Application Note

Implementing MIDI Using the Z8F1680 MCU

AN035001-1213

Abstract

This application demonstrates how to transmit and receive MIDI data from the Z8F1680 MCU's UART module into a host PC or MIDI program. The MIDI data specified in this document refers to a *MIDI message*. The supported MIDI messages for transmission include *Note On* and *Note Off* only. For MIDI reception, the supported messages are *Note On*, *Note Off*, *Polyphonic Aftertouch* and *Pitch Bend*. MIDI messages are displayed on an LCD panel, while the Linux Multimedia Studio program outputs the equivalent MIDI sound.

Note: The source code file associated with this application note, AN0350-SC01.zip, is available free for download from the Zilog website. This source code has been tested with version 5.0.0 of ZDSII for Z8 Encore! XP MCUs. Subsequent releases of ZDSII may require you to modify the code supplied with this application note.

Features

This application offers the following features:

- Two modes of operation: MIDI IN and MIDI OUT
- Transmission of MIDI messages for playback in Linux MultiMedia Studio
- Reading MIDI messages from Linux MultiMedia Studio
- Displays MIDI receive and transmit messages on an LCD panel

Discussion

The Musical Instrument Digital Interface (MIDI) is an industry specification for electronic musical instruments such as audio sequencers, digital synthesizers, etc., to communicate with one another. MIDI communication occurs via an asynchronous serial interface with a data stream of 31250 baud.

The data passing across MIDI devices are called MIDI messages, which consist of one status byte followed by one or two data bytes, as shown in Figure 1. MIDI messages are classified into two categories: *channel messages* and *system messages*. Channel messages address each channel independently of one another, while system messages affect the entire MIDI system. This application note only discusses channel messages.

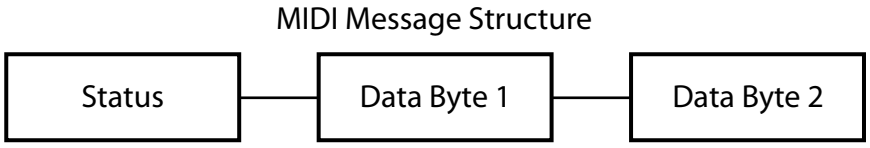

Figure 1. The MIDI Message Structure

Two types of channel messages that control the overall functionality of a MIDI device: *channel voice* and *channel mode*. Channel voice messages are defined by the upper nibble of a status byte; the lower nibble of this status byte (*n*) defines which MIDI channel the message is sent to (e.g., valid values range from 1–16). Four MIDI channel voice messages are used in this application: Note On, Note Off, Polyphonic Aftertouch, and Pitch Bend.

In this application, the value ranges for the four MIDI messages used are:

Note Off. The value of *k* ranges from `0x0C` to `0x77`, and the value of *v* is fixed at `0x00`.

Note On. The value of *k* ranges from `0x0C` to `0x77`, and value of *v* is fixed at `0x7F`.

Polyphonic Aftertouch. The value of *k* ranges from `0x0C` to `0x77`, and value of *p* ranges from `0x00` to `0x7F`.

Pitch Bend. The values *f* and *c* range from `0x00` to `0x7F`.

The value of *n* is hard-coded to Channel 1 (*n* = 0000) for MIDI OUT operation. For MIDI IN operation, this value receives MIDI messages from any of the 16 channels.

Table 1 lists the supported MIDI messages for this application.

Table 1. MIDI Messages

Status Byte	Data Byte 1	Data Byte 2	Message	Legend
1000nnnn	0kkkkkkk	0vvvvvvv	Note Off	n=channel k=key # 0x00 to 0x7F (3C=middle C) v=velocity (0x00 to 0x7F)
1001nnnn	0kkkkkkk	0vvvvvvv	Note On	n=channel k=key # 0x00 to 0x7F (3C=middle C) v=velocity (0x00 to 0x7F)
1010nnnn	0kkkkkkk	0ppppppp	Polyphonic Aftertouch	n=channel k=key # 0x00 to 0x7F (3C=middle C) p=pressure (0x00 to 0x7F)
1110nnnn	0fffffff	0ccccccc	Pitch Bend	n=channel f=fine c=coarse (f+c = 14-bit resolution)

Hardware Implementation

This application uses the development board from the Z8 Encore! XP F1680 Series (28-Pin) Development Kit (Z8F16800128ZCOG) to communicate with the MIDI device through a UART. A rotary encoder is used to select the octave group. Pushbuttons are

used to select the musical keys. The combination of an octave group and a key produces a MIDI message. A slide switch is used to select the mode of operation, and MIDI information is displayed on an LCD. A MIDI-USB cable is used to interface the MIDI device and the PC. The MIDI IN circuit receives MIDI messages from the PC, and the MIDI OUT circuit transmits MIDI messages to the PC.

The block diagram in Figure 2 provides an overview of the MIDI application hardware. For details about the hardware connections, see Appendix A. Schematics on page 22.

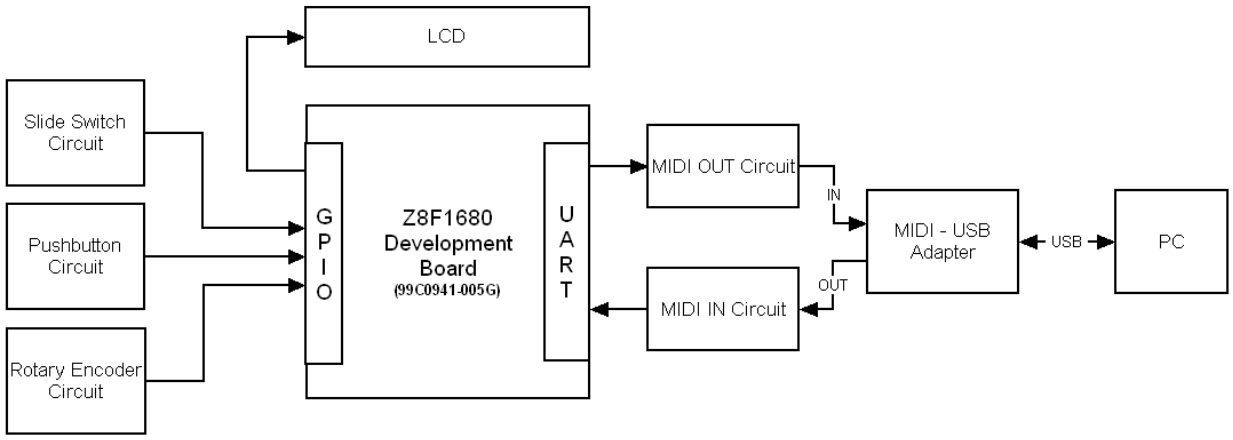

Figure 2. MIDI Hardware Block Diagram

Slide Switch

The slide switch controls each of two modes of operation, the MIDI OUT and MIDI IN modes. When this switch is closed, the application operates as a MIDI output (i.e., MIDI OUT); otherwise, operation is MIDI IN. Figure 3 presents a schematic for this slide switch. The 100Ω resistor in this diagram is used for series pin protection to prevent excess current that could damage Port B5 (PB5) of the MCU.

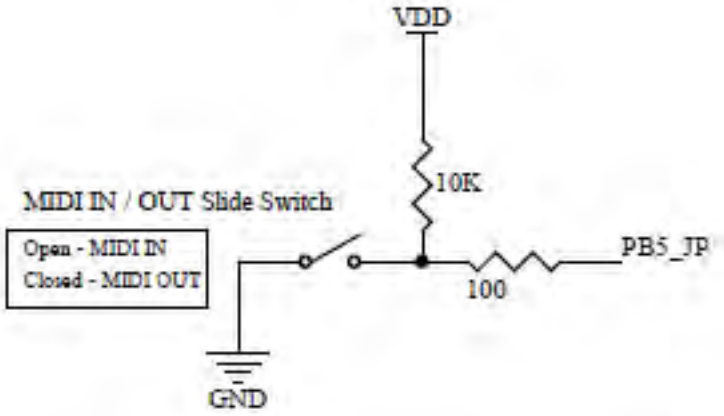

Figure 3. Slide Switch Circuit

LCD

Figure 4 represents a 4x20 LCD display in a 4-bit data interface. MIDI messages and events are shown in this display.

