ШАБЛОН ДЛЯ ТАЙМЕРА:

!!!НЕ ЗАБУДЬ РАЗРЕШИТЬ ГЛОБАЛЬНЫЕ ПРЕРЫВАНИЯ!!!

1. Организовать ввод данных в порт P2 и вывод через порт P1 введенных данных.

3. Составить алгоритм и программу вывода в параллельный порт значений 1,2,3,...с интервалом времени 0.5 с.

4. Составить алгоритм и программу вывода сигналов на 2 светодиода с интервалом времени 1 с.

5. Составить алгоритм и программу вывода сигналов на 2 светодиода поочередно с интервалом времени 0.3 с.

6. Составить алгоритм и программу вывода сигналов на 1 светодиод с интервалом времени 0.2 с.

7. Составить алгоритм и программу вывода сигналов на 1 светодиод с интервалом времени 2 с.

8. Организовать ввод информации в порт P2 и вывод через порт P1 введенных данных с интервалом 1 с.

9. Составить алгоритм и программу для переключений вывода P1.1 порта P1 с заданной частотой.

10. Составить алгоритм и программу для переключений выводов P1.0, P1.1 порта P1 с заданной частотой.

11. Составить алгоритм и программу для переключений вывода P1.0 порта P1 с частотой 16 Гц.

12. Составить алгоритм и программу для переключений выводов P1.0, P1.1 порта P1 с частотой 32 Гц.

13. Составить алгоритм и программу формирования последовательности импульсов частотой 200 Гц.

14. Составить алгоритм и программу формирования последовательности импульсов частотой 160 Гц.

15. Составить алгоритм и программу формирования последовательности импульсов с заданным периодом и со скважностью 2.

16. Составить алгоритм и программу вывода двух сигналов заданной частоты

через P1.0 и P1.1 в противофазе.

17. Составить алгоритм и программу вывода двух сигналов заданной частоты

через P1.0 и P1.1 со сдвигом на четверть периода.

18. Составить алгоритм и программу формирования последовательности

импульсов частотой 40 Гц на выходе P2.5.

19. Составить алгоритм и программу формирования последовательности прямоугольных импульсов на выходе P2.1.

20. Составить алгоритм и программу формирования импульсов на выходе P1.0 в режиме непрерывного счета.

21. Составить алгоритм и программу формирования импульсов на выходе P1.0 в режиме счета вверх.

22. Составить алгоритм и программу формирования на P1.0 сигнала частоты

40 Гц, а на P1.1 – 80 Гц.

23. Составить алгоритм и программу формирования сигнала частотой 8 кГц.

24. Составить алгоритм и программу для вывода значений температуры ядра процессора в двоичном коде через 500 мс.

25. Организовать одиночное одноканальное аналого-цифровое преобразование.

26. Составить алгоритм и программу формирования повторяющегося одноканального аналого-цифровое преобразования.

27. Составить алгоритм и программу формирования 1 на выходе P1.1

при достижении заданной температуры ядра процессора.

28. Составить подпрограмму прерывания для вывода ADC12MEM0 в

порты P1, P2

Заняты - оранжевый
Свободны - синий
Готовы - чёрный

ШАБЛОН ДЛЯ ТАЙМЕРА:

```
void timer_A_init(){
    TACTL = TASSEL0 + MC0; // (SSEL = 01) ACLK - 32768 Гц и режим
вверх
    // в режиме вверх период прерываний равен TACCR0 + 1
    //ТУТ ПОМЕНЯТЬ НА НУЖНОЕ ВРЕМЯ
    TACCR0 = 32767;   // - 1 секунда (32768 тиков в секунду)
    TACCTL0 = CCIE;     // разрешить прерывание от TACCR0
}

// обработчик прерываний от таймера A
#pragma vector=TIMERA0_VECTOR
__interrupt void timer_A_interrupt(void){
    //do smth
```

```
        // флаг прерывания сбрасывается автоматически
    }
```

!!!НЕ ЗАБУДЬ РАЗРЕШИТЬ ГЛОБАЛЬНЫЕ ПРЕРЫВАНИЯ!!!

