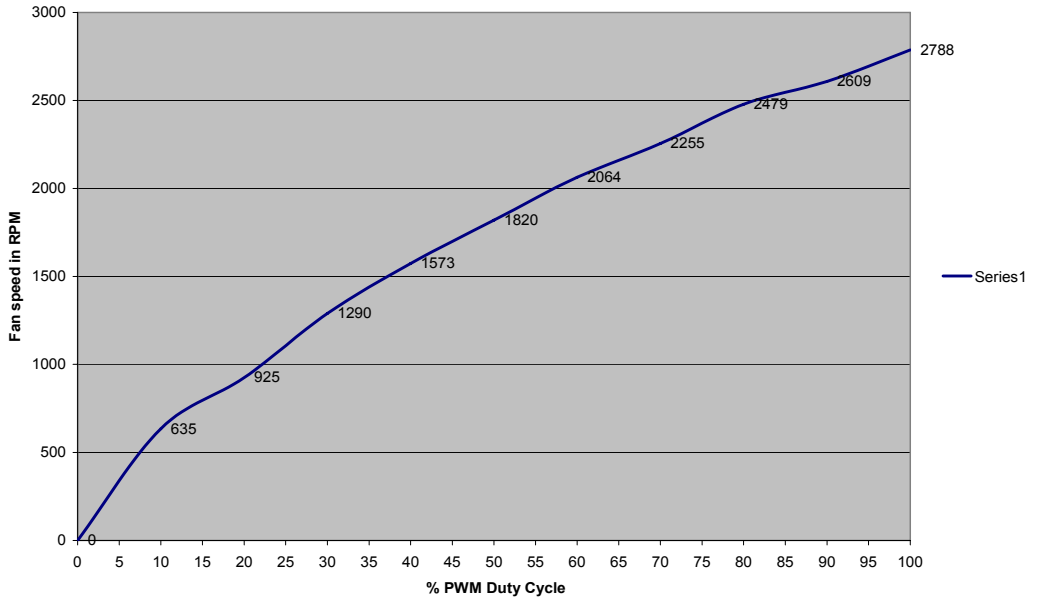

Figure 2. Fan Speed in RPM v/s % PWM Duty Cycle

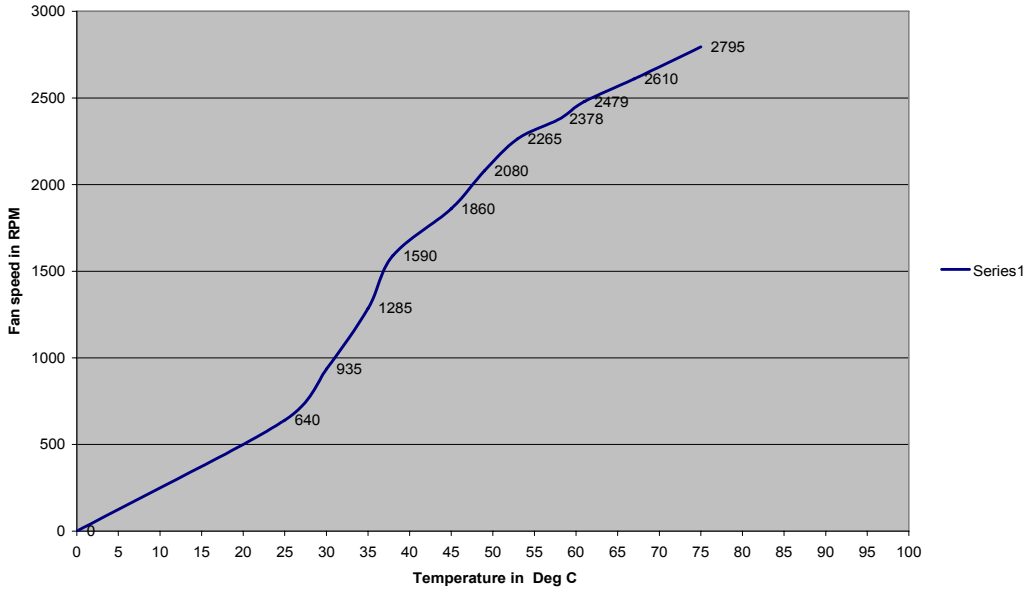

Figure 3. Fan Speed in RPM v/s Temperature

The testing is also performed to see if the fans are being switched on sequentially with a predetermined delay between each fan switching. First FAN1 is switched on followed by FAN2, FAN3 and FAN4 with some delay in between. This will ensure the reduction of system inrush currents. The waveform is captured using Tektronix TLA 611 logic analyzer. It displays the sequential switching from FAN1 to FAN4.

