Raport kodu dla mikrokontrolera PIC24FJ128GA010

Imię i Nazwisko: Jakub Dmochowski
Numer albumu: 169236

Platforma: MPLAB X IDE + XC16
Mikrokontroler: PIC24FJ128GA010

Przegląd funkcji

Nr	Funkcja	Opis działania
1	f1()	8-bitowy licznik binarny zliczający w górę od 0 do 255
2	f2()	8-bitowy licznik binarny zliczający w dół od 255 do 0
3	f3()	8-bitowy licznik w kodzie Graya (0 do 255)
4	f4()	8-bitowy licznik w kodzie Graya (255 do 0)
5	f5()	2x4-bitowy licznik BCD od 0 do 99
6	f6()	2x4-bitowy licznik BCD od 99 do 0
7	f7()	"Wężyk" 3-bitowy poruszający się w lewo i prawo
8	f8()	Efekt kolejki zapalających się kolejno diod
9	f9()	Generator pseudolosowy z pomocniczą funkcją helper9()

Konfiguracja sprzętowa

Program konfiguruje mikrokontroler w następujący sposób:

```
#pragma config POSCMOD = XT // XT Oscillator mode selected
#pragma config OSCIOFNC = ON // OSC2/CLKO/RC15 as port I/O (RC15)
#pragma config FCKSM = CSDCMD // Clock Switching and Monitor disabled
#pragma config FNOSC = PRI // Primary Oscillator (XT, HS, EC)
#pragma config IESO = ON // Int Ext Switch Over Mode enabled
```

Istotne jest również ustawienie watchdog timera oraz konfiguracja portów I/O:

```
AD1PCFG = 0xFFFF;
TRISA = 0x0000;
```

AD1PCFG // Ustawienie wszystkich pinów jako cyfrowe

TRISA // Port A jako wyjście (do sterowania diodami LED)

Analiza kodu

1. Funkcja główna main()

Funkcja main() zawiera nieskończoną pętlę, która sprawdza, który przycisk został naciśnięty i odpowiednio wywołuje jedną z dziewięciu funkcji:

```
int main(void) {
    AD1PCFG = 0xFFFF;
    TRISA = 0x0000;
    char button = 0b00000001;
    while(1){
        if(BUTTON_IsPressed (BUTTON_S3) == true){
            button--;
            if(button == 0b00000000) button = 0b00001001;
            __delay32(1000000);
        } else if(BUTTON_IsPressed (BUTTON_S4) == true){
            button++;
            if(button == 0b00001010) button = 0b00000001;
            __delay32(1000000);
        }
        switch(button){
            case 1:
                f1();
                break;
            case 2:
                f2();
                break;
            case 3:
                f3();
                break;
            case 4:
                f4();
                break;
```

Każde naciśnięcie przycisku S3 powoduje przejście do poprzedniej funkcji, a S4 do następnej. Dodatkowo pozwala na przejście z funkcji 1 do 9 i odwrotnie.

2. **Obsługa przycisków**

buttons.h definiuje funkcję BUTTON_IsPressed() do wykrywania stanu przycisków. Implementacja sprawdza stan pinów portu D:

```
bool BUTTON_IsPressed ( BUTTON button )
{
    switch(button)
    {
        case BUTTON_S3:
            return ( (S3_PORT == BUTTON_PRESSED) ? true : false);

        case BUTTON_S6:
            return ( (S6_PORT == BUTTON_PRESSED) ? true : false);

        case BUTTON_S5:
            return ( ( S5_PORT == BUTTON_PRESSED ) ? true : false ) ;

        case BUTTON_S4:
            return ( ( S4_PORT == BUTTON_PRESSED ) ? true : false ) ;

        default:
            return false;
    }

    return false;
}
```

Mechanizm Przycisków

Każda z funkcji zawiera mechanizm sprawdzający, czy przycisk został naciśnięty, co pozwala na przerwanie aktualnie działającej funkcji

```
    if(BUTTON_IsPressed (BUTTON_S3) == true){
        button--;
        if(button == 0b00000000) button = 0b00001001;
        __delay32(1000000);
    } else if(BUTTON_IsPressed (BUTTON_S4) == true){
        button++;
        if(button == 0b00001010) button = 0b00000001;
        __delay32(1000000);
    }
```

3. Analiza funkcji

Funkcja f1() - Licznik binarny rosnący

```
int f1(){
    unsigned char value = 0b00000000;
    LATA = value;
    while(1){
        __delay32(1000000);
        LATA = ++value;
        if(BUTTON_IsPressed (BUTTON_S3) || BUTTON_IsPressed (BUTTON_S4)) return 0;
    }
}
```

Funkcja inicjalizuje licznik wartością 0, a następnie w nieskończonej pętli inkrementuje licznik i wyświetla jego wartość na diodach LED. Licznik automatycznie wykonuje overflow z 255 na 0 ze względu na typ unsigned char(8 bitów).

Funkcja f2() - Licznik binarny malejący

```
int f2(){
    unsigned char value = 0b11111111;
    LATA = value;
    while(1) {
        __delay32(1000000);
        LATA = --value;
        if(BUTTON_IsPressed (BUTTON_S3) || BUTTON_IsPressed (BUTTON_S4)) return 0;
    }
}
```

Funkcja działa podobnie jak f1(), ale rozpoczyna od wartości maksymalnej 255 i dekrementuje licznik.

