Application Note

Active IR Washroom Designs Using the Z8 Encore!® F0830 MCU

Z8 Encore!®

AN026901-0108

Abstract

This application note describes a Z8 Encore!®-based active infrared washroom appliance typically used for towel dispensing, faucet control, or dryer control in a touch-free washroom. Utilizing the Z8 Encore! F0830's low-power features, this design has an average battery drain of less than 70 µA.

Note: *The source code associated with this application note (AN0269-SC01) is available on* www.zilog.com.

Theory of Operation

The IR detection system in this application note uses an IR LED to illuminate the area in front of the appliance and an IR photodiode to measure the IR light being reflected back to the appliance. When a target is close to the appliance, it will reflect light from the LED into the photodiode. In order to prevent false triggering from other IR sources, the photodiode is AC coupled to the microcontroller, and the IR LED is only pulsed for a very short period of time. Using this method, the photodiode must see a large enough difference between ambient IR light and the IR light measured when the LED is pulsed on. There are several power-saving attributes to this detection system; because the photodiode is AC coupled, leakage from the photodiode will not bias the ADC input to the microcontroller. Also, the LED is only pulsed for a brief period of time, which reduces the overall current draw.

Developing the Application with the Z8 Encore!®

Hardware Architecture

Thanks to the extensive feature set of the Zilog Z8 Encore!® microcontroller, very little external hardware is required for this design. The overall design consists of the Z8 Encore!® microcontroller, IR transmitter, and IR receiver.

The Z8 Encore!® microcontroller has two internal clock sources, freeing this design of any external clock sources or components. Because of the unique low-power requirements, this design will need to use both clock sources. The main clock is the 5.53-MHz Internal Precision Oscillator, which has an accuracy of ±4% and typically draws 700 µA. The second internal clock source is the 10-KHz Watch Dog Timer, which has an accuracy of ±50% and typically draws 2 µA. The WDT has the ideal power consumption for this design, but the accuracy is too poor for adequate performance. On the other hand, the IPO has the accuracy required, but its power consumption is far too high to be used in this design. Fortunately, the IPO can be used to measure the frequency of the WDT and then use the compensated WDT as an accurate low-current clock source.

The Z8 Encore!® microcontroller also has several other features that can be utilized for a very low power design. Key to this design is the power-saving HALT mode of operation. In the HALT

mode, the CPU is turned off, but the oscillator, timers, and other peripherals continue to operate. The Z8 Encore!® is returned to NORMAL operation through an interrupt or external event on a port pin. The washroom appliance will spend most of its time in the low-current HALT mode, measuring time with the WDT, and then return to NORMAL operation when a timer expires. When the timer expires, the microcontroller will switch back to using the 5.5-MHz IPO. Although the microcontroller is consuming more current in the NORMAL operation mode, it is now running at 5.5 MHz and will quickly be able to complete the task of detecting a target, cycling the load as required, switching back to the WDT, and then returning to the HALT mode.

To further reduce power, STOP mode can be used with the watchdog timer running.

The next design requirement is a fast, high-resolution Analog-to-Digital Converter. The Z8 Encore!® F0830 Series features a 10-bit ADC that can complete a conversion in less than 50 µS while using the 5.5-MHz IPO or 12 µS when using a 20-MHz clock. (The Z8 Encore! F083A is also available if more performance for other functions is needed.) The fast ADC will allow the IR pulse from the LED to be very short, keeping current low and allowing the microcontroller to read the photodiode quickly, which helps to eliminate outside interference. The ADC will also be used to read the battery voltage, so the appliance will not attempt to run the motor when the battery voltage has become too low to complete a cycle.

