```
/*
 * File:   seq.c
 * Author: Maarten Hofman
 *
 * Created on October 1, 2022, 2:17 PM
 */

// CONFIG
#pragma config FOSC = INTOSCIO  // Oscillator Selection bits (INTOSCIO oscillator:
I/O function on RA4/OSC2/CLKOUT pin, I/O function on RA5/OSC1/CLKIN)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config MCLRE = OFF      // MCLR Pin Function Select bit (MCLR pin function
is digital input, MCLR internally tied to VDD)
#pragma config CP = OFF         // Code Protection bit (Program memory code
protection is disabled)
#pragma config CPD = OFF        // Data Code Protection bit (Data memory code
protection is disabled)
#pragma config BOREN = OFF      // Brown Out Detect (BOR disabled)
#pragma config IESO = OFF       // Internal External Switchover bit (Internal
External Switchover mode is disabled)
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enabled bit (Fail-Safe
Clock Monitor is disabled)

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

#include <xc.h>
#include <stdlib.h>

#define _XTAL_FREQ 8000000

unsigned int read(unsigned char location) {
  int result = 0;
  EEADR = location * 2 + 1;
  EECON1bits.RD = 1;
  result = EEDAT;
  result = result * 256;
  EEADR = location * 2;
  EECON1bits.RD = 1;
  result = result + EEDAT;
  return result;
}

void write(unsigned char location, unsigned int v) {
  EEDAT = v & 255;
  EEADR = location * 2;
  EECON1bits.WREN = 1;
  PIR1bits.EEIF = 0;
  while (INTCONbits.GIE) {
    INTCONbits.GIE = 0;
  }
  EECON2 = 0x55;
  EECON2 = 0xaa;
  EECON1bits.WR = 1;
  while (!PIR1bits.EEIF) {
    // Wait for write to complete.
  }
  EEDAT = v / 256;
```

```
  EEADR = location * 2 + 1;
  PIR1bits.EEIF = 0;
  EECON2 = 0x55;
  EECON2 = 0xaa;
  EECON1bits.WR = 1;
  while (!PIR1bits.EEIF) {
    // Wait for write to complete.
  }
  PIR1bits.EEIF = 0;
  INTCONbits.GIE = 1;
}

unsigned previous;
unsigned switch1 = 0;
unsigned switch2 = 0;
unsigned char clock;
unsigned char changed = 0;
unsigned char half = 1;
unsigned char quarter;
unsigned int artificialclock = 1;
unsigned char ac = 0;
unsigned char externalclock;
unsigned char counter = 0;
unsigned char kick = 0;
unsigned char snare = 0;
unsigned char pulse = 0;
unsigned char hihat = 0;

unsigned long pattern[8];

typedef union {
    struct {
        unsigned kick:1;
        unsigned snare:1;
        unsigned hihat:1;
        unsigned metal:1;
        unsigned kick2:1;
        unsigned snare2:1;
    };
    uint8_t byte;
} drum_t;

typedef union {
    struct {
        unsigned kick:1;
        unsigned dummy:4;
        unsigned snare:1;
    };
    uint8_t byte;
} drum2_t;

drum_t drums;
drum2_t drums2;

void randomdrum() {
    if (counter > 31) {
        counter = 0;
    }
    drums.kick = (pattern[0] >> counter) & 1;
```

```
    drums.kick2 = (pattern[1] >> counter) & 1;
    drums.hihat = (pattern[2] >> counter) & 1;
    drums.snare = (pattern[3] >> counter) & 1;
    drums.snare2 = (pattern[4] >> counter) & 1;
    drums.metal = (pattern[5] >> counter) & 1;
    drums2.kick = (pattern[6] >> counter) & 1;
    drums2.snare = (pattern[7] >> counter) & 1;
}

void drum15() {
    if (counter > 29) {
        counter = 0;
    }
    drums.byte = 0;
    drums2.byte = 0;
    if ((counter == 0) || (counter == 6) || (counter == 12) || (counter == 18)
            || (counter == 24)) {
        drums.kick = 1;
    }
    if ((counter == 5) || (counter == 15) || (counter == 25)) {
        drums.snare = 1;
    }
