```
#include <hidboot.h>
#include <usbhub.h>

// Satisfy the IDE, which needs to see the include statment in the ino too.
#ifdef dobogusinclude
#include <spi4teensy3.h>
#endif
#include <SPI.h>
#include "Keyboard.h"
#include <EEPROM.h>
#define KEY_DOWN 0
#define KEY_UP 1
#define MOD_CHANGE 2

// include the library code:
#include <LiquidCrystal.h>

// initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 12, en = 2, d4 = 3, d5 = 4, d6 = 5, d7 = 6;
LiquidCrystal lcd(A0, en, d4, d5, d6, d7);

const int SetMacroButton = A2;     // the number of the pushbutton pin
int MacroButtonState = HIGH;         // variable for reading the pushbutton status
int ButtonStatePrev = HIGH;
bool ProgramMacro = false;
bool TypeMacro = false;
struct ScanCode{
  uint8_t mod;
  uint8_t key;
  uint8_t type;
};
struct Macro{
  uint8_t count;
  uint8_t mod;
  uint8_t key;
  ScanCode sc[6];
};
Macro MacroArray [10];
int idx = -1;
int curr = -1;
Macro testMacro{0};
bool MacroPressed = false;
```

```
KeyReport report;

void edit_modifier(uint8_t mod){
  report.modifiers = mod;
}
void report_add(uint8_t mod, uint8_t key) {
  uint8_t i;
  report.modifiers = mod;
  if (report.keys[0] != key && report.keys[1] != key &&
              report.keys[2] != key && report.keys[3] != key &&
              report.keys[4] != key && report.keys[5] != key) {
    for (i = 0; i < 6; ++i) {
      if (report.keys[i] == 0) {
        report.keys[i] = key;
        break;
      }
    }
  }
}

void report_remove(uint8_t mod, uint8_t key) {
  uint8_t i;
  report.modifiers = mod;
  for (i = 0; i < 6; ++i) {
      if (report.keys[i] == key) {
        report.keys[i] = 0;
        break;
      }
    }
}

class KbdRptParser : public KeyboardReportParser
{
  protected:
    void OnControlKeysChanged(uint8_t before, uint8_t after);
    void OnKeyDown  (uint8_t mod, uint8_t key);
    void OnKeyUp  (uint8_t mod, uint8_t key);
};
```

```
void KbdRptParser::OnKeyDown(uint8_t mod, uint8_t key)
{
  if(ProgramMacro == true && TypeMacro == false){
    idx = (idx+1)%10;
    for(int i; i < 10; i++){
      if (mod == MacroArray[i].mod && key == MacroArray[i].key){
        idx = i;
      }
    }
    MacroArray[idx].mod = mod;
    MacroArray[idx].key = key;
  }
  else if(TypeMacro == true && MacroArray[idx].count < 6){
    MacroArray[idx].sc[MacroArray[idx].count].mod = mod;
    MacroArray[idx].sc[MacroArray[idx].count].key = key;
    MacroArray[idx].sc[MacroArray[idx].count].type = KEY_DOWN;
    MacroArray[idx].count++;
  }
  else{
  for(int i; i < 10; i++){
    if (mod == MacroArray[i].mod && key == MacroArray[i].key){
      MacroPressed = true;
      curr = i;
      return;
    }
  }

  report_add(mod,key);
  Keyboard.sendReport(&report);
  uint8_t c = OemToAscii(mod, key);

  if (c){
    lcd.setCursor(0, 0);
    lcd.print("Key ");
    lcd.print((char)c);
    lcd.print(" pr");
    lcd.setCursor(0, 1);
    lcd.print("essed   ");
  }
  }
}


void KbdRptParser::OnControlKeysChanged(uint8_t before, uint8_t after) {
  if(ProgramMacro == true){
    if(TypeMacro == true && MacroArray[idx].count < 6){
      MacroArray[idx].sc[MacroArray[idx].count].mod = after;
      MacroArray[idx].sc[MacroArray[idx].count].type = MOD_CHANGE;
      MacroArray[idx].count++;
    }
  }
  else{
  edit_modifier(after);
  Keyboard.sendReport(&report);

  MODIFIERKEYS beforeMod;
  *((uint8_t*)&beforeMod) = before;

  MODIFIERKEYS afterMod;