Hardware Details

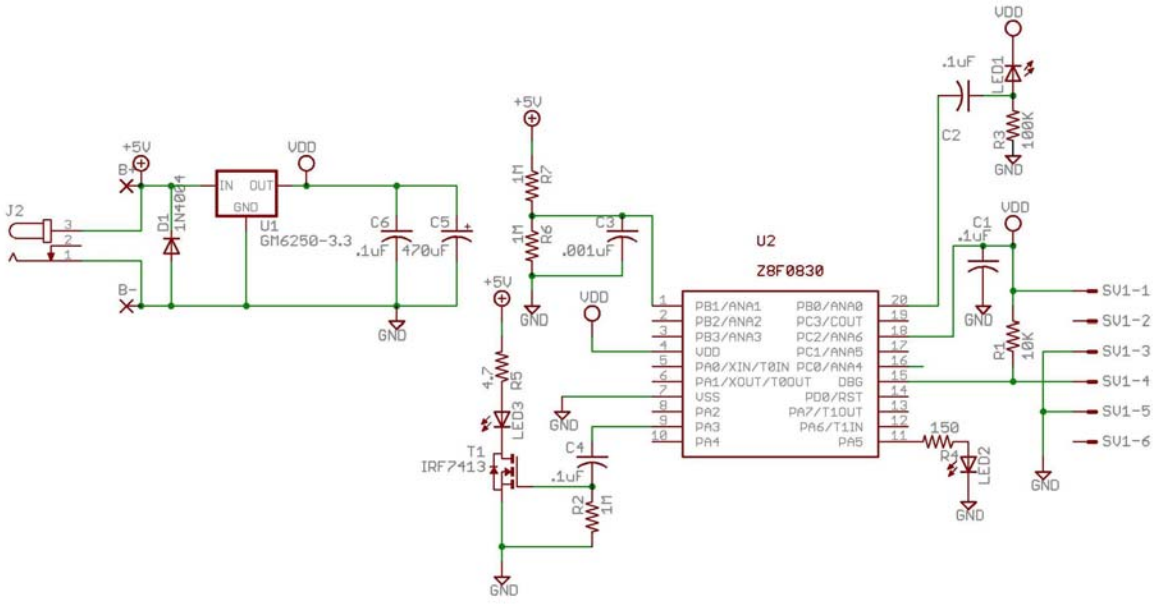

Figure 1. Active IR Washroom Schematic

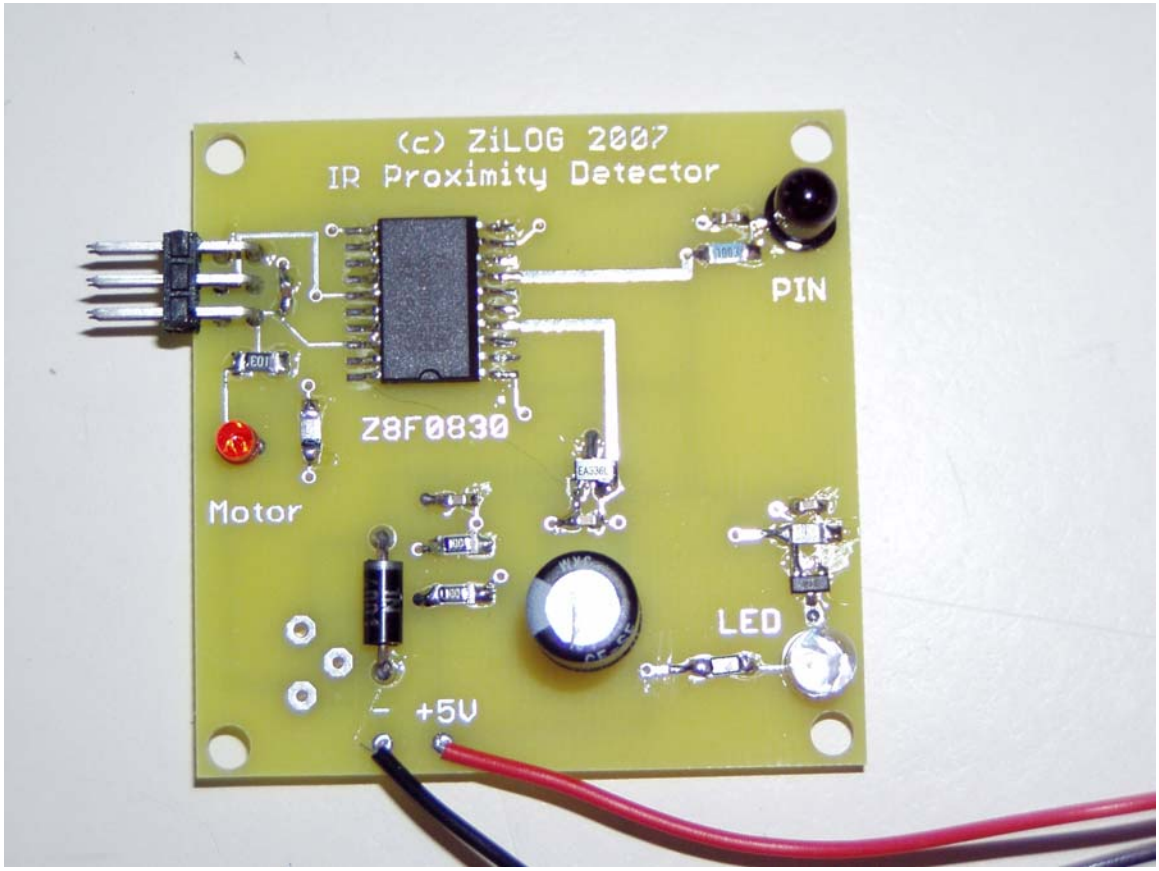

Figure 2. Active IR Washroom Design Board

The IR transmitter consists of an IR LED, LED1, and FET driver, T1. LED3 is driven with a 100-mA pulse, which is kept very narrow to prevent damage to the LED and to keep the average current draw low. As a precaution, T1 is AC coupled to prevent damage to the FET and LED if the port pin should remain in the driving state. This can happen during debugging if a breakpoint is placed at that portion of the software and can also happen while the Z8 Encore!® is being programmed.

The IR receiver consists of a photodiode, LED1, its load resistor R3, and a decoupling capacitor C2. LED1 is biased in the photoconductive mode, which is responsive enough for a 100-µS IR pulse, and the AC coupling allows the appliance to push the port pin low when it is not used, which reduces the overall current draw.

Resistors R6 and R7 form a voltage divider that is used to measure the battery voltage. If the battery is near the end of life, the heavy drain from turning on the appliance's load might cause the battery voltage to fall below the operating level of the microcontroller. To prevent this failure, the battery voltage is measured before cycling the load, and the load is not cycled if the battery is below a safe level.

LED2 and its current limiting resistor, R4, are used as the simulated load. In this demonstration, LED2 will light when the appliance detects a target and should cycle.

Software Implementation

After power-up and initialization, the software's Main Loop will wait in the low-power HALT mode for ¼ of a second, measuring this time interval with the WDT. When the timer expires, the microcontroller will return to the NORMAL mode and switch to the faster 5.5-MHz IPO. Once in the NORMAL mode, the software will test for the presence of a target and run the load if one has been detected.