    if (pulse) {
        drums.hihat = 1;
    }
    if ((counter == 3) || (counter == 23)) {
        drums.hihat = 1;
    }
    if ((counter == 4) || (counter == 24)) {
        drums.hihat = 0;
    }
    if ((counter == 14) || (counter == 27) || (counter == 29)) {
        drums.metal = 1;
    }
    if ((counter == 2) || (counter == 4) || (counter == 8) || (counter == 10)
            || (counter == 14) || (counter == 16) || (counter == 20)
            || (counter == 22) || (counter == 26) || (counter == 28)) {
        drums.snare2 = 1;
    }
    if (!drums.kick) {
        drums.kick2 = 1;
    }
    if (!drums.snare) {
        drums2.snare = 1;
    }
    if (!drums.snare2) {
        drums2.kick = 1;
    }
}

void drum16() {
    if (counter > 31) {
        counter = 0;
    }
    drums.byte = 0;
    drums2.byte = 0;
    if (counter & 2) {
        drums.kick = 0;
    } else {
```

```
        drums.kick = 1;
    }
    if (counter & 4) {
        drums.snare = 1;
    }
    if (pulse) {
        drums.hihat = 1;
    }
    if ((counter == 3) || (counter == 20)) {
        drums.hihat = 1;
    }
    if ((counter == 4) || (counter == 21)) {
        drums.hihat = 0;
    }
    if ((counter == 15) || (counter == 31) || (counter == 29)) {
        drums.metal = 1;
    }
    if ((counter == 0) || (counter == 3) || (counter == 6) || (counter == 12)
            || (counter == 16) || (counter == 19) || (counter == 22)
            || (counter == 26) || (counter == 30)) {
        drums.kick2 = 1;
    }
    if ((counter == 4) || (counter == 12) || (counter == 20)
            || (counter == 28)) {
        drums.snare2 = 1;
        if (pulse) {
            drums2.kick = 1;
        }
    }
    if ((counter == 0) || (counter == 5) || (counter == 8) || (counter == 13)
            || (counter == 16) || (counter == 21) || (counter == 24)
            || (counter == 29)) {
        drums2.kick = 1;
    }
    if ((counter & 3) == 3) {
        drums2.snare = 1;
    }
    if ((counter & 3) == 2) {
        if (pulse) {
          drums2.snare = 1;
        }
    }
}

void randomize() {
  for (int i = 0; i < 8; i++) {
      pattern[i] = ((rand() * 65536) + (rand() * 4)) + rand() % 4;
  }
}

void main(void) {
  OSCCON = 0b01110110;
  ANSEL = 0b00000000;   // All digital
  TRISC = 0b00000000; // PORTC All Output
  TRISA = 0b11011110; // PORTA All Input except RA0 and RA5
  CMCON0 = 7;

  randomize();
  previous = PORTAbits.RA2;
```

```
  while(1) {
      externalclock = PORTAbits.RA3;
      switch1 = PORTAbits.RA1;
      switch2 = PORTAbits.RA2;
      if (externalclock) {
          artificialclock = 0;
      }
      if (artificialclock) {
          artificialclock++;
          if (artificialclock > 4000) {
              if (ac) {
                  ac = 0;
              } else {
                  ac = 1;
              }
              artificialclock = 1;
          }
          clock = ac;
      } else {
          clock = externalclock;
      }

      if (clock != changed) {
          pulse = 1;
          counter = counter + 1;
      }
      if (pulse > 0) {
          pulse++;
      }
      changed = clock;
      if (!switch2) {
          if (previous) {
              randomize();
          }
          randomdrum();
      } else {
        if (switch1) {
            drum16();
        } else {
            drum15();
        }
      }
      PORTC = drums.byte;
      PORTA = drums2.byte;
      previous = switch2;
  }
  return;
}
```