|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 8 |

**Название:**

Микроконтроллер AT91SAM7

**Дисциплина:** Микропроцессорные системы.

.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-62Б |  |  | Ашуров Д. Н. Марчук И. С. |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  |  |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2022

**Цель работы:**

* знакомство с архитектурой микроконтроллеров SAM7;
* изучение контроллеров ввода-вывода PIO, прерываний AIC, последовательного канала USART и программных примеров, иллюстрирующих их работу;
* программирование контроллеров.

**Ход работы:**

Схема контроллера PIO представлена на рисунке 1.

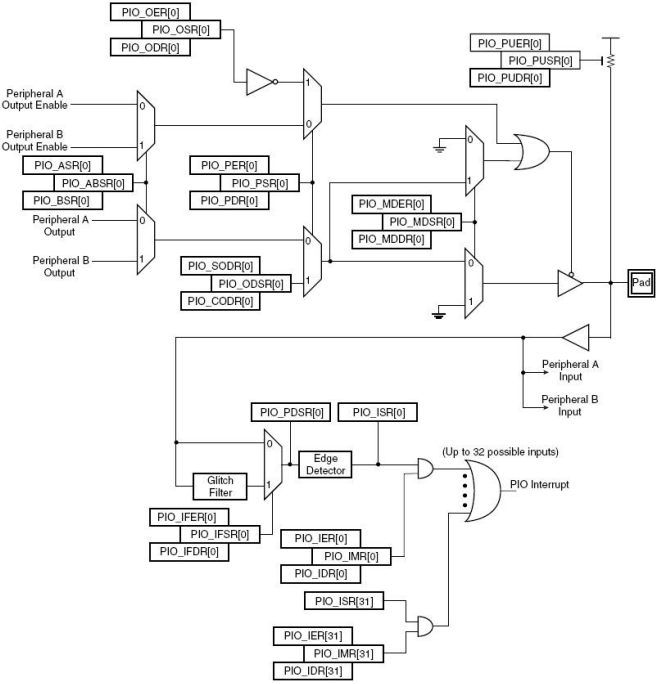


Рисунок 1 – Схема контроллера PIO

**Задание 1.** Изменить программу переключения светодиодов LED1 и LED2 так, чтобы частота переключения LED1 вдвое превышала частоту переключения LED2.

Код исходной программы с измененной частотой переключения светодиодов представлен ниже:

// Мигание индикаторов

#include "AT91SAM7S256.h" // библиотека определений для AT91SAM7S256

#include "SAM7-P256.h" // библиотека определений для периферии SAM7S-P256

void delay (int i); //формирование задержки

static void led\_config (void); //конфигурирование индикаторов

//процедура формирования задержки-------------------------------------------------------//

void delay (int n)

{

int i;

for (i=0; i<=n; i++)

{

asm("nop");

}

}

//предварительная настройка индикаторов-------------------------------------------------//

static void led\_config (void)

{

volatile AT91PS\_PIO pPIO = AT91C\_BASE\_PIOA; //указатель на контроллер ввода/вывода

pPIO->PIO\_PER = LED\_MASK; //разрешение работы индикаторов (выводы 18, 17)

pPIO->PIO\_OER = LED\_MASK; //выводы 18, 17 как выходы

pPIO->PIO\_PPUDR = LED\_MASK; //отключение подтягивающих резисторов выводов 18, 17

pPIO->PIO\_CODR = LED2; //включение индикатора 2 (вывод 17)

pPIO->PIO\_CODR = LED1; //включение индикатора 1 (вывод 18)

}

//главная процедура---------------------------------------------------------------------//

int main()

{

volatile AT91PS\_PIO pPIO = AT91C\_BASE\_PIOA; //указатель на контроллер ввода/вывода

led\_config ();

while (1) //вначале горЯт

{

delay (40000); //выкл1

pPIO->PIO\_SODR = LED1;

delay (40000);

pPIO->PIO\_CODR = LED1; //вкл1

pPIO->PIO\_SODR = LED2; //выкл2

delay (40000);

pPIO->PIO\_SODR = LED1; //выкл1

delay (40000);

pPIO->PIO\_CODR = LED1; //вкл1

pPIO->PIO\_CODR = LED2; //вкл2

}

return 0;

}

**Задание 2.** Изменить программу, включив в нее проверку значения speed и, в случае выхода значения за границы выбранного диапазона, зажигая индикатор LED2.