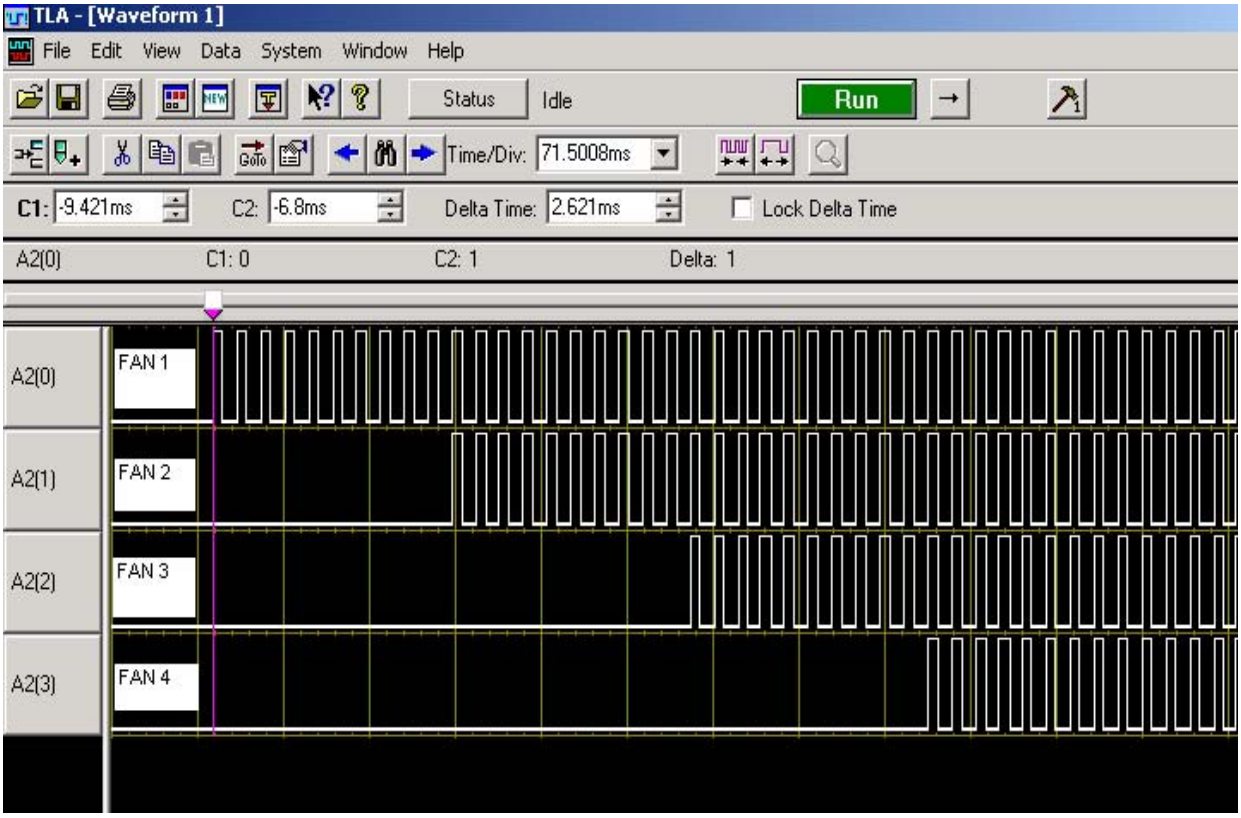

Figure 4. Waveform Showing the Sequential Switching of Fans

Summary

This application note demonstrates the use of on-chip temperature sensor of Z8 Encore! XP. Using the on-chip temperature sensor of the Z8 Encore! XP MCU for temperature ranging from 20 ^{o}C to 75 ^{o}C, will provide good accuracy and eliminates the need for using the external temperature sensor. This reduces the total system cost and the development time of the product.

This application note also demonstrates the method of configuring and using the on-chip ADC and the timer to generate the PWM output. The PWM output is generated based on the simulated input from the temperature sensor, which controls the speed of the BLDC fan and ensures that the temperature of an object is maintained at certain pre-defined levels.

Appendix A—Photograph

Figure 5 displays photograph of reference design kit.

Figure 5. Photograph of Reference Design Kit

Appendix B—Schematics

Appendix B displays the schematics of Z8 Encore XP MCU.

