Application Note

8-Direction Digital Compass Using Z8 Encore!® MCU

AN016502-0608

Abstract

This Application Note demonstrates a simple 8-direction digital compass application using Zilog's Z8 Encore!® MCU and an external sensor-compass hardware. Communication ports are provided for the digital compass to receive commands and send status on the I^2C bus as well as the UART, from/to an external host. According to the command received, the digital compass toggles an output that generates an interrupt for a specific direction.

This Application Note provides the schematics and software to implement the Digital Compass application. It also provides host-side Application Programming Interface (APIs) to communicate with the Z8 Encore! MCU-based digital compass. The host can be any processor with UART and/or I^2C peripherals. However, the APIs are written based on using Z8 Encore! MCU as an external host.

Note: *Following source code files associated with this application note are available for download at* www.zilog.com:

- `AN0165-SC01.zip` (Source Code files for the external host).
- `AN0165-SC02.zip` (Source Code files for the Digital Compass unit).

Z8 Encore! Flash Microcontrollers

Z8 Encore! MCU products are based on the Zilog's 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.

Featuring Zilog's eZ8 CPU, the Z8 Encore! microcontrollers 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 various applications including motor control, security systems, home appliances, personal electronic devices, and sensors.

Discussion

This section describes the Digital Compass basics and the digital compass sensor used in this application.

Theory of Operation

A compass is an instrument used to determine the directions. It works on the principle that a suspended magnet remains in the North-South direction under the influence of the Earth's magnetic field.

The *digital* compass, built using Z8 Encore! MCU, works on the same principle as a magnetic/mechanical compass; however, it features some additional functionality to communicate its status on a request from the host. A digital compass is useful in the applications where direction assumes significance like navigation and robotics.

The resolution of the digital compass is determined by the sensor it is based on.

There are two types of sensors:

- Digital Sensor
- Analog Sensor

A digital sensor's four digital outputs indicate the four cardinal directions. When two cardinal directions are simultaneously excited, it indicates a heading direction in-between the two cardinal directions. For example, if the directions North (N) and East (E) are output at the same time, the actual heading direction is North-East (NE). Thus, a digital compass based on a digital sensor is valid for resolving 8 directions, with a resolution of 45°.

An analog sensor is more precise than a digital sensor, with a resolution of 1°. Analog sensors normally use a bridge network with the bridge arms formed of resistors with magneto-resistive material. A change in resistance unbalances the bridge; the output is sensed with a differential amplifier and fed to an A/D converter for further processing.

This Application Note describes the implementation of a digital sensor to resolve the 8 main directions namely, the 4 cardinal directions - North (N), South (S), East (E), West (W), and the 4 intermediate directions North-East (NE), South-East (SE), South-West (SW), and North-West (NW).

Description of Components

The digital compass as described in this Application Note uses the digital sensor #1490 from Dinsmore Sensors (see References on page 8). The sensor contains four open-collector outputs, one for each direction, to facilitate an easy interface with a host of microcontrollers. The sense mechanism works between 6 V to 18 V; the voltage is unregulated but is polarity- and spike-protected. The outputs are damped to resemble a regular compass.

With damping, it takes approximately 3.5 seconds to change the output for a 90° change in the orientation. The outputs also contain built-in hysterisis to avoid flutters in the output. For a pin-out diagram and further details, refer to the #1490 Digital Compass Sensor.

Developing the Z8 Encore!® Digital Compass Application

The section provides the details of the hardware and software implementation to develop the Z8 Encore! -based Digital Compass application.

Hardware Architecture

Figure 1 on page 3 displays the hardware architecture of the Digital Compass application.