The software utilizes conditional compilation to make debugging a little easier. It is not possible to debug if the microcontroller is in the HALT mode or is running from the WDT. Because of this, the macro name LowPower is used to conditionally compile code to run from the WDT or to compile code that remains operating from the IPO.

IdlePorts()

This function will configure the ports pins for low-current operation. The pin used for the battery voltage detect is idled as a high output because the pin will normally be closer to this level and will pull less current if it is high. All other port pins are idled as low outputs.

UseInternalOsc()

This function switches the oscillator source to the internal oscillator.

MeasureTime(int* time)

This function compiles differently depending on the condition of LowPower. If LowPower has been defined, the function will measure the number of WDT cycles that expire within a given time period, *time*, in milliseconds. The WDT typically pulls 2 µA, but it only has an accuracy ±50%. The WDT is, however, fairly stable over temperature when used with a regulated supply. MeasureTime() will use the IPO as a clock source to measure WDT pulses on all the fixed intervals used in the application. When the WDT is used as a clock source, a timer will be loaded with the interval values measured by this function.

If LowPower has not been defined, then the function will simply exit, leaving *time* with the delay value in milliseconds.

Wait(int* time)

This function compiles differently depending on the condition of LowPower. If LowPower has been defined, then all peripherals will be turned off, and the microcontroller will switch to the WDT clock source. Next, a timer will be loaded with the argument *time*, and the microcontroller will be placed into the HALT mode. When the timer expires, the microcontroller will return to the NORMAL operating mode and return the clock source to the IPO.

If LowPower has not been defined, then a timer will be configured to measure the argument *time*, and the function will wait for the timer to time out.

FullSpeedWait(unsigned int time)

Similar to *Wait(int* time)*, this function will wait the argument *time* in milliseconds but without switching clock sources or changing to the HALT mode.

unsigned int ReadADC(unsigned char channel)

This function will read the ADC channel in the argument *channel* and return the value of the conversion.

InitDurations()

Will repeatedly call *MeasureTime(int* time)* to intervals from milliseconds WDT ticks.

unsigned int CalcDifference(unsigned int x, unsigned int y)

Returns the difference of two unsigned integers, limiting the lowest result to zero. This function is used to calculate the difference in IR light seen by the photodiode.

unsigned char TestRec()

This function performs the actual test for sensing a target and returns TRUE if a target has been detected. First, the pin used for the photodiode is returned to an input state, given time to settle, and the ReadADC() is called to measure the output of the photodiode while the IR LED is off. This is the dark reading. Next, the IR LED is turned on, and ReadADC() is called to measure the output of the photodiode with the LED on; this is the light reading. Finally, CalcDifference() is called to measure the difference between the dark reading and the light reading. If the difference is greater than the TargetLevel, then the function returns TRUE, signifying an object has been detected.

Wait2Leave()

This function is called after a target has been detected and will wait for the target to leave before continuing. The ports are returned to their low current, idle state, and TestRec() is called at regular intervals until it returns FALSE.

RunMotor()

This function will turn the load (motor) on for a specific time interval and then return the ports to their idle state. Since the load already pulls a considerable amount of current, there is no need to place the microcontroller in the HALT mode.

BatteryOK()

This function will return TRUE if the battery voltage is higher than the threshold. The function sets the battery pin as an input, waits for the voltage on the pin to settle, and then calls ReadADC() to read the battery voltage before returning the battery pin to its idle state.

Main()

Looking at the entire project, Main calls IdlePorts() to configure the ports for the correct direction and place them in a low current state. Next, InitDurations() is called to measure all of the time intervals, based upon the WDT. At this point, the microcontroller will enter a simple delay loop. This loop is critical to debugging success and should be considered in all projects that switch clock sources or use a low-power operating mode. The delay loop allows time for the microcontroller to communicate with the debug interface. Without this delay, the microcontroller would switch to the WDT clock or HALT mode, and it would be impossible to reflash the part or perform any debugging.

The Main Loop calls Wait(), which places the microcontroller in the HALT mode for ¼ second. Next, the appliance will determine if a target has been detected by calling TestRec(). If a target has been detected, BatteryOK() returns true; the load will be cycled by calling RunMotor(). The appliance will then call Wait2Leave(), which will pause execution until the target is no longer detected. After the target leaves, the ports are returned to idle with IdlePorts(), and program execution returns to the top of the Main Loop.

Testing

Functionality

The circuit described in Figure 1 was built on an etched circuit board and programmed with the software described in this application note. A gray card was used as a target and placed in front of the LED so that it could reflect IR light back to the photodiode. Detection testing was started with the card 50 cm from the LED, and the distance was decreased until the target was detected. This circuit had a detection distance of 22 cm in standard office lighting conditions. Because the photodiode is not linear while operating in the photoconductive mode, range will be decreased if the photodiode is in direct sunlight. The range of this circuit was 10 cm in direct sunlight.

Current Draw

Low current draw is critical to the success of this application. The actual current draw has two profiles, Idle and Detecting. In the Idle state, the microcontroller is in the HALT mode, running from the WDT, and all ports are idled. This state

has the lowest operating current, and the appliance spends the bulk of its time in this state. In the Detecting state, the microcontroller is in the NORMAL mode running from the IPO, and the IR LED is turned on. This operating condition is kept as short as possible to keep the average current draw low.

The average current draw is calculated by measuring the current draw of the two operating modes and the time the microcontroller spends in these two modes. The current draw of the idle state was measured using an ammeter with the WAIT() function in an infinite loop. The current draw for the detecting state was calculated by first measuring the LED current and pulse width with an oscilloscope across R5. Next, the operating current of the rest of the circuit was measured using an ammeter during the power-up delay. The total operating current for this state is then the sum of the LED current and the power-up delay current.

