Санкт-Петербургский Политехнический Университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

Отчет по лабораторному практикуму

**Дисциплина:** Аппаратные платформы встраиваемых систем

|  |  |  |
| --- | --- | --- |
| Выполнили студенты гр. 13541/1 | \_\_\_\_\_\_\_\_ | Никитин А.Е. |
|  | (подпись) |  |
|  | \_\_\_\_\_\_\_\_ | Баринов М.С. |
|  | (подпись) |  |
| Руководитель | \_\_\_\_\_\_\_\_ | Васильев А.Е. |
|  | (подпись) |  |
|  |  | «\_\_»\_\_\_\_\_\_ 2019г. |

г. Санкт-Петербург

2019г.

Оглавление

[Лабораторная работа №1 «IAR, CMSIS, SPL, GPIO» 3](#_Toc2556722)

[Цель работы 3](#_Toc2556723)

[Программа работы 3](#_Toc2556724)

[Алгоритм переключения светодиодов 3](#_Toc2556725)

[Ход работы 4](#_Toc2556726)

[Выводы 4](#_Toc2556727)

# **Лабораторная работа №1 «IAR, CMSIS, SPL, GPIO»**

## **Цель работы**

Ознакомиться с интегрированной средой разработки IAR Embedded Workbench for ARM, а также функциями CMSIS и MDRSPL, получить навыки создания и отладки программного обеспечения для целевой платформы на примере разработки программ, взаимодействующих с портами ввода-вывода.

## **Программа работы**

1. Создать проект-заготовку для последующих лабораторных работ. Листинг демонстрационной программы приведен ниже.
2. Подключить к проекту библиотеку CMSIS. Объяснить назначение и содержание файлов библиотеки. Объяснить назначение и содержание файла startup\_MDR32F9Qx.c.
3. Подключить к проекту библиотеку MDR32F9Qx Standart Peripherals Library.
4. Настроить параметры отладчика для запуска демонстрационного примера на отладочной плате. Собрать проект, продемонстрировать его исполнение «по шагам».
5. Разработать программу, включающую светодиоды на плате при нажатии кнопок; алгоритм согласовать с руководителем.

Код демонстрационного примера приведен в листинге 1.