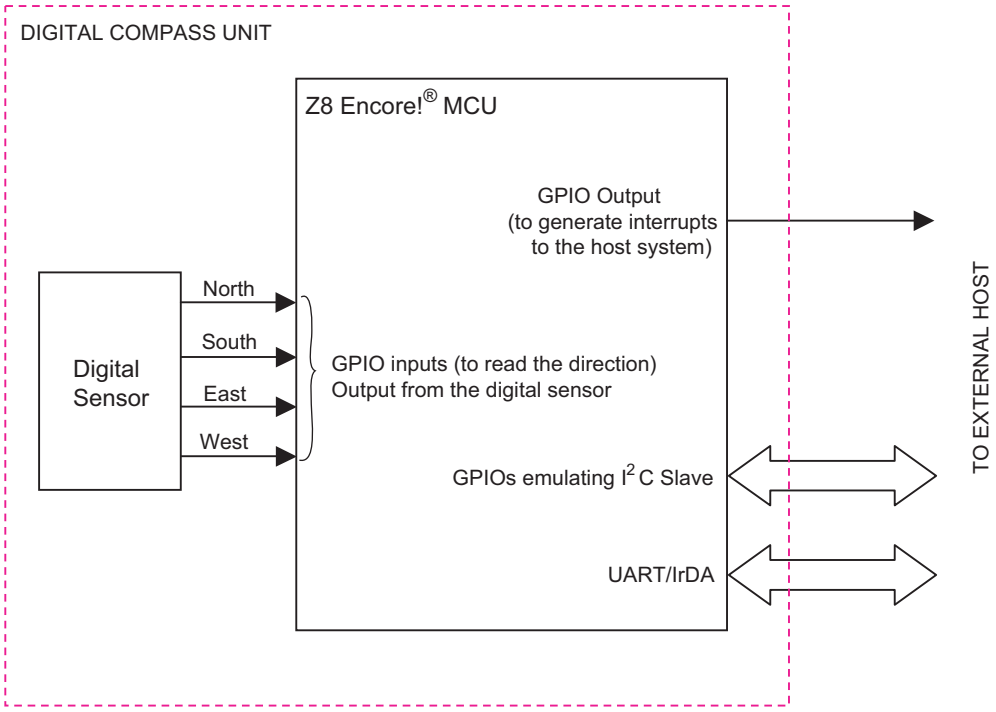

Figure 1. Hardware Architecture of the Digital Compass Unit

In Figure 1, the dotted lines enclose the block diagram for the Digital Compass unit.

The Digital Compass unit consists of two major components:

- Digital sensor
- Z8 Encore! microcontroller.

The digital sensor's four open-collector outputs are connected to four GPIO pins on the Z8 Encore! MCU. The Z8 Encore! MCU communicates with an external host through its on-chip I^2C and UART peripherals. In the I^2C mode, the external host acts as the I^2C master, while the Digital Compass unit acts as an I^2C slave. While using the UART interface (RS-232), the system is independent of I^2C and there is no master/slave relationship between the Digital Compass unit and the external host. You can select one of these modes/interfaces during run time.

A GPIO on the Z8 Encore! MCU is used as the output pin to indicate the orientation to a particular direction as programmed by the host. This GPIO pin is configured in the open-drain mode.

Software Implementation

The Digital Compass application can perform the following tasks:

- Read the digital sensor outputs.
- Determine the orientation (heading direction) based on two successive readings of the digital sensor data, ensuring that the heading direction is stable and not a crossover.
- Read commands received from the host, using either the UART or the I^2C interfaces.
- On request, provide status information to the host on either the UART or the I^2C interfaces.

- Based on the command received, toggle a GPIO pin on attaining a certain direction. Toggling the GPIO pin acts as a source of interrupt for the host.

Additional features supported by the Digital Compass application include:

- Asserting an interrupt on attaining a particular direction
- Disabling interrupt assertion
- Sending the status upon request

The additional features are implemented by defining following two 8-bit variables:

- Digital Compass control register
- Digital Compass status register

Table 1 provides the Digital Compass Control Register Details.

Table 1. Digital Compass Control Register Details

BITS	B7	B6	B5	B4	B3	B2	B1	B0
NAMES	X	X	COM 1	COM 0	X	IRDR2	IRDR1	IRDR0
DEFAULT	0	0	1	1	0	1	1	1

Table 2. Inference for the COM 1 and COM 0 Signals

COM 1	COM 0	Inference
0	0	Set Interrupt
0	1	Reset Interrupt
1	0	Send Status
1	1	Undefined

In Table 1, there are three commands defined for the Digital Compass. The bits COM 0 (B4) and COM 1 (B5) form the basis for the command. The bits B3, B6, and B7 are undefined and are set to zero. The bits B0, B1, and B2 are the direction bits; they provide the direction for which the interrupt, if enabled, is to be generated.