The current for the idle state was found to be 6 µA for a period of 250 mS (1.5 µA seconds). The current for the detecting state was found to be 736 mV/4.7 Ohms = 160 mA for 91.6 µS or 14.7 µA seconds. The average current is then 64.8 µA. From these measurements, it can be seen that the greatest contributor to average current draw is the LED pulse current.

Summary

The Z8 Encore! F0830's low-power capabilities and fast ADC make it possible to design a battery-operated active IR washroom appliance with very low external component count and an average operating current of less than 70 µA.

References

Topic	Document Name
Z8 Encore!® MCU	*Z8 Encore!® F0830 Series Product Specification* (PS0251)
	Z8 Encore!® MCU User Manual (UM0128)
ZDS II	*Zilog Developer Studio II—Z8 Encore!® User Manual* (UM0130)

Appendix A—Flowcharts

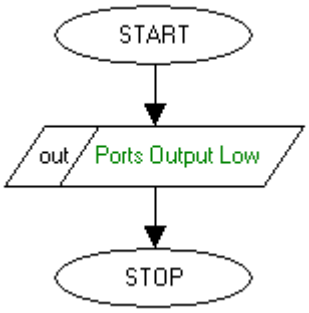

Figure 3. IdlePorts()

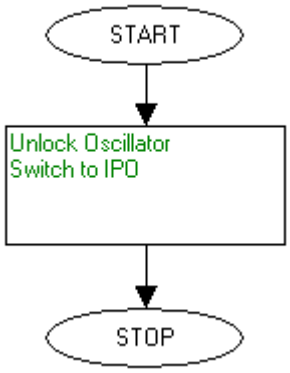

Figure 4. Use InternalOsc()

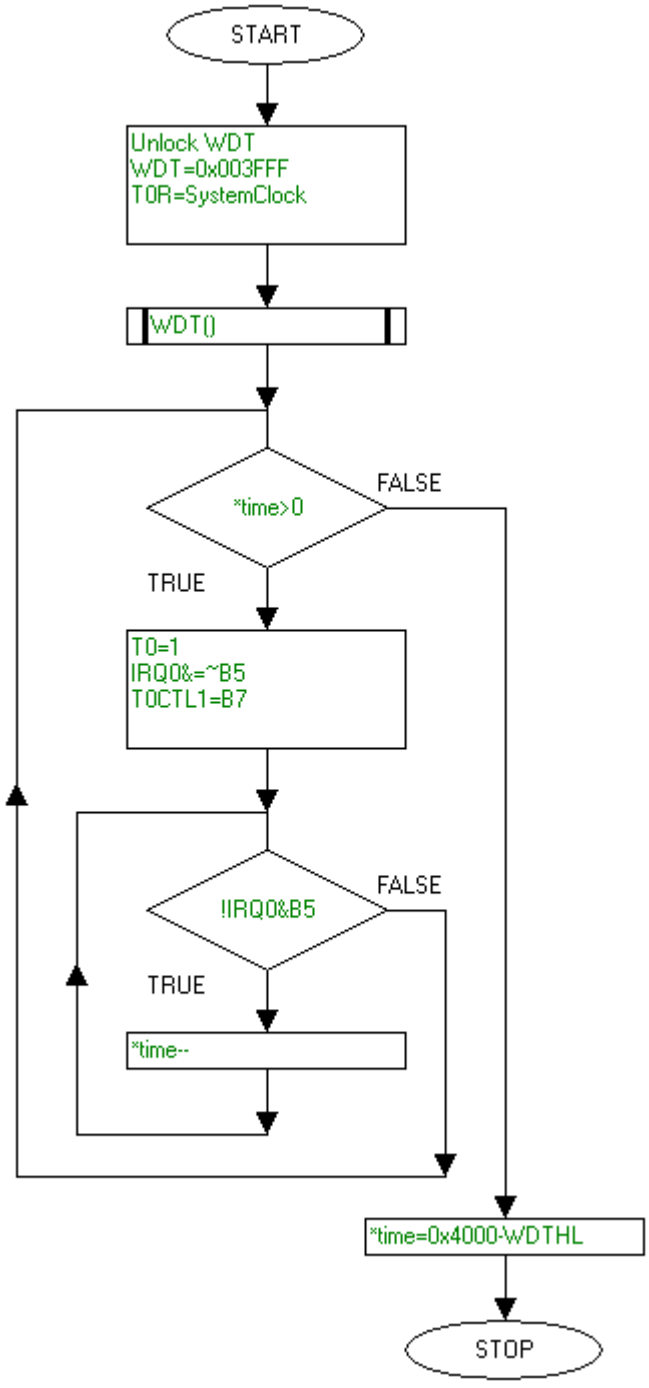

Figure 5. MeasureTime(int* time)

