



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

О Т Ч Е Т

по лабораторной работе № 8

Название: Микроконтроллер AT91SAM7

Дисциплина: Микропроцессорные системы.

Студенты

ИУ6-62Б

(Группа)

(Подпись, дата)

С.В. Астахов

(И.О. Фамилия)

Д.И. Вариханов

(И.О. Фамилия)

(Подпись, дата)

Преподаватель

(Подпись, дата)

(И.О. Фамилия)

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.

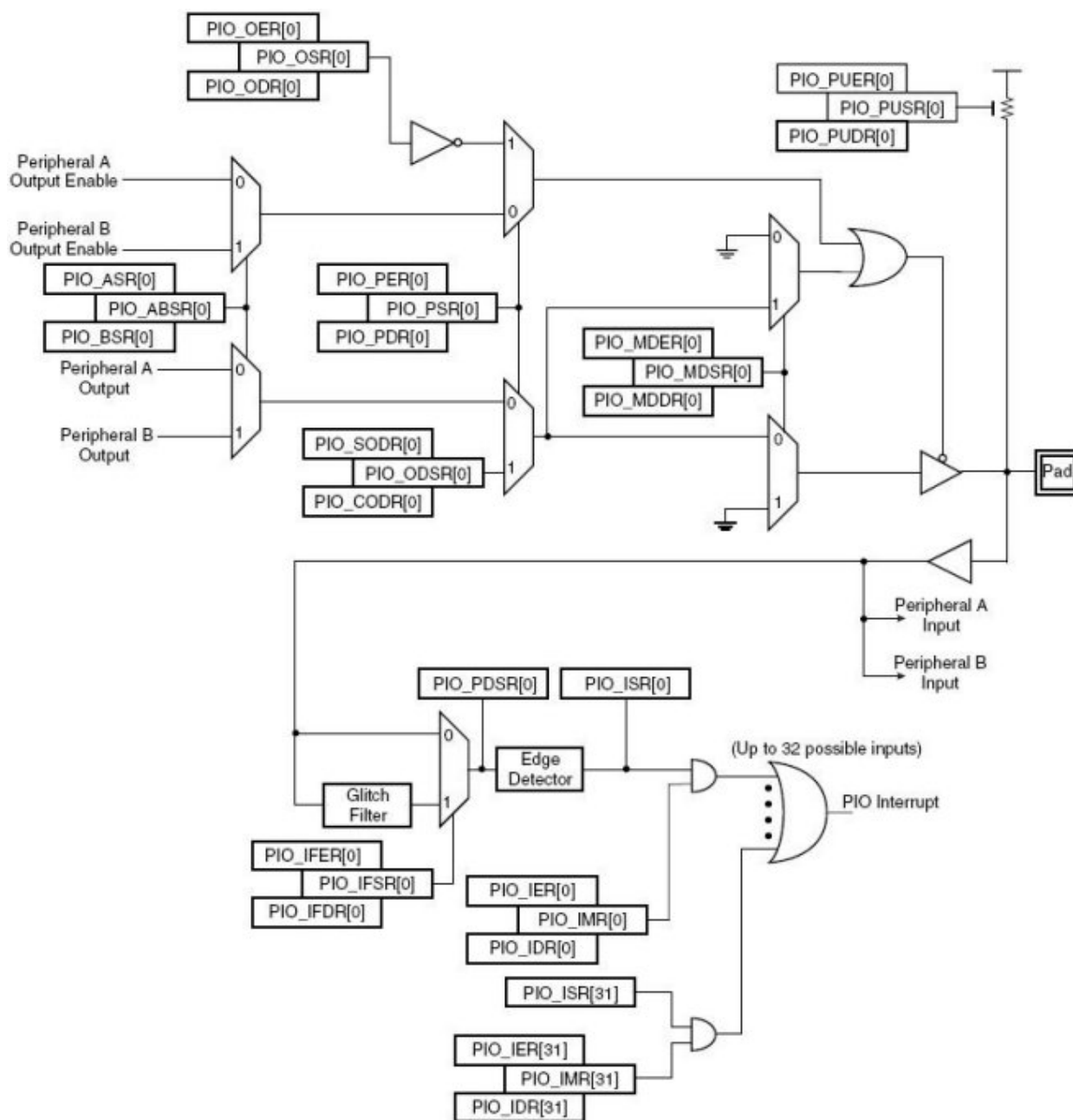


Рисунок 1 – Схема контроллера PIO

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 подавлениядребезга
}

//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_MOSCEEN ));
    //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,                                   // то его надо
        pPIO->PIO_SODR = LED1;                   // выключить
    else                                           //
    иначе                                         //
        if ((pPIO->PIO_PDSR & LED1) == LED1)       // если индикатор
    LED1 не горит,                               // то его надо
        pPIO->PIO_CODR = LED1;                   // 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.