Код исходной программы с добавленной проверкой на выход за границы частоты представлен ниже:

// Мигание индикаторов с регулировкой скорости мигания

#include "AT91SAM7S256.h" // библиотека определений для AT91SAM7S256

#include "SAM7-P256.h" // библиотека определений для периферии SAM7S-P256

void delay (int i); //формирование задержки

static void wate (void); //процедура проверки нажатия кнопки

static void button\_config (void); //конфигурирование кнопок

static void led\_config (void); //конфигурирование индикаторов

int speed; //величина задержки, для регулирования скорости мигания

//процедура формирования задержки-------------------------------------------------------//

void delay (int n)

{

int i;

for (i=0; i<=n; i++)

{

asm("nop");

}

}

//процедура проверки нажатия клавиши----------------------------------------------------//

static void wate (void)

{

unsigned int pio\_pdsr; //регистр состояния выводов контроллера ввода вывода (PDSR)

volatile AT91PS\_PIO pPIO = AT91C\_BASE\_PIOA;//указатель на контроллер ввода/вывода

pio\_pdsr=pPIO->PIO\_PDSR; //считываем текущие значения со всех выводов PIO0 - PIO32

if (((pio\_pdsr&SW1) == 0) && (speed > 40000)) //проверка нажатия кнопки SW1

{

speed=speed-20000; //если нажата кнопка SW1, то увеличиваем скорость мигания

}

else

if (((pio\_pdsr&SW2) == 0) && (speed < 140000)) //если нажата кнопка SW2, то уменьшаем скорость мигания

{

speed=speed+20000;

}

if ((speed > 139999) || (speed < 40001)) {

pPIO->PIO\_CODR = LED2;

} else {

pPIO->PIO\_SODR = LED2;

}

}

//процедура настройки кнопок------------------------------------------------------------//

static void button\_config (void)

{

volatile AT91PS\_PIO pPIO = AT91C\_BASE\_PIOA; //указатель на контроллер ввода/вывода

volatile AT91PS\_PMC pPMC = AT91C\_BASE\_PMC; //указатель на контроллер управления питанием

pPMC->PMC\_PCER = (1<<AT91C\_ID\_PIOA); //разрешение тактирования контроллера ввода/вывода

pPIO->PIO\_PER = SW\_MASK; //подключение кнопок (выводы 19, 20)

pPIO->PIO\_ODR = SW\_MASK; //выводы 19, 20 предназначены для ввода информации

pPIO->PIO\_IFER = SW\_MASK; //разрешение работы схемы подавления дребезга

}

//процедура настройки индикаторов-------------------------------------------------------//

static void led\_config (void)

{

volatile AT91PS\_PIO pPIO = AT91C\_BASE\_PIOA;//указатель на контроллер ввода/вывода

pPIO->PIO\_PER = LED\_MASK; //подключение индикаторов (выводы 17, 18)

pPIO->PIO\_OER = LED\_MASK; //выводы 17, 18 предназначены для вывода информации

pPIO->PIO\_PPUDR = LED\_MASK; //отключение подтягивающих резисторов

pPIO->PIO\_SODR = LED2; //включение индикатора 2

pPIO->PIO\_SODR = LED1; //выключение индикатора 1

}

//главная процедура---------------------------------------------------------------------//

int main()