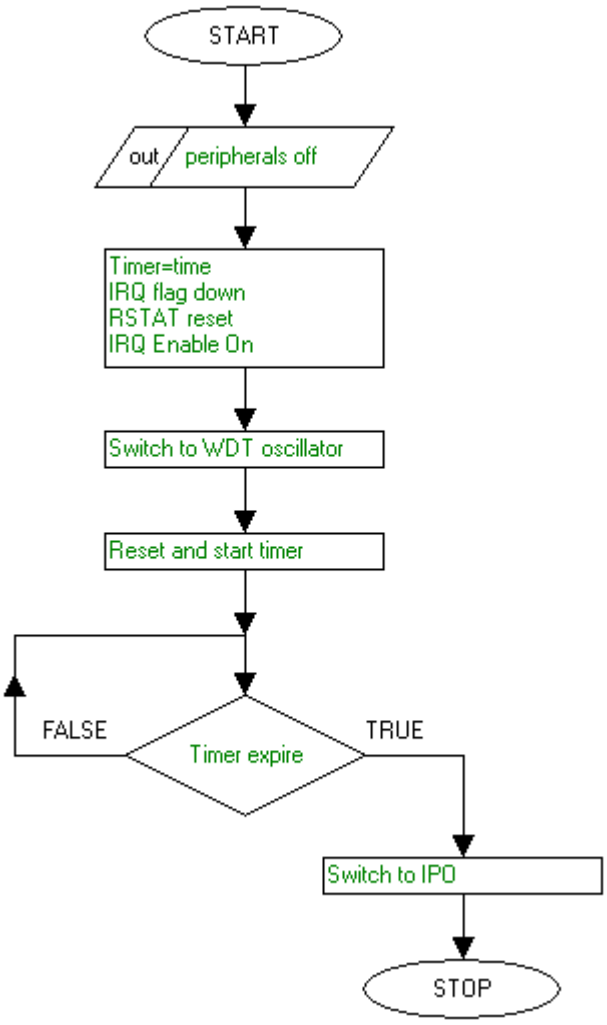

Figure 6. Wait(*int time)

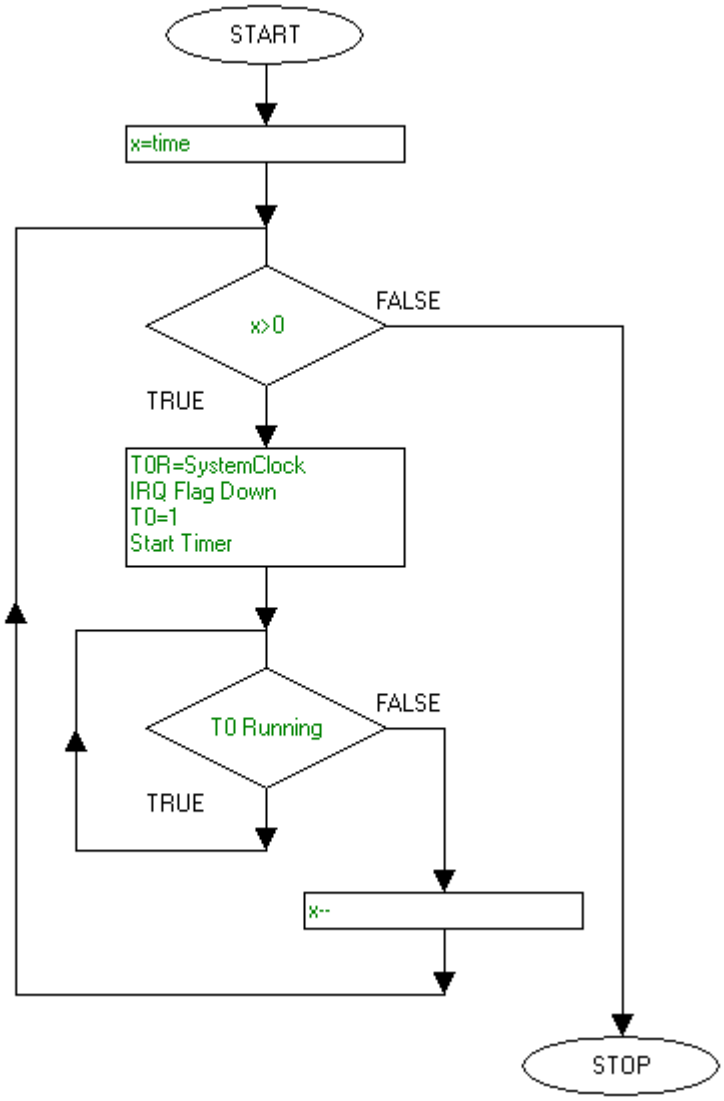

Figure 7. FullSpeedWait(unsigned int)