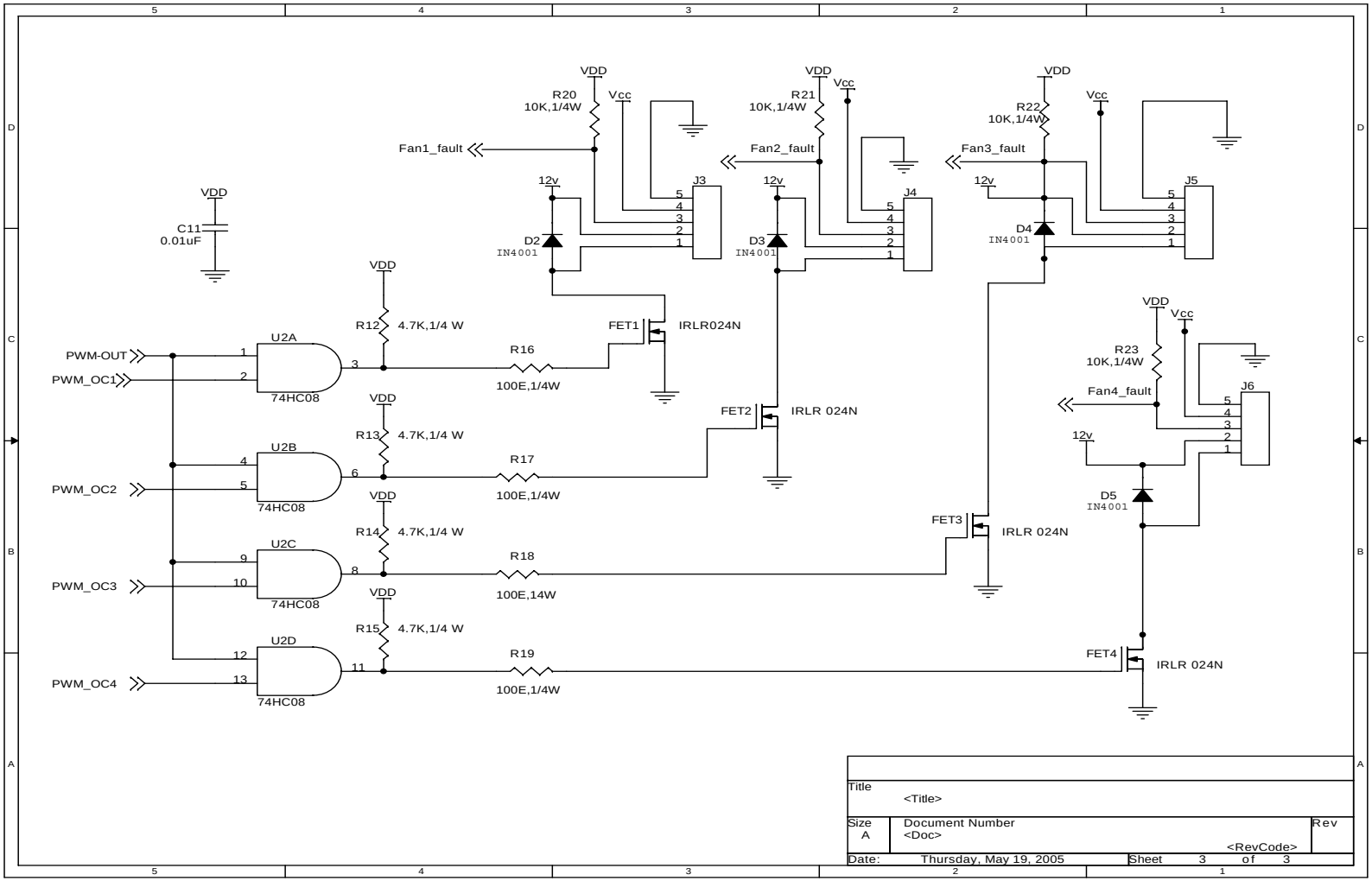

Figure 6. Schematic of Z8 Encore! XP Processor Circuit for Speed Control of BLDC Fans