{

volatile AT91PS\_PIO pPIO = AT91C\_BASE\_PIOA; //указатель на контроллер ввода/вывода

button\_config (); //настройка кнопок

led\_config (); //настройка индикаторов

speed = 80000; //начальная задержка

while (1) //бесконечный цикл

{

pPIO->PIO\_CODR = LED1; //включение индикатора 1

wate(); //проверка нажатия кнопки

delay (speed); //включение задержки

pPIO->PIO\_SODR = LED1; //выключение индикатора 1

wate(); //проверка нажатия кнопки

delay (speed); //включение задержки

}

return 0;

}

**Задание 3.** Настройте программу для проверки пароля из 8-и символов. Проверьте работу программы.

Код исходной программы с измененной длиной пароля представлен ниже:

// Обмен информацией по USART0

#include "AT91SAM7S256.h" // библиотека определений для AT91SAM7S256

#include "SAM7-P256.h" // библиотека определений для периферии SAM7-P256

#include "usart.h" // библиотека функций для работы с USART

//настройка индикаторов---------------------------------------------------------//

void led\_config(void)

{

AT91PS\_PIO pPIO = AT91C\_BASE\_PIOA; // указатель на контроллер PIO

pPIO->PIO\_PER = LED\_MASK; // подключение индикаторов к PIO

pPIO->PIO\_OER = LED\_MASK; // настройка выводов на вывод

pPIO->PIO\_PPUDR = LED\_MASK; // отключение резисторов

pPIO->PIO\_SODR = LED\_MASK; // выключение индикаторов

}

//процедура формирования задержки-----------------------------------------------//

void delay()

{

int i; // счетчик

for (i = 0; i < 24000000; i++) // цикл задержки

asm("nop");

}

//главная функция---------------------------------------------------------------//

int main()

{

AT91PS\_PIO pPIO = AT91C\_BASE\_PIOA; // указатель на контроллер PIO

AT91PS\_USART pUSART = AT91C\_BASE\_US0; // указатель на USART0

unsigned char passw[]=""; // буфер для вводимого пароля

unsigned char psw\_ok[9]="vsempriv"; // правильный пароль

unsigned int i=0x0; // вспомогательный счетчик

unsigned char ch; // вспомогательная переменная

led\_config(); // настройка индикаторов

InitUSART0();

while (1) // бесконечный цикл

{

i=0x0; // обнуление счетчика

\*passw=""; // очистка буфера

write\_str\_USART0("\nEnter the password:\n"); //вывод приглашения на ввод

ch=read\_char\_USART0(); // чтение первого символа вводимого пароля

while (ch != 0xD) // если не ENTER, то

{

passw[i]=ch; // добавление очередного символа в буфер

write\_char\_USART0(ch); // вывод введенного символа

i++; // переход к следующему символу

ch=read\_char\_USART0(); // чтение следующего символа

}

passw[i]='\0'; // добавление признака конца строки

ch=psw\_ok[i];

while ((passw[i] == ch) & (i != -1)) // проверка пароля

{

i--;

ch=psw\_ok[i];

}

pPIO->PIO\_SODR = LED\_MASK; // выключение индикаторов

if (i == -1) // при совпадении паролей

{

pPIO->PIO\_CODR = LED1; // включить зеленый индикатор

write\_str\_USART0("\nThe password is correct (green LED must be turned on).\n");

}

else // при несовпадении паролей

{

pPIO->PIO\_CODR = LED2; // включить желтый индикатор

write\_str\_USART0("\nError (yellow LED must be turned on).\n");

}

delay(); // задержка перед новой попыткой ввода пароля

}

return 0;

}

Код программы, работающей с прерыванием IRQ, представлен ниже без изменений:

// LED Blink test by Adam Pierce http://www.doctort.org/adam

// This is my first ever ARM program.

// 17-Mar-2007

#include "AT91SAM7S256.h" // Definitions of the ARM chip and on-chip peripherals.

