Application Note

Using the Z8 Encore!® MCU as a Tachometer

AN018301-0604

ZiLOG Worldwide Headquarters • 532 Race Street • San Jose, CA 95126
Telephone: 408.558.8500 • Fax: 408.558.8300 • www.ZiLOG.com

This publication is subject to replacement by a later edition. To determine whether a later edition exists, or to request copies of publications, contact:

ZiLOG Worldwide Headquarters
532 Race Street
San Jose, CA 95126
Telephone: 408.558.8500
Fax: 408.558.8300
www.zilog.com

ZiLOG is a registered trademark of ZiLOG Inc. in the United States and in other countries. All other products and/or service names mentioned herein may be trademarks of the companies with which they are associated.

Information Integrity

The information contained within this document has been verified according to the general principles of electrical and mechanical engineering. Any applicable source code illustrated in the document was either written by an authorized ZiLOG employee or licensed consultant. Permission to use these codes in any form, besides the intended application, must be approved through a license agreement between both parties. ZiLOG will not be responsible for any code(s) used beyond the intended application. Contact the local ZiLOG Sales Office to obtain necessary license agreements.

Document Disclaimer

©2004 by ZiLOG, Inc. All rights reserved. Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZiLOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZiLOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. Except with the express written approval ZiLOG, use of information, devices, or technology as critical components of life support systems is not authorized. No licenses or other rights are conveyed, implicitly or otherwise, by this document under any intellectual property rights.

AN018301-0604

Table of Contents

List of Figures iv
List of Tables iv
Abstract 1
Z8 Encore!® Flash Microcontrollers 1
Discussion 1
Photoreflective Sensor 2
Developing the Z8 Encore!® Tachometer 2
Hardware Architecture 4
Software Implementation 4
Testing 5
Summary 7
Appendix A—References 8
Appendix B—Flowcharts 9
Appendix C—Source Code 11
C Files 11
Header Files 24
Appendix D—Alexan IR Emitting Diode 27
Appendix E—Alexan WPT-440F IR Photo Transistor 29

List of Figures

Figure 1. Circuitry for the Photoreflective Sensor 2
Figure 2. Z8 Encore!® MCU-based Tachometer Setup 3
Figure 3. Photoreflective Sensor Circuit Board Details 3
Figure 4. Hardware Block Diagram for the Z8 Encore!® Tachometer 4
Figure 5. Flowchart for the Main Routine 9
Figure 6. Flowchart for the Port D Interrupt Routine 10
Figure 7. Flowchart for the Timer0 Interrupt Routine 10

List of Tables

Table 1. Displayed and Actual Values from the Z8 Encore!® Tachometer 7
Table 2. List of References 8

Abstract

This Application Note describes how to use the Z8 Encore!® MCU as a tachometer to measure the number of revolutions of a rotating object over a specified period of time. A locally available phototransistor is used to capture infrared light that is beamed onto a flywheel which is attached to the rotating shaft of a motor. The signals from the phototransistor are fed to the Z8 Encore!® MCU's GPIO port; the Z8 Encore!® on-chip timer and an interrupt vector are used to compute the revolutions per minute of the motor.

The source code associated with this Application Note is contained in the `AN0183-SC01.zip` file, which is available on the ZiLOG website.

Z8 Encore!® Flash Microcontrollers

ZiLOG's Z8 Encore!® products are based on the new eZ8 CPU and introduce Flash memory to ZiLOG's extensive line of 8-bit microcontrollers. Flash memory in-circuit programming capability allows for faster development time and program changes in the field. The high-performance register-to-register based architecture of the eZ8 core maintains backward compatibility with ZiLOG's popular Z8 MCU.

Z8 Encore!® MCUs combine a 20 MHz core with Flash memory, linear-register SRAM, and an extensive array of on-chip peripherals. These peripherals make the Z8 Encore!® MCU suitable for a variety of applications including motor control, security systems, home appliances, personal electronic devices, and sensors.

Discussion

A tachometer is a device that measures the rotational speed of motors, wheels, or engines. The signal received by the tachometer is directly proportional to the rotational speed of the motor, wheel, or engine. Tachometers are widely used to measure and monitor the rotational speed of various engines such as those found in cars and airplanes.

A number of techniques are used to measure the rotational speed of a motor. One method is to measure the time taken by a motor to generate high and low pulses. Another method is to measure rotational speeds without direct contact with the device being measured. Light sensors are placed beside the motor and the rotational speed is measured by the reflection of light transmitted to the wheel.

The Z8 Encore!® MCU-based tachometer employs the method where a light-emitting device coupled with a phototransistor is used to capture reflected light from a flywheel attached to the rotating shaft of a motor.

Photoreflective Sensor

Photoreflective sensors combine a photoemitting device with a phototransistor. The phototransistor converts the reflected light beams (obtained as input) into signals for processing. A locally available infrared-emitting diode and a phototransistor were used to form the photoreflective sensor for the Z8 Encore!® MCU-based tachometer described in this Application Note.

Note: Any photoemitter and phototransistor unit can be used.

The data sheets for the IR Emitting Diode (Module No: IE-0575HLG) and the phototransistor (Module No: WPT-440F) are presented in Appendices D and E on pages 27 and 29, respectively. Figure 1 illustrates the circuitry for the photoreflective sensor using these devices.

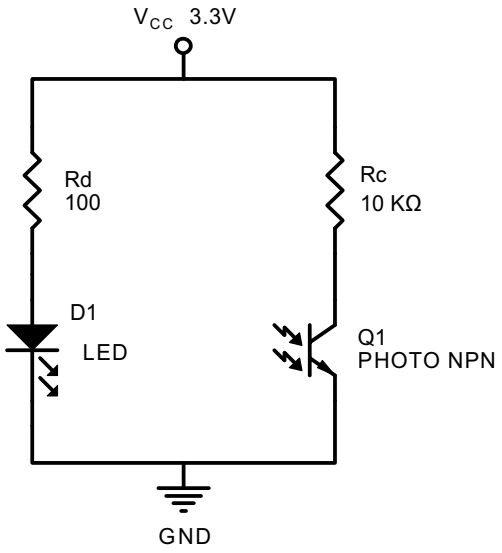


Figure 1. Circuitry for the Photoreflective Sensor

Developing the Z8 Encore!® Tachometer

The Z8 Encore!® MCU-based tachometer uses a flywheel encoder, which is attached to the rotating shaft of a motor. The flywheel is divided into 20 parts that are alternately black and white. This configuration is used to drive a phototransistor to saturation and cut-off alternately. The output of the phototransistor is fed to bit 2 of Port D on the Z8 Encore!® MCU. Figures 2 and 3 are snapshots of the flywheel encoder, the photoreflective sensor circuit board, and the Z8 Encore!® Development Board.