Table 3 lists the direction bits and their definitions. These definitions hold true for Status and Control registers. All other combinations are undefined and result in an error condition. For the `Send Status` command, the B0, B1, and B2 bits are *don't care*.

Table 3. Heading Direction Definitions for the Compass Control Register

HDDR2	HDDR1	HDDR0	Inference
IRDR2	IRDR1	IRDR0	
0	0	0	North
0	0	1	East
0	1	0	South
0	1	1	West
1	0	0	North-East
1	0	1	South-East
1	1	0	South-West
1	1	1	North-West

The Status register indicates the following items:

- The current heading direction (whether interrupt generation is enabled).
- The direction for which the interrupt was set by the last command.

The Status register is updated every 1 second, and is sent to the host on receiving the `Send Status` command.

Table 4 provides the bit definitions of the Digital Compass Status register.

Table 4. Digital Compass Status Register Definitions

BITS	B7	B6	B5	B4	B3	B2	B1	B0
FLAG	INTR	IRDR2	IRDR1	IRDR0	ERR	HDDR2	HDDR1	HDDR0
RESET	0	0	0	0	0	0	0	0

In Table 4, the bit B7 indicates that the interrupt generation mechanism was enabled or disabled by the last received host command. If enabled, the bits B4-B6 define the direction for which the interrupt is sent to the host. Bit B3 is for error indication and is activated when an invalid command is received from the host or when the sensor data reading is ambiguous. Bits B0-B2 indicate the current/last modified heading direction.

Note: *The heading direction (HDDR) definitions are identical to the interrupt direction (IRDR) definitions.*

I²C Communication

The Digital Compass application is capable of I²C communication, and implements an I²C slave functionality while the external host functions as the I²C master.

At the outset of the I²C communication, the external host generates a START condition (S) and follows it by the 7-bit address of the Digital Compass unit, and a *write* bit. On matching the address, the Digital Compass application sends the acknowledgement (ACK) back to the host. On receiving the ACK, the external host sends an 8-bit command to the Digital Compass application. The Digital Compass application now acknowledges the 8-bit command by asserting another ACK.

For the `set interrupt` or `reset interrupt` command, the external host sends a STOP (P) condition after receiving the ACK for the command byte. However, for the `send status` command, the external host initiates a second (master read) transaction by generating a repeat START condition (Sr) followed by the compass address and a *read* bit. The Digital Compass application, in response, sends an ACK followed by the 8-bit status register. The external host then sends a NACK, signifying the end of transfer, and follows it by a STOP condition.

Table 5 provides the `set interrupt/reset interrupt` command format.

Table 6 provides the `send status` command format.

Table 5. `set interrupt/reset interrupt` **Command Format**

S	ADDRESS	W	ACK	COMMAND	ACK	P
Host			Compass	Host	Compass	Host

Table 6. `send status` **Command Format**

S	ADDRESS	R	ACK	STATUS	NACK	P
Host			Compass		Host	

Note: *The transmitting devices are indicated below the command details.*

UART Communication

The Digital Compass application is capable of an RS-232 communication. For such communication, the Z8 Encore! MCU UART is used in the INTERRUPT mode. The UART receives the commands specified earlier and responds accordingly to the host. You can easily set the UART baud rate. The command and status format are same for the I^2C and the UART communications.

LED Display

The scrolling LED display mechanism is used for the Digital Compass display. The current heading direction is scrolled continuously. When any error occurs, the message, `!Err,` is blinked till the error-source is removed.

Interrupts and Interrupt Service Routines (ISRs)

The Digital Compass application uses interrupts for its operation as well as for communicating with the host. The required interrupts and the associated ISRs are listed below:

- Timer0 interrupt, every 1 ms, to maintain the LED display and to keep count of digital sensor data updates.
- Port C interrupts (pins PC0, PC1) to implement the I^2C communication (slave mode).
- UART 0 interrupt to implement the UART communication.