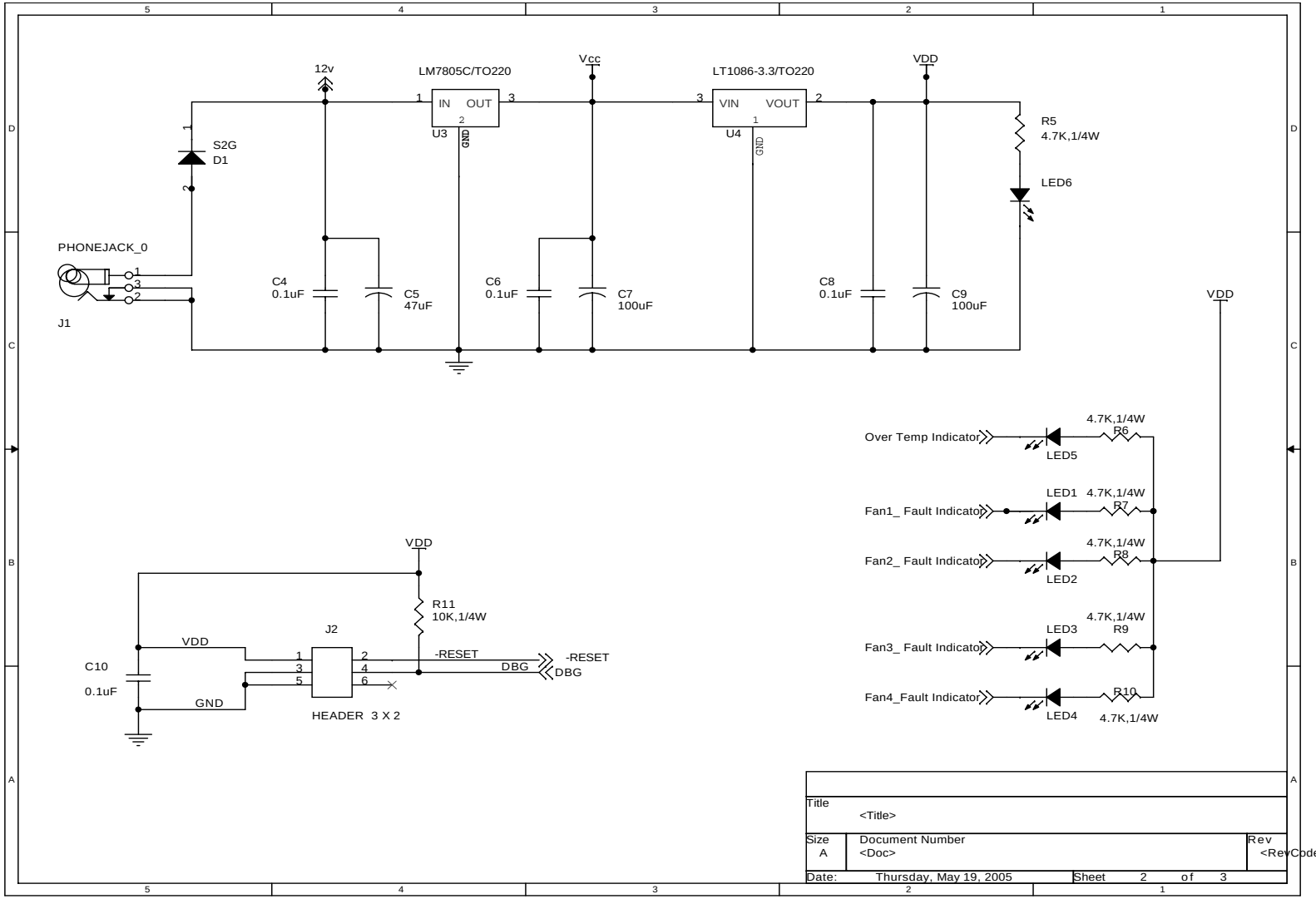

Figure 7. Schematic of Power and Debug Circuit for Speed Control of BLDC Fans

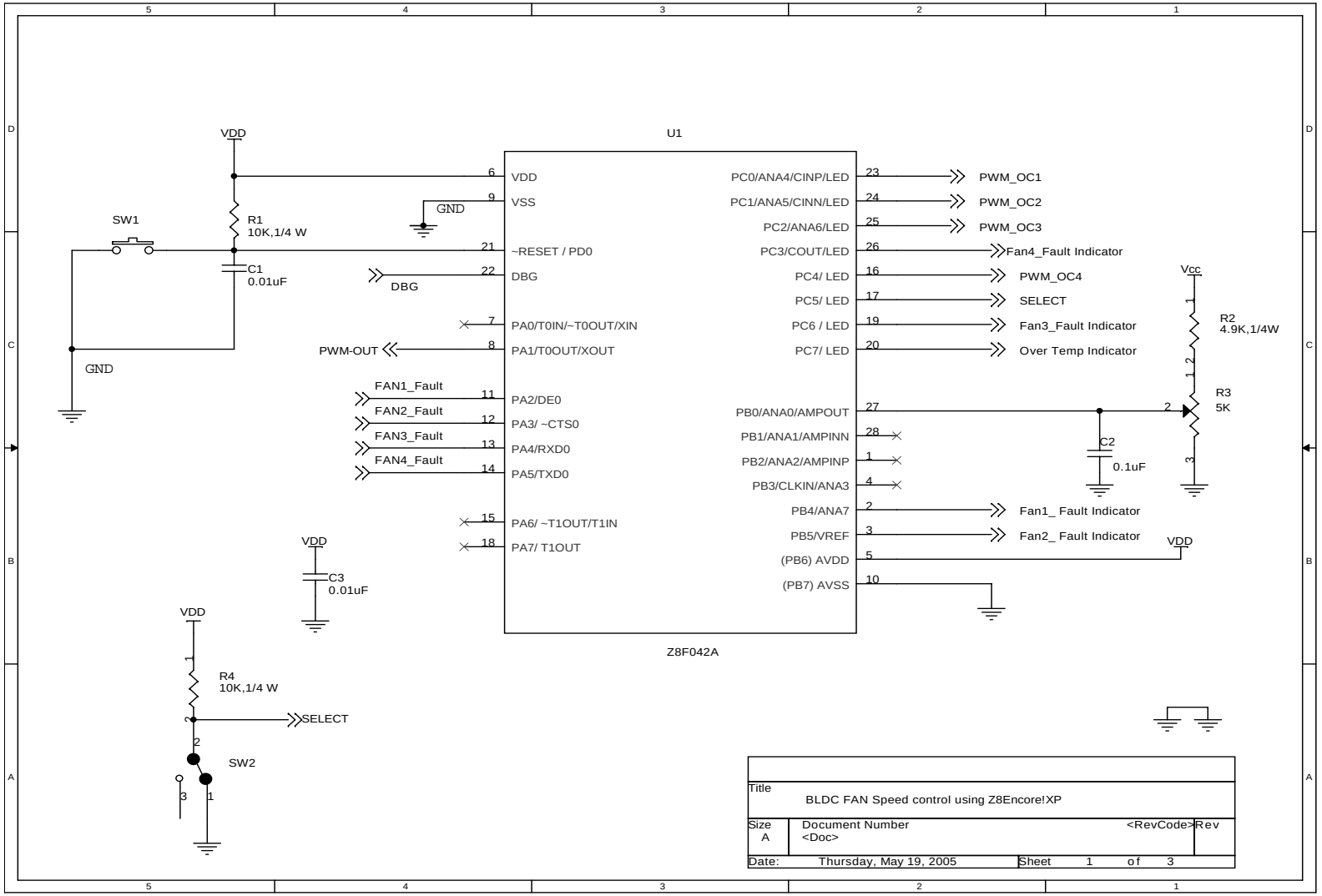

Figure 8. Schematic for Speed Control of BLDC Fans using Z8 Encore! XP

Appendix C—Flowcharts

This Appendix displays the flowchart of Main function, ADC ISR, and Timer0 ISR function.