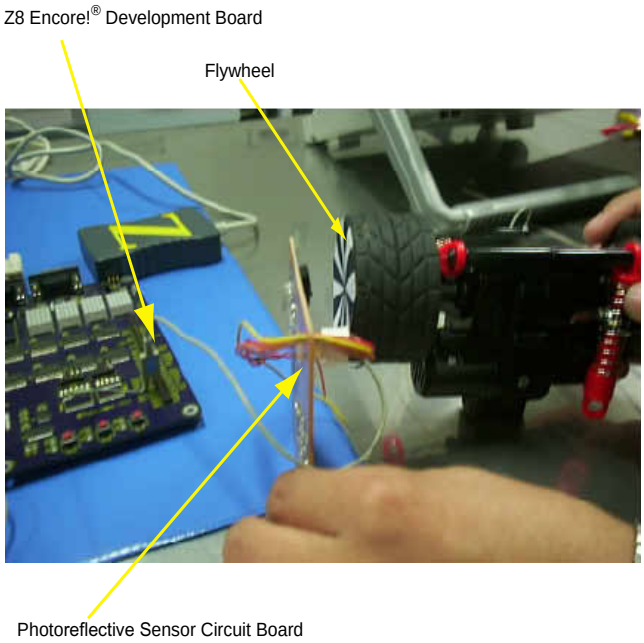


Figure 2. Z8 Encore!® MCU-based Tachometer Setup

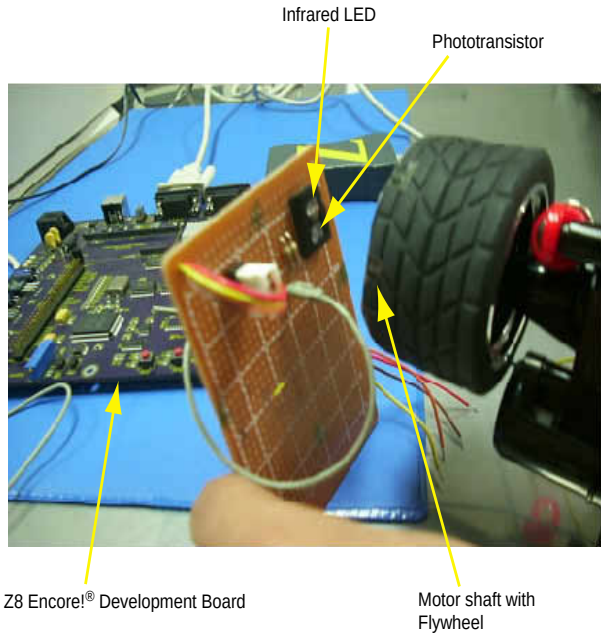


Figure 3. Photoreflective Sensor Circuit Board Details

The hardware and software used to create this application are discussed in the next section.

Hardware Architecture

This application uses a photoreflective sensor composed of a phototransistor (Module no: WPT-440F) and an infrared emitting diode (Module no: IE-0575HLG). The photoreflective sensor detects the infrared light reflected by the wheel encoder. Figure 4 illustrates the hardware block diagram for the Z8 Encore!® tachometer.

Note: Any general-purpose phototransistor and infrared light-emitting diode can be used for the tachometer application. See Appendix A on page 8 for sites that provide data sheets for these devices.

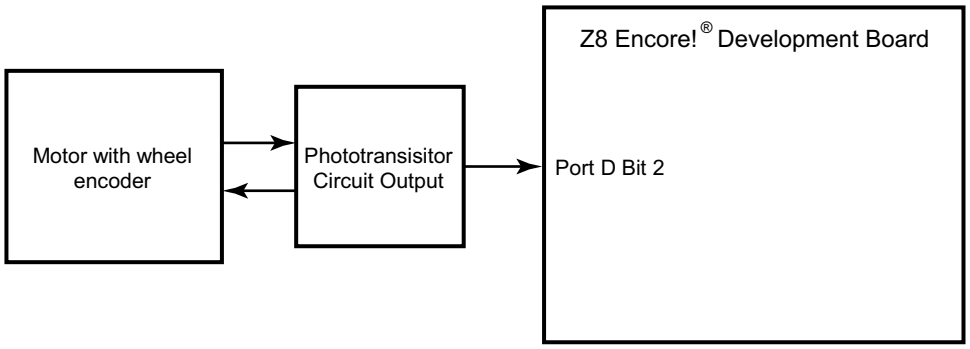


Figure 4. Hardware Block Diagram for the Z8 Encore!® Tachometer

The flywheel mounted on the motor shaft reflects infrared light onto the phototransistor. The emitted infrared light does not fall directly onto the phototransistor due to the acceptance angle of the phototransistor. The infrared light that bounces off the flywheel causes the phototransistor to switch on and off. This switching sends pulses to the Z8 Encore!® MCU via Port D bit 2 (PD2). For each falling-edge signal encountered at PD2, an interrupt is generated.

Infrared light is reflected only when it strikes the white portion of the flywheel. The circuit is mounted away from the center of the flywheel so that the waveform has sufficient spacing to ensure proper triggering of the phototransistor.

Software Implementation

Z8 Encore!® Flash microcontrollers feature four built-in timers that can be used as interrupt sources or event counters, or can be used to generate pulse-width mod-

ulation (PWM). Using these timers, the Z8 Encore!® MCU is used to measure rotational speed, which is the basic function of a tachometer.

Z8 Encore!® microcontrollers feature both internal and external interrupt sources. The interrupts are of the following types: communications peripheral interrupts, timer interrupts, and external-edge-triggered interrupts. With these interrupts, various applications are possible.