---

1. Организовать ввод данных в порт P2 и вывод через порт P1 введенных данных.

```
    #include <msp430.h>
    #include "system_define.h"
    #include "system_variable.h"
    #include "function_prototype.h"

    void main(void) {
        WDTCTL = WDTPW + WDTHOLD;  // Остановка watchdog timer
        Init_System_Clock();            // НАДО ЛИ
        Init_System();                  // НАДО ЛИ
        P1DIR = 255;  // P1 настраиваем на вывод
        P2DIR = 0;    // P2 настраиваем на ввод
        while(1)
            P1OUT = P2IN;  //из регистра ввода порта 2 записываем
        в регистр вывода порта 1
    }
```

2. Составить алгоритм и программу записи в P1 членов геометрической прогрессии 1,2,4, ...с интервалом 1 с.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"

void timer_A_init(){
    TACTL = TASSEL0 + MC0; // (SSEL = 01) ACLK - 32768 Гц и режим
вверх
    // в режиме вверх период прерываний равен TACCR0 + 1
    TACCR0 = 32767;     // - 1 секунда (32768 тиков в секунду)
    TACCTL0 = CCIE;     // разрешить прерывание от TACCR0
}

unsigned int k = 1;
void main(void) {
    WDTCTL = WDTPW + WDTHOLD;  // Остановка watchdog timer
    Init_System_Clock();
    Init_System();
    __enable_interrupt();   //разрешение глобальных прерываний (бит
GIE)
    P1DIR = 255;  // P1 настраиваем на вывод
    timer_A_init();
    while(1);
}

// обработчик прерываний от таймера A
#pragma vector=TIMERA0_VECTOR
__interrupt void timer_A_interrupt(void){
```

```
    P1OUT = k;
    if (k == 32768) // чтоб не переполнялось
        k = 1;
    else
        k *= 2;
    // флаг прерывания сбрасывается автоматически
}
```

3. Составить алгоритм и программу вывода в параллельный порт значений 1,2,3,...с интервалом времени 0.5 с.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"

void timer_A_init(){
    TACTL = TASSEL0 + MC0; // (SSEL = 01) ACLK - 32768 Гц и режим
вверх
    // в режиме вверх период прерываний равен TACCR0 + 1
    TACCR0 = 16383;   // - 0.5 секунды (32768 тиков в секунду => 16384
тиков в 0.5 sec)
    TACCTL0 = CCIE;     // разрешить прерывание от TACCR0
}

void main(void) {
    WDTCTL = WDTPW + WDTHOLD;  // Остановка watchdog timer
    Init_System_Clock();
    Init_System();
    P1DIR = 255;
    __enable_interrupt();   //разрешение глобальных прерываний (бит
GIE)
    timer_A_init();
    //P1OUT = 1 ;
    while(1);
}

unsigned int i=0;
// обработчик прерываний от таймера A
#pragma vector=TIMERA0_VECTOR
__interrupt void timer_A_interrupt(void){
    P1OUT = i++;
    //P1OUT++;  // можно и так, PxOUT доступны для чтения
    // флаг прерывания сбрасывается автоматически
}
```

4. Составить алгоритм и программу вывода сигналов на 2 светодиода с интервалом времени 1 с.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"

void timer_A_init(){
    TACTL = TASSEL0 + MC0; // (SSEL = 01) ACLK - 32768 Гц и режим
вверх
```

```
    // в режиме вверх период прерываний равен TACCR0 + 1
    TACCR0 = 32767;   // - 1 секунда (32768 тиков в секунду)
    TACCTL0 = CCIE;     // разрешить прерывание от TACCR0
}

unsigned int k = 1;
void main(void) {
    WDTCTL = WDTPW + WDTHOLD;  // Остановка watchdog timer
    Init_System_Clock();
    Init_System();
    __enable_interrupt();   //разрешение глобальных прерываний (бит
GIE)
    Init_I2C();
    timer_A_init();
    while(1);
}

// обработчик прерываний от таймера A
#pragma vector=TIMERA0_VECTOR
__interrupt void timer_A_interrupt(void){
    LED_change(1);    //меняем состояния 1 и 2 светодиода
    LED_change(2);
    // флаг прерывания сбрасывается автоматически
}
```

5. Составить алгоритм и программу вывода сигналов на 2 светодиода поочередно с интервалом времени 0.3 с.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"

void timer_A_init(){
    TACTL = TASSEL0 + MC0; // (SSEL = 01) ACLK - 32768 Гц и режим
вверх
    // в режиме вверх период прерываний равен TACCR0 + 1
    TACCR0 = 9829;   // - 0.3 секунды (32768*0.3 -1 = 9829.4)
    TACCTL0 = CCIE;     // разрешить прерывание от TACCR0
}

unsigned int k = 1;
void main(void) {
    WDTCTL = WDTPW + WDTHOLD;  // Остановка watchdog timer
    Init_System_Clock();
    Init_System();
    __enable_interrupt();   //разрешение глобальных прерываний (бит
GIE)
    Init_I2C();
    timer_A_init();
    LED_set(1);     // включим первый светодиод, чтобы горели по
очереди
    while(1);
}

// обработчик прерываний от таймера A
#pragma vector=TIMERA0_VECTOR
__interrupt void timer_A_interrupt(void){
```

```
    LED_change(1);    //меняем состояния 1 и 2 светодиода
    LED_change(2);
    // флаг прерывания сбрасывается автоматически
}
```