The Timer0 ISR is invoked every ms to fetch fresh data to maintain the LED display. For more details on LED display using Z8 Encore! MCU, refer to the *5x7 LED Matrix Display with Z8 Encore!® Application Note (AN0144)*. The timer interrupt is also used to flag the sensor update required after a user-defined time interval has elapsed (default is 1 second).

The Port C ISRs are required for the I^2C slave implementation. For I^2C slave implementation details, refer to the *Using a Z8 Encore!® MCU as an I^2C Slave Application Note (AN0139)*.

A UART0 interrupt indicates to the Digital Compass application that a fresh command was received from the external host. The Digital Compass application then interprets and executes the command. For more details on the UART implementation, refer to the *Direct Memory Access on the Z8 Encore!® UART Application Note (AN0142)*.

Digital Compass Application Main Routine

The pseudo code for the `main()` program for the Digital Compass application is provided below.

```
main()
{
    initialize_compass();// compass
    init plus all
    // other initializations
    EI();

    While (1)
    {
        if (sensor_timeout == TRUE)
        // default 1 second
        {
           direction = read_sensor();
           update status();
           display direction();
        }
        if (interrupt_generation ==
        enabled)
        {
           if (heading_direction ==
           interrupt direction)
           {
              toggle GPIO to interrupt
              the host;
           }
        }
        ... ...
        ... ...
        code for updating LED
        ... ...
```

```
        ... ...
    }
}
```

Testing the Digital Compass Application

This section provides the setup and procedure details to test the Digital Compass application using a Z8 Encore! MCU.

Setup

The test setup consists of two Z8 Encore! MCU Development Kits: one contains the Digital Compass unit, and the other functions as the external host.

The digital sensor is interfaced to the Digital Compass according to the schematic diagram in Appendix B—Schematics on page 11.

The Digital Compass unit uses a console port, UART0, and Pins C0 and C1 to interface with the external host in the UART and I^2C (Slave mode) type of communications. The external host's console port is used to connect to the HyperTerminal application, while the external host's modem port is used to interface with the Digital Compass application. The external host, which is the Z8 Encore!® MCU, implements the I^2C master functionality through the on-chip I^2C peripheral. (see Figure 2).

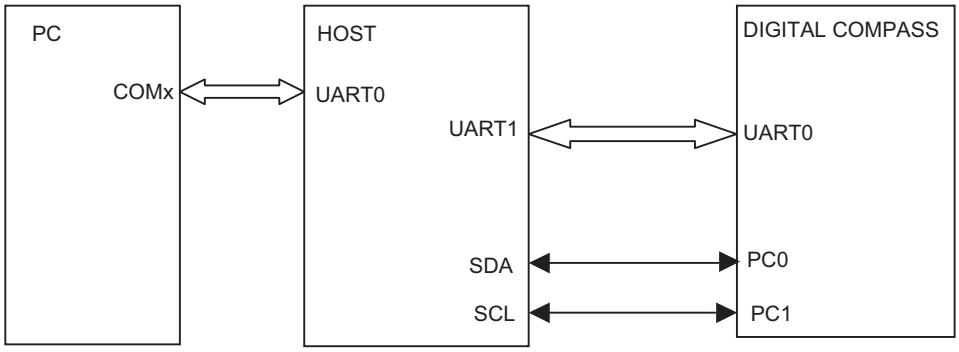

Figure 2. Setup Diagram for Z8 Encore! Digital Compass Testing

Equipments Used

The equipment used to test the Digital Compass application include:

- Two Z8 Encore! Development Kits (Z8ENCORE000ZC0)
- ZDS-II for Z8 Encore! v4.6.0
- Digital Compass sensor from Dinsmore, #1490
- PC with the HyperTerminal application (HyperTerminal settings: 9600 Baud, 1 stop bit, no parity, no flow control)
- Oscilloscope - Tektronix TDS3032B

Procedure

Follow the steps below to test the Digital Compass application:

1. Set up the Digital Compass-Z8 Encore! Development Board unit according to the schematic provided in Appendix B—Schematics on page 11.
2. Connect the modem port of the other Z8 Encore! Development Board (external host) to the Digital Compass-Z8 Encore! Development Board.