#include "SAM7-P256.h" // Definitions of peripherals on the Olimex dev board.

void init()

{

//MAIN POINTER

AT91PS\_RSTC pRSTC = AT91C\_BASE\_RSTC;

AT91PS\_PMC pPMC = AT91C\_BASE\_PMC;

AT91PS\_USART pUSART = AT91C\_BASE\_US0;

AT91PS\_PDC pPDC = AT91C\_BASE\_PDC\_US0;

AT91PS\_MC pMC = AT91C\_BASE\_MC;

AT91PS\_AIC pAIC = AT91C\_BASE\_AIC;

// Get a pointer to the PIO data structure. The PIO is the "Peripheral input/output

// controller" and is the part of the ARM chip which can access input/output (GPIO) pins.

AT91PS\_PIO pPIO = AT91C\_BASE\_PIOA;

unsigned char i=0;

//Watchdog Disable

AT91C\_BASE\_WDTC->WDTC\_WDMR= AT91C\_WDTC\_WDDIS;

// Reset Disable

pRSTC->RSTC\_RMR=0xFFFF<<8||0xA5<<24;

//Enabling the Main Oscillator:

//SCK = 1/32768 = 30.51 uSecond

//Start up time = 8 \* 6 / SCK = 56 \* 30.51 = 1,46484375 ms

pPMC->PMC\_MOR = (( AT91C\_CKGR\_OSCOUNT & (0x06 <<8) | AT91C\_CKGR\_MOSCEN ));

//Wait the startup time

while(!(pPMC->PMC\_SR & AT91C\_PMC\_MOSCS));

//Setting PLL and divider:

//- div by 5 Fin = 3,6864 =(18,432 / 5)

//- Mul 25+1: Fout = 95,8464 =(3,6864 \*26)

//for 96 MHz the erroe is 0.16%

//Field out NOT USED = 0

//PLLCOUNT pll startup time estimate at : 0.844 ms

//PLLCOUNT 28 = 0.000844 /(1/32768)

pPMC->PMC\_PLLR = ((AT91C\_CKGR\_DIV & 3) | (AT91C\_CKGR\_PLLCOUNT & (28<<8)) | (AT91C\_CKGR\_MUL & (24<<16)));

// Wait the startup time

while(!(pPMC->PMC\_SR & AT91C\_PMC\_LOCK));

while(!(pPMC->PMC\_SR & AT91C\_PMC\_MCKRDY));

//Selection of Master Clock and Processor Clock

//select the PLL clock divided by 2

pPMC->PMC\_MCKR = AT91C\_PMC\_CSS\_PLL\_CLK | AT91C\_PMC\_PRES\_CLK\_32 ;

while(!(pPMC->PMC\_SR & AT91C\_PMC\_MCKRDY));

for (i = 0; i < 8 ; i++) {

pAIC->AIC\_EOICR = 0;

}

return;

}

Код программы, работающей с прерыванием FIQ, представлен ниже без изменений:

// Обработка FIQ прерывания

#include "AT91SAM7S256.h" // библиотека определений для AT91SAM7S256

#include "SAM7-P256.h" // библиотека определений для периферии SAM7S-P256

#include "interrupt\_utils.h" // библиотека для работы с прерываниями

#define ERAM (1) // память RAM

#ifdef ERAM // директивы компилятору, если вызываемая процедура

#define ATTR RAMFUNC // обработки прерывания находится в памяти RAM

#else

#define ATTR

#endif

#if 0

#define IENABLE /\* Nested Interrupts Entry \*/ \

\_\_asm { MRS LR, SPSR } /\* Copy SPSR\_irq to LR \*/ \

\_\_asm { STMFD SP!, {LR} } /\* Save SPSR\_irq \*/ \

\_\_asm { MSR CPSR\_c, #0x1F } /\* Enable IRQ (Sys Mode) \*/ \

\_\_asm { STMFD SP!, {LR} } /\* Save LR \*/

#define IDISABLE /\* Nested Interrupts Exit \*/ \

\_\_asm { LDMFD SP!, {LR} } /\* Restore LR \*/ \

\_\_asm { MSR CPSR\_c, #0x92 } /\* Disable IRQ (IRQ Mode) \*/ \

\_\_asm { LDMFD SP!, {LR} } /\* Restore SPSR\_irq to LR \*/ \

\_\_asm { MSR SPSR\_cxsf, LR } /\* Copy LR to SPSR\_irq \*/

#endif

//обработчик FIQ прерывания-------------------------------------------------------------//

void NACKEDFUNC ATTR fiq\_int (void)

