БГУИР

Кафедра ЭВМ

Отчет по лабораторной работе № 3
Тема: «Тактирование. Питание. Энергопотребление»
Вариант №8

Выполнил:
студент группы 150503 Ходосевич М.А.

Проверил:
ассистент каф. ЭВМ _____ Шеменков В.В.

Минск
2024

1. Постановка задачи

Написать программу, переключающую режимы тактирования, питания и энергопотребления в соответствии с заданием. Задание варианта 2: LPM2, Vcore = 1,9 – 1,6 В, DCOCLK (источник для всех) = REFOCLK, смена частоты DCO на выходе 10 / 60 частот входа.

2. Теоретические сведения

Серия MSP430 разработана для использования в устройствах с низким энергопотреблением. Микроконтроллер поддерживает 7 режимов пониженного энергопотребления: LPM0 – LPM4, LPM3.5, LPM4.5. Режимы LPM0 – LPM4 конфигурируются битами CPUOFF, OSCOFF, SCG0, SCG1 регистра состояния микроконтроллера SR:

бит 4 – CPUOFF. Установка бита отключает процессор микроконтроллера;

бит 5 – OSCOFF. Установка бита отключает генератор частот, если только он не используется для формирования тактовой частоты MCLK или SMCLK;

биты 6, 7 – SCG0, SCG1 — отключение системного тактового генератора 0 (1), какого именно, зависит от серии устройства, например FLL или DCO.

Комбинации бит регистра состояния, соответствующие режимам, а также какие сигналы микроконтроллера при этом отключаются, приведены в таблице:

Таблица 3.1. Режимы пониженного энергопотребления

Режим	Сигналы				Биты SR			
	Vcore	CPU/MCLK	FLL	ACLK	CPUOFF	OSCOFF	SCG0	SCG1
LPM0	On	Off	On	On	1	0	0	0
LPM1	On	Off	Off	On	1	0	1	0
LPM2	On	Off	Off	On	1	0	0	1
LPM3	On	Off	Off	On	1	0	1	1
LPM4	On	Off	Off	Off	1	1	1	1

Как видим, во всех режимах отключается ЦПУ. При этом в LPM0 и в LPM1 тактирование периферии разрешено, DCO разрешен, если является источником для частот ACLK или SMCLK; в LPM2 и LPM3 разрешена только одна тактовая частота периферии ACLK (SMCLK отключена), DCO разрешен, если является источником для частоты ACLK; в LPM4 – все тактовые частоты запрещены, отключены SMCLK, ACLK, DCO. Самоактивация возможна для

режимов LPM0 – LPM3, в этих режимах доступны следующие источники прерываний: таймеры, включая WDT и RTC, АЦП, DMA, USART, цифровые входы-выходы, компаратор, внешние прерывания, USCI, а также другая периферия микроконтроллера и внешние прерывания. В режиме LPM4 самоактивация невозможна, доступны только внешние прерывания.

Установка любого из вышеназванных битов регистра состояния сразу же переводит микроконтроллер в требуемый режим. Предыдущий режим сохраняется в стеке (так как это регистр состояния) при выполнении прерывания. Биты CPUOFF, OSCOFF, SCG1 автоматически сбрасываются при обработке прерывания. Управление вернется к предыдущему режиму, если в обработчике прерываний сохраненное значение SR не изменяется, иначе при возврате установится новый режим. Периферия может быть отключена не только общим запретом тактирования, но и индивидуально установками в своих регистрах управления.

В режимах LPM цифровые порты ввода-вывода и ОЗУ и регистры не изменяются. В режимах LPMx.5 (LPM3.5, LPM4.5) отключается регулятор напряжения блока управления питанием PMM, содержимое регистров и ОЗУ теряется, но состояние цифровых входов-выходов сохраняется. Выход из этих режимов возможен включением питания, сигналом сброса, а также отдельными периферийными модулями. LPM3.5, в отличие от LPM4.5, позволяет выйти из режима по сигналу от RTC. Выход из LPMx.5 приводит к сбросу, поэтому требуется повторное конфигурирование устройств. Режим LPMx.5 устанавливается битом PMMREGOFF и битами регистра статуса LPM4.