3. Connect the console port of the external host to the PC with the HyperTerminal application.
4. Launch the HyperTerminal application with settings 9600-8-N-1, and no flow control.
5. Using ZDS II, build and download the digital compass software (AN0165-SC02.zip) and the external host software (AN0165-SC01.zip) to the Z8 Encore! Development Platforms, respectively.
6. Run the code on the Digital Compass-Z8 Encore! Development Board and the external host system, and follow the instructions on the HyperTerminal.
7. Observe the heading direction on the LED display on the Digital Compass-Z8 Encore! Development Board.
8. Observe the interrupts generated on the external host with the CRO.
9. Rotate the sensor in the horizontal plane and observe the change in LED display.

Results

The 8-direction Digital Compass unit displays the current heading direction.

Summary

This Application Note describes the simple 8-direction Digital Compass unit which receives and reads the directions indicated by the digital compass sensor, and displays them on the LED matrix display. The digital compass features include UART and I^2C interfaces to an external host for direction indication and control.

The digital compass, with its 45° resolution, is ideal for simple toys and gadgets. The I^2C and UART peripherals can be used for remote sensing or controlling the direction.

References

The documents associated with Digital Compass Sensor, Z8 Encore! MCU, and the Application Notes referenced in the document are provided below:

- Digital Compass Sensor — 1490 Digital Compass Sensor, 1490spec.htm, available at www.dinsmoresensors.com.
- Z8 Encore! XP® F64XX Series Product Specification (PS0199)
- I2C Slave Application — Using a Z8 Encore! MCU as an I2C Slave Application Note (AN0139)
- UART Application — Direct Memory Access on the Z8 Encore! UART Application Note (AN0142)
- LED Matrix Display Application — 5x7 LED Matrix Display with Z8 Encore! Application Note (AN0144)

Appendix A—Flowcharts

Figure 3 displays the flowchart for the main routine for the Digital Compass application.