6. Составить алгоритм и программу вывода сигналов на 1 светодиод с интервалом времени 0.2 с.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"

void timer_A_init(){
    TACTL = TASSEL0 + MC0; // (SSEL = 01) ACLK - 32768 Гц и режим
вверх
    // в режиме вверх период прерываний равен TACCR0 + 1
    TACCR0 = 6553;   // - 0.2 секунды (32768*0.2 -1 = 6552.6)
    TACCTL0 = CCIE;     // разрешить прерывание от TACCR0
}

unsigned int k = 1;
void main(void) {
    WDTCTL = WDTPW + WDTHOLD;  // Остановка watchdog timer
    Init_System_Clock();
    Init_System();
    __enable_interrupt();   //разрешение глобальных прерываний (бит
GIE)
    Init_I2C();
    timer_A_init();
    while(1);
}

// обработчик прерываний от таймера A
#pragma vector=TIMERA0_VECTOR
__interrupt void timer_A_interrupt(void){
    LED_change(1);    //меняем состояния 1 светодиода
    // флаг прерывания сбрасывается автоматически
}
```

7. Составить алгоритм и программу вывода сигналов на 1 светодиод с интервалом времени 2 с.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"

void timer_A_init(){
    TACTL = TASSEL0 + MC0; // (SSEL = 01) ACLK - 32768 Гц и режим
вверх
    // в режиме вверх период прерываний равен TACCR0 + 1
    TACCR0 = 65535;     // - 2 секунды (32768*2 -1 = 65535)
    TACCTL0 = CCIE;     // разрешить прерывание от TACCR0
}

unsigned int k = 1;
void main(void) {
    WDTCTL = WDTPW + WDTHOLD;  // Остановка watchdog timer
```

```
    Init_System_Clock();
    Init_System();
    __enable_interrupt();   //разрешение глобальных прерываний (бит
GIE)
    Init_I2C();
    timer_A_init();
    while(1);
}

// обработчик прерываний от таймера A
#pragma vector=TIMERA0_VECTOR
__interrupt void timer_A_interrupt(void){
    LED_change(1);    //меняем состояния 1 светодиода
    // флаг прерывания сбрасывается автоматически
}
```

8. Организовать ввод информации в порт P2 и вывод через порт P1 введенных данных с интервалом 1 с.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"

void timer_A_init(){
    TACTL = TASSEL0 + MC0; // (SSEL = 01) ACLK - 32768 Гц и режим
вверх
    // в режиме вверх период прерываний равен TACCR0 + 1
    TACCR0 = 32767;   // - 1 секунда (32768 тиков в секунду)
    TACCTL0 = CCIE;     // разрешить прерывание от TACCR0
}

unsigned int k = 1;
void main(void) {
    WDTCTL = WDTPW + WDTHOLD;  // Остановка watchdog timer
    Init_System_Clock();
    Init_System();
    __enable_interrupt();   //разрешение глобальных прерываний (бит
GIE)
    P1DIR = 255;  // P1 настраиваем на вывод
    P2DIR = 0;    // P2 настраиваем на ввод
    timer_A_init();
    while(1);
}

// обработчик прерываний от таймера A
#pragma vector=TIMERA0_VECTOR
__interrupt void timer_A_interrupt(void){
    P1OUT = P2IN;     //из регистра ввода порта 2 записываем в регистр
вывода порта 1
    // флаг прерывания сбрасывается автоматически
}
```

9. Составить алгоритм и программу для переключений вывода P1.1 порта P1 с заданной частотой.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
```

```
#include "function_prototype.h"

unsigned int freq = 100;   // задаваемая частота в Гц

void timer_A_init(){
    TACTL = TASSEL0 + MC0; // (SSEL = 01) ACLK - 32768 Гц и режим
вверх
    // в режиме вверх период прерываний равен TACCR0 + 1
    TACCR0 = 32768 / freq - 1;   // TACCR0 соответствует freq Гц
(можно еще заморочиться с округлением для точности)
    TACCTL0 = CCIE;     // разрешить прерывание от TACCR0
}

unsigned int k = 1;
void main(void) {
    WDTCTL = WDTPW + WDTHOLD;  // Остановка watchdog timer
    Init_System_Clock();
    Init_System();
    __enable_interrupt();   //разрешение глобальных прерываний (бит
GIE)
    P1DIR |= BIT1;  // P1.1 настраиваем на вывод
    timer_A_init();
    while(1);
}

// обработчик прерываний от таймера A
#pragma vector=TIMERA0_VECTOR
__interrupt void timer_A_interrupt(void){
    P1OUT ^= BIT1;     //переключение значения на ножке P1.1
}
```

10. Составить алгоритм и программу для переключений выводов P1.0, P1.1 порта P1 с заданной частотой.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"

unsigned int freq = 100;   // задаваемая частота в Гц

void timer_A_init(){
    TACTL = TASSEL0 + MC0; // (SSEL = 01) ACLK - 32768 Гц и режим
вверх
    // в режиме вверх период прерываний равен TACCR0 + 1
    TACCR0 = 32768 / freq - 1;   // TACCR0 соответствует freq Гц
(можно еще заморочиться с округлением для точности)
    TACCTL0 = CCIE;     // разрешить прерывание от TACCR0
}

unsigned int k = 1;
void main(void) {
    WDTCTL = WDTPW + WDTHOLD;  // Остановка watchdog timer
    Init_System_Clock();
    Init_System();
    __enable_interrupt();   //разрешение глобальных прерываний (бит
GIE)
    P1DIR |= BIT1 + BIT0;  // P1.1 и P1.0 настраиваем на вывод
```

```
    timer_A_init();
    while(1);
}

// обработчик прерываний от таймера A
#pragma vector=TIMERA0_VECTOR
__interrupt void timer_A_interrupt(void){
    P1OUT ^= BIT1 + BIT0;     //переключение значения на ножках P1.1 и
P1.0
}
```

11. Составить алгоритм и программу для переключений вывода P1.0 порта P1 с частотой 16 Гц.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"

void timer_A_init(){
    TACTL = TASSEL0 + MC0; // (SSEL = 01) ACLK - 32768 Гц и режим
вверх
    // в режиме вверх период прерываний равен TACCR0 + 1
    TACCR0 = 2047;   // TACCR0 соответствует 16 Гц (32768 / 16 - 1)
    TACCTL0 = CCIE;     // разрешить прерывание от TACCR0
}

unsigned int k = 1;
void main(void) {
    WDTCTL = WDTPW + WDTHOLD;  // Остановка watchdog timer
    Init_System_Clock();
    Init_System();
    __enable_interrupt();   //разрешение глобальных прерываний (бит
GIE)
    P1DIR |= BIT0;  // P1.0 настраиваем на вывод
    timer_A_init();
    while(1);
}

// обработчик прерываний от таймера A
#pragma vector=TIMERA0_VECTOR
__interrupt void timer_A_interrupt(void){
    P1OUT ^= BIT0;     //переключение значения на ножке P1.0
}
```

12. Составить алгоритм и программу для переключений выводов P1.0, P1.1 порта P1 с частотой 32 Гц.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"

void timer_A_init(){
    TACTL = TASSEL0 + MC0; // (SSEL = 01) ACLK - 32768 Гц и режим
вверх
    // в режиме вверх период прерываний равен TACCR0 + 1
    TACCR0 = 1023;   // TACCR0 соответствует 32 Гц (32768 / 32 - 1)
    TACCTL0 = CCIE;     // разрешить прерывание от TACCR0
```

```
}

unsigned int k = 1;
void main(void) {
    WDTCTL = WDTPW + WDTHOLD;  // Остановка watchdog timer
    Init_System_Clock();
    Init_System();
    __enable_interrupt();   //разрешение глобальных прерываний (бит
GIE)
    P1DIR |= BIT1 + BIT0;  // P1.1 и P1.0 настраиваем на вывод
    timer_A_init();
    while(1);
}

// обработчик прерываний от таймера A
#pragma vector=TIMERA0_VECTOR
__interrupt void timer_A_interrupt(void){
    P1OUT ^= BIT1 + BIT2;     //переключение значения на ножках P1.1 и
P1.0
}
```

13. Составить алгоритм и программу формирования последовательности импульсов частотой 200 Гц.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"

void timer_A_init(){
    TACTL = TASSEL0 + MC0; // (SSEL = 01) ACLK - 32768 Гц и режим
вверх
    // в режиме вверх период прерываний равен TACCR0 + 1
    TACCR0 = 1023;   // TACCR0 соответствует 32 Гц (32768 / 32 - 1)
    TACCTL0 = CCIE;     // разрешить прерывание от TACCR0
}

unsigned int k = 1;
void main(void) {
    WDTCTL = WDTPW + WDTHOLD;  // Остановка watchdog timer
    Init_System_Clock();
    Init_System();
    __enable_interrupt();   //разрешение глобальных прерываний (бит
GIE)
    P1DIR |= BIT1 + BIT0;  // P1.1 и P1.0 настраиваем на вывод
    timer_A_init();
    while(1);
}

// обработчик прерываний от таймера A
#pragma vector=TIMERA0_VECTOR
__interrupt void timer_A_interrupt(void){
    P1OUT ^= BIT1 + BIT2;     //переключение значения на ножках P1.1 и
P1.0
}
```

14. Составить алгоритм и программу формирования последовательности

импульсов частотой 160 Гц.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"

void timer_A_init(){
    TACTL = TASSEL0 + MC0; // (SSEL = 01) ACLK - 32768 Гц и режим
вверх
    // в режиме вверх период прерываний равен TACCR0 + 1
    TACCR0 = 203;   // TACCR0 соответствует 160 Гц (32768 / 160-
1=204,8-1 =~203)
    TACCTL0 = CCIE;     // разрешить прерывание от TACCR0
}

void main(void) {
    WDTCTL = WDTPW + WDTHOLD;  // Остановка watchdog timer
    Init_System_Clock();
    Init_System();
    __enable_interrupt();   //разрешение глобальных прерываний (бит
GIE)
    P1DIR |= BIT0;  // P1.0 настраиваем на вывод
    timer_A_init();
    while(1);
}

// обработчик прерываний от таймера A
#pragma vector=TIMERA0_VECTOR
__interrupt void timer_A_interrupt(void){
    P1OUT ^= BIT0;     //переключение значения на ножке P1.0
}
```

15. Составить алгоритм и программу формирования последовательности импульсов с заданным периодом и со скважностью 2.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"

void main(void) {
    WDTCTL = WDTPW|WDTHOLD;
    Init_System_Clock();
    // P1.1 выводит OUT0 таймера A
    P1SEL |= BIT1;
    // ACLK тактирует таймер
    TACTL |= TASSEL_1;
    // частота
    TACCR0 = 500;
    TACCR1 = 250;
    TACCTL0 |= OUTMOD_3;
    TACTL |= MC_1;
};
```

16. Составить алгоритм и программу вывода двух сигналов заданной частоты через P1.0 и P1.1 в противофазе.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"

unsigned int freq = 100;   // задаваемая частота в Гц

void timer_A_init(){
    TACTL = TASSEL0 + MC0; // (SSEL = 01) ACLK - 32768 Гц и режим
вверх
    // в режиме вверх период прерываний равен TACCR0 + 1
    TACCR0 = 32768 / freq - 1;   // TACCR0 соответствует freq Гц
(можно еще заморочиться с округлением для точности)
    TACCTL0 = CCIE;     // разрешить прерывание от TACCR0
}

unsigned int k = 1;
void main(void) {
    WDTCTL = WDTPW + WDTHOLD;  // Остановка watchdog timer
    Init_System_Clock();
    Init_System();
    __enable_interrupt();   //разрешение глобальных прерываний (бит
GIE)
    P1DIR |= BIT1 + BIT0;  // P1.1 и P1.0 настраиваем на вывод
    P1OUT = BIT0; // Здесь мы выдаем сигнал на 0-й ножке, 1-я ножка
отдыхает
    timer_A_init();
    while(1);
}

// обработчик прерываний от таймера A
#pragma vector=TIMERA0_VECTOR
__interrupt void timer_A_interrupt(void){
    P1OUT ^= BIT1 + BIT0;     //переключение значения на ножках P1.1 и
P1.0 на противоположные и 0-ой порт отдыхает, а 1-ый выдаёт сигнал и
так постоянно будет меняться.
}
```

17. Составить алгоритм и программу вывода двух сигналов заданной частоты через P1.0 и P1.1 со сдвигом на четверть периода.

18. Составить алгоритм и программу формирования последовательности импульсов частотой 40 Гц на выходе P2.5.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"

void timer_A_init(){
    TACTL = TASSEL0 + MC0; // (SSEL = 01) ACLK - 32768 Гц и режим
вверх
    // в режиме вверх период прерываний равен TACCR0 + 1
    TACCR0 = 818;   // TACCR0 соответствует 40 Гц (32768 /40-1=819,2-1
=~818)
    TACCTL0 = CCIE;     // разрешить прерывание от TACCR0
```

```
}

void main(void) {
    WDTCTL = WDTPW + WDTHOLD;  // Остановка watchdog timer
    Init_System_Clock();
    Init_System();
    __enable_interrupt();   //разрешение глобальных прерываний (бит
GIE)
    P2DIR |= BIT5;  // P2.5 настраиваем на вывод
    timer_A_init();
    while(1);
}

// обработчик прерываний от таймера A
#pragma vector=TIMERA0_VECTOR
__interrupt void timer_A_interrupt(void){
    P2OUT ^= BIT5;     //переключение значения на ножке P2.5
}
```

19. Составить алгоритм и программу формирования последовательности прямоугольных импульсов на выходе P2.1.

20. Составить алгоритм и программу формирования импульсов на выходе P1.0 в режиме непрерывного счета.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"

void timer_A_init(){
    TACTL = TASSEL0 + MC1; // (SSEL = 01) ACLK - 32768 Гц и режим
вверх
    // в непрерывном режиме (MC1): таймер считает от 0000h до 0FFFFh;
    TACCTL0 = CCIE;     // разрешить прерывание от TACCR0
}

void main(void) {
    WDTCTL = WDTPW + WDTHOLD;  // Остановка watchdog timer
    Init_System_Clock();
    Init_System();
    __enable_interrupt();   //разрешение глобальных прерываний (бит
GIE)
    P1DIR |= BIT0;  // P1.0 настраиваем на вывод
    timer_A_init();
    while(1);
}

// обработчик прерываний от таймера A
#pragma vector=TIMERA0_VECTOR
__interrupt void timer_A_interrupt(void){
    P1OUT ^= BIT0;     //переключение значения на ножке P1.0
}
```

21. Составить алгоритм и программу формирования импульсов на выходе P1.0 в режиме счета вверх.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"

void timer_A_init(){
    TACTL = TASSEL0 + MC0; // (SSEL = 01) ACLK - 32768 Гц и режим
вверх
    // в режиме счета вверх, период прерываний равен TACCR0 + 1
    TACCR0 = 10000;
    TACCTL0 = CCIE;     // разрешить прерывание от TACCR0
}

void main(void) {
    WDTCTL = WDTPW + WDTHOLD;  // Остановка watchdog timer
    Init_System_Clock();
    Init_System();
    __enable_interrupt();   //разрешение глобальных прерываний (бит
GIE)
    P1DIR |= BIT0;  // P1.0 настраиваем на вывод
    timer_A_init();
    while(1);
}

// обработчик прерываний от таймера A
#pragma vector=TIMERA0_VECTOR
__interrupt void timer_A_interrupt(void){
    P1OUT ^= BIT0;     //переключение значения на ножке P1.0
}
```

22. Составить алгоритм и программу формирования на P1.0 сигнала частоты 40 Гц, а на P1.1 – 80 Гц.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"
#include "I2C.h"
#include "main.h"

/*
 * 22. Составить алгоритм и программу формирования на P1.0 сигнала
частоты 40 Гц, а на P1.1 – 80 Гц.
 * Идея в том, чтобы заставить таймер считать вверх-вниз, при этом на
частота 40 Гц будет генерироваться при достижении TACCR0,
 * а частота 80 Гц - при достижении TACCR1 (в два раза чаще)
 * Аналогично можно было еще сделать один TACCR0 с частотой 80 Гц, и
просто напросто каждое второе прерывание дергать порт P1.0
 */
// обработчик прерываний от таймера A
// Убрать прерывание из uninitialized_vectors !!!
#pragma vector=TIMERA0_VECTOR
__interrupt void timerA0_interrupt(void){
     char port = P1OUT;
     port &= 0xFE + ~(port & 0x01);
     P1OUT = port;
}
```

```
#pragma vector=TIMERA1_VECTOR
__interrupt void timerA1_interrupt(void){
      if (TAIV & TAIV_TACCR1){   // Если у нас тут прерывание по поводу
TACCR1, то обрабатываем
      char port = P1OUT;
      port &= 0xFD + ~(port & 0x02);
      P1OUT = port;
      }
}

void timer_init(){
      // TASSEL0,1: 00 - TACLK, 01 - ACLK, 10 - SMCLK, 11 - INCLK
      // MC0,1: 00 - останолвен, 01 - вверх к TACCR0, 10 - непрерывный
до 0FFFFh, 11 - вверх/вниз к TACCR0 - 0000h
      // TACLR - очистка счетчика
      // ID_x - делитель /1, /2, /4, /8
      TACTL = TASSEL_1 + MC_1 + TACLR;
      TACCR0 = 32767 / 40; // Если у нас ACLK (32768 Гц), то при TACCR0
= 32767 прерывание будет происходить раз в секунду (1 Гц)
      TACCR1 = 32767 / 40 / 2;
      TACCTL0 = CCIE; // разрешить прерывания от TACCR0, TACCR1
      TACCTL1 = CCIE;
}