Общие принципы построения систем с низким энергопотреблением:

– максимально длительное время нахождения в режимах пониженного энергопотребления (особенно LPM3 и LPM4);

– использование прерываний для управления ходом выполнения программ;

– включение периферии только по мере необходимости;

– использование интегрированной периферии с низким потреблением энергии вместо программного выполнения функций;

– вычисление ветвей и использование таблиц значений вместо опроса флагов и длительных программных вычислений;

– избегать частого вызова функций и процедур из-за дополнительных затрат;

– использовать однотактные регистры ЦПУ в длинных процедурах;

– отключать недоступные и неиспользуемые сегменты памяти при помощи регистра управления ОЗУ RCCTL0.

Модуль управления питанием PMM обладает следующими характеристиками:

– входное напряжение 1,8-3,6 В;

– 4 уровня напряжения для питания ядра (Vcore), задаваемых программно;

– супервизор уровня напряжения (SVS - Supple Voltage Supervisor) с программируемым порогом как для выходного, так и для входного напряжения;

– монитор уровня напряжения (SVM - Supple Voltage Monitor) с программируемым порогом как для выходного, так и для входного напряжения;

– сброс при низком питании (BOR);

– программно доступные индикаторы сбоя питания;

– защита выводов от сбоя по питанию.

Монитор изменения напряжения SVM только отслеживает выход за пороговое значение, супервизор SVS еще и генерирует сигнал сброса.

При изменении уровня Vcore нельзя увеличивать частоту MCLK, пока не установится новый уровень напряжения. Для проверки уровня Vcore используется SVML. При снижении уровня питания необходимо убедиться, что его будет достаточно для установленной частоты. Vcore можно изменять только на один уровень за 1 раз.

При увеличении уровня напряжения необходимо выполнить следующую последовательность действий:

1. Устанавливаются новые уровни SVMH, SVSH, чтобы убедиться, что входное напряжение DVCC выше планируемого Vcore. Установить новый уровень SVML, дождаться установки флага SVSMLDLYIFG.

2. Установить PMMCOREV для определения нового уровня Vcore.

3. Дождаться установки флага SVMLVLRIFG.

4. Установить новый уровень SVSL. При снижении уровня напряжения необходимо выполнить следующее:

5. Устанавливаются SVML, SVSL для нового уровня и ожидается установка флага SVSMLDLYIFG.

6. Программируем новый уровень Vcore, задав PMMCOREV.

Для управления подсистемой питания используются регистры:

PMMCTL0 (0120h) - управление PMM;

PMMCTL1 (0122h) - управление PMM;

SVSMHCTL (0124h) - управление SVS, SVM на входе;

SVSMLCTL (0126h) - управление SVS, SVM на выходе;

PMMIFG (012Ch) - флаги прерываний;

PMMIE (012Eh) - разрешение прерываний;

PM5CTL0 (0130h) - управление режимом LPMx.5.

В микроконтроллере MSP430F5529 имеются следующие источники синхросигналов:

– внутренний генератор низкой частоты со сверхмалым потреблением (VLO), около 9,4 КГц;

– внутренний низкочастотный генератор (REFO), 32 Кгц;

– 32 КГц кварцевый генератор (XT1 LF);

– интегрированный внутренний цифровой управляемый генератор (DCO), стабилизируется с помощью цифровой автоподстройки частоты (FLL

– frequency locked loop);

– высокочастотный кварцевый генератор (XT2) 4 – 32 МГц. На плате используется резонатор 4 МГц.

Три синхросигнала выбираются из этих источников: – вспомогательная тактовая частота (ACLK);

– главная частота (MCLK), используется для тактирования процессора;

– SMCLK (Sub-Main), используется для тактирования периферии;

– буферный выход частоты ACLK/n (n = 1, 2, 4, 8, 16, 32).

Генератор VLO активен, если он используется в качестве источника для любой из частот MCLK, SMCLK, ACLK. REFO — если он установлен как источник: для частоты ACLK в активном режиме, либо LPM0 – LPM3; частоты MCLK в активном режиме; частоты SMCLK в активном режиме и LPM0, LPM1; частоты FLLREFCLK и при этом DCO выбран источником для трех вышеперечисленных вариантов. XT1 и XT2 разрешаются для тех же 6 вариантов, что и REFO, а также в случаях, когда поля XT1OFF = 0 (запрещено отключение XT1) либо XT2OFF = 0 соответственно.

После сброса устанавливается следующий режим:

– источником для ACLK выбирается XT1CLK;

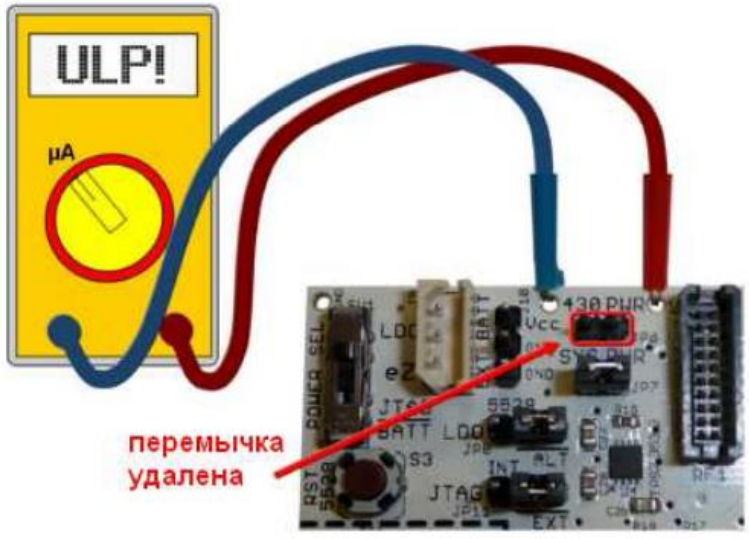
– источником для MCLK, SMCLK выбирается DCOCLKDIV;

– разрешается автоподстройка, источником для FLL выбирается XT1CLK;

– для F5529 выводы кварцевых генераторов разделяются с цифровыми I/O, поэтому после сброса эти выводы конфигурируются как цифровые, и сигналы от кварца недоступны.

Для измерения тока потребления используются перемычки JP6 и JP7.

Для измерения тока потребления микроконтроллера MSP430F5529, при отключенном напряжении питания экспериментальной платы, перемычка JP6 удаляется, щупы амперметра либо мультиметра подключаются непосредственно к освободившимся контактам либо к отверстиям на краю печатной платы по соседству с ними:



Для измерения тока потребления экспериментальной платы MSPEXP430F5529, при отключенном напряжении питания экспериментальной платы, перемычка JP7 удаляется, щупы амперметра либо мультиметра подключаются непосредственно к освободившимся контактам. Перемычка JP7 находится непосредственно под перемычкой JP6.

3. Выполнение работы

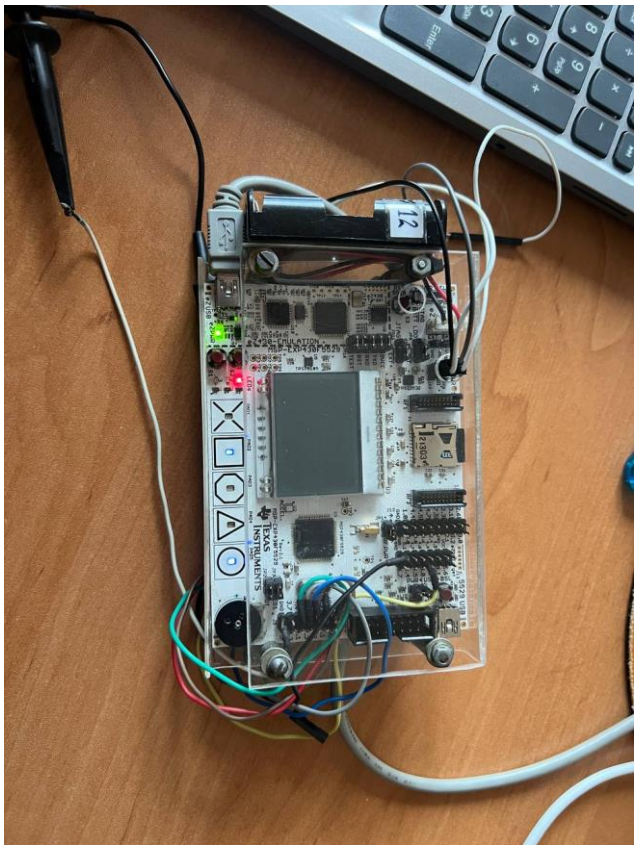
Программа инициализирует порты ввода-вывода микроконтроллера MSP430: светодиоды настраиваются как выходы, а кнопки (S1 и S2) — как входы с подтягивающими резисторами для стабилизации их состояния и предотвращения ложных срабатываний. Для обеих кнопок активируются прерывания, которые срабатывают при нажатии, и вызывают соответствующие обработчики событий.

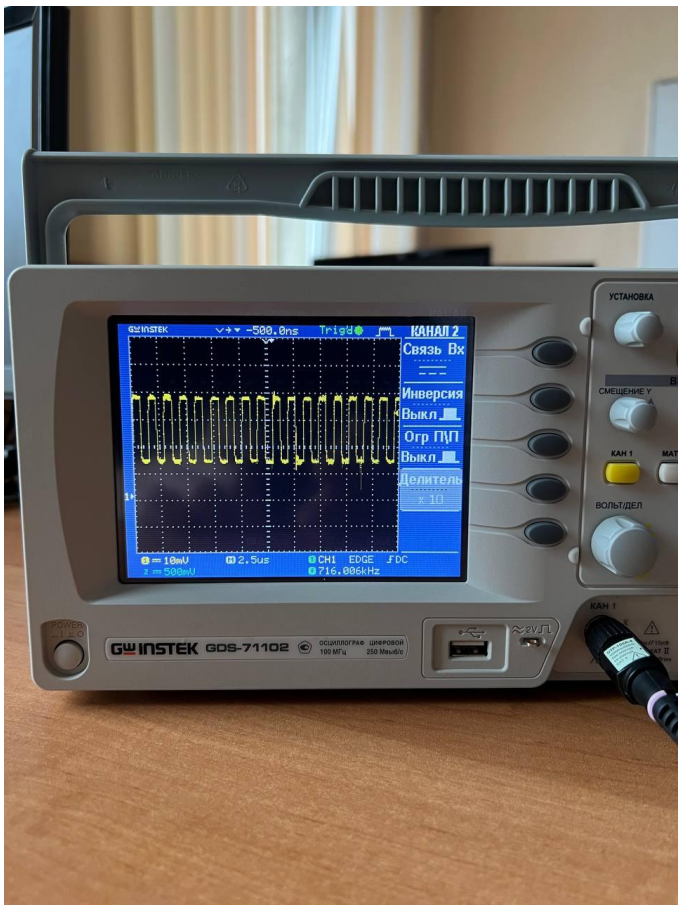
При нажатии первой кнопки (S1) происходит прерывание, в котором программа переключает микроконтроллер между режимом нормального потребления и режимом пониженного энергопотребления (LPM0).

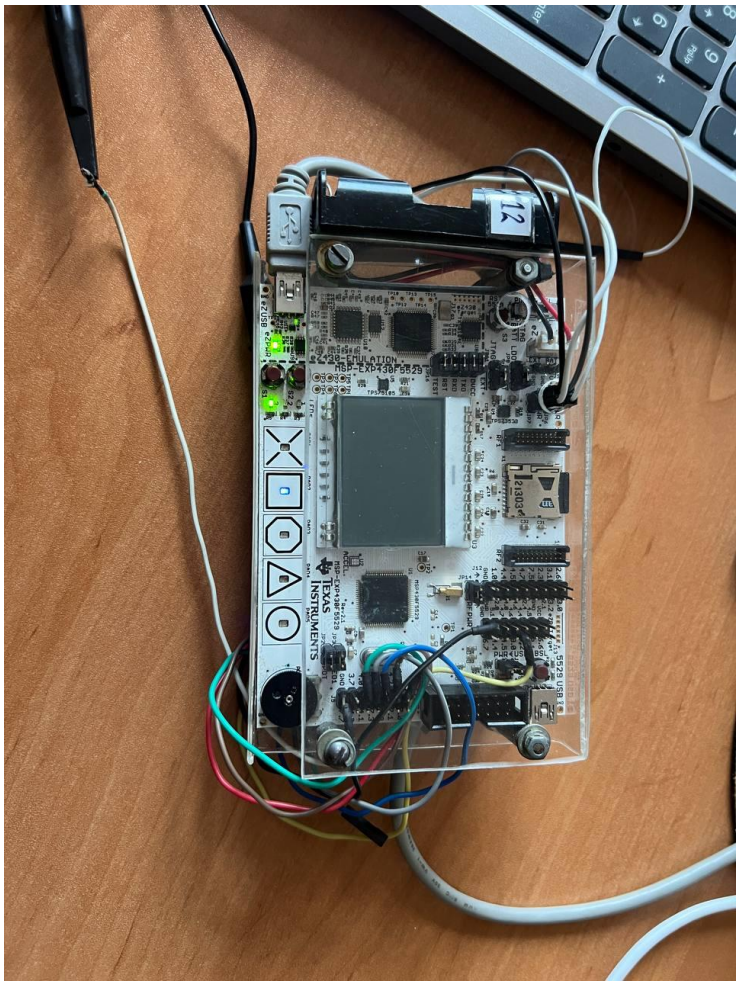
Если микроконтроллер был в режиме пониженного энергопотребления (LPM0), он возвращается в нормальный режим, и загорается светодиод LED1. В это время LED3, индицирующий режим низкого потребления, выключается.

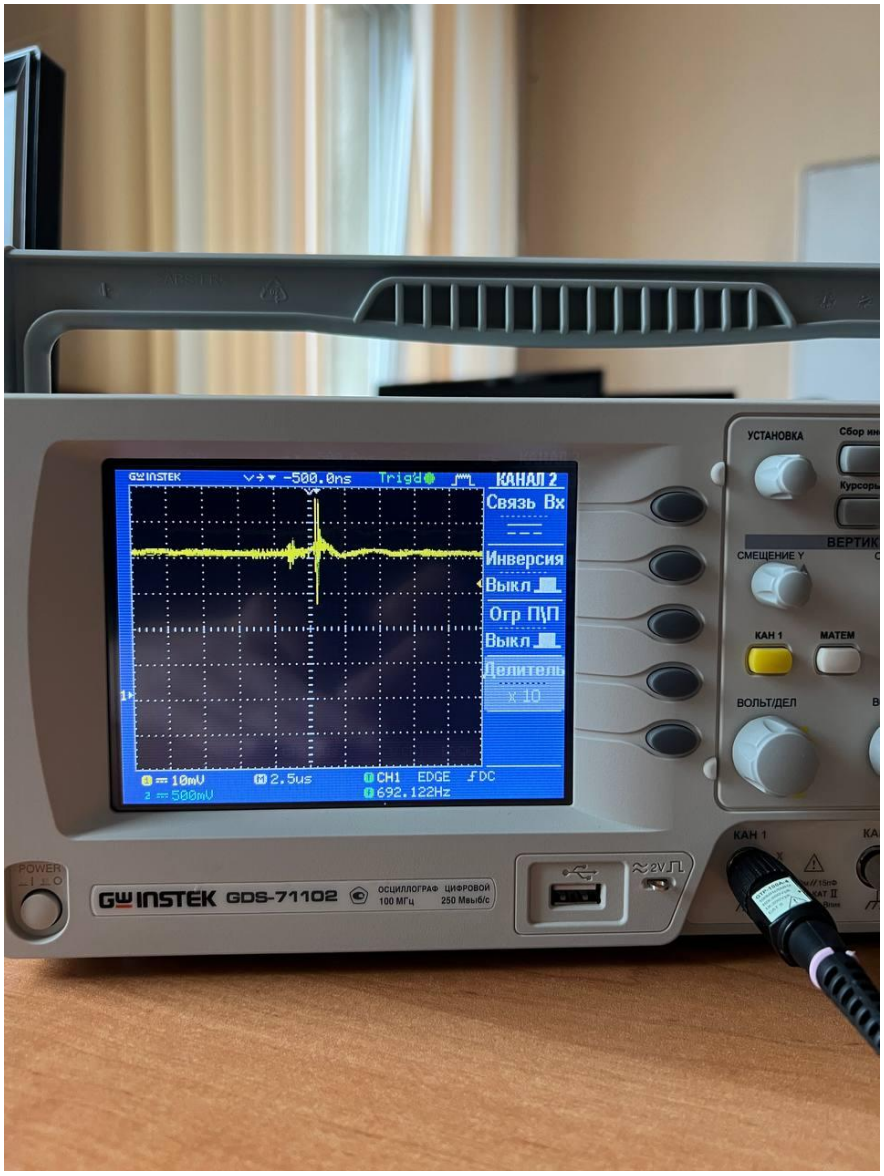
При нажатии второй кнопки (S2) происходит прерывание, в котором программа переключает частоту SMCLK между 1 МГц и 2 МГц.

Результат работы:




КАНАЛ 2
Связь Вх
Инверсия
Выкл
Огр ПП
Выкл
Делитель
GW INSTEK GDS-71102





4. Листинг программы

```
#include <msp430.h>
volatile short CHANGED_FREQUENCY_MODE = 0;
volatile short LOW_POWER_MODE_CORE_0_MODE = 0;
volatile short DEFAULT_DELAY_IN_INTERRUPT = 1000000;
enum Voltage {
 LEVEL0,
 LEVEL1,
 LEVEL2,
 LEVEL3
};
inline void SetVCoreUp(const enum Voltage level) {
 // Open PMM registers for write access
 PMMCTL0_H = PMMPW_H;
 // Set SVS/SVM high side new level
 SVSMHCTL = SVSHE + SVSHRVL0 * level + SVMHE + SVSMHRRL0 * level;
 // Set SVM low side to new level
 SVSMLCTL = SVSLE + SVMLE + SVSMLRRL0 * level;
 // Wait till SVM is settled
 while ((PMMIFG & SVSMLDLYIFG) == 0);
```

```
 // Clear already set flags
 PMMIFG &= ~(SVMLVLRIFG + SVMLIFG);
 // Set VCore to new level
 PMMCTL0_L = PMMCOREV0 * level;
 // Wait till new level reached
 if ((PMMIFG & SVMLIFG))
 while ((PMMIFG & SVMLVLRIFG) == 0);
 // Set SVS/SVM low side to new level
 SVSMLCTL = SVSLE + SVSLRVL0 * level + SVMLE + SVSMLRRL0 * level;
 // Lock PMM registers for write access
 PMMCTL0_H = 0x00;
}
inline void SetVCoreDown(const enum Voltage level) {
 // Open PMM registers for write access
 PMMCTL0_H = PMMPW_H;
 // Set SVS/SVM low side to new level
 SVSMLCTL = SVSLE + SVSLRVL0 * level + SVMLE + SVSMLRRL0 * level;
 // Wait till SVM is settled
 while ((PMMIFG & SVSMLDLYIFG) == 0);
 // Clear already set flags
 PMMIFG &= ~(SVMLVLRIFG + SVMLIFG);
 // Set VCore to new level
 PMMCTL0_L = PMMCOREV0 * level;
 // Wait till new level reached
 if ((PMMIFG & SVMLIFG))
 while ((PMMIFG & SVMLVLRIFG) == 0);
 // Set SVS/SVM high side new level
 SVSMHCTL = SVSHE + SVSHRVL0 * level + SVMHE + SVSMHRRL0 * level;
 // Lock PMM registers for write access
 PMMCTL0_H = 0x00;
}
// Interrupt service routine for button S1
#pragma vector = PORT1_VECTOR
__interrupt void PORT1_S1(void) {
 volatile int i;
 volatile int k = 1;
 for (i = 0; i < DEFAULT_DELAY_IN_INTERRUPT; i++) {
 k++;
 }

 if (LOW_POWER_MODE_CORE_0_MODE) {
 P1OUT |= BIT0;
 P8OUT &= ~BIT2;
 _bic_SR_register_on_exit(LPM0_bits);
 LOW_POWER_MODE_CORE_0_MODE = 0;
 } else {
 P8OUT |= BIT2;
 P1OUT &= ~BIT0;
 LOW_POWER_MODE_CORE_0_MODE = 1;
 _bis_SR_register_on_exit(LPM0_bits);
 }
 P1IFG &= ~BIT7;
}
#pragma vector = PORT2_VECTOR
__interrupt void PORT2_S2(void) {
 volatile int i;
 volatile int k = 1;
 for (i = 0; i < DEFAULT_DELAY_IN_INTERRUPT; i++) {
 k++;
 }
 if (CHANGED_FREQUENCY_MODE) {
 P1OUT |= BIT1;
 P1OUT &= ~BIT2;
```

```
 SetVCoreUp(LEVEL1);
 SetVCoreUp(LEVEL2);
 SetVCoreUp(LEVEL3);
 // Set SMCLK to 1 MHz
 UCSCTL1 = DCORSEL_3; // Select DCO range for 1 MHz
 UCSCTL2 = FLLD__1 | 30; // FLL multiplier for 1 MHz
 CHANGED_FREQUENCY_MODE = 0;
 } else {
 P1OUT |= BIT2;
 P1OUT &= ~BIT1;
 SetVCoreDown(LEVEL2);
 SetVCoreDown(LEVEL1);
 SetVCoreDown(LEVEL0);
 // Set SMCLK to 2 MHz
 UCSCTL1 = DCORSEL_4; // Select DCO range for 2 MHz
 UCSCTL2 = FLLD__1 | 60; // FLL multiplier for 2 MHz
 CHANGED_FREQUENCY_MODE = 1;
 }
 P2IFG &= ~BIT2; // Clear interrupt flag
}
#pragma vector = TIMER0_A0_VECTOR
__interrupt void TIMER0_A0_ISR(void) {
 P1OUT ^= BIT5; // Toggle LED8 (P1.5)
}
int main(void) {
 WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer

 P1DIR &= ~BIT7;
 P1OUT |= BIT7;
 P1REN |= BIT7;
 P2DIR &= ~BIT2;
 P2OUT |= BIT2;
 P2REN |= BIT2;
 P1DIR |= BIT5;
 P1OUT &= ~BIT5;
 // Configure other LEDs
 P1DIR |= BIT0; // LED1
 P1OUT &= ~BIT0;
 P8DIR |= BIT2; // LED3
 P8OUT &= ~BIT2;
 P1DIR |= BIT1; // LED4
 P1OUT &= ~BIT1;
 P1DIR |= BIT2; // LED5
 P1OUT &= ~BIT2;
 __bis_SR_register(GIE);
 // Configure interrupts for buttons S1 and S2
 P1IES |= BIT7; // Interrupt on high-to-low transition
 P1IFG &= ~BIT7; // Clear interrupt flag
 P1IE |= BIT7; // Enable interrupt for S1
 P2IES |= BIT2; // Interrupt on high-to-low transition
 P2IFG &= ~BIT2; // Clear interrupt flag
 P2IE |= BIT2; // Enable interrupt for S2
 // Initialize SMCLK to 1 MHz
 UCSCTL3 |= SELREF__REFOCLK; // Select REFOCLK as reference
 UCSCTL4 = SELM__DCOCLK | SELS__DCOCLK; // Set DCOCLK as source for MCLK
and
SMCLK
 UCSCTL1 = DCORSEL_3; // Set DCO range for 1 MHz
 UCSCTL2 = FLLD__1 | 30; // Set FLL multiplier for 1 MHz
 // Configure timer TA0 for SMCLK
 TA0CCR0 = 50000; // Value to count to (for LED8 blinking)
 TA0CCTL0 |= CCIE; // Enable interrupt for CCR0
 TA0CTL = TASSEL_2 | MC_1 | TACLR; // SMCLK source, Up mode, clear timer
```

```
 return 0;
}
```

5. Заключение

В ходе выполнения лабораторной работы были изучены принципы построения систем с низким энергопотреблением на базе микроконтроллера MSP430F5529. Удалось написать программу, переключающую режимы тактирования, питания и энергопотребления в соответствии с заданием.