In this application, the software uses one internal (Timer0) and one external (Port D) interrupt source. Timer0 is configured to generate an interrupt every 100 milliseconds. Port D is configured to generate an interrupt every time it encounters a rising edge from its input pin. With this configuration, a counter is incremented every time it enters the interrupt service routine (ISR) for Port D. The counter increments until Timer0 generates an interrupt. When Timer0 generates an interrupt, it calls functions that convert the counter value and display the result on the 5x7 LED Matrix display of the Z8 Encore!® Development Board.

Figure 5 on page 9 illustrates the flowchart for the main routine of the tachometer program. The main routine initializes Ports E and G, which are used by the 5x7 LED matrix display. It also initializes and enables the interrupts Timer0 and Port D. The program enters a continuous loop where its function is to display the value measured.

Figure 6 on page 10 illustrates the interrupt service routine of the rising edge interrupt on Port D. Its main function is to increment a counter whenever a rising edge is encountered.

Figure 7 on page 10 illustrates the interrupt service routine of Timer0. It computes and updates the value to be displayed on the 5x7 LED matrix display. The Timer0 ISR also resets the value of the counter to reinitialize its count.

Testing

This section contains the details necessary for testing the functionality of the Z8 Encore!® MCU-based tachometer. The setup diagram is illustrated in Figure 4 on page 4.

For schematics and further details about the Z8 Encore!® Development Board, please refer to the *Z8 Encore!® Flash Microcontroller Development Kit User Manual (UM0146)*.

The compiled code, available in the `AN0183-SC01.zip` file (on ZiLOG.com), is downloaded to Z8 Encore!® MCU Flash memory using the ZDSII IDE for Z8 Encore!® v4.8.0[1].

1. For details about how to download the code to the Flash memory on the Z8 Encore!® MCU, please refer to the ZiLOG Developer Studio II–Z8 Encore!® User Manual (UM0130).

Equipment Used

- The Z8 Encore!® Development Kit includes:
  - Z8 Encore!® Development Board
    - Z8 Encore!® MCU (Z8F642x)
    - LED array with four 5x7 LED matrices
  - Smart Cable to connect a PC to the Z8 Encore!® Development Board
  - Universal 5VDC Power Supply
  - ZDSII–Z8 Encore!®
- Photoreflective Sensor Circuit Board
- DC Motor with a flywheel encoder attached to its shaft

Procedure

Instructions for using the Z8 Encore!® MCU-based tachometer are provided below.

1. Refer to Figure 1 on page 2 and Figure 4 on page 4 to connect the Z8 Encore!®-based tachometer circuitry.
2. Launch ZDSII and open the `tachometer.pro` program. Download this program to the Z8 Encore!® Development Board and reset the Board.
3. Start the DC motor and observe the displayed value on the 5x7 LED matrix display.

Results

The measured revolutions per minute is displayed on the Z8 Encore!® Development Board's 5x7 LED matrix display. The value is displayed as follows:

LED1—Thousands Value

LED2—Hundreds Value

LED3—Tens Value

LED4—Ones Value

For example, if the display reads 2000, the actual value is 2000 revolutions per minute (2000RPM). Actual values were verified using simple calculations to convert cycles per second to revolutions per minute. Table 1 is a sample of the displayed and actual values for motor speeds obtained during testing.

Table 1. Displayed and Actual Values from the Z8 Encore!® Tachometer

Actual Motor Speed (in RPM)	Displayed Motor Speed (in RPM)
405	0405
750	0750
2250	2250
7800	7800
9600	9600

Summary

This Application Note demonstrates the capability of ZiLOG's Z8 Encore!® Flash microcontrollers with their built-in timers to measure the speed of a rotating body with simple programming and some discrete devices such as a phototransistor and an infrared light-emitting diode.

The development of the hardware and software to implement the Z8 Encore!® MCU-based tachometer is straightforward. The hardware uses few discrete devices, thus making it easy to design. The software utilizes simple interrupt routines, thus reducing code size and complexity. Additionally, non-contact tachometers are easy to install without requiring a change to existing motor configuration.

Appendix A—References

Further details about the Z8 Encore!® products, ZDS II, photo transistors and light emitting diodes can be found in the references listed in Table 2.

Table 2. List of References

Topic	Document Name
Z8 Encore!® MCU	Z8 Encore!® Flash Microcontroller Development Kit User Manual (UM0146)
	Z8 Encore!® Z8F642x Flash Microcontroller Development Kit User Manual (UM0151)
	Z8 Encore!® 8K/4K Series Development Kit User Manual (UM0150)
	Z8 Encore!® Z8F640 Family of 8-Bit Microcontrollers Product Specification (PS0176)
	Z8 Encore!® 64K Series High-Performance 8-Bit Microcontrollers Product Specification (PS0199)
eZ8 CPU	eZ8 CPU User Manual (UM0128)
ZDS II-IDE	ZiLOG Developer Studio II–Z8 Encore!® User Manual (UM0130)
Photo transistor	Phototransistor S2829 data sheet available on www.hamamatsu.com.
Infrared light-emitting diode	IR LED XTH153W850 available on www.sunled.com.

Appendix B—Flowcharts

This appendix contains all the flowcharts associated with the Z8 Encore!®-based tachometer application described in this Application Note.

The flowchart for the main routine is illustrated in Figure 5.

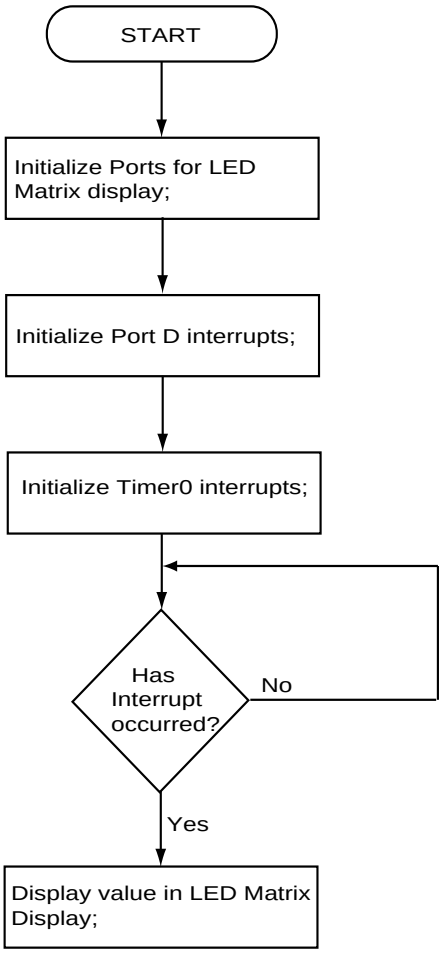


Figure 5. Flowchart for the Main Routine

The flowchart for the Port D interrupt routine is illustrated in Figure 6.