void main(void) {
      WDTCTL = WDTPW+WDTHOLD; // Отключить сторожевой таймер
      _enable_interrupt();  // Включить прерывание
      Init_System_Clock();  // Запустить тактирование
      Init_System();        // Настройка портов, итд
      P1DIR += BIT0 + BIT1;   // Порт P1.0, P1.1 -на выход
      while(1);
}
```

23. Составить алгоритм и программу формирования сигнала частотой 8 кГц.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"

void timer_A_init(){
    TACTL = TASSEL0 + MC0; // (SSEL = 01) ACLK - 32768 Гц и режим
вверх
    // в режиме вверх период прерываний равен TACCR0 + 1
    TACCR0 = 3;   // TACCR0 соответствует 8000 Гц (32768 / 8000-
1=4,096-1 =~3)
    TACCTL0 = CCIE;     // разрешить прерывание от TACCR0
}

void main(void) {
    WDTCTL = WDTPW + WDTHOLD;  // Остановка watchdog timer
    Init_System_Clock();
    Init_System();
     __enable_interrupt();   //разрешение глобальных прерываний (бит
GIE)
    P1DIR |= BIT0;  // P1.0 настраиваем на вывод
```

```
    timer_A_init();
    while(1);
}

// обработчик прерываний от таймера A
#pragma vector=TIMERA0_VECTOR
__interrupt void timer_A_interrupt(void){
    P1OUT ^= BIT0;     //переключение значения на ножке P1.0
}
```

24. Составить алгоритм и программу для вывода значений температуры ядра процессора в двоичном коде через 500 мс.

(мог накосячить немного с настройкой АЦП, требуется адекватная проверка)

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"

void timer_A_init(){
    TACTL = TASSEL0 + MC0; // (SSEL = 01) ACLK - 32768 Гц и режим
вверх
    // в режиме вверх период прерываний равен TACCR0 + 1
    TACCR0 = 16384;   // - 0,5 секунды
    TACCTL0 = CCIE;     // разрешить прерывание от TACCR0
}

void ADC_init(){
    ADC12CTL1 = SHP + CSTARTADD_0; //  режим "одноканальный с одним
преобразованием", таймер выборки и адрес одиночного преобразования -
ADC12MEM0, тактирование - ADC12OSC (~5МГц)
    ADC12CTL0 = SHT00 + SHT01 + SHT02; // делитель = 192 (5МГц/192 ~=
38мкс, надо минимум 30мкс для температурного датчика)
    // выбор опорного напряжения - Vr+ = VeREF+ = 3.3В, Vr- = AVss =
0В
    //    и входного канала для ячейки памяти ADC12MEM0
    ADC12MCTL0 = SREF_3 + INCH_3 +  INCH_1;
//или с внутренним опорным напряжением (всё равно для термодатчика он
активируется))
    //ADC12MCTL0 = INCH_3 +  INCH_1;
    ADC12IE = BIT0;  // разрешить прерывание от ADC12MEM0
    ADC12CTL0 |= ADC12ON + ENC;    // включение АЦП, преобразование
разрешено
}

void main(void) {
    WDTCTL = WDTPW + WDTHOLD;
    Init_System_Clock();
    __enable_interrupt();
    ADC_init();
    timer_A_init();
    P1DIR = 255;   // P1 настраиваем на вывод
    while(1);
}


// обработчик прерываний от таймера A
#pragma vector=TIMERA0_VECTOR
```

```
__interrupt void timer_A_interrupt(void){
    ADC12CTL0 |= ADC12SC;   // страрт преобразования
}

#pragma vector = ADC12_VECTOR
__interrupt void ADC_interrupt(void){
    ADC12CTL0 &= ~ENC;    // преобразование запрещено
    float V_in = ADC12MEM0 * 3.3 / 4095;
    byte temperature = (V_in - 0.986)/0.00355;
    P1OUT = temperature;
    ADC12CTL0 |= ENC;       // преобразование разрешено
}
```

25. Организовать одиночное одноканальное аналого-цифровое преобразование.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"
#include "main.h"
unsigned result = 0;
// Будем преобразовывать данные с датчика тока, потому что почему бы и нет
void main(void) {
  WDTCTL = WDTPW + WDTHOLD;
  Init_System_Clock();
  Init_System();
  _enable_interrupt();
  P6SEL |= BIT1;              // Выбираем АЦП ADC1, к которому подключен датчик тока
  ADC12CTL1 = SHP + CSTARTADD_0;  // Режим одиночного одноканального
преобразования, начальный адрес преобразования - ADCMEM0
  ADC12MCTL0 = SREF_3 + INCH_1;   // Выбор опорного напряжения и входного канала
ADC
  ADC12IE |= BIT1;            // Включить прерывания от АЦП
  ADC12CTL0 |= ENC;             // Разрешить преобразования
  ADC12CTL0 |= ADC12ON;         // Включить АЦП
  ADC12CTL0 |= ADC12SC;         // Запуск преобразования
  while ((ADC12IFG & BIT1) == 0); // Ожидаем результат преобразования
  result = ADC12MEM0;           // Сохраняем результат преобразования
  ADC12CTL0 = 0;              // Все выключай
  while (1);
}
```

26. Составить алгоритм и программу формирования повторяющегося одноканального аналого-цифровое преобразования.