  *((uint8_t*)&afterMod) = after;
  }
}
```

```
void KbdRptParser::OnKeyUp(uint8_t mod, uint8_t key)
{
  if(ProgramMacro == true){
    if(TypeMacro == false){
      if(mod == MacroArray[idx].mod && key == MacroArray[idx].key){
        TypeMacro = true;
        MacroArray[idx].count = 0;
        lcd.setCursor(0, 0);
        lcd.print("Type Mac");
        lcd.setCursor(0, 1);
        lcd.print("ro      ");
      }
    }
    else if(MacroArray[idx].count < 6){
      MacroArray[idx].sc[MacroArray[idx].count].mod = mod;
      MacroArray[idx].sc[MacroArray[idx].count].key = key;
      MacroArray[idx].sc[MacroArray[idx].count].type = KEY_UP;
      MacroArray[idx].count++;
    }
  }
  else{
  if(MacroPressed == true && mod == MacroArray[curr].mod && key == MacroArray[curr].key){
    lcd.setCursor(0, 0);
    lcd.print("Macro ");
    lcd.print((char)(curr + 48));
    lcd.print(" ");
    lcd.setCursor(0, 1);
    lcd.print("Pushed  ");
    for(uint8_t i = 0; i<MacroArray[curr].count; i++){
      if(MacroArray[curr].sc[i].type == KEY_DOWN){
        report_add(MacroArray[curr].sc[i].mod,MacroArray[curr].sc[i].key);
        Keyboard.sendReport(&report);
        delay(100);
      }
      if(MacroArray[curr].sc[i].type == MOD_CHANGE){
        edit_modifier(MacroArray[curr].sc[i].mod);
        Keyboard.sendReport(&report);
        delay(100);
      }
      if(MacroArray[curr].sc[i].type == KEY_UP){
        report_remove(MacroArray[curr].sc[i].mod,MacroArray[curr].sc[i].key);
        Keyboard.sendReport(&report);
        delay(100);
      }
    }
    MacroPressed = false;
  }

  else {
    report_remove(mod,key);
    Keyboard.sendReport(&report);
    uint8_t c = OemToAscii(mod, key);
    if (c){
      lcd.setCursor(0, 0);
      lcd.print("Key ");
      lcd.print((char)c);
      lcd.print(" re");
      lcd.setCursor(0, 1);
      lcd.print("leased  ");
    }
  }
  }
}
```

```
USB     Usb;
//USBHub     Hub(&Usb);
HIDBoot<USB_HID_PROTOCOL_KEYBOARD>    HidKeyboard(&Usb);

KbdRptParser Prs;

void setup()
{
  // set up the LCD's number of columns and rows:
  lcd.begin(8, 2);

  // initialize the pushbutton pin as an input:
  pinMode(SetMacroButton, INPUT_PULLUP);

  EEPROM.get( 0, MacroArray );

  if (Usb.Init() == -1)
    //Serial.println("OSC did not start.");

  delay( 200 );

  HidKeyboard.SetReportParser(0, &Prs);
}

void loop()
{
  Usb.Task();
   // read the state of the pushbutton value:
  MacroButtonState = digitalRead(SetMacroButton);

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (MacroButtonState == LOW) {
    if(ProgramMacro == false){
       lcd.setCursor(0, 0);
       lcd.print("Press Ke");
       lcd.setCursor(0, 1);
       lcd.print("y       ");
    }
    else{
      lcd.setCursor(0, 0);
      lcd.print("Macro Se");
      lcd.setCursor(0, 1);
      lcd.print("t       ");
    }
  }
  if(ButtonStatePrev != MacroButtonState && MacroButtonState == HIGH){
    ProgramMacro = !ProgramMacro;
    if(ProgramMacro == false && TypeMacro == false){
        memset(MacroArray, 0, 10 * sizeof(Macro));
        EEPROM.put(0, MacroArray);
        idx = -1;
        curr = -1;
        lcd.setCursor(0, 0);
        lcd.print("Macros R");
        lcd.setCursor(0, 1);
        lcd.print("eset    ");
    }
  }
  if(ProgramMacro == false && TypeMacro == true){
    TypeMacro = ProgramMacro;
    EEPROM.put(0, MacroArray);
  }
  ButtonStatePrev = MacroButtonState;
}
```