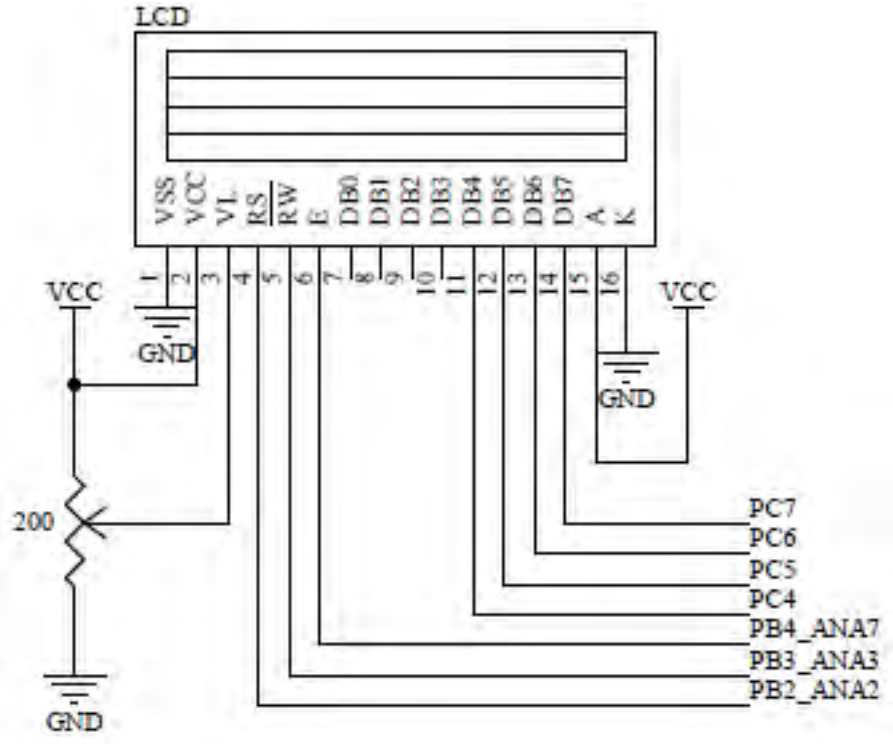

Figure 4. 4x20 LCD

MIDI OUT Implementation

A series of seven pushbuttons represent piano key inputs; each of these pushbuttons represents a musical note. When a pushbutton is pressed, it produces a Note On message. When the pushbutton is released, a Note Off message is produced. Figure 5 shows the circuit diagram for these pushbuttons.

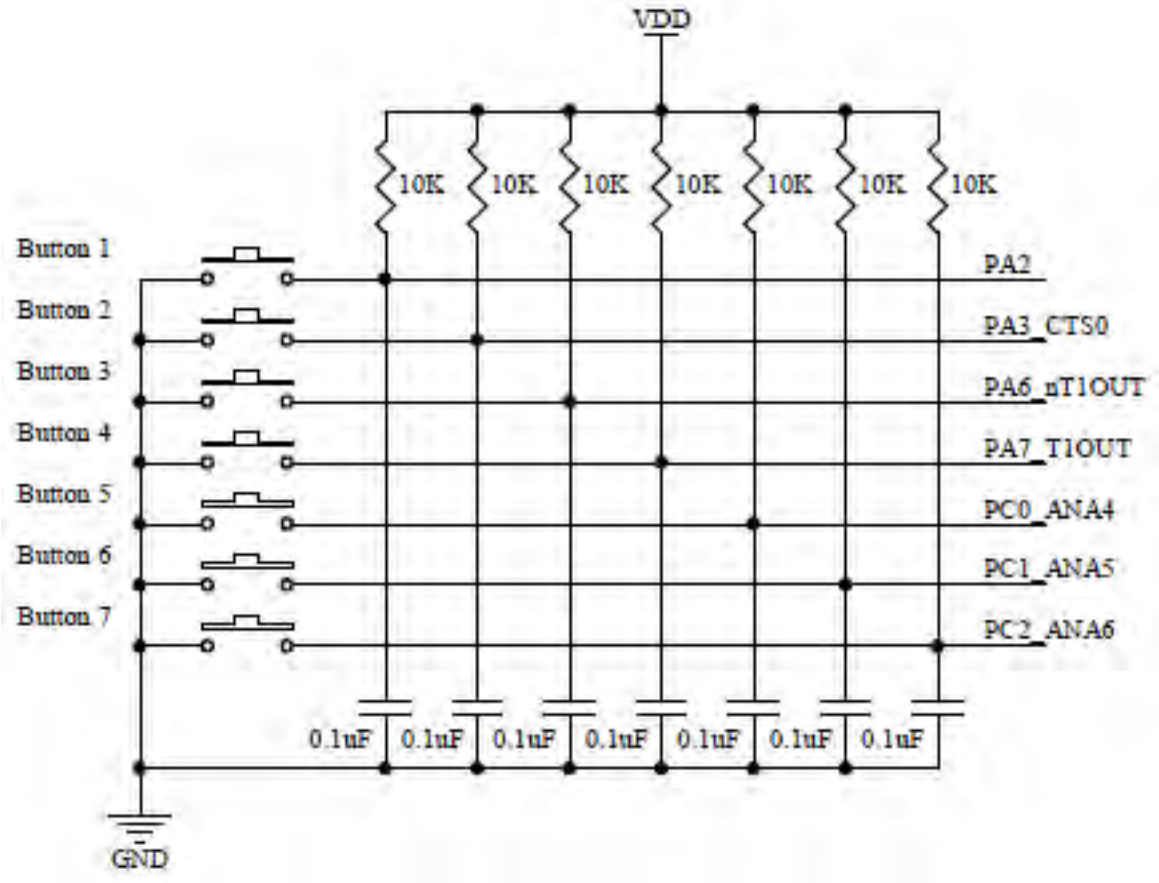

Figure 5. Input Pushbutton Circuit

The rotary encoder can alternate between nine octave levels (i.e., from 0 to 8) to produce additional ranges of notes. Figure 6 shows the circuit diagram for this rotary encoder.

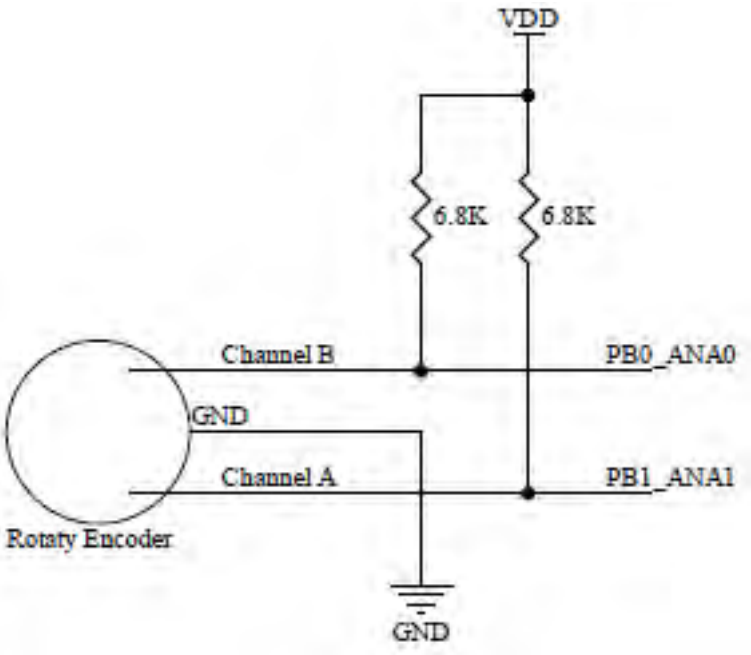

Figure 6. Rotary Encoder Circuit

The MIDI OUT circuit, shown in Figure 7, passes a transmitted MIDI message from the UART into the MIDI OUT port.