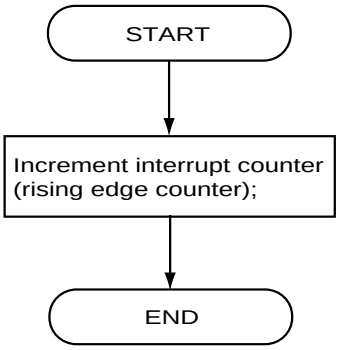


Figure 6. Flowchart for the Port D Interrupt Routine

The flowchart for the Timer0 interrupt routine is illustrated in Figure 7.

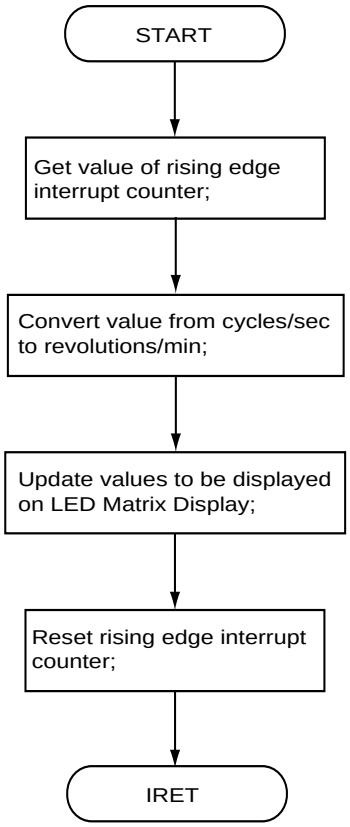


Figure 7. Flowchart for the Timer0 Interrupt Routine

Appendix C—Source Code

This appendix provides a listing of all source code associated with the application described in this document. The source code files are in the `AN0183-SC01.zip` file available for download on the ZiLOG website.

C Files

The following C files are listed in this section:

- main.c
- functions.c
- gpio.c
- interrupt.c

```
/*************************************************************
* File: main.c
* Description: This file contains all the initializations
* of ports used and interrupts.
*
* Copyright 2004 ZiLOG Inc. ALL RIGHTS RESERVED.
*
* The source code in this file was written by an
* authorized ZiLOG employee or a licensed consultant.
* The source code has been verified to the fullest
* extent possible.
*
* Permission to use this code is granted on a royalty-free
* basis. However users are cautioned to authenticate the
* code contained herein.
*
* ZiLOG DOES NOT GUARANTEE THE VERACITY OF THE SOFTWARE.
*************************************************************/
#include <stdio.h>
#include "gpio.h"
#include "interrupt.h"
#include "functions.h"
#include "Z8Encore.h"

extern char num_string[4];

main()                                  //Main Function
{
```

```
  init_led_gpio();            //Initialize Port E and Port G
                              //for 5x7 LED Matrix
  init_timer0();              //Initialize Timer0 Interrupt
  init_p2ad();                //Initialize Port D bit 2 for
                              //rising edge interrupt
  T0CTL |= 0x80;              //Enable Timer0
  EI();                       //Enable Interrupt
  turn_off_led();             //Turn off all LED

  while(1)                    //Continuous Loop
 {
  LedDisplay(num_string);     //Display value in 5x7 LED Matrix
 }
}/*main*/

/*****************************************************************
*********************End of main.c file*************************
*****************************************************************/



/*********************************************************
* File: functions.c
* Description: This file contains the functions that
* compute and update the values to be
* dislayed on the 5x7 LED Matrix Display
*
* Copyright 2004 ZiLOG Inc. ALL RIGHTS RESERVED.
*
* The source code in this file was written by an
* authorized ZiLOG employee or a licensed consultant.
* The source code has been verified to the fullest
* extent possible.
*
* Permission to use this code is granted on a royalty-free
* basis. However users are cautioned to authenticate the
* code contained herein.
*
* ZiLOG DOES NOT GUARANTEE THE VERACITY OF THE SOFTWARE.
*****************************************************************/
#include <stdio.h>
#include "z8encore.h"

#define mark_num 10                 //define the number of markers on
```

```
                                        //the wheel
#define rpm_ratio(60/mark_num)          // ratio for converting
                                        // frequencies to
                                        // revolutions

extern const char Display_Num[10];      // Array Initialization
extern char num_string[4];

int revolution = 0;
int wheel_mark = 0;                     // Initialize wheel counter to
                                        // zero
int ones_digit = 0;                     // Initialize counter1 to zero
int tens_digit = 0;                     // Initialize counter2 to zero
int hundreds_digit = 0;                 // Initialize counter3 to zero
int thousands_digit = 0;                // Initialize counter4 to zero
/*
*********************************************************************
** Routine: Update Display
** Parameters: void
** Return: void
** Purpose: This function updates the values to be displayed
** on the 5x7 LED matrix display
*********************************************************************
*/
void Update_Display(void)               //Function that updates the
                                        //measured value
{
  int i;
  for(i=0;i<10;i++)                     //loop that finds the appropriate
                                        //value to be displayed on the
                                        //5x7 led matrix display
 {
  if(i == ones_digit)
  {
   num_string[3] = Display_Num[i];      //Value to be displayed
                                        //on LED4 (most significant
                                        // digit)
  }
  if(i == tens_digit)
  {
   num_string[2] = Display_Num[i];      //Value to be displayed on
                                        //LED3
  }
  if(i == hundreds_digit)
```

```
  {
  num_string[1] = Display_Num[i];    //Value to be displayed on
                                     //LED2
  }
  if(i == thousands_digit)
  {
  num_string[0] = Display_Num[i];    //Value to be displayed on
                                     //LED1 (least significant
                                     //digit)
  }
 }
}/*Update_Display*/

/*
**********************************************************************
** Routine: Arrange Display
** Parameters: void
** Return: void
** Purpose: This function arranges the numbers to be displayed
**                                    on the 5x7 LED matrix display
**********************************************************************
*/
void Arrange_Display(void)
{
  revolution = wheel_mark * rpm_ratio;      //Computes the revolutions
                                            //per minute of the motor
  thousands_digit = (revolution ÷ 1000);    //Gets the value of the
                                            //Thousands Digit
  hundreds_digit = (revolution % 1000) ÷ 100;//Gets the value for the
                                            //hundreds position
  tens_digit = (revolution % 100) / 10;     //Gets the value for the
                                            //tens position
  ones_digit = (revolution % 10);           //Gets the value for the
                                            //ones position
}/*Arrange_Display*/

/*
**********************************************************************
** Routine: Reset_Digits
** Parameters: void
** Return: void
** Purpose: This function resets the variables used in
**                                    udating the display
```

```
*********************************************************************
*/
void Reset_Digits(void)
{
  ones_digit = 0;                    //reset counter1
  tens_digit = 0;                    //reset counter2
  hundreds_digit = 0;                //reset counter3
  thousands_digit = 0;               //reset counter4
  wheel_mark = 0;                    //reset wheel mark counter
  revolution = 0;                    //reset the revolution counter
}/*Reset_Digits*/

/*****************************************************************/
/*********************End of functions.c file*******************/
/*****************************************************************/



/*********************************************************
* File: gpio.c
* Description: This file contains the functions to be used
* to initialize the ports to be used by
* the 5x7 LED matrix display.
*
* Copyright 2004 ZiLOG Inc. ALL RIGHTS RESERVED.
*
* The source code in this file was written by an
* authorized ZiLOG employee or a licensed consultant.
* The source code has been verified to the fullest
* extent possible.
*
* Permission to use this code is granted on a royalty-free
* basis. However users are cautioned to authenticate the
* code contained herein.
*
* ZiLOG DOES NOT GUARANTEE THE VERACITY OF THE SOFTWARE.
*************************************************************/
#include <stdio.h>
#include "Z8Encore.h"
#include <string.h>

void LatchLed(int Inst);

#define SLEN 10
```

```
const char Zero_array[7] = {0x11, 0x06, 0x0A, 0x0A, 0x0C, 0x0E, 0x11};
//Condition of cathode of the LED display for Zero Digit

const char One_array[7] = {0x1B, 0x13, 0x1B, 0x1B, 0x1B, 0x1B, 0x11};
//Condition of cathode of the LED display for One Digit

const char Two_array[7] = {0x11, 0x0E, 0x0E, 0x11, 0x0F, 0x0F, 0x00};
//Condition of cathode of the LED display for Two Digit

const char Three_array[7] = {0x11, 0x0E, 0x1E, 0x11, 0x1E, 0x0E, 0x11};
//Condition of cathode of the LED display for Three Digit

const char Four_array[7] = {0x0F, 0x0E, 0x0E, 0x00, 0x1E, 0x1E, 0x1E};
//Condition of cathode of the LED display for Four Digit

const char Five_array[7] = {0x00, 0x0E, 0x0F, 0x01, 0x1E, 0x0E, 0x11};
//Condition of cathode of the LED display for Five Digit

const char Six_array[7] = {0x11, 0x0E, 0x0F, 0x01, 0x0E, 0x0E, 0x11};
//Condition of cathode of the LED display for Six Digit

const char Seven_array[7] = {0x00, 0x0E, 0x1E, 0x1D, 0x1B, 0x1B, 0x1B};
//Condition of cathode of the LED display for Seven Digit

const char Eight_array[7] = {0x11, 0x0E, 0x0E, 0x11, 0x0E, 0x0E, 0x11};
//Condition of cathode of the LED display for Eight Digit

const char Nine_array[7] = {0x11, 0x0E, 0x0E, 0x10, 0x1E, 0x0E, 0x11};
//Condition of cathode of the LED display for Nine Digit

const char Display_Num[10] = {'0', '1', '2', '3', '4', '5', '6', '7',
'8', '9'};                          //Array of digits from 0–9

char Run_array[SLEN][7];

/*************************************************************/