{

AT91PS\_PIO pPIO = AT91C\_BASE\_PIOA; // указатель на контроллер ввода/вывода PIO

ISR\_ENTRY(); // вспомогат. действия при входе в обработчик прерывания

if ((pPIO->PIO\_PDSR & LED1) == 0) // если индикатор LED1 горит,

pPIO->PIO\_SODR = LED1; // то его надо выключить

else // иначе

if ((pPIO->PIO\_PDSR & LED1) == LED1) // если индикатор LED1 не горит,

pPIO->PIO\_CODR = LED1; // то его надо включить

ISR\_EXIT(); // возврат в основную процедуру

}

//предварительная настройка кнопок и контроллера прерываний AIC ------------------------//

static void button\_config(void)

{

volatile AT91PS\_PIO pPIO = AT91C\_BASE\_PIOA;// указатель на контроллер ввода/вывода PIO

volatile AT91PS\_AIC pAIC = AT91C\_BASE\_AIC; // указатель на контроллер прерываний AIC

volatile AT91PS\_PMC pPMC = AT91C\_BASE\_PMC; // указатель на контроллер управления питанием

pPMC->PMC\_PCER = (1<<AT91C\_ID\_PIOA); // разрешение тактирования PIO для работы кнопок

pPIO->PIO\_ODR = SW1\_MASK; // вывод PIN19 работает на ввод (SW1)

pPIO->PIO\_PER = SW1\_MASK; // подключение кнопки SW1

pAIC->AIC\_IDCR = (1<<AT91C\_ID\_FIQ); // запрещение FIQ прерываний

pAIC->AIC\_SVR[AT91C\_ID\_FIQ] = (unsigned long) fiq\_int; // вектор FIQ прерывания

pAIC->AIC\_SMR[AT91C\_ID\_FIQ] = AT91C\_AIC\_SRCTYPE\_EXT\_NEGATIVE\_EDGE | 0; // режим обработки FIQ

pAIC->AIC\_ICCR = (1<<AT91C\_ID\_FIQ); // установка в 0 бита FIQ прерывания

pAIC->AIC\_IECR = (1<<AT91C\_ID\_FIQ); // разрешение FIQ прерываний

}

//предварительная настройка индикаторов-------------------------------------------------//

static void led\_config(void)

{

volatile AT91PS\_PIO pPIO = AT91C\_BASE\_PIOA;// указатель на контроллер ввода/вывода PIO

pPIO->PIO\_OER = LED\_MASK; // выводы 17, 18 как выходы

pPIO->PIO\_PER = LED\_MASK; // подключение индикаторов

pPIO->PIO\_PPUDR = LED\_MASK; // отключение pull-up резисторов для индикаторов

pPIO->PIO\_SODR = LED1; // выключение индикатора LED1

pPIO->PIO\_CODR = LED2; // включение индикатора LED2

}

//главная процедура---------------------------------------------------------------------//

int main()

{

led\_config(); // вызов процедуры настройки индикаторов

button\_config(); // вызов процедуры настройки PIO и AIC

while (1) // бесконечный цикл ожидания прерывания

{

;

}

}

**Выводы:** в результате выполнения лабораторной работы были получены знания об архитектуре микроконтроллеров SAM7, были изучены контроллеры ввода-вывода PIO, прерывания IRQ и FIQ. Также еще раз была проведена работа с последовательным каналом USART.