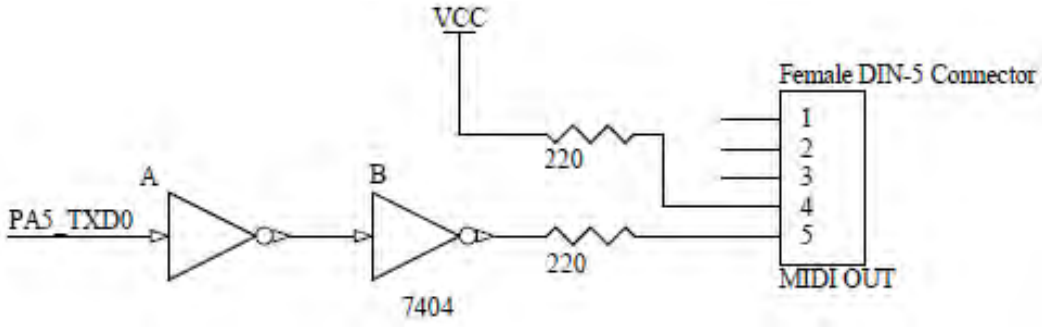

Figure 7. MIDI OUT Circuit

MIDI IN Implementation

The circuit used for MIDI reception requires an optocoupler to ensure that the MIDI interface is isolated from the remainder of the interface. Figure 8 diagrams this MIDI IN circuit.

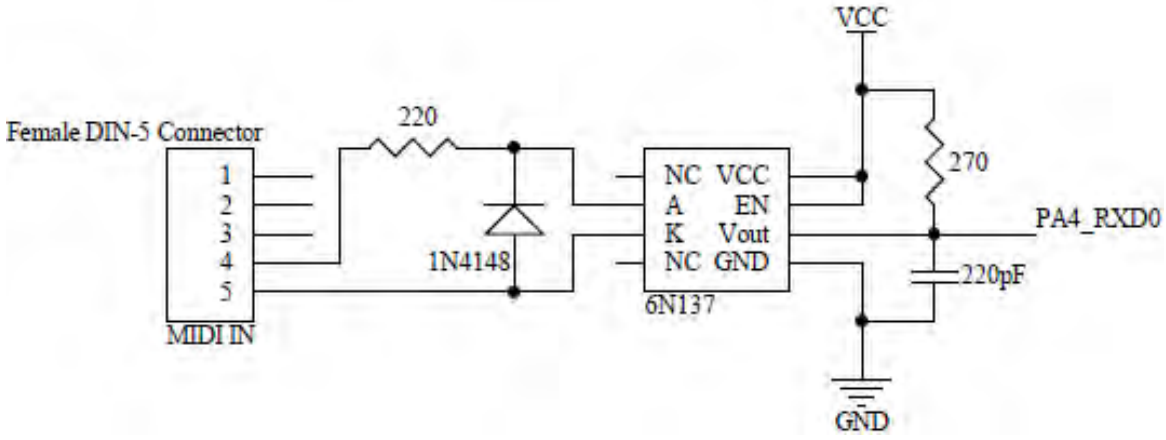

Figure 8. MIDI IN Circuit

Software Implementation

The firmware developed for this application, contained in the AN0350-SC01.zip file, is designed to be used with the ZDSII IDE for Z8 Encore!, version 5.0.0. Table 2 describes these firmware files.

Table 2. List of Source Files

Filename	Description
gpio.c	GPIO port initializations for the rotary encoder, slide switch, pushbuttons, and LCD assigned at selected pin numbers of Port A[7:6],[3,2] Port B[5:0], and Port C[7:4],[2:0].
lcd_api.c	Contains the API for the LCD.
lcd_hextoascii.c	Converts Hex to ASCII and prints to LCD.
midi_main.c	The main calling program that calls the various functions and routines
midi_rx.c	Contains implementation for the MIDI IN message from Linux MultiMedia Studio.
midi_tx.c	Contains the MIDI OUT routine. MIDI_Out will transmit three-byte data to a MIDI device or PC using the Linux Multimedia Studio to produce a MIDI message or an equivalent sound depending on the device or software. The pushbuttons will produce a note while the rotary encoder will change the octave level.
midi_uart.c	This contains the standard UART routines. The baud rate used is 31250 to comply with MIDI protocol.
timer.c	Contains Timer0 implementation and it is used for the timing requirements of the LCD and main application.

UART Setup for MIDI Compliance

This section discusses how the UART peripheral's baud rate generator is configured to comply with the MIDI standard. The following equation calculates the MIDI data rate.

$$\text{UART Data Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{16 \times \text{Baud Rate Divisor Value}}$$

Because the required data rate for MIDI is 31250 baud, and because the frequency of the Z8F1680 MCU's system clock is 20 MHz, plugging these two values into the above equation, as seen in the following equation, yields a baud rate divisor value of 40.

$$31250 = \frac{20{,}000{,}000}{16 \times \text{Baud Rate Divisor Value}}$$

MIDI Operation Mode

The application uses the slide switch to select the MIDI mode of operation. When the user closes the slide switch, the program will execute the MIDI OUT routine. Otherwise, the program will execute the MIDI IN routine. For an illustration of this process, see Appendix B. Flowcharts on page 23.

MIDI OUT Routines

During MIDI OUT operation, the application continuously polls for a rotary encoder value to set an octave group number. When there is a pushbutton event (i.e., whether pressed or released), the application will search for its key value (shown in Table 3). The resulting data will be displayed on the LCD panel and, at the same time, will be transmitted to the PC running Linux MultiMedia Studio.

Table 3 lists the musical note value which is the result of the combination of the rotary encoder and the pushbutton.

Table 3. Octaves and Notes Used in MIDI OUT

	Musical Note						
Rotary Encoder	**Button 1**	**Button 2**	**Button 3**	**Button 4**	**Button 5**	**Button 6**	**Button 7**
Octave	C	D	E	F	G	A	B
00	0C	0E	10	11	13	15	17
01	18	1A	1C	1D	1F	21	23
02	24	26	28	29	2B	2D	2F
03	30	32	34	35	37	39	3B
04	3C	3E	40	41	43	45	47
05	48	4A	4C	4D	4F	51	53
06	54	56	58	59	5B	5D	5F
07	60	62	64	65	67	69	6B
08	6C	6E	70	71	73	75	77

Example. When Button 1 is pressed and the Rotary Encoder is set to Octave number 04, the following MIDI message will be displayed on the LCD.

Transmitted MIDI Message:	90 3C 7F

The values shown in the above MIDI message can be interpreted as:

90 = Status: Note On at Channel 1, as shown in Table 1 on page 2.

3C = Note: C at Octave 04, as shown in Table 3.

7F = Velocity is at the maximum value.

When Button 1 is released, an updated MIDI message will be displayed on the LCD, as follows:

Transmitted MIDI Message:	80 3C 00

The values shown in the above MIDI message can be interpreted as:

80 = Status: Note Off, as shown in Table 1 on page 2.

3C = Note: C at Octave 04, as shown in Table 3.

00 = Velocity is at the minimum value.

These MIDI OUT routines can be found in the `midi_tx.c` file, which is contained in the AN0350-SC01.zip file. Table 4 lists the routines that are included in this file.

Table 4. MIDI OUT Routines

Function Name	Description
void MIDITX_SendMessage(CHAR cCommand, CHAR cByte1, CHAR cByte2)	Sends a three-byte MIDI message.
void interrupt isr_KeyC(void) _At P2AD	Interrupt service routines for the seven key pushbuttons.
void interrupt isr_KeyD(void) _At P3AD	
void interrupt isr_KeyE(void) _At P6AD	
void interrupt isr_KeyF(void) _At P7AD	
void interrupt isr_KeyG(void) _At PC0_IVECT	
void interrupt isr_KeyA(void) _At PC1_IVECT	
void interrupt isr_KeyB(void) _At PC2_IVECT	
UCHAR MIDITX_GetREncData(void)	Acquires values from the rotary encoder.
void MIDITX_DisplayNoteStat(UCHAR MIDI_ucVelocity, UCHAR MIDI_ucNote, UCHAR MIDI_ucOctave)	Displays the sent MIDI message to the LCD panel.
void MIDI_Out(void)	Waits for pushbutton and rotary encoder events and sends corresponding MIDI messages via the UART.

MIDI IN Routines

The MIDI IN operation interprets MIDI messages received from the UART. The supported MIDI messages for this application are Note On, Note Off, Polyphonic Aftertouch, and Pitch Bend.

Table 5 lists the musical notes available to read in the MIDI receiver.

Table 5. Octaves and Notes for MIDI IN

	Musical Note											
Octave	**C**	**C#**	**D**	**D#**	**E**	**F**	**F#**	**G**	**G#**	**A**	**A#**	**B**
00	0C	0D	0E	0F	10	11	12	13	14	15	16	17
01	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23
02	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
03	30	31	32	33	34	35	36	37	38	39	3A	3B
04	3C	3D	3E	3F	40	41	42	43	44	45	46	47
05	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53
06	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
07	60	61	62	63	64	65	66	67	68	69	6A	6B
08	6C	6D	6E	6F	70	71	72	73	74	75	76	77

Example. The hexadecimal value `0x18` represents Octave 01 and Note C.

The reception of MIDI messages occurs when the MCU interprets the MIDI message from the Linux Multimedia Studio.

When a key is pressed in the Linux Multimedia Studio, it sends a MIDI message, as shown below:

Transmitted MIDI Message from PC:	90 18 7F

The values shown in the above MIDI message can be interpreted as:

90 = Status: Note On at Channel 1, as shown in Table 1 on page 2.

18 = Note: C at Octave 01, as shown in Table 5.

7F = Velocity is at the maximum value.

A Polyphonic Aftertouch event occurs when a key in the Linux Multimedia Studio is clicked and dragged on a certain note, as shown in the following example.

Transmitted MIDI Message from PC:	A0 18 64

The values shown in the above MIDI message can be interpreted as:

A0 = Status: Polyphonic Aftertouch at Channel 1, as shown in Table 1 on page 2.

18 = Note: C at Octave 01, as shown in Table 5.

64 = Pressure value (range covered as shown in Table 1).

A Pitch Bend event occurs when the Pitch button is adjusted, as shown in the following example.

Transmitted MIDI Message from PC:	E0 32 00

The values shown in the above MIDI message can be interpreted as:

E0 = Status: Pitch Bend at Channel 1, as shown in Table 1 on page 2.

32 = Fine (range covered as shown in Table 1).

00 = Coarse (range covered as shown in Table 1).

These MIDI IN routines are contained in the `midi_rx.c` file. Table 6 describes these routines.

Table 6. MIDI IN Routines

Function Name	Description
void MIDIRX_PrintKey(void)	Identifies the note and octave to be displayed to LCD.
void MIDI_In(void)	Receives MIDI data from UART, translates what type of message and displays MIDI message to LCD.

Equipment Used

The tools used to develop this application are:

- Z8 Encore! XP F1680 Series (28-Pin) Development Kit (Z8F16800128ZCOG), which contains the following items:
  - Z8 Encore! XP F1680 Series Development Board
  - DC 5V power supply
  - USB SmartCable
- MIDI USB cable (available on Amazon.com)
- Windows-based computer
- A circuit design developed for this application (see Appendix A. Schematics on page 22)

Setting up the Application

This section presents procedures for setting up the hardware, configuring the software, and for demonstrating the application.

Hardware Setup

Observe the following procedure to prepare the hardware.

1. Remove the following components from the Z8F1680 Development Board:
   - R7
   - R8
   - D2
   - D3
2. Insert a jumper on JP3 (DIS RS-232).
3. Plug the MIDI USB adapter into the PC's USB port. If you are a first-time user, the installation of the MIDI driver will occur.
4. Connect the MIDI OUT cable of the adapter into the MIDI IN socket of the MIDI IN circuit.
5. Connect the MIDI IN cable of the adapter into the MIDI OUT socket of the MIDI OUT circuit.
6. Connect the other circuits as shown in Figure 9.

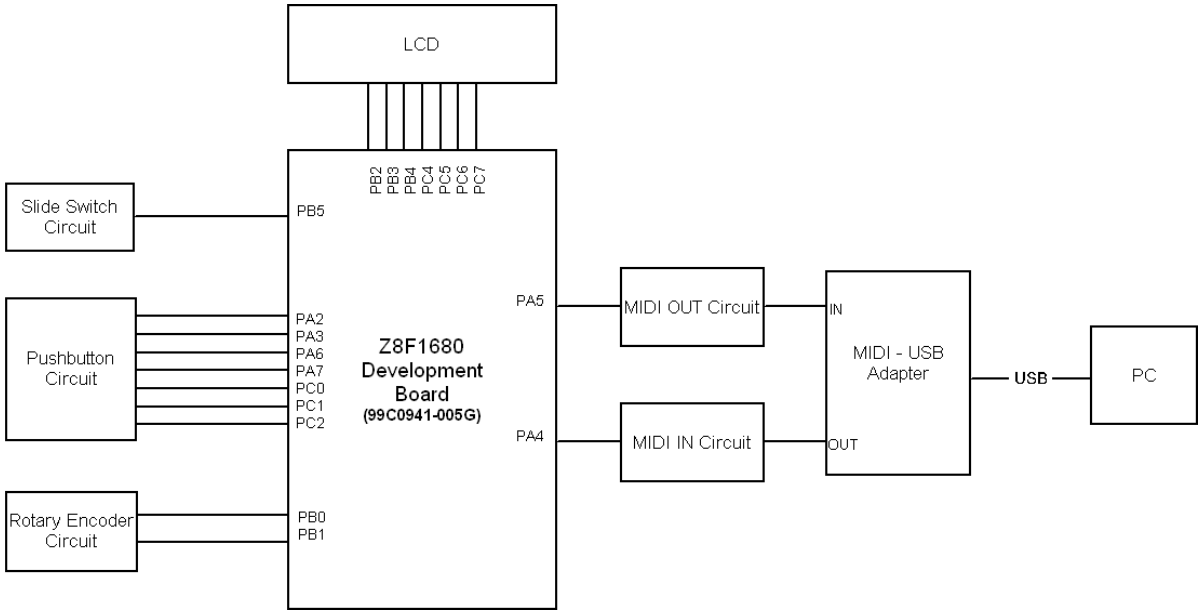

Figure 9. MIDI OUT/IN Setup

Software Setup

This section provides instructions for loading the source code onto the Z8 Encore! XP F1680 Series Development Board using ZDSII, and to configure Linux Multimedia Studio.

Load the Source Code Using Zilog Development Studio II

Observe the following brief procedure to load and build the MIDI application code to the Development Board.

1. Download the AN0350-SC01.zip source code file, which can be obtained free from the Zilog website. Unzip this file to an appropriate location on your PC's hard drive.
2. Launch ZDSII, and navigate via the **File** menu to the location of the source code files you just downloaded, and open the `AN0350-SC01.zdsproj` project file.
3. Click the **Build All** button to compile the code. After the build is complete, click the **Go** button to run the code.

Linux MultiMedia Studio Setup

Observe the following brief procedure to set up and configure Linux Multimedia Studio.

1. Download Linux MultiMedia Studio to your PC from the SourceForge open source website at http://lmms.sourceforge.net/download.php.
2. Install Linux Multimedia Studio on your PC. After installation is completed, launch the program.
3. Figure 10 shows the Linux Multimedia Studio main screen. Click the **Default preset** button indicated in this figure.

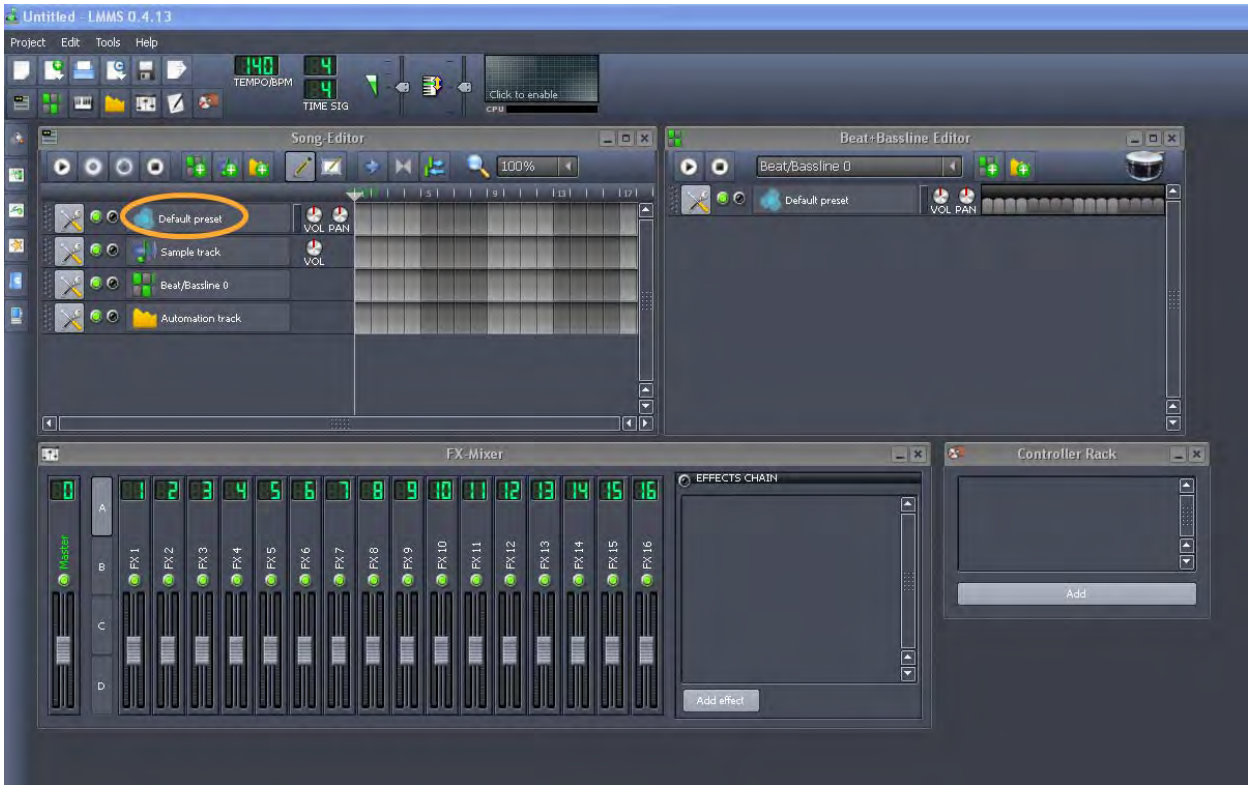

Figure 10. Linux Multimedia Studio Environment

4. The General Settings window appears, as shown in Figure 11. Click the **MIDI** tab indicated in this figure to select the USB port assignments.

Figure 11. The General Settings Window

5. To set the USB port assignment for MIDI IN, click the upper piano icon for MIDI input, which is indicated in Figure 12.

Figure 12. The MIDI Tab

6. A drop-down menu will appear. From this menu, select **USB Audio Device**. To set the USB port assignment for MIDI OUT, click the lower piano icon for MIDI output, also indicated in Figure 12. In the drop-down menu that appears, select **USB Audio Device[2]**.

Note: The names of USB devices may differ from PC to PC.

7. After setting the USB port assignments, go to the channel area indicated in Figure 13 and set it to MIDI Input Channel 1. Click Enable MIDI Input then scroll up/down over channel field to change channels.

Figure 13. MIDI Channel Setup for Input

8. For MIDI output, go at the channel area indicated in Figure 14 and set to Channel 1. Click Enable MIDI Output then scroll up/down over channel field to change channels.

Figure 14. MIDI Channel Setup for Output

Octave Groups in Linux Multimedia Studio

Linux Multimedia Studio uses a different naming convention for grouping musical notes than the octave groups that display on the LCD, as follows:

- As displayed on the LCD, octave groups range from Group 00 to Group 08
- Linux Multimedia Studio names octave groups –01 to 07 as groups C0 to C8 (`0x00` to `0x6B`)

Table 7 compares these musical note groupings between the LCD and Linux Multimedia Studio.

Table 7. Octave Group Comparison Between LCD and Linux Multimedia Studio

LCD Octave Group	LMS Octave Group
00	C1
01	C2
02	C3
03	C4
04	C5
05	C6
06	C7
07	C8
08	N/A

Note: Because the LCD will only display octave groups 00 to 08, octave groups –01 and C0 are not shown in Table 7.

Demonstrating the Application

There are two modes of operation for demonstrating this application: MIDI OUT and MIDI IN, as this section describes.

MIDI OUT Mode

Observe the following brief procedure to set up and configure MIDI OUT Mode.

1. Set up the hardware according to the MIDI application setup in Figure 9 on page 12.
2. Set the slide switch to the CLOSE position and wait for the LCD to display `MIDI OUT`.
3. Launch the Linux Multimedia Studio program and set it up according to the steps provided in the Linux Multimedia Studio Setup.
4. Press any of the seven pushbuttons. The resulting MIDI message data will be displayed on the LCD.

   Example 1. When Button 1 is pressed, the LCD displays the following data:

   ```
   Data: 90 3C 7F
   Status: Note On
   Note: C    Octave: 04
   Velocity: 7F
   ```

```
When Button 1 is released, the LCD displays the following
Data: 80 3C 00
Status: Note Off
Note: C    Octave: 04
Velocity: 00
```

5. To change the octave level, turn the rotary encoder. A clockwise direction increases the octave level. Next, repeat step 4.
6. Observe the MIDI message changes, as referenced in Tables 1 and 3.

MIDI IN Mode

Observe the following brief procedure to set up, configure, and demonstrate MIDI IN Mode for each of the four message types.

Note On and Note Off Demonstration

1. Set up the hardware according to the MIDI application setup in Figure 9 on page 12.
2. Set the slide switch to OPEN position and wait for the LCD to display `MIDI IN`.
3. Launch the Linux Multimedia Studio program and set up according to the steps provided in Linux Multimedia Studio Setup.
4. In Linux Multimedia Studio, click any of the piano keys in the Default Preset window (for reference, see Figure 11 on page 15).
5. Observe the MIDI Message changes on the LCD as referenced in Tables 1 and 5.
6. If changing of MIDI channel is desired, go to the channel area indicated in Figure 14 and change it to any of the channels from 1 to 16.
7. Repeat steps 4 through 6.

Polyphonic Aftertouch Demonstration

1. Set up the hardware according to the MIDI application setup in Figure 9 on page 12.
2. Set the slide switch to OPEN position and wait for the LCD to display `MIDI IN`.
3. Launch the Linux Multimedia Studio program and set it up according to the steps provided in the Linux Multimedia Studio Setup.
4. In Linux Multimedia Studio, click and drag any of the piano keys in the Default Preset window.
5. Observe the Aftertouch message changes on the LCD, as referenced in Table 1 on page 2.
6. Repeat steps 4 through 5.