void init_led_gpio(void)            //Initialize Port E and G for 5x7
                                    //LED Matrix Display
{
  PEA = ALT_FUN;                    //Port E Alternate Function
  PEC = 0x00;                       //No Alternate Function
  PGA = ALT_FUN;                    //Port G Alternate Function
```

```
  PGC = 0x00;                        //No Alternate Function

  PEA = OUT_CTL;                     //Port E Output Control
  PEC = 0x00;                        //Output configured as push pull
  PGA = OUT_CTL;                     //Port G Output Control
  PGC = 0x00;                        //Output configured as push pull

  PEA = DATA_DIR;                    //Port E Data Direction
  PEC = 0x00;                        //Port E is Output
  PGA = DATA_DIR;                    //Port G Data Direction
  PGC = 0x00;                        //Port G is Output
 }
/*****************************************************************/

void turn_off_led(void)//Turn off all LED
{
                                     //LED1
  PEO |= 0x1f;
  PGO &= 0x80;                       //load data

  PEO &= 0xdf;
  PEO |= 0x20;                       //latch data
                                     //LED2
  PEO &= 0x7f;
  PEO |= 0x80;                       //latch data
                                     //LED3
  PGO &= 0x7f;
  PGO |= 0x80;                       //latch data
                                     //LED3
  PEO &= 0xbf;
  PEO |= 0x40;                       //latch data
}

/*****************************************************************/

void GetNumArray(char ch, char array[7])
{
  int k;
  if ( ch == '0')
  {
  for (k=0; k<7; k++)
  array[k] = Zero_array[k];
  }
  else if ( ch == '1')
```

```
{
for (k=0; k<7; k++)
array[k] = One_array[k];
}
else if ( ch == '2')
{
for (k=0; k<7; k++)
array[k] = Two_array[k];
}
else if ( ch == '3')
{
for (k=0; k<7; k++)
array[k] = Three_array[k];
}
else if ( ch == '4')
{
for (k=0; k<7; k++)
array[k] = Four_array[k];
}
else if ( ch == '5')
{
for (k=0; k<7; k++)
array[k] = Five_array[k];
}
else if ( ch == '6')
{
for (k=0; k<7; k++)
array[k] = Six_array[k];
}
else if ( ch == '7')
{
for (k=0; k<7; k++)
array[k] = Seven_array[k];
}
else if ( ch == '8')
{
for (k=0; k<7; k++)
array[k] = Eight_array[k];
}
else if ( ch == '9')
{
for (k=0; k<7; k++)
array[k] = Nine_array[k];
}
```

```
}

/*****************************************************************/

void LedDisplay(char string[])
{
  int k, kk;
  char temp_array[7], ch;
  int slen, delay;

  slen = strlen(string);

  for(kk=0;kk<slen; kk++)
  {
  ch = string[kk];
  GetNumArray(ch, temp_array);

  for (k=0; k<7; k++)
  Run_array[kk][k] = temp_array[k];
   }

  for(delay=0; delay <100; delay++)
  {
  PGO = 0x01;
  for (k=0; k<7; k++)
  {
  for(kk=0; kk<4; kk++)
  {
  PEO = Run_array[kk][k];
  LatchLed(kk+1);
  }
  PGO = (PGO <<1);
  }
                              // turn-off led D2
  PEO |= 0x1f;
  PGO &= 0x80;                // load data
  PGO &= 0x7f;
  PGO |= 0x80;                // latch data
 }
}

/*****************************************************************/

void LatchLed(int Inst)
```

```
{
  switch (Inst)
  {
  case 4:                    // Port E6
  PEO &= 0xBF;
  PEO |= 0x40;
  PEO &= 0xBF;
  break;

  case 3:                    // Port E5
  PEO &= 0xDF;
  PEO |= 0x20;
  PEO &= 0xDF;
  break;

  case 2:                    // Port G7
  PGO &= 0x7F;
  PGO |= 0x80;
  PGO &= 0x7F;
  break;

  case 1:                    // Port E7
  PEO &= 0x7F;
  PEO |= 0x80;
  PEO &= 0x7F;
  break;

  case 0:                    // Latch all LED Displays at same
                             // time
                             // Port E5, E6, E7, G7
  PEO &= 0x1F;
  PGO &= 0x7F;

  PEO |= 0xE0;
  PGO |= 0x80;

  PEO &= 0x1F;
  PGO &= 0x7F;
  break;
  }
}

/*****************************************************************/
/*********************End of gpio.c.c file***********************/
```

```
/*****************************************************************/



/*********************************************************
* File: interrupt.c
* Description: This file contains all the functions used
* to initialize the Interrupts.
*
* Copyright 2004 ZiLOG Inc. ALL RIGHTS RESERVED.

*
* The source code in this file was written by an
* authorized ZiLOG employee or a licensed consultant.
* The source code has been verified to the fullest
* extent possible.
*
* Permission to use this code is granted on a royalty-free
* basis. However users are cautioned to authenticate the
* code contained herein.
*
* ZiLOG DOES NOT GUARANTEE THE VERACITY OF THE SOFTWARE.
*****************************************************************/
#include <stdio.h>
#include <string.h>
#include "functions.h"
#include "Z8Encore.h"

extern void isr_timer0(void);
extern void isr_p2ad(void);

#define delay 10                    //Update tachometer reading every
                                    //second

extern const char Display_Num[10];  //Array Initialization
char num_string[4];                 //Initialize array for display
int Update_Delay;                   //Delay for RPM update

extern int revolution;              //Initialize revolution counter
extern int wheel_mark;              //Initialize wheel counter
extern int ones_digit;              //Initialize counter1
extern int tens_digit;              //Initialize counter2
extern int hundreds_digit;          //Initialize counter3
extern int thousands_digit;         //Initialize counter4
```

```
/*****************************************************************/

void init_p2ad(void)
{
  SET_VECTOR(P2AD, isr_p2ad);//Pass the vector number and the
                             //ISR address to be placed into
                             //the interrupt table

  PDA = ALT_FUN;             //Port D Alternate Function
  PDC |= 0x00;               //No Alternate Function

  PDA = DATA_DIR;            //Port D Data Direction
  PDC |= 0x04;               //Port D bit 2 is configured as
                             //Input

  IRQ1E0 &= 0xFB;            //Port D interrupt is
  IRQ1E1 |= 0x04;            //nominal in priority

  IRQES &= 0x04;             //Interrupt is rising edge
                             //triggered
  PS |= 0x04;                //Port D is selected over Port A
} /*init_p2ad*/

/*****************************************************************/

#pragma interrupt
void isr_p2ad(void)          //Interrupt service routine for
                             //Port D bit2
{
  wheel_mark++;              //increment counter if received
                             //interrupt
}/*isr_p2ad*/

/*****************************************************************/

void init_timer0(void)               //Initialize Timer0 to have a
                                     //time-out of 1 millisecond
{
  SET_VECTOR(TIMER0, isr_timer0);    //Pass the vector number and the
                                     //ISR address to be placed into
                                     //the interrupt table

  T0H = 0x00;                        //Timer0 High Byte count value
```

```
  T0L = 0x00;                        //Timer0 Low Byte count value

  /*for a 20Mhz clock source, use 0x3D09 for T0CPH and T0CPL*/

  T0CPH = 0x38;                      //Timer0 Reload High Byte
                                     //register
  T0CPL = 0x40;                      //Timer0 Reload Low Byte
                                     //register

  T0CTL = 0x39;                      //Timer0 is in continuous mode
                                     //operation with prescale of 128
  IRQ0E0 |= 0x20;                    //Timer0 is configured
  IRQ0E1 |= 0x20;                    //for highest priority interrupt

  num_string[0]='0';                 //Initialize 5x7 LED1 Matrix
                                     //Display
  num_string[1]='0';                 //Initialize 5x7 LED2 Matrix
                                     //Display
  num_string[2]='0';                 //Initialize 5x7 LED3 Matrix
                                     //Display
  num_string[3]='0';                 //Initialize 5x7 LED4 Matrix
                                     //Display
 }/*init_timer0*/

/*****************************************************************/

#pragma interrupt
void isr_timer0(void)                //Interrupt service routine for
                                     //Timer0
{
  Update_Delay++;
  if(Update_Delay==delay)
  {
  Arrange_Display();                 //call function Arrange_Display
  Update_Display();                  //call function update
  Reset_Digits();                    //reset all digits/variables
  Update_Delay = 0;
  }
} /*isr_timer0*/

/*****************************************************************
********************End of interrupt.c file********************
*****************************************************************/
```