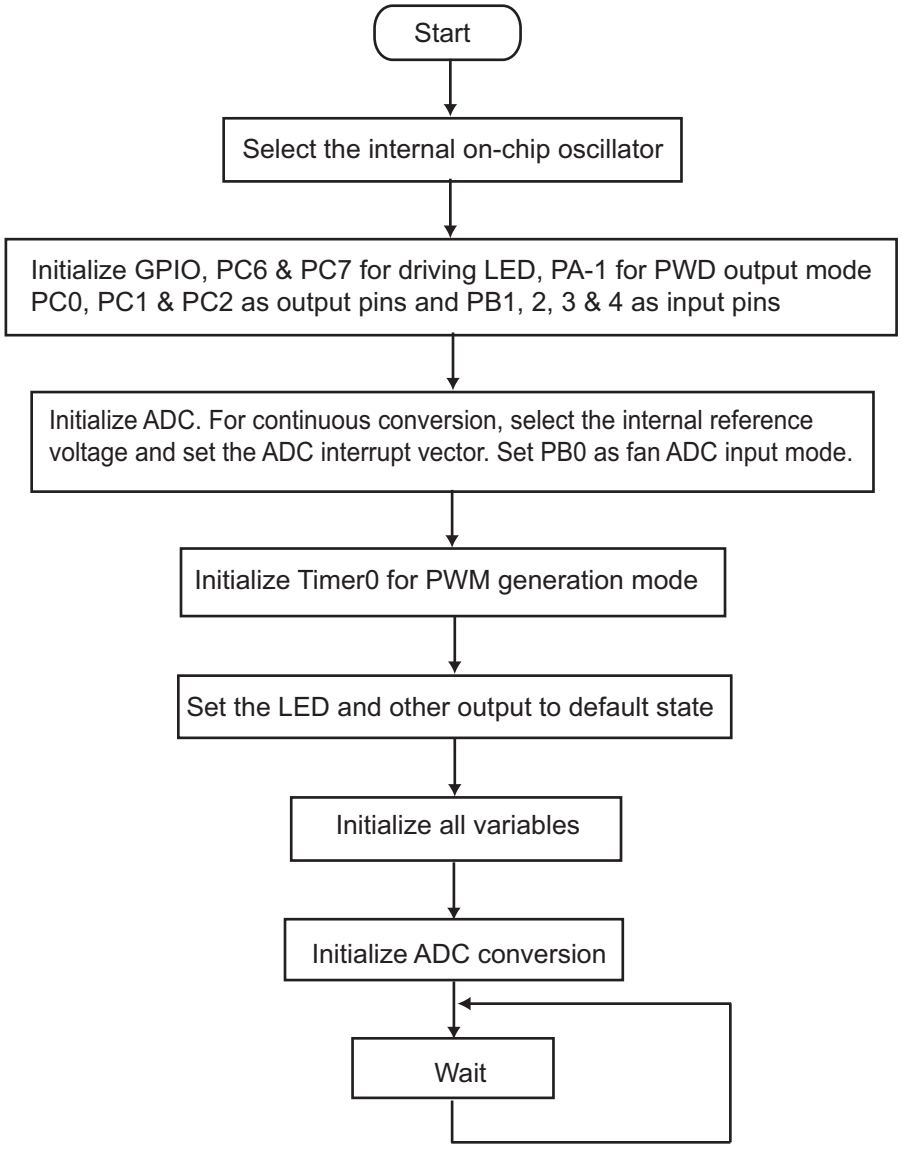

Figure 9. Main Routine to Initialize the Z8 Encore! XP Peripheral

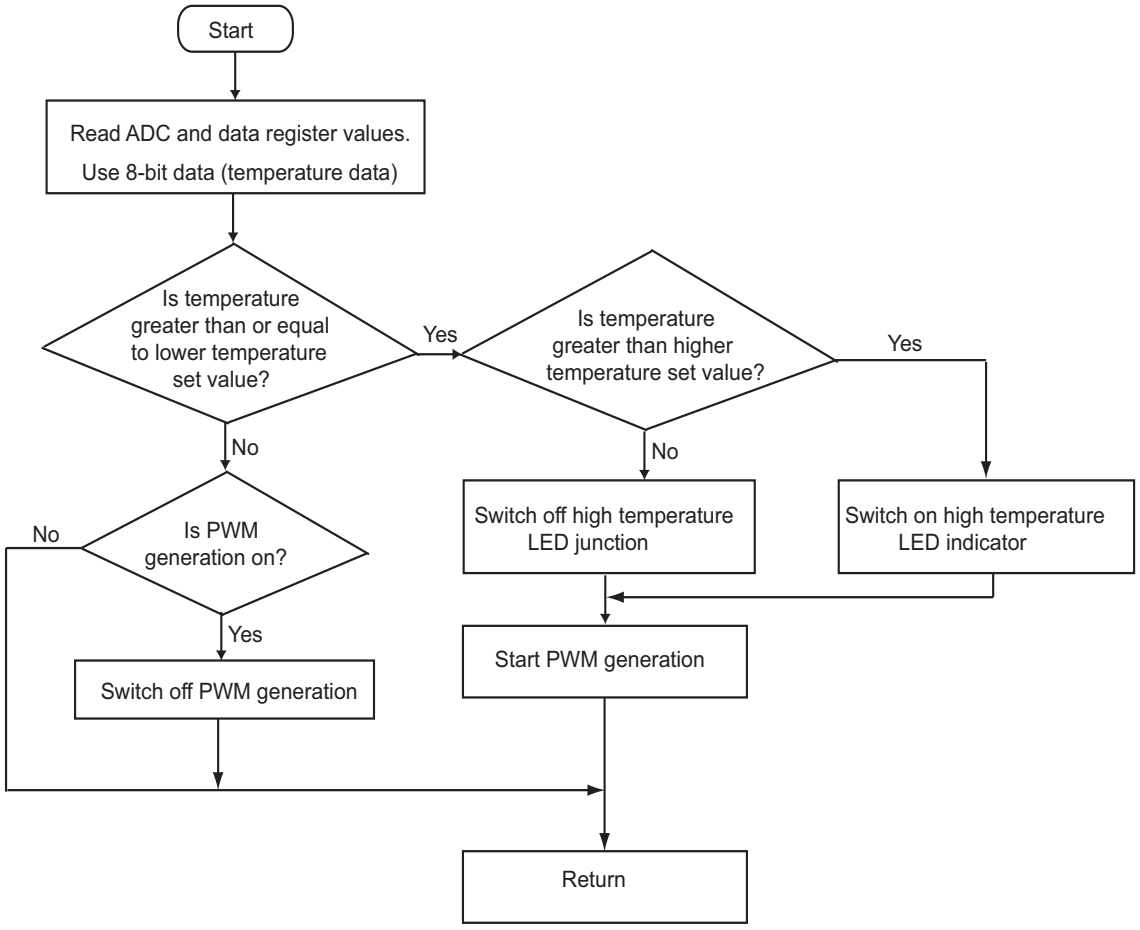