Pitch Bend Demonstration

1. Set up the hardware according to the MIDI application setup in Figure 9 on page 12.
2. Set the slide switch to OPEN position and wait for the LCD to display `MIDI IN`.
3. Launch the Linux Multimedia Studio program and set it up according to the steps provided in the Linux Multimedia Studio Setup.
4. In the Default Preset window, click the **Pitch** knob to adjust the pitch values.
5. Observe the Pitch Bend message on the LCD, as referenced in Table 1 on page 2.
6. Repeat steps 4 through 5.

Results

This application demonstrates that the Z8F1680 MCU's UART module can be used as a MIDI device. The MCU is able to accurately transmit and receive MIDI messages.

Summary

This application implements the MIDI protocol for Zilog's Z8F1680 MCU. The transmission and reception of MIDI messages is made possible by properly configuring the UART baud rate to 31250. MIDI messages are displayed on an LCD panel for both transmit and receive operations.

References

Documents that support this Z8F1680 MCU-based application are listed below, in addition to external references. Zilog documentation can be downloaded for free from the Zilog website by clicking the link associated with its document number where indicated.

- eZ8 CPU Core User Manual (UM0128)
- Z8 Encore! XP F1680 28-Pin Series Development Kit User Manual (UM0203)
- Z8 Encore! XP F1680 Series Product Specification (PS0250)
- An Interrupt-Driven UART for Z8 Encore! XP and Z8 Encore! MC MCUs Application Note (AN0330)
- Interfacing an Incremental Rotary Encoder with Z8F64xx Series MCUs Application Note (AN0348)
- midi.org
- lmms.sourceforge.net

Appendix A. Schematics

Figure 15 presents the schematic diagram for the MIDI application.

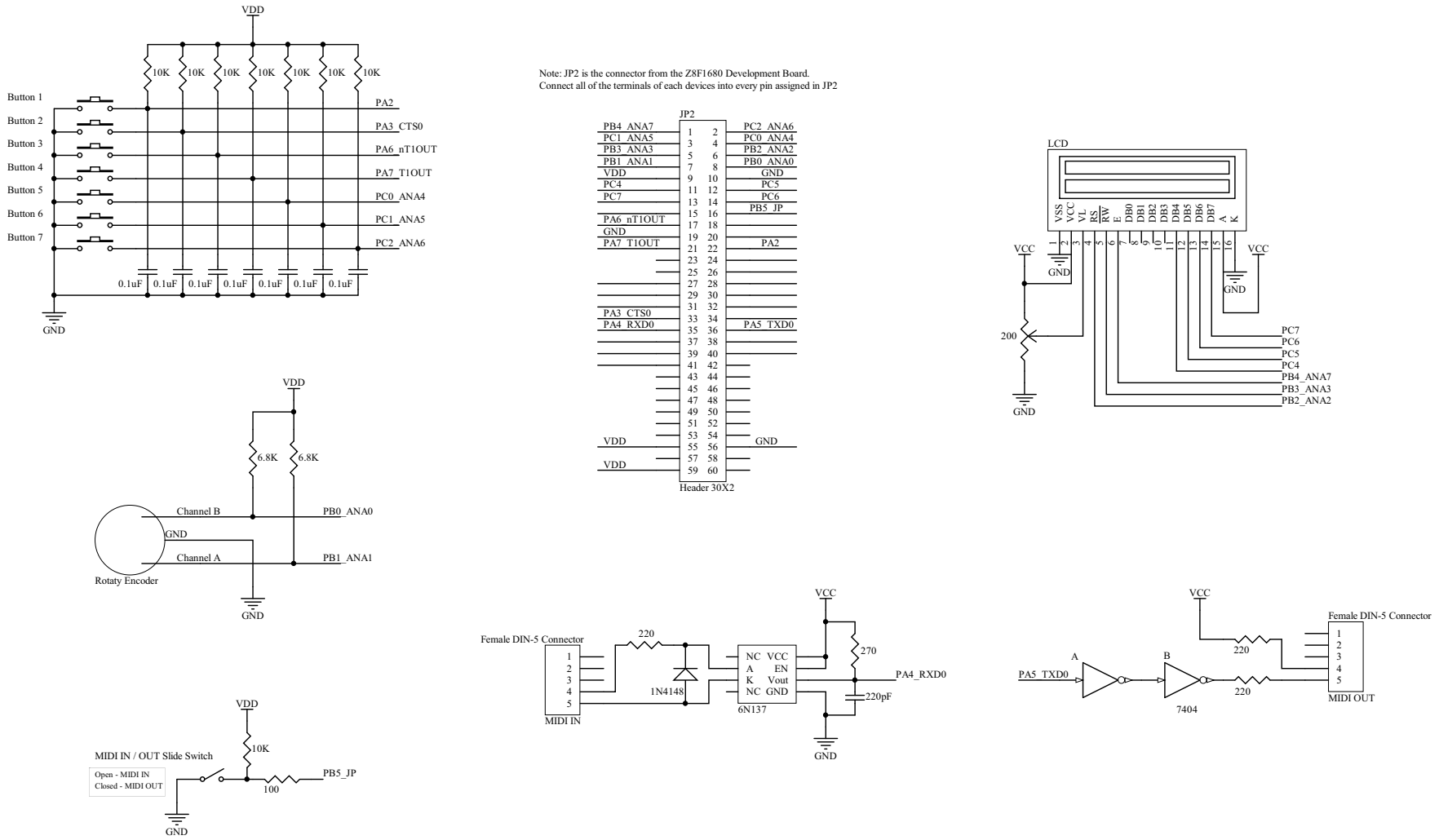

Figure 15. Schematic Diagram for the MIDI Application

Appendix B. Flowcharts

Figures 16 through 18 present flowcharts of the MIDI program activity.

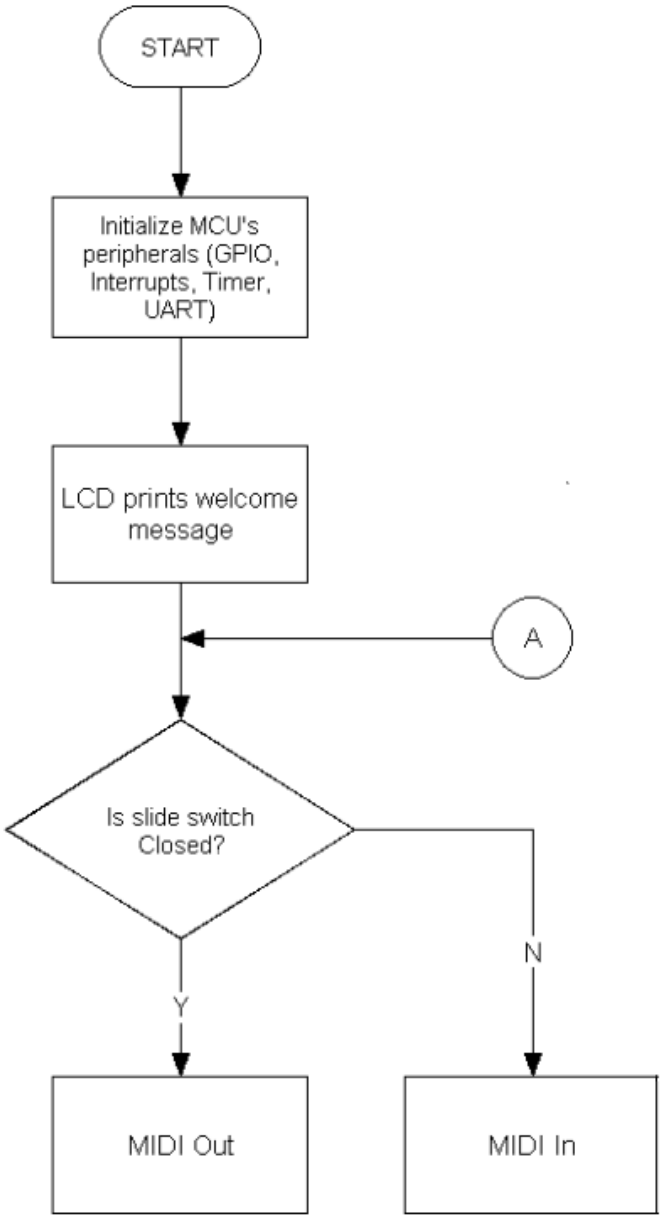

Figure 16. MIDI Main Program Flow

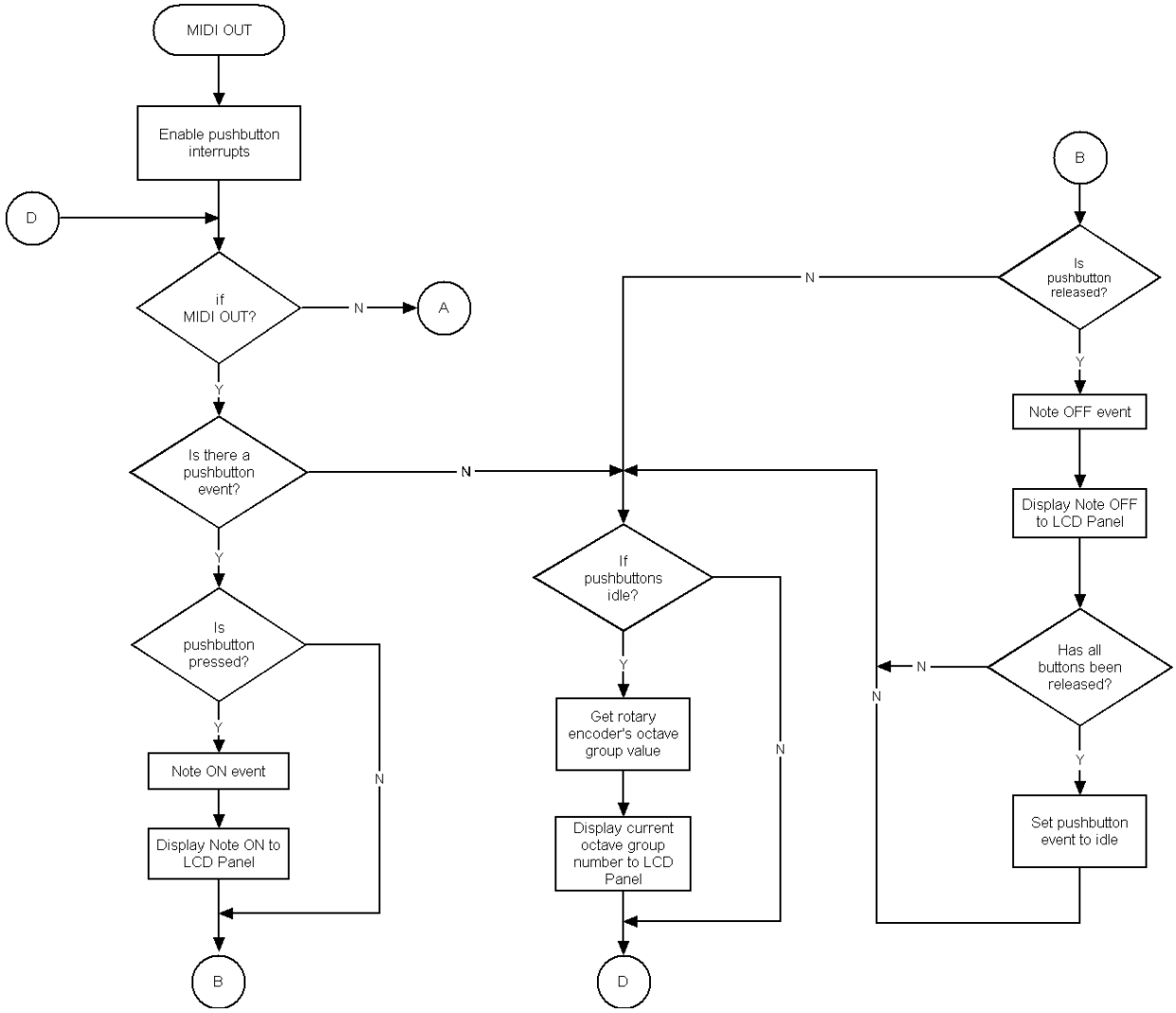

Figure 17. MIDI OUT Routine

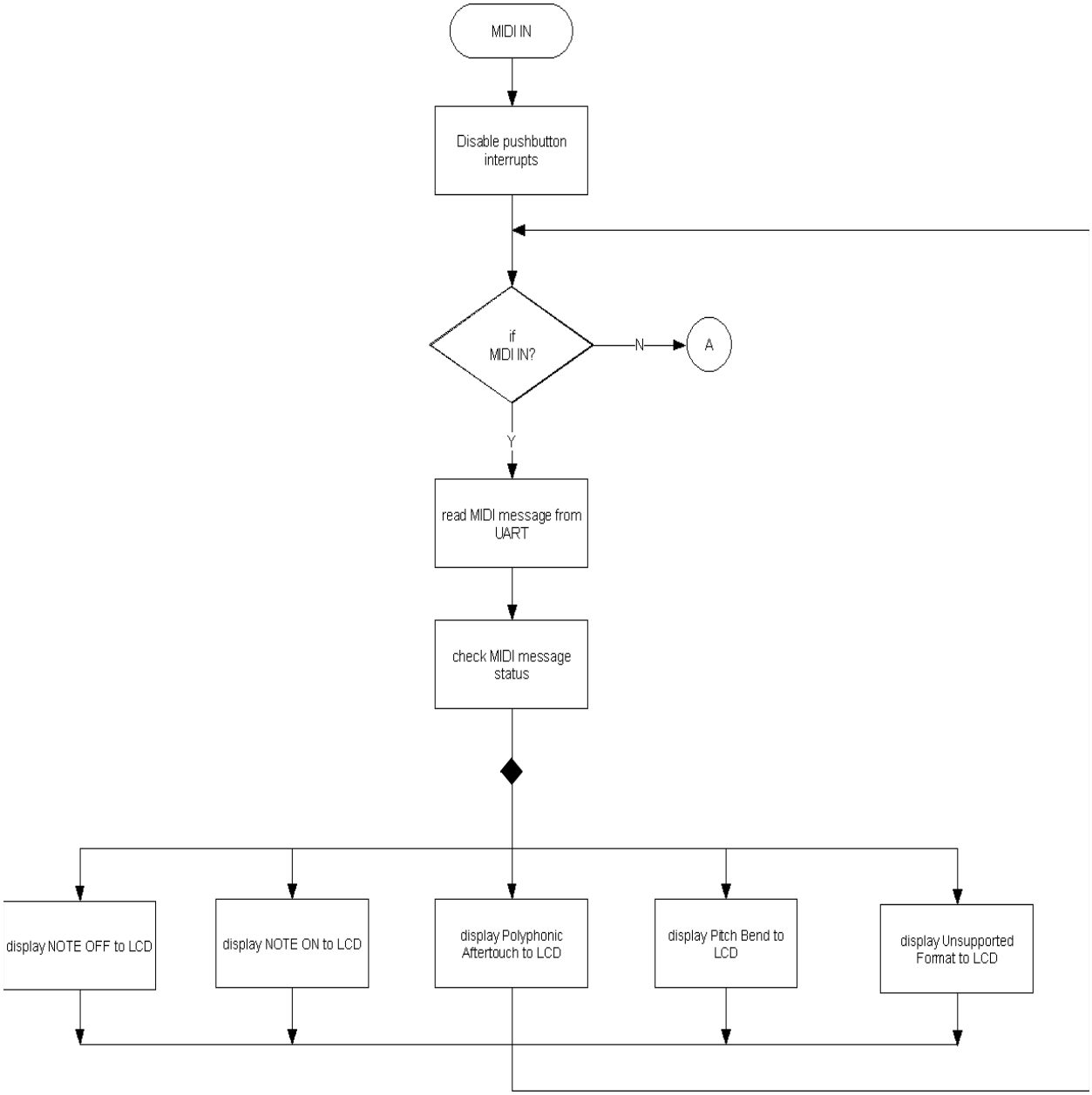

Figure 18. MIDI IN Routine

Figures 19 and 20 show flowcharts for the pushbutton interrupt routines.