Header Files

The following header files are listed in this section:

- functions.h
- gpio.h
- interrupt.h

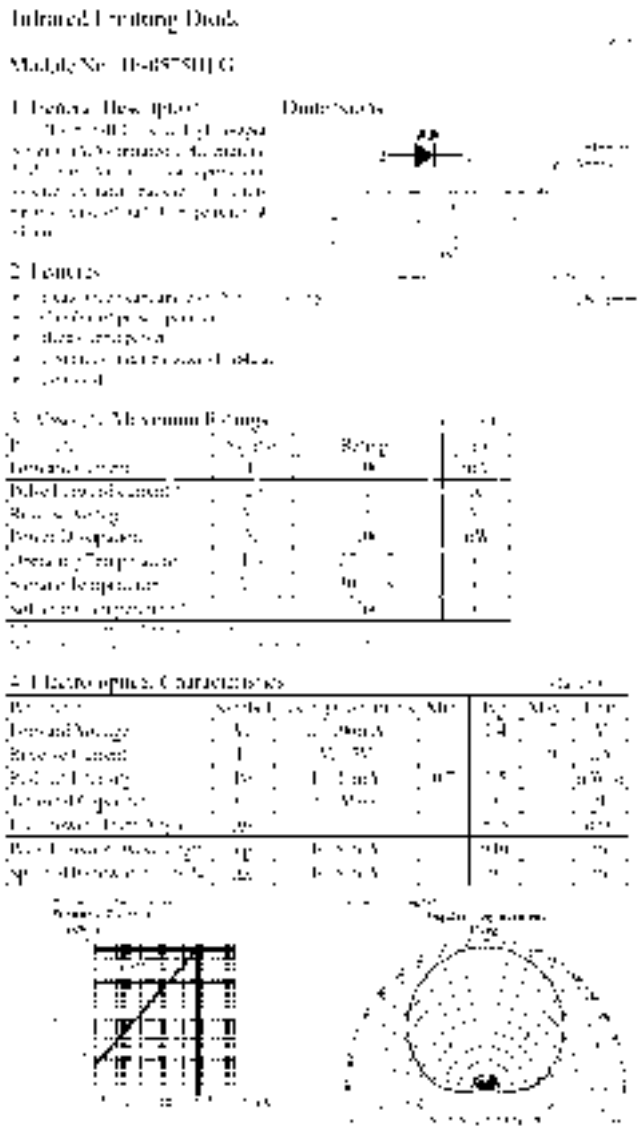
```
/*********************************************************
* File: functions.h
* Description: This file contains function prototypes.
*
* Copyright 2004 ZiLOG Inc. ALL RIGHTS RESERVED.

*
* The source code in this file was written by an
* authorized ZiLOG employee or a licensed consultant.
* The source code has been verified to the fullest
* extent possible.
*
* Permission to use this code is granted on a royalty-free
* basis. However users are cautioned to authenticate the
* code contained herein.
*
* ZiLOG DOES NOT GUARANTEE THE VERACITY OF THE SOFTWARE.
*************************************************************/

#include <stdio.h>
#include "z8encore.h"

void Update_Display(void);
void Arrange_Display(void);
void Reset_Digits(void);

/*************************************************************
********************End of function.h file*******************
*************************************************************/


/*********************************************************
* File: gpio.h
* Description: This file contains function prototypes.
*
* Copyright 2004 ZiLOG Inc. ALL RIGHTS RESERVED.
```

```
*
* The source code in this file was written by an
* authorized ZiLOG employee or a licensed consultant.
* The source code has been verified to the fullest
* extent possible.
*
* Permission to use this code is granted on a royalty-free
* basis. However users are cautioned to authenticate the
* code contained herein.
*
* ZiLOG DOES NOT GUARANTEE THE VERACITY OF THE SOFTWARE.
*****************************************************************/

#include <stdio.h>

void init_led_gpio(void);
void turn_off_led(void);
void GetNumArray(char ch, char array[7]);
void LedDisplay(char string[]);
void Arrange_Display(void);
void Reset_Digits(void);

/*****************************************************************
********************End of gpio.h file********************
*****************************************************************/



/*******************************************************
* File: Interrupt.h
* Description: This file contains function prototypes.
*
* Copyright 2004 ZiLOG Inc. ALL RIGHTS RESERVED.
*
* The source code in this file was written by an
* authorized ZiLOG employee or a licensed consultant.
* The source code has been verified to the fullest
* extent possible.
*
* Permission to use this code is granted on a royalty-free
* basis. However users are cautioned to authenticate the
* code contained herein.
*
```