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"
#include "main.h"

unsigned result = 0;

void ADC_init(){
  P6SEL |= BIT1;                  // Выбираем АЦП ADC1, к которому подключен датчик тока
  ADC12CTL1 = SHP + CSTARTADD_0 + CONSEQ1; // таймер выборки и стартовый адрес
преобразования - ADC12MEM0, повторяющийся одноканальный режим
  ADC12MCTL0 = SREF_3 + INCH_1; // Выбор опорного напряжения и входного канала
для adc12mem0
```

```
    ADC12IE |= BIT1;              // Включить прерывания от АЦП
    ADC12CTL0 |= ENC;             // Разрешить преобразования
    ADC12CTL0 |= ADC12ON + ADC12SC;         // Включить АЦП, старт преобразований
}

// Будем преобразовывать данные с датчика тока, потому что почему бы и нет
void main(void) {
    WDTCTL = WDTPW + WDTHOLD;
    _enable_interrupt();
    Init_System_Clock();
    Init_System();
    ADC_init();
    while(1);
}

#pragma vector = ADC12_VECTOR
__interrupt void ADC_interrupt(void){
    ADC12CTL0 &= ~ENC;  // преобразование запрещено
    result = ADC12MEM0; // сохраняем результат для чего нибудь прикольного
    ADC12CTL0 |= ENC;   // преобразование разрешено
}
```

27. Составить алгоритм и программу формирования 1 на выходе P1.1 при достижении заданной температуры ядра процессора.

(мог накосячить немного с настройкой АЦП, требуется адекватная проверка)

```
#include <msp430.h>
#include "system_define.h"
#include "system_variable.h"
#include "function_prototype.h"

void ADC_init(){
    ADC12CTL1 = CONSEQ1 + SHP + CSTARTADD_0; //  режим "повторный одноканальный", таймер выборки и адрес одиночного преобразования - ADC12MEM0, тактирование - ADC12OSC (~5МГц)
    ADC12CTL0 = SHT00 + SHT01 + SHT02; // делитель = 192 (5МГц/192 ~= 38мкс, надо минимум 30мкс для температурного датчика)
    // выбор опорного напряжения - Vr+ = VeREF+ = 3.3В, Vr- = AVss = 0В
    //    и входного канала для ячейки памяти ADC12MEM0
    ADC12MCTL0 = SREF_3 + INCH_3 +  INCH_1;
//или с внутренним опорным напряжением (всё равно для термодатчика он активируется))
//ADC12MCTL0 = INCH_3 +  INCH_1;
    ADC12IE = BIT0;  // разрешить прерывание от ADC12MEM0
    ADC12CTL0 |= ADC12ON + ENC + ADC12SC;    // включение АЦП, преобразование разрешено, стрварт преобразования
}

int temperature = 70;   // задаваемая температур процессора
float t_V = 0;

// напряжение, соответствующее заданной температуре

void main(void) {
    WDTCTL = WDTPW + WDTHOLD;
    t_V = 0.00355 * temperature + 0.986;
    Init_System_Clock();
```

```
    Init_System();
    __enable_interrupt();
    ADC_init();
    P1DIR |= BIT1;   // P1.1 настраиваем на вывод
    while(1);
}

#pragma vector = ADC12_VECTOR
__interrupt void ADC_interrupt(void){
    ADC12CTL0 &= ~ENC;    // преобразование запрещено
    float V_in = ADC12MEM0 * 3.3 / 4095;
       if (V_in >= t_V)
           P1OUT |= BIT1;   // установка бита P1.1
       else
           P1OUT &= ~BIT1;   // сброс бита P1.1
    ADC12CTL0 |= ENC;       // преобразование разрешено
}
```

28. Составить подпрограмму прерывания для вывода ADC12MEM0 в порты P1, P2

Разрешить глобальные прерывания (__enable_interrupt();)
Настроить АЦП, для обработки прерывания записи в ADC12MEM0 надо разрешить это прерывание (ADC12IE = BIT0;)
Настроить порты P1 и P2 на вывод (единицы)
Подпрограмма обработки прерывания:

```
#pragma vector = ADC12_VECTOR
__interrupt void ADC_interrupt(void){
    ADC12CTL0 &= ~ENC;
        P1OUT = (byte)ADC12MEM0;
        P2OUT = (byte)ADC12MEM0 >> 8;
    ADC12CTL0 |= ENC;       // преобразование разрешено
}
```