Figure 10. ADC Interrupt Service Routine

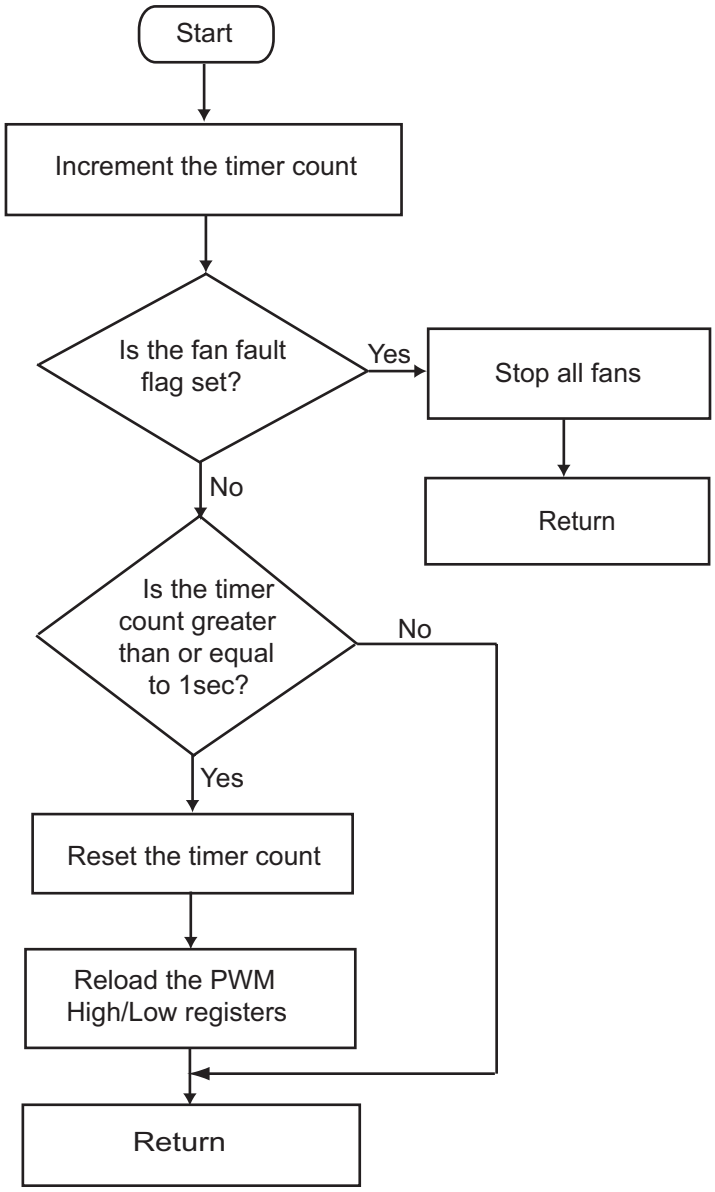

Figure 11. Timer0 (in PWM Mode) Interrupt Service Routine

Warning: DO NOT USE IN LIFE SUPPORT

LIFE SUPPORT POLICY

ZILOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZILOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer