```
// Author Glen Popiel, KW5GP
// uses Morse Library by Erik Linder, Errors fixed and modified by Glen
Popiel, KW5GP
// uses PS2Keyboard Library written by Christian Weichel
<info@32leaves.net>, Errors fixed and modified by Glen Popiel, KW5GP
/*

 This program is free software: you can redistribute it and/or modify
 it under the terms of the version 3 GNU General Public License as
 published by the Free Software Foundation.

 This program is distributed in the hope that it will be useful,
 but WITHOUT ANY WARRANTY; without even the implied warranty of
 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 GNU General Public License for more details.

 You should have received a copy of the GNU General Public License
 along with this program.  If not, see <http://www.gnu.org/licenses/>.

 */

#include <avr/eeprom.h>  // Include the AVR EEPROM Library

// Define the EEPROM data format
#define eeprom_read_to(dst_p, eeprom_field, dst_size)
eeprom_read_block(dst_p, (void *)offsetof(__eeprom_data, eeprom_field),
MIN(dst_size, sizeof((__eeprom_data*)0)->eeprom_field))
#define eeprom_read(dst, eeprom_field) eeprom_read_to(&dst, eeprom_field,
sizeof(dst))
#define eeprom_write_from(src_p, eeprom_field, src_size)
eeprom_write_block(src_p, (void *)offsetof(__eeprom_data, eeprom_field),
MIN(src_size, sizeof((__eeprom_data*)0)->eeprom_field))
#define eeprom_write(src, eeprom_field) { typeof(src) x = src;
eeprom_write_from(&x, eeprom_field, sizeof(x)); }

#define MIN(x,y) ( x > y ? y : x )  // Define a MIN function

#include <Morse.h>   // Include the Morse Library
#include <PS2Keyboard.h>  // Include the PS2Keyboard Library
#include <Wire.h>  // Include the Wire Communication Library
#include <LiquidCrystal_I2C.h>  // Include the LiquidCrystal I2C Library

extern int __bss_end;  // Used by Free Memory function
extern int *__brkval;

const int buflen = 41; // Set size of Macros to 40 characters
const int lcd_end = 16; // set width of LCD
const int lcd_home = 0;
const int comm_speed = 9600; // Set the Serial port speed
const int current_id = 18; // EEPROM ID - used to verify EEPROM data is
valid
const int lcd_address = 0x27; // I2C LCD Address
const int lcd_lines = 1; // Number of lines on LCD
const int beep_pin = 11; // Pin for CW tone
```

```
const int key_pin = 12; // Pin for CW Key
const int beep_on = 1; // 0 = Key, 1 = Beep
const String EEPROM_invalid = "EEPROM data not valid";  // Define the
text data
const String ready = "Keyer Ready";
const String free_mem = "Free Mem:";
const String spd = "Speed = ";
const String f1key = "F1";
const String f2key = "F2";
const String f3key = "F3";
const String f4key = "F4";
const String f5key = "F5";
const String macro = " Macro";
const String selected = " Entry";
const boolean one = 1;
const boolean zero = 0;
const int start_speed = 15; // Starting CW speed
const String mode = " Mode";
const int DataPin = 5; // Set PS2 Keyboard Data Pin
const int IRQpin =  3; // Set PS2 Keyboard Clock Pin
char c;  // variable to hold the CW character to send
int key_speed = start_speed;  // the current keying speed
int cursor_pos = lcd_home;  // the current cursor position
int cursor_line = lcd_home;  // the current cursor line
boolean create_macro = zero;  // create macro flag
boolean macro_select = zero;  // the current macro being created
String macro_data;  // the current macro data
String add_data;  // the new macro data to replace
char macro_key;  // The macro entry key
String F1_data = "";  // The macro key data
String F2_data = "";
String F3_data = "";
String F4_data = "";
String F5_data = "";
boolean morse_beep = one;  // Use to select keying or beep mode. ) = key,
1 = beep
char xF[buflen];
String fdata;
int id;
int morse_speed;

/*
 * __eeprom_data is the magic name that maps all of the data we are
 * storing in our EEPROM
 */
struct __eeprom_data     // The structure of the EEPROM data
{
  char eF1[buflen];  // the F1 macro
  char eF2[buflen];  // the F2 macro
  char eF3[buflen];  // the F3 macro
  char eF4[buflen];  // the F4 macro
  char eF5[buflen];  // the F5 macro
  int e_speed;  // the CW speed
  boolean e_beep;  // beep or key mode
```

```
  int EEPROM_ID;  // the EEPROM ID
  char * F;
};

PS2Keyboard keyboard;  // define the PS2Keyboard

LiquidCrystal_I2C lcd(lcd_address,lcd_end,lcd_lines);  // set the LCD
address to 0x27 for a 16 chars and 1 line display
// The display is a 2 line display, but we only want scrolling on one
line

Morse morse(beep_pin, key_speed, beep_on); //default to beep on pin 11

void setup()
{
  keyboard.begin(DataPin, IRQpin);  // Start the Keyboard
  lcd.init(); // initialize the lcd
  lcd.backlight();  // Turn on the LCD backlight
  lcd.home();  // Go to Home position on LCD
  lcd.print(ready);  // Show Ready on LCD
  lcd.setCursor(lcd_end,lcd_home);  // Move to the end of Line 0
  lcd.autoscroll();  // Enable Autoscroll

  char F1[buflen], F2[buflen], F3[buflen], F4[buflen], F5[buflen];  //
Define the function keys

  eeprom_read(id,EEPROM_ID); // Read the EEPROM
  if (id != current_id)   // We put a set value in the EEPROM data to
indicate the data is valid. If it's not there, the data is invalid
  {
     // Data not valid - keep the default settings
  } else {
     // valid EEPROM DATA - Read EEPROM data
     eeprom_read(F1,eF1);  // Read the EEPROM data to the function keys
     eeprom_read(F2,eF2);
     eeprom_read(F3,eF3);
     eeprom_read(F4,eF4);
     eeprom_read(F5,eF5);
     eeprom_read(morse_speed,e_speed);  // Read the saved Speed
     eeprom_read(morse_beep,e_beep);  // Read the saved Mode

     // Save EEPROM data to variables
     F1_data = String(F1);
     F2_data = String(F2);
     F3_data = String(F3);
     F4_data = String(F4);
     F5_data = String(F5);

     key_speed = morse_speed;  // Set the Key Speed
     mode_set();     // Set the Mode (beep or key)
  }
}  // End Setup Loop

void loop()
```

```
{
  if (keyboard.available())    // Check the keyboard to see if a key has
been pressed
  {
    c = keyboard.read();      // read the key
    // check for some of the special keys

    switch (c)  // Case Statement to determine which key was pressed
    {
      case PS2_ENTER:  // The ENTER key
        break;

      case PS2_TAB:  // The TAB key
        break;

      case PS2_ESC:  // The ESC key
        break;

      case PS2_PAGEDOWN:  // The PageDown key
        break;

      case PS2_PAGEUP:  // The PageUp key
        break;

      case PS2_LEFTARROW:  // The Left Arrow key
        lcd.noBacklight(); // Turn backlight off
        break;

      case PS2_RIGHTARROW:  // The Right Arrow key
        lcd.backlight(); // Turn backlight on
        break;

      case PS2_UPARROW:  // The Up arrow key
        key_speed = key_speed ++;  // Increase CW Speed by 1wpm
        lcd.noAutoscroll();  // Clear the LCD and display current speed
        lcd.clear();
        lcd.home();
        lcd.print(spd + String(key_speed));
        lcd.setCursor(lcd_end,lcd_home);
        lcd.autoscroll();
        increase_speed();  // increase the CW speed
        break;

      case PS2_DOWNARROW:  // The Down Arrow key
        key_speed = key_speed --;  // decrease CW speed by 1wpm
        lcd.noAutoscroll();// Clear the LCD and display current speed
        lcd.clear();
        lcd.home();
        lcd.print(spd + String(key_speed));
        lcd.setCursor(lcd_end,lcd_home);
        lcd.autoscroll();
        decrease_speed();  // decrease the CW speed
        break;
```

```
      case PS2_DELETE:  // The DELETE key
        morse.sendmsg("EEE");// Send 3 dits for error
        break;

      case PS2_F1:  // Macro Key F1
        if (F1_data !="" and !create_macro)
        {
          send_macro(F1_data);  // Send the Macro data if we're not
creating it
          break;

        }
        if (create_macro)   // If we're creating the macro for F1
        {
          // Check Macro_select and select this key if not already
selected
          if (macro_select)
          {
            // We've already selected it, break out - we want to add data
            break;
          } else {
            // Select F1 Macro
            lcd.noAutoscroll();
            lcd.clear();
            lcd.home();
            lcd.print(f1key + macro + selected);  // Display the macro
key selected
            lcd.setCursor(lcd_end,lcd_home);
            lcd.autoscroll();
            macro_select = one;  // A macro key has been selected
            macro_key = PS2_F1;  // The macro key we're creating is F1
          }
        } else {
          break;
        }
        break;

        case PS2_F2:  // Macro Key F2
          if (F2_data !="" and !create_macro)
          {
            send_macro(F2_data);  // Send the Macro data if we're not
creating it
            break;
          }
          if (create_macro)
          {
            // Check Macro_select and select this key if not already
selected
            if (macro_select)
            {
              // We've already selected it, break out - we want to add
data
              break;
            } else {
```

```
              // Select F2 Macro
              lcd.noAutoscroll();
              lcd.clear();
              lcd.home();
              lcd.print(f2key + macro + selected);  // Display the macro
key selected
              lcd.setCursor(lcd_end,lcd_home);
              lcd.autoscroll();
              macro_select = one;  // A macro key has been selected
              macro_key = PS2_F2;  // The macro key we're creating is F2
            }
          } else {
            break;
          }
          break;

        case PS2_F3: // Macro Key F3
          if (F3_data !="" and !create_macro)
          {
            send_macro(F3_data);// Send the Macro data if we're not
creating it
            break;
          }
          if (create_macro)
          {
            // Check Macro_select and select this key if not already
selected
            if (macro_select)
            {
              // We've already selected it, break out - we want to add
data
              break;
            } else {
              // Select F3 Macro
              lcd.noAutoscroll();
              lcd.clear();
              lcd.home();
              lcd.print(f3key + macro + selected);  // Display the macro
key selected
              lcd.setCursor(lcd_end,lcd_home);
              lcd.autoscroll();
              macro_select = one;  // A macro key has been selected
              macro_key = PS2_F3;  // The macro key we're creating is F3
            }
          } else {
            break;
          }
          break;

        case PS2_F4:
          if (F4_data !="" and !create_macro)
          {
            send_macro(F4_data);  // Send the Macro data if we're not
creating it
```

```
            break;
          }
          if (create_macro)
          {
            // Check Macro_select and select this key if not already
selected
            if (macro_select)
            {
              // We've already selected it, break out - we want to add
data
              break;
            } else {
              // Select F4 Macro
              lcd.noAutoscroll();
              lcd.clear();
              lcd.home();
              lcd.print(f4key + macro + selected);  // Display the macro
key selected
              lcd.setCursor(lcd_end,lcd_home);
              lcd.autoscroll();
              macro_select = one;  // A macro key has been selected
              macro_key = PS2_F4;  // The macro key we're creating is F4
            }
          } else {
            break;
          }
          break;

        case PS2_F5:
          if (F5_data !="" and !create_macro)
          {
            send_macro(F5_data);  // Send the Macro data if we're not
creating it
            break;
          }
          if (create_macro)
          {
            // Check Macro_select and select this key if not already
selected
            if (macro_select)
            {
              // We've already selected it, break out - we want to add
data
              break;
            } else {
              // Select F5 Macro
              lcd.noAutoscroll();
              lcd.clear();
              lcd.home();
              lcd.print(f5key + macro + selected);  // Display the macro
key selected
              lcd.setCursor(lcd_end,lcd_home);
              lcd.autoscroll();
              macro_select = one;  // A macro key has been selected
```

```
            macro_key = PS2_F5;  // The macro key we're creating is F5
          }
        } else {
          break;
        }
        break;

      case PS2_F6:  // The F6 key
        break;

      case PS2_F7:  // The F7 key
        break;

      case PS2_F8:  // the F8 key
        break;

      case PS2_F9:  // The F9 key
        break;

      case PS2_F10:  // The F10 key
        save_macro();  // Save the macro to EEPROM
        break;

      case PS2_F11: // F11 key - Turn on Beep Mode
        lcd.noAutoscroll();
        lcd.clear();
        lcd.home();
        lcd.print("Beep" + mode);  // Clear the LCD and display Beep
Mode
        lcd.setCursor(lcd_end,lcd_home);
        lcd.autoscroll();
        morse_beep = one;
        mode_set();  // Set the Mode to Beep
        break;

      case PS2_F12: // F12 key - Turn on Key Mode
        lcd.noAutoscroll();
        lcd.clear();
        lcd.home();
        lcd.print("Key" + mode);  // Clear the LCD and display Keying
Mode
        lcd.setCursor(lcd_end,lcd_home);
        lcd.autoscroll();
        morse_beep = zero;
        mode_set();  // Set the Mode to Keying
        break;

      case PS2_INSERT: // INSERT key - Enter Macro Key Data
        if (!create_macro and !macro_select)  // If we want to create a
macro but haven't selected a key yet
        {
          lcd.noAutoscroll();
          lcd.clear();
          lcd.home();
```

```
            lcd.print("Select" + macro + " Key");  // Display the Key
selected
            lcd.setCursor(lcd_end,lcd_home);
            lcd.autoscroll();
            create_macro = one;  // Set the Create macro flag
            macro_data ="";  // Clear the macro data
          }
            break;

        case PS2_END: // END key - End Macro Data Entry
          if (create_macro)  // If we're creating it, check to see if a
key was selected
          {
            // Abort or Save the Macro
            lcd.noAutoscroll();
            lcd.clear();
            lcd.home();
            if (macro_select)  // If a macro key has been selected, save
it
            {
              save_macro();  // Save the Macro data to EEPROM
              lcd.print(macro + " Saved");  // Display "Saved" on the LCD
            } else {
              // Aborting
              lcd.print(macro + " Aborted");  // Don't save and display
"Aborted" on the lCD
            }
          }
          lcd.setCursor(lcd_end,lcd_home);  // We're all done with the
Macro, clear the screen and reset the flags
          lcd.autoscroll();
          create_macro = zero;
          macro_select = zero;
          break;

        default:  // Otherwise do the default action, which is to
continue on

      if (create_macro) // If we're creating a macro
      {
        // check to see if we have selected key
        if (macro_select) // If we have a valid key, get the macro data
        {
          lcd.noAutoscroll();
          lcd.clear();
          lcd.home();
          lcd.print("Enter" + macro);  // Display that we're ready to
enter Macro data for the selected key
          c = toupper(c);  // convert the character to uppercase

          if (macro_data.length()<(buflen - 1))   // As long as the macro
is <= max length, add to the macro data
          {
            macro_data = String(macro_data) + String(c);
```

```
          }
          lcd.setCursor(lcd_end,lcd_home);
          lcd.autoscroll();
        } else {
          // Select the Macro Key
          if (c == PS2_END)   // Check to see if we want out
          {
            macro_select = zero;
            create_macro = zero;
          }
        }
        } else {
          // If we're not doing the Macro thing just print and send all
normal characters

          lcd.write(toupper(c)); // Display the character on the LCD

          morse.send(c);  // Send the character
      }
    }
  }
}  // End Main Loop

int increase_speed()  // Function to Increase CW Speed
{
  if (morse_beep)
  {
    Morse morse(beep_pin, key_speed, one); //beep
  } else {
    Morse morse(key_pin, key_speed, zero); // no beep
  }
  return key_speed;
}

int decrease_speed()   // Function to decrease CW Speed
{
  if (morse_beep) {
    Morse morse(beep_pin, key_speed, one); // beep
  } else {
    Morse morse(key_pin, key_speed, zero); // keyer
  }
  return key_speed;
}

void save_macro()   // Function to Save the macro to variable -
eventually write it to EEPROM
{
  if (macro_data.length() > (buflen - 1))
  {
    macro_data = macro_data.substring(0,(buflen - 1));
  }
  switch (macro_key) // Determine which key to save the data to
  {
    case PS2_F1:
```

```
      F1_data = macro_data;
      break;

    case PS2_F2:
      F2_data = macro_data;
      break;

    case PS2_F3:
      F3_data = macro_data;
      break;

    case PS2_F4:
      F4_data = macro_data;
      break;

    case PS2_F5:
      F5_data = macro_data;
      break;
  }

  id = 18;  // set the EEPROM ID Byte
  morse_speed = key_speed;
  morse_beep = 1;
  fdata = F1_data;
  m_data();  // Format the data for writing to EEPROM
  eeprom_write_from(xF, eF1,buflen);  // Write the Data to EEPROM

  fdata = F2_data;
  m_data();  // Format the data for writing to EEPROM
  eeprom_write_from(xF, eF2,buflen);  // Write the Data to EEPROM

  fdata = F3_data;
  m_data();  // Format the data for writing to EEPROM
  eeprom_write_from(xF, eF3,buflen);  // Write the Data to EEPROM

  fdata = F4_data;
  m_data();  // Format the data for writing to EEPROM
  eeprom_write_from(xF, eF4,buflen);  // Write the Data to EEPROM

  fdata = F5_data;
  m_data();  // Format the data for writing to EEPROM
  eeprom_write_from(xF, eF5,buflen);  // Write the Data to EEPROM

  eeprom_write(morse_speed,e_speed);   // Write the current speed to
EEPROM
  eeprom_write(morse_beep,e_beep);  // Write the current mode to EEPROM

  eeprom_write(id,EEPROM_ID);  // Write the EEPROM ID to EEPROM
  fdata = "";

}

void m_data()   // Function to convert data to EEPROM writable format
{
```

```
  // This routine figures out where to write the variable data in the
EEPROM
  char yF1[buflen];
  int i = 0;
  if (fdata.length() > (buflen - 1)) // trim the data to the right size
  {
    fdata = fdata.substring(0,(buflen - 1));
  }
  if (fdata.length() > 0) // if it has data, assign it to the variable
one character at a time for each Macro key
  {
    while (i <= fdata.length() -1)
    {
      yF1[i] =  fdata.charAt(i);
      i++;
    }
    yF1[i] = 0;
    i = 0;
    while (yF1[i] != 0) // place the data in the variable one character
at time to write to the EEPROM
    {
      xF[i] = yF1[i];
      i++;
      }
    }
  xF[i] = 0;
}

void send_macro(String F_Key_data)   // Function to Send the Macro data
{
  char send_data[buflen];
  int i = 0;

  lcd.print(F_Key_data);
  while (i <= F_Key_data.length() -1) // Fill the character array with
each character in the macro one character at a time
  {
    send_data[i] = F_Key_data.charAt(i);
    i++;
  }
  send_data[i] = 0;
  morse.sendmsg(send_data);  // send the whole message
}

int get_free_memory()  // Function to get free memory
{
  int free_memory;

  if((int)__brkval == 0)
    free_memory = ((int)&free_memory) - ((int)&__bss_end);
  else
    free_memory = ((int)&free_memory) - ((int)__brkval);

  return free_memory;
```

```
}

void mode_set()   // Function to Set the mode to beep or keying
{
  if (morse_beep)
  {
    Morse morse(beep_pin, key_speed, one); //default to beep on pin 11
  } else {
    Morse morse(key_pin, key_speed, zero); //default to key on pin 12
  }
}
```