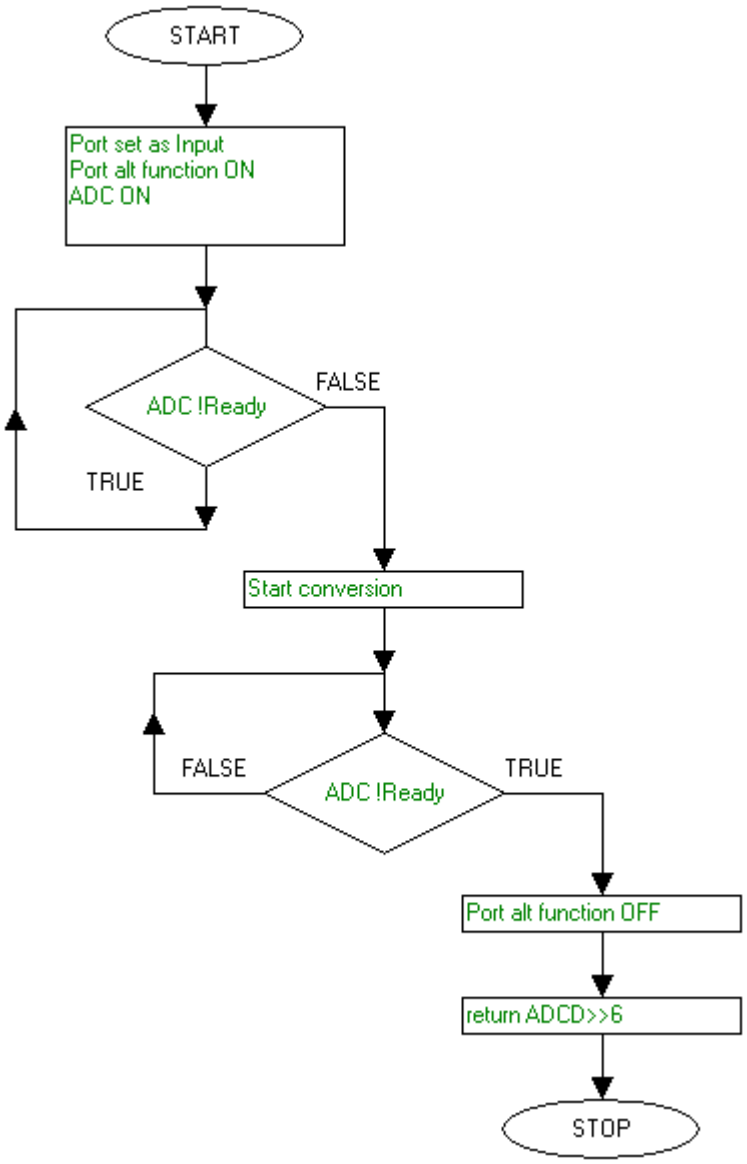

Figure 8. unsigned int ReadADC()

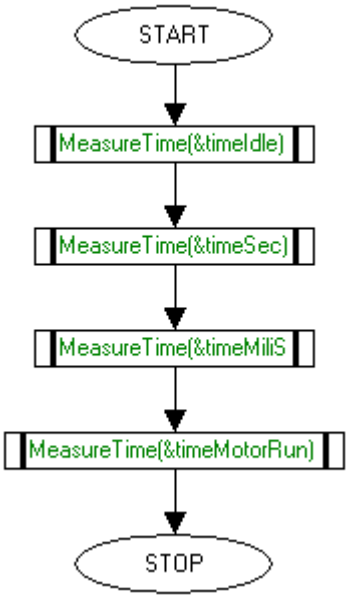

Figure 9. InitDurations()

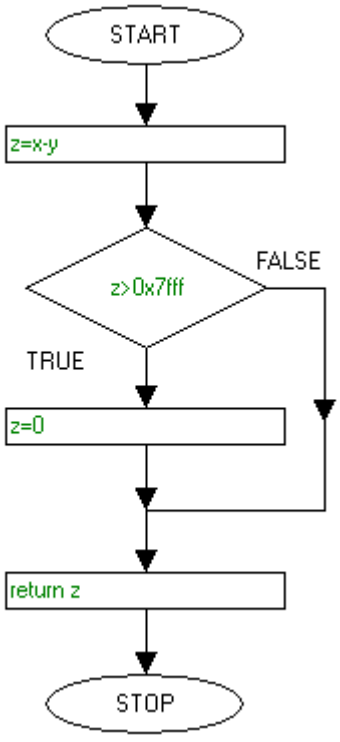

Figure 10. unsigned int CalDifference(unsigned int x, unsigned int y)

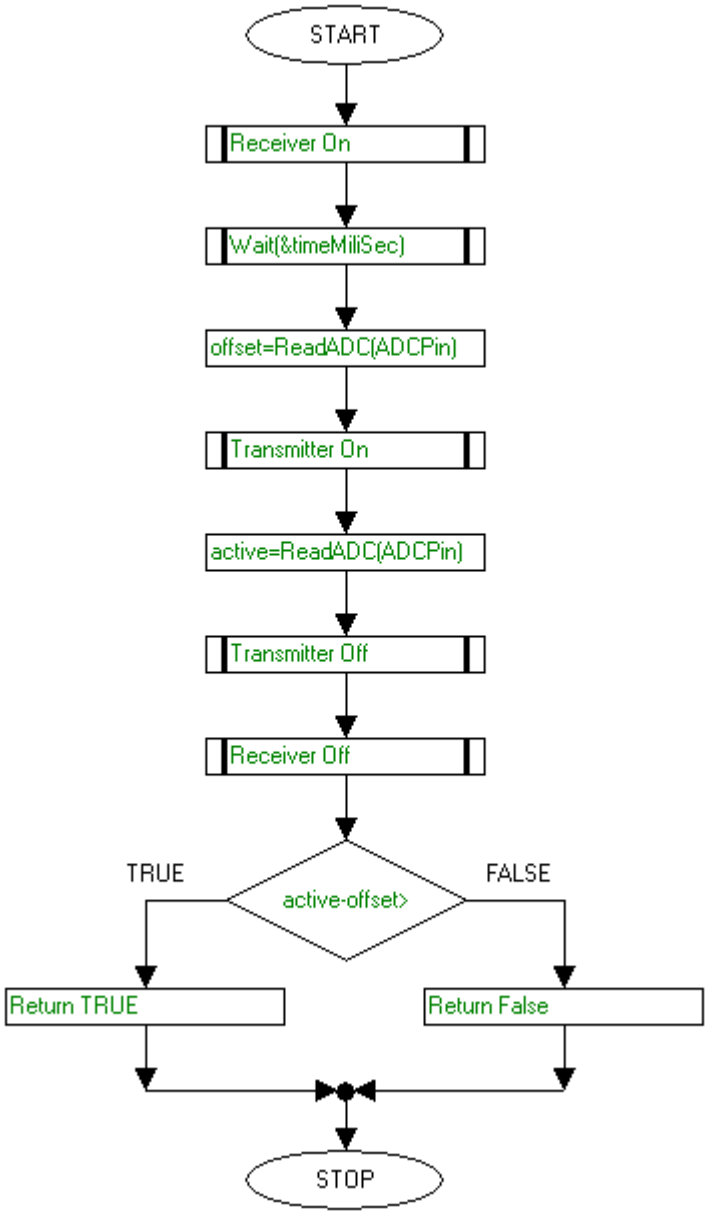

Figure 11. unsigned char TestRec()

