**U1161851 Carl Mills**

**Microcontrollers: Embedded Systems Assignment**

**Report**

**Overview**

I have designed and implemented a fully operational MIDI-based controller system programmable to a PIC18F4520 microcontroller. The design features 5 digital inputs and 5 analogue inputs that allow the user to change various parameters of the MIDI messages transmitted.

**User Instructions**

**LCD Display**

MIDIChannel = 0

This represents the channel that the MIDI messages will be sent.

MIDI Instrument

Number: 12

Represents the MIDI instrument number. Displayed when RB4 or RB5 is pressed.

This displays the value of a particular MIDI parameter. Whilst it is displayed this can be adjusted using the appropriate control knob. This also applies to Velocity, Pan, Modulation and Aftertouch.

Pitch = 53

**Buttons/Control Knobs**

Digital and analogue inputs are individually detailed in the diagram below:



RB4

RB0

RA4

RC5

VRA1

VRA2

VRA3

RA0

RB5

**RB0 (Digital):** This button scrolls through MIDI parameters volume, pitch, pan and modulation. Each parameter and its value are displayed on the LCD screen when selected.

**RA4 (Digital):** When pressed, this button sends a Note on MIDI message containing all information from the adjusted MIDI parameters. Once the button is let go, a note off MIDI message will be sent.

**RB4 (Digital):** This decrements the MIDI instrument number. It sends a MIDI program change message along with the new number, and whilst the button is held displays the new value on the LCD screen.

**RB5 (Digital):** This is similar to RB4’s function, however this increments the MIDI instrument value. This also displays the new value whilst the button is held.

**RC5 (Digital):** This increments the MIDI channel number. The new MIDI channel number is displayed on LCD. If the value reaches 15 it cycles back to 0. This also alters information sent for MIDI messages to the new MIDI channel.

**RA0 (Analog):** Thiscontrol alters the modulation value of the MIDI note when the parameter is selected.

**VRA1 (Analog):** Thiscontrols the velocity value of the MIDI note when the parameter is selected and RA4 is off. When RA4 is on, VRA1 becomes the parameter for Aftertouch, allowing continuous control of the value whilst the note plays.

**VRA2 (Analog):** This controls the pitch of the MIDI note when RB0 selects the pitch channel (LCD will display pitch value e.g. “Pitch = 64”).

**VRA3 (Analog):** This controls the pan settings of the MIDI information. This and each parameter above holds a value between 0-127 selected by the user.

**Software Flow Diagram 229**

**ADC.c**

When prompted by Main.c, ADCConvert function begins; creating a 10-bit value that is then scaled to a range of 0-127. The ADCON0 register is set in Main prior to the function call, which enables the converter and specifies the channel.

**USART.c**

Defines the functions outline in USART.h. This also sets the baud rate in relation to the 4MHz crystal and the MIDI standard baud rate (31250 bits per second).

**Main.h**

This holds all MIDI message functions that are implemented using the USART code. A pause function is defined that allows a pause value to be defined (measured in milliseconds). Main.h also contains declarations of all variables used in Main.c.

**ADC.h**

Declares the ADCConvert function as well as the variables required for conversion e.g. ADRESH & ADRESL variables.

**LCD.h**

Declares functions used to program LCD screen. These include the selecting the line to write text, writing the text and clearing the screen.

**Main.c**

Sets Port Direction Registers and ADCON settings. If statements for digital inputs control functions defined in Main.h. If statements for analog inputs trigger an ADC conversion function defined in ADC files. These also trigger changes in LCD displays.

**LCD.c**

Defines the functions that are declared within LCD.h. Initializes the LCD screen and sets it to 4-bit mode.

**USART.h**

Declares various functions used for sending byte information. The most used for this system is the USARTWriteByte ();

**MIDI Data Commands**

Triggered when RA4 is pressed. 144 = 0x90, meaning the MIDI channel desired will be selected through adding it to 0x90. Pitch and Velocity from the conversions are also sent to this function

*Void NoteOn (int Pitch, int Velocity, int MIDIch)*

*{*

*USARTWriteByte (MIDIch+144);*

*USARTWriteByte (Pitch);*

*USARTWriteByte (Velocity);*

*}*

*Void NoteOff (int Pitch, int MIDIch)*

Triggered when RA4 is released. This uses the same calculation with the MIDI channel, using 128 (0x80)

*{*

*USARTWriteByte (MIDIch+128);*

*USARTWriteByte (Pitch);*

*USARTWriteByte (0x00);*

*}*

This is implemented in the RA4 if statement **before** NoteOn to ensure the pan setting is adjusted before the note is sounded to the user

*Void PanChange (int Pan, int MIDIch)*

*{*

*USARTWriteByte (MIDIch+176);*

*USARTWriteByte (0x0A);*

*USARTWriteByte (Pan);*

*}*

This is also implemented before note on to prevent delay of the effect when sounded. With modulation and pan each variable is sent to the function and used in the last byte.

*Void ModulationLevel (int Modulation, int MIDIch)*

*{*

*USARTWriteByte (MIDIch+176);*

*USARTWriteByte (0x01);*

*USARTWriteByte (Modulation);*

*}*

*Void ProgramChange (int MIDIch, int instrument)*

Program change requires only two byte messages, the status to signify a program change, and the data byte to update the instrument number. This is triggered by RB4 & RB5.

*{*

*USARTWriteByte (MIDIch+192);*

*USARTWriteByte (instrument);*

*}*

*Void AfterT (int MIDIch, int Pitch, int Pressure)*

*{*

Aftertouch is implemented whilst RA4 is on and straight after the conversion function, allowing continuous manipulation whilst the note is held.

*USARTWriteByte (MIDIch+160);*

*USARTWriteByte (Pitch);*

*USARTWriteByte (Pressure);*

*}*