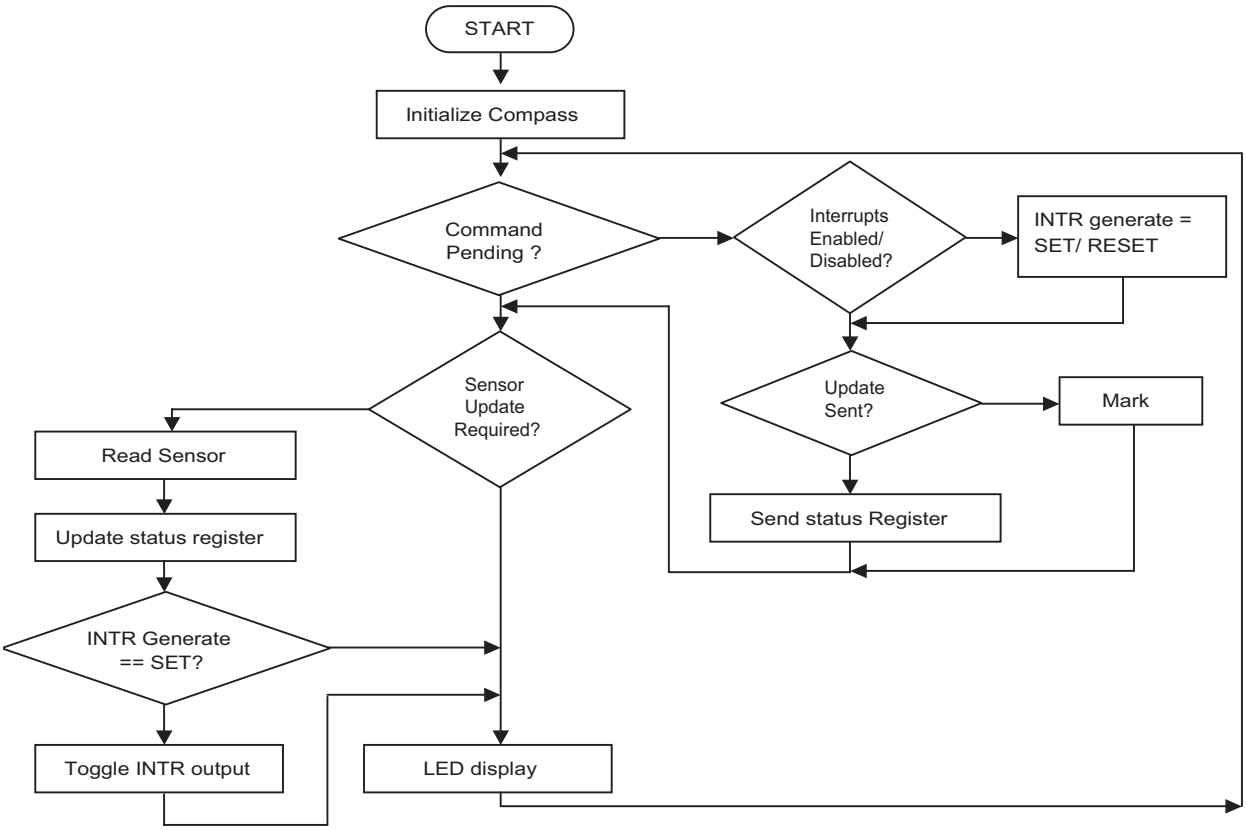

Figure 3. Main Routine

Figure 4 displays the flowchart for the UART ISR.

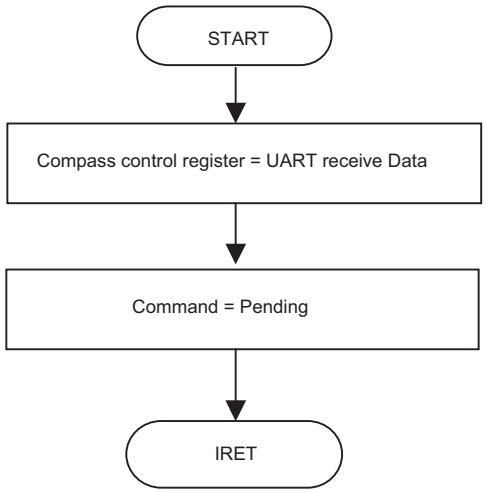

Figure 4. Flowchart for the UART ISR

Note: *For the I^2C implementation flowcharts, refer to the Using a Z8 Encore!® MCU as an I^2C Slave Application Note (AN0139).*

Appendix B—Schematics

Figure 5 displays a schematic diagram for the Digital Compass application.