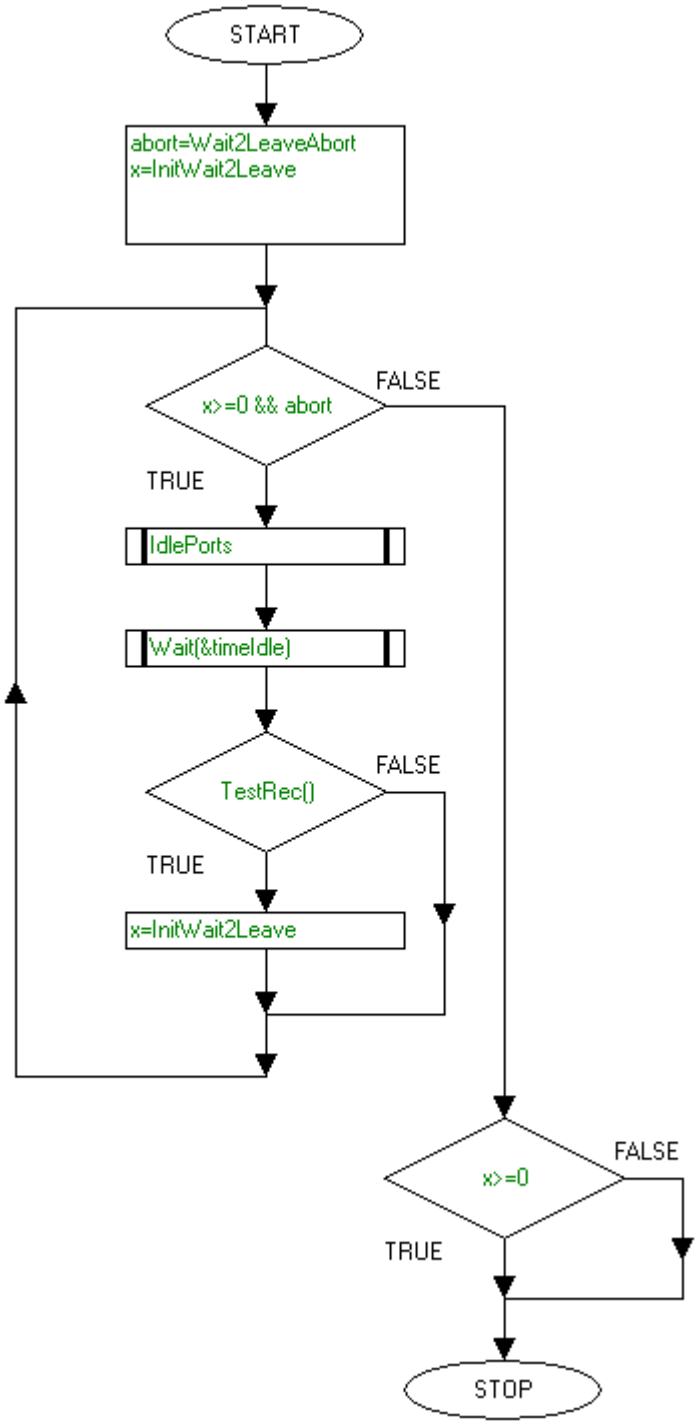

Figure 12. Wait2Leave()

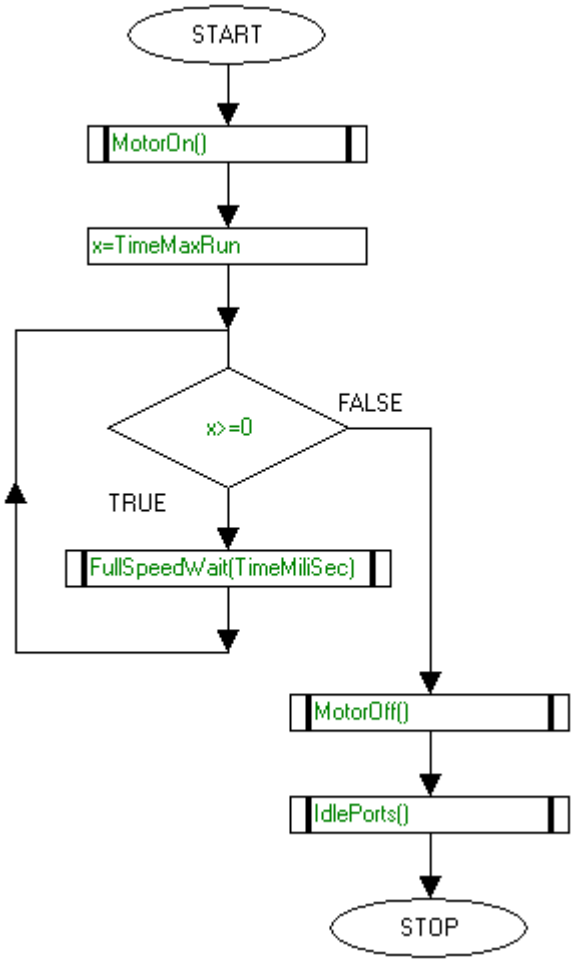

Figure 13. RunMotor()