Funkcja f3() - Kod Graya rosnący

```
int f3(){
        unsigned char value = 0b00000000;
        unsigned char gray = 0b00000000;
        LATA = gray;
        while(1){
            __delay32(1000000);
            value++;
            gray = value ^ (value >> 1);
            LATA = gray;
            if(BUTTON_IsPressed (BUTTON_S3) || BUTTON_IsPressed (BUTTON_S4)) return 0;
        }
}
```

Funkcja implementuje licznik w kodzie Graya, który jest specjalnym typem kodu binarnego, gdzie dwie kolejne wartości różnią się tylko jednym bitem. Konwersja z kodu binarnego na kod Graya odbywa się poprzez operację XOR na liczbie i tej samej liczbie przesuniętej o 1 bit w prawo.

Funkcja f4() - Kod Graya malejący

```
int f4(){
        unsigned char value = 0b11111111;
        unsigned char gray = 0b10000000;
        LATA = gray;
        while(1){
            __delay32(1000000);
            value--;
            gray = value ^ (value >> 1);
            LATA = gray;
            if(BUTTON_IsPressed (BUTTON_S3) || BUTTON_IsPressed (BUTTON_S4)) return 0;
        }
}
```

Funkcja działa podobnie jak f3(), ale rozpoczyna od wartości maksymalnej i dekrementuje licznik

Funkcja f5() - Licznik BCD rosnący

```
int f5(){
        unsigned char value = 0b00000000;
        unsigned char bcd = 0b00000000;
        LATA = bcd;
        while(1){
            __delay32(1000000);
            value++;
            bcd = (((value / 10) << 4) | (value % 10));
            LATA = bcd;
            if(BUTTON_IsPressed (BUTTON_S3) || BUTTON_IsPressed (BUTTON_S4)) return 0;
        }
}
```

Funkcja implementuje licznik w kodzie BCD (Binary-Coded Decimal), gdzie każda z cyfr dziesiętnych jest reprezentowana na 4 bitach. Starsze 4 bity reprezentują dziesiątki, a młodsze 4 bity jednostki.

Funkcja f6() - Licznik BCD malejący

```
int f6(){
        unsigned char value = 0b01100011;
        unsigned char bcd = 0b10011001;
        LATA = bcd;
        while(1){
            __delay32(1000000);
            value--;
            bcd = (((value / 10) << 4) | (value % 10));
            LATA = bcd;
            if(BUTTON_IsPressed (BUTTON_S3) || BUTTON_IsPressed (BUTTON_S4)) return 0;
        }
}
```

Funkcja działa podobnie jak f5(), ale rozpoczyna od wartości 99 i dekrementuje licznik.

Funkcja f7() - Wężyk

```
int f7(){
    unsigned char snake = 0b00000111;
    while(snake != 0b11100000){
        LATA = snake;
        snake <<= 1;
        __delay32(1000000);
        if(BUTTON_IsPressed (BUTTON_S3) || BUTTON_IsPressed (BUTTON_S4)) return 0;
    }
    while(snake != 0b00000011){
        LATA = snake;
        snake >>= 1;
        __delay32(1000000);
        if(BUTTON_IsPressed (BUTTON_S3) || BUTTON_IsPressed (BUTTON_S4)) return 0;
    }
}
```

Funkcja implementuje efekt "wężyka" - 3-bitowej sekwencji poruszającej się najpierw w prawo, a po osiągnięciu krawędzi - w lewo. Efekt ten jest realizowany przez przesunięcia bitowe.

Funkcja f8() - Kolejka

```
int f8(){
    while(1){
        unsigned char value = 0b00000000;
        for(int i = 0; i < 8; i++){
            unsigned char temp = 0b00000001;
            for(int j = i + 1; j < 8; j++){
                LATA = value + temp;
                temp <<= 1;
                __delay32(1000000);
                if(BUTTON_IsPressed (BUTTON_S3) || BUTTON_IsPressed (BUTTON_S4)) return 0;
            }
            LATA = value + temp;
            value += temp;
            __delay32(1000000);
            if(BUTTON_IsPressed (BUTTON_S3) || BUTTON_IsPressed (BUTTON_S4)) return 0;
        }
    }
}
```

Funkcja tworzy efekt "kolejki" diod LED, gdzie każda kolejna dioda zapala się, a zapalona dioda pozostaje włączona. Implementacja wykorzystuje zagnieżdżone pętle i operacje przesunięcia bitowego.

Funkcja f9() - Generator pseudolosowy

```
int f9(){
    unsigned char value = 0b01100100;
    while(1){
        value = helper9(value);
        LATA = value;
        __delay32(1000000);
        if(BUTTON_IsPressed (BUTTON_S3) || BUTTON_IsPressed (BUTTON_S4)) return 0;
    }
}

unsigned char helper9(unsigned char seed){
    return (((seed & (1 << 6)) ^ (seed & (1 << 5)) ^
        (seed & (1 << 4)) ^ (seed & (1 << 1)) ^ (seed & 1)) << 6) | (seed >> 1);
}
```

Funkcja implementuje generator liczb pseudolosowych oparty na liniowym rejestrze przesuwnym. Funkcja pomocnicza helper9() oblicza następny stan rejestru na podstawie aktualnego stanu, używając operacji XOR na wybranych bitach.