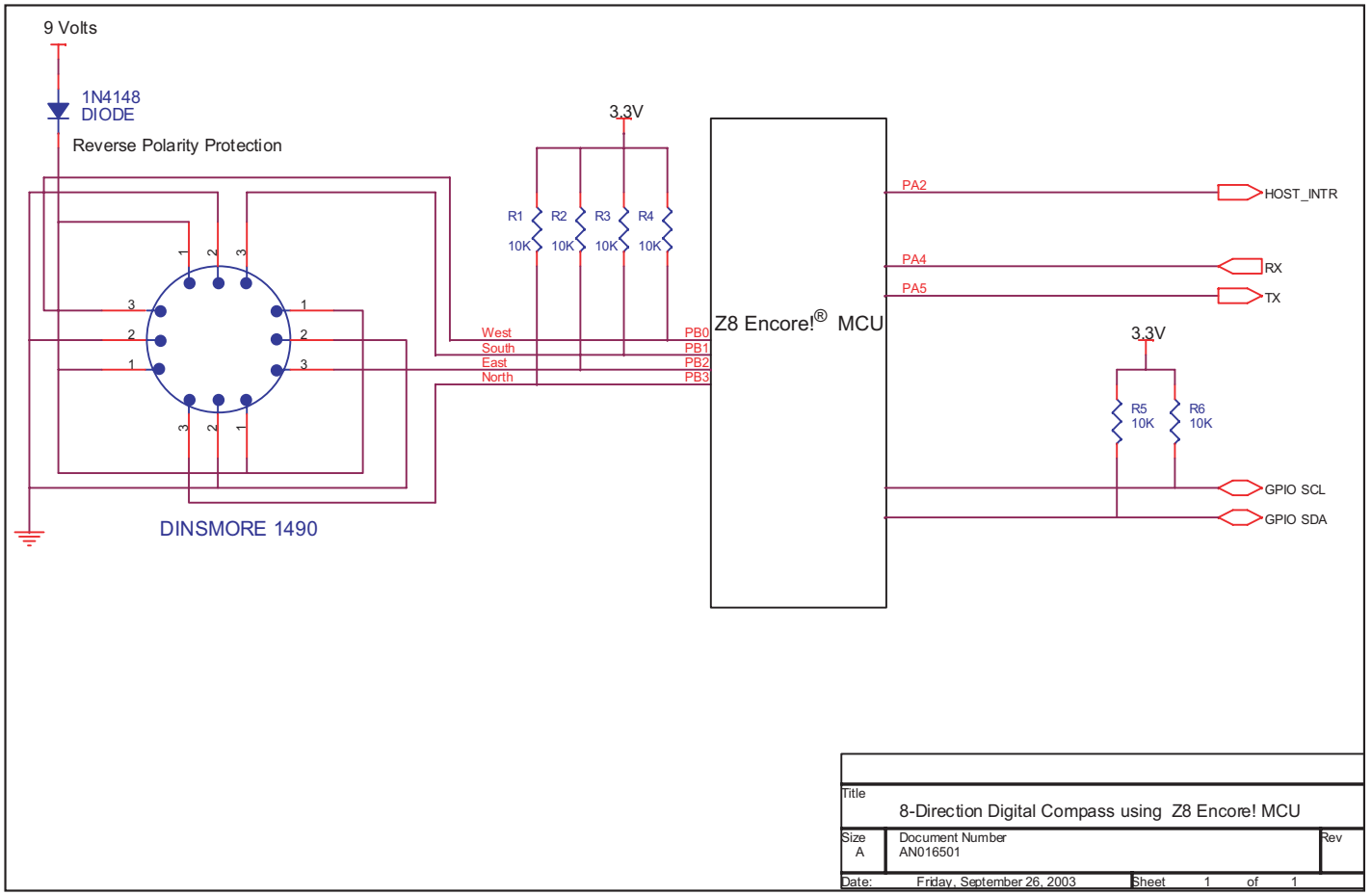

Figure 5. Schematic for the Z8 Encore! 8-direction Digital Compass

Appendix C—API Descriptions

This appendix provides a description of the APIs used to interface the Z8 Encore!® MCU (as an external host) with the Digital Compass application described in this Application Note.

Table 7 lists the Digital Compass application APIs for quick reference.

Table 7. Digital Compass Application APIs

API Name	Description
Read_Z8Encore_slave()	Requests the Digital Compass to send status.
Write_Z8Encore_slave()	Sends the control word to the Digital Compass.

Description for the Digital Compass application APIs are provided below.

Read_Z8Encore_slave()

Syntax

```
char Read_Z8Encore_slave(COMPASS_READ_ADDR)
```

Description

The `Read_Z8Encore_slave()` API reads the Digital Compass status register.

Argument(s)

char	`COMPASS_READ_ADDR`	The address of the status register.

Return(s)

char	`compass_status_register`	Data contained in the status register.

Usage

```
status = Read_Z8Encore_slave(COMPASS_READ_ADDR);
```

Write_Z8Encore_slave()

Syntax

```
void Write_Z8Encore_slave(COMPASS_WRITE_ADDR, command)
```

Description

The `Write_Z8Encore_slave()` API sends the command on the I^2C port to the Digital Compass application.

Argument(s)

char	`COMPASS_WRITE_ADDR`	The address where the command is to be written.
char	`command`	The command to be written.

Return(s)

void

Usage

```
Write_Z8Encore_slave(COMPASS_WRITE_ADDR, command);
```

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