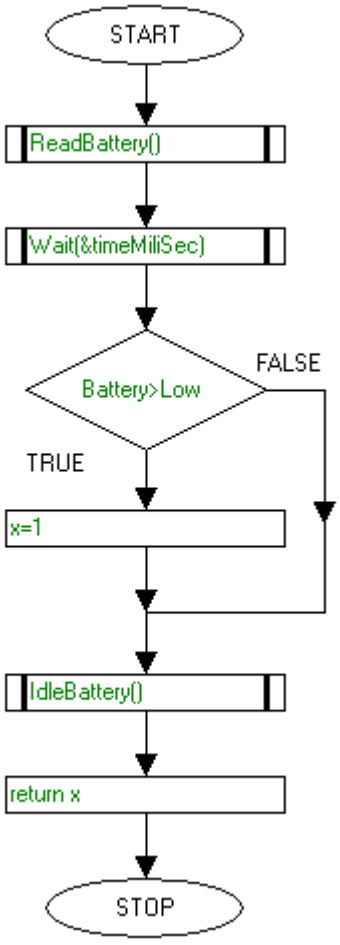

Figure 14. unsigned char BatteryOK()

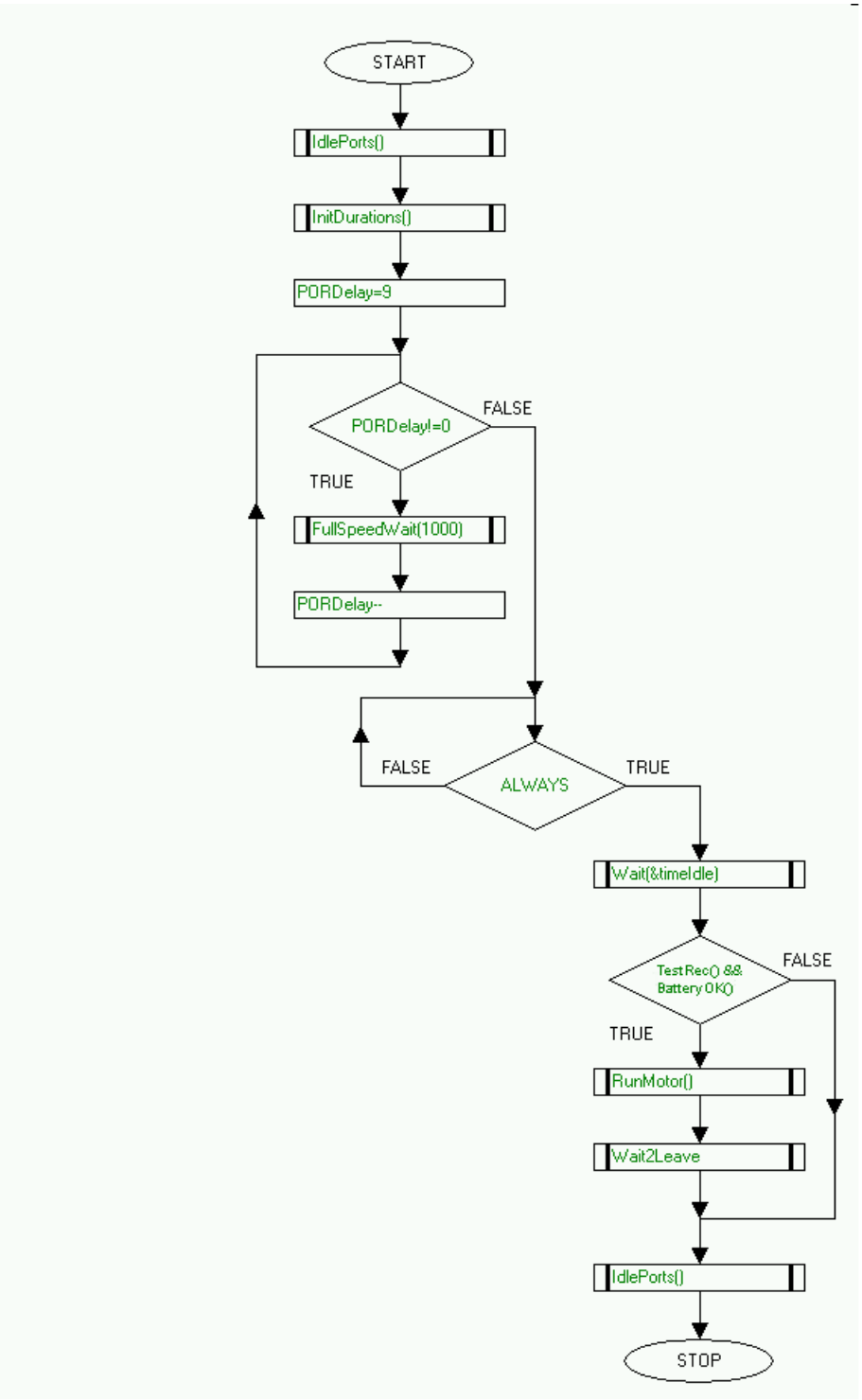

Figure 15. main()

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