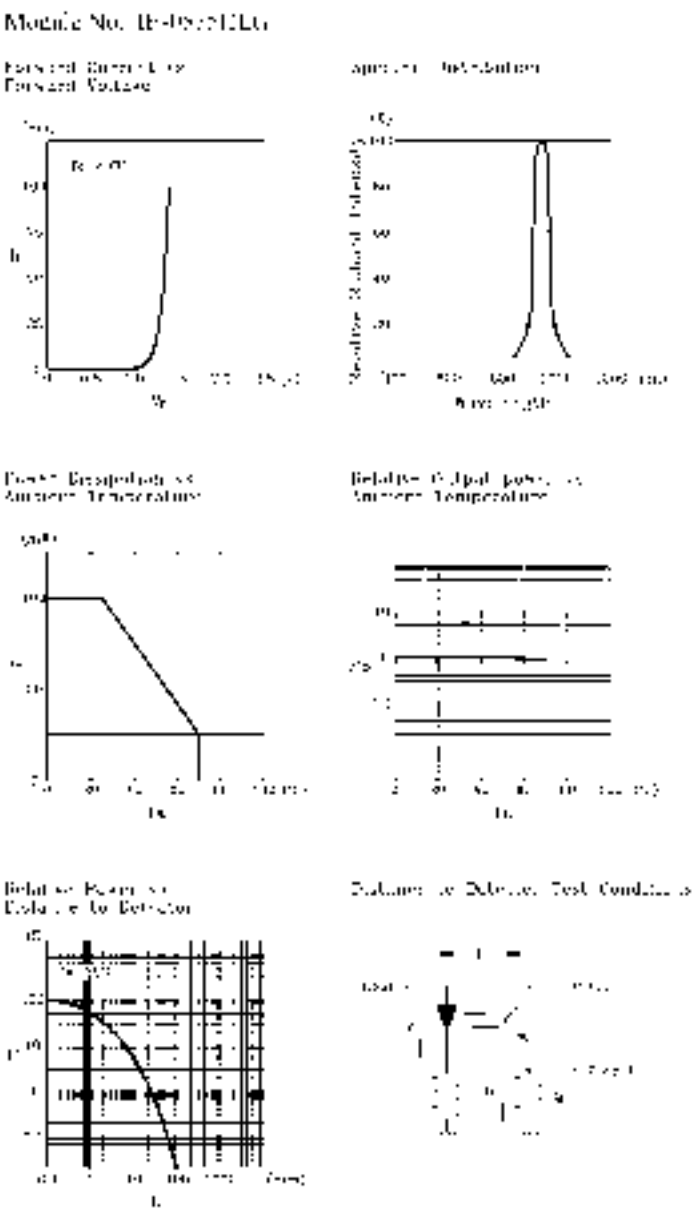
```
* ZiLOG DOES NOT GUARANTEE THE VERACITY OF THE SOFTWARE.
*************************************************************/

#include<stdio.h>
#include"z8encore.h"

void init_timer0(void);
void init_p2ad(void);

/*************************************************************
********************End of Interrupt.h file*******************
*************************************************************/
```

Appendix D—Alexan IR Emitting Diode

This appendix presents the data sheet for the Alexan IR Emitting Diode used in the Z8 Encore!®-based tachometer application.



Infrared Emitting Diode



Appendix E—Alexan WPT-440F IR Photo Transistor

This appendix presents the data sheet for the Alexan WPT-440F IR Photo Transistor used in the Z8 Encore!® MCU-based tachometer application.

Photo Transistor

Module No. WPT-440F

1. General Description

Dimensions

2. Features

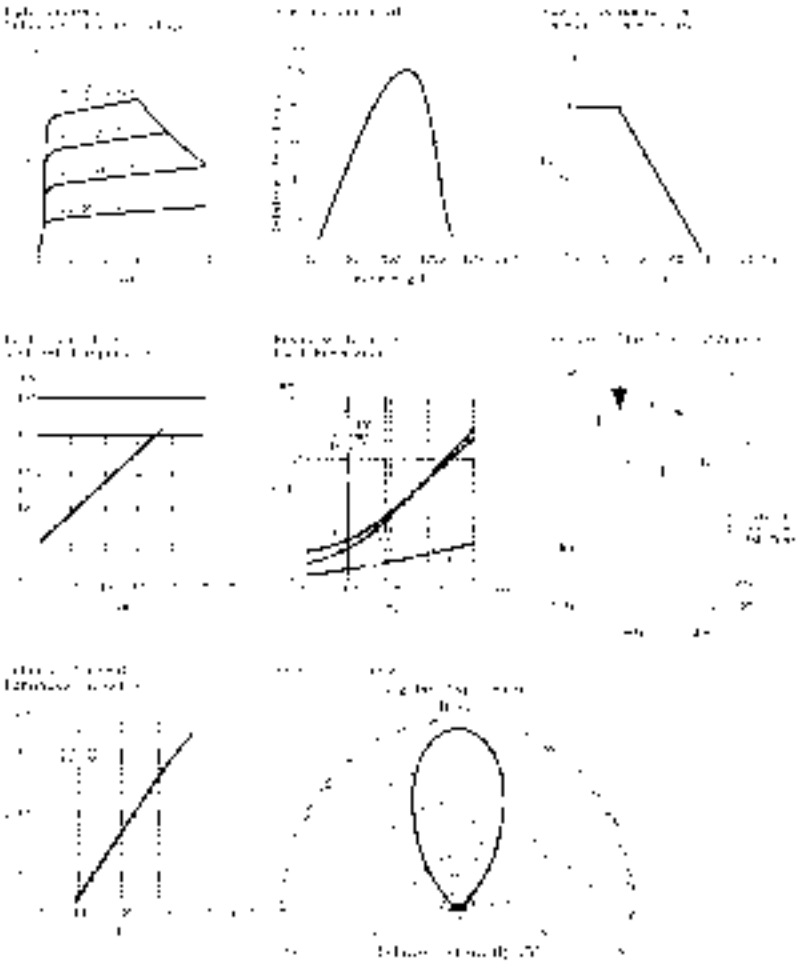
3. Applications

4. Absolute Maximum Ratings

Parameter	Symbol	Ratings	Unit
Collector-Emitter Voltage	[illegible]	[illegible]	[illegible]
Emitter-Collector Voltage	[illegible]	[illegible]	[illegible]
Collector Current	[illegible]	[illegible]	mA
Collector Power Dissipation	[illegible]	[illegible]	mW
Operating Temperature	[illegible]	[illegible]	[illegible]
Storage Temperature	[illegible]	[illegible]	[illegible]
Soldering Temperature [illegible]	[illegible]	[illegible]	[illegible]

5. Electro-optical Characteristics

Parameter	Symbol	Testing Conditions	Min	Typ	Max	Unit
Collector Light Current	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	mA
Collector Dark Current	[illegible]	[illegible]		[illegible]	[illegible]	nA
Collector Emitter Saturation Voltage	[illegible]	[illegible]		0.2	0.4	V
Peak Sensitivity Wavelength	[illegible]			[illegible]		nm
Spectral Sensitivity	[illegible]		[illegible]	[illegible]		nm
Angular Response	[illegible]			[illegible]		deg
Rising Response Time	[illegible]	[illegible]		[illegible]		µs
Falling Response Time	[illegible]	[illegible]		[illegible]		µs

Photo Transistor

Module No.: WPT-440F