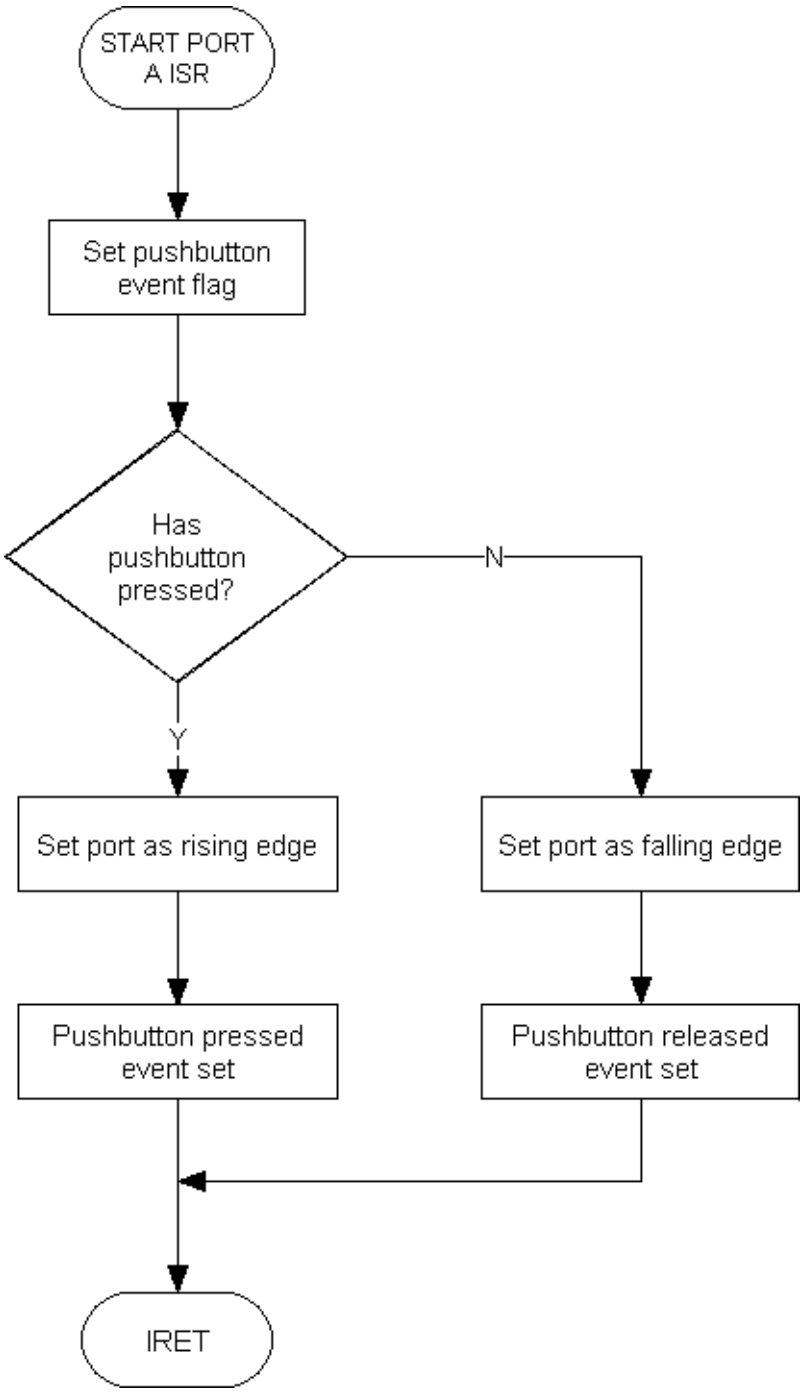

Figure 19. Interrupt Service Routine for the Pushbuttons at Port A

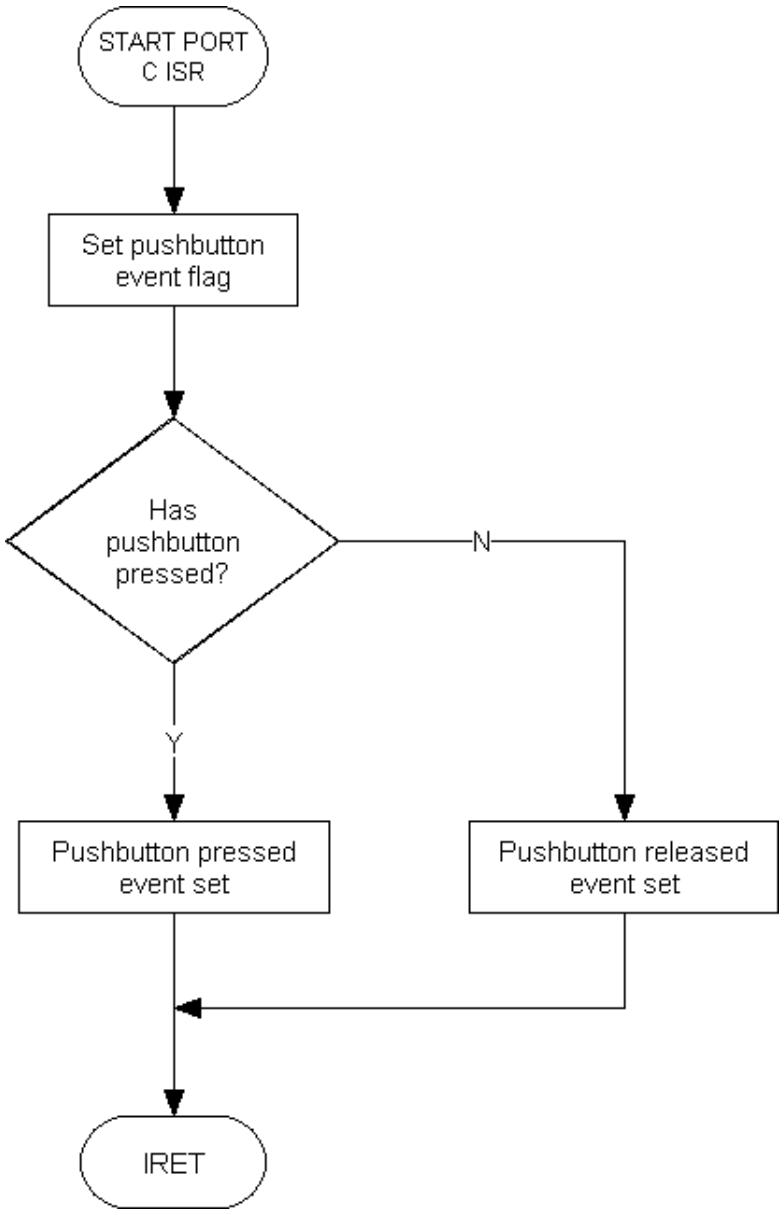

Figure 20. Interrupt Service Routine for the Pushbuttons at Port C

Customer Support

To share comments, get your technical questions answered, or report issues you may be experiencing with our products, please visit Zilog's Technical Support page at http://support.zilog.com.

To learn more about this product, find additional documentation, or to discover other facets about Zilog product offerings, please visit the Zilog Knowledge Base at http://zilog.com/kb or consider participating in the Zilog Forum at http://zilog.com/forum.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, please visit the Zilog website at http://www.zilog.com.

Warning: DO NOT USE THIS PRODUCT IN LIFE SUPPORT SYSTEMS.

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

 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. The information contained within this document has been verified according to the general principles of electrical and mechanical engineering.

Z8 Encore! and Z8 Encore! XP are trademarks or registered trademarks of Zilog, Inc. All other product or service names are the property of their respective owners.