*Листинг 1 ­ Код демонстрационного примера.*

|  |
| --- |
| /\* Includes ------------------------------------------------------------------\*/  #include "MDR32Fx.h"  #include "MDR32F9Qx\_config.h"  #include "MDR32F9Qx\_port.h"  #include "MDR32F9Qx\_rst\_clk.h"  /\* Private typedef -----------------------------------------------------------\*/  /\* Private define ------------------------------------------------------------\*/  #define DELAY\_MIN 250000  #define DELAY\_MAX 500000  #define step 10000  /\* Private macro -------------------------------------------------------------\*/  /\* Private variables ---------------------------------------------------------\*/  /\* Private function prototypes -----------------------------------------------\*/  static void Delay( uint32\_t delay );  static void PeriphInit( void );  static int Button( int state);  /\* Private functions ---------------------------------------------------------\*/  int main()  {    int delay = DELAY\_MAX;  PeriphInit();  while(1)  {  uint8\_t flag = PORT\_ReadInputDataBit(MDR\_PORTC,PORT\_Pin\_2);  uint8\_t up = PORT\_ReadInputDataBit(MDR\_PORTB,PORT\_Pin\_5);  uint8\_t down = PORT\_ReadInputDataBit(MDR\_PORTE,PORT\_Pin\_1);  if (flag == 0) {  delay = DELAY\_MIN;  }  if (up == 0 & delay>step)  delay -= step;  if (down == 0 & delay<DELAY\_MAX\*2)  delay += step;  Delay( delay );  PORT\_SetBits( MDR\_PORTC, PORT\_Pin\_1 );  PORT\_ResetBits( MDR\_PORTC, PORT\_Pin\_0 );  Delay( delay );  PORT\_SetBits( MDR\_PORTC, PORT\_Pin\_0 );  PORT\_ResetBits( MDR\_PORTC, PORT\_Pin\_1 );  }  }  static int upButton (int delay) {  }  static void Delay( uint32\_t delay )  {  while( --delay )  {  \_\_NOP();  }  }  static void PeriphInit( void )  {  PORT\_InitTypeDef PORT\_InitStruct;  /\* Включение тактирования порта C \*/  RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTC | RST\_CLK\_PCLK\_PORTB | RST\_CLK\_PCLK\_PORTE, ENABLE);  /\* Настройка порта C.0 для вывода в дискретном режиме. \*/  PORT\_InitStruct.PORT\_Pin = PORT\_Pin\_1 | PORT\_Pin\_0;  PORT\_InitStruct.PORT\_OE = PORT\_OE\_OUT;  PORT\_InitStruct.PORT\_FUNC = PORT\_FUNC\_PORT;  PORT\_InitStruct.PORT\_MODE = PORT\_MODE\_DIGITAL;  PORT\_InitStruct.PORT\_SPEED = PORT\_SPEED\_SLOW;  PORT\_InitStruct.PORT\_PULL\_UP = PORT\_PULL\_UP\_OFF;  PORT\_InitStruct.PORT\_PULL\_DOWN = PORT\_PULL\_DOWN\_OFF;  PORT\_InitStruct.PORT\_PD\_SHM = PORT\_PD\_SHM\_OFF;  PORT\_InitStruct.PORT\_PD = PORT\_PD\_DRIVER;  PORT\_InitStruct.PORT\_GFEN = PORT\_GFEN\_OFF;  PORT\_Init(MDR\_PORTC, &PORT\_InitStruct);    PORT\_InitStruct.PORT\_Pin = PORT\_Pin\_2;  PORT\_InitStruct.PORT\_OE = PORT\_OE\_IN;  PORT\_Init(MDR\_PORTC, &PORT\_InitStruct);  PORT\_InitStruct.PORT\_Pin = PORT\_Pin\_1;  PORT\_Init(MDR\_PORTE, &PORT\_InitStruct);  PORT\_InitStruct.PORT\_Pin = PORT\_Pin\_5;  PORT\_Init(MDR\_PORTB, &PORT\_InitStruct);    } |

## **Алгоритм переключения светодиодов**

«алгоритм согласовать с руководителем»

Конечный автомат состояний программы представлен на рисунке 1, где st0 – состояние ожидания прерывания, а при переходе в состояние st1 вызывается подпрограмма обработки этого прерывания.

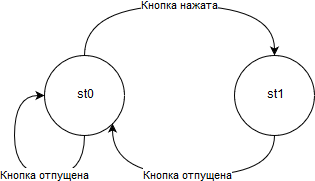


Рисунок 1­ ­­– Конечный автомат состояний программы.

## **Ход работы**

После настройки среды разработки IAR Embedded Workbench for ARM для работы с микросхемой Milandr, подключения необходимых библиотек и запуска демонстрационного проекта, код программы был запущен и протестирован на работоспособность. Затем были внесены изменения в соответствии с заданием преподавателя. Для этого был разработан конечный автомат, схема которого приведена выше.

Код программы, разработанной в соответствии с индивидуальным заданием руководителя приведен в листинге 2.

*Листинг 1 ­ Код демонстрационного примера.*

|  |
| --- |
| /\* Includes ------------------------------------------------------------------\*/  #include "MDR32Fx.h"  #include "MDR32F9Qx\_config.h"  #include "MDR32F9Qx\_port.h"  #include "MDR32F9Qx\_rst\_clk.h"  /\* Private typedef -----------------------------------------------------------\*/  /\* Private define ------------------------------------------------------------\*/  #define DELAY\_MIN 250000  #define DELAY\_MAX 500000  #define step 10000  /\* Private macro -------------------------------------------------------------\*/  /\* Private variables ---------------------------------------------------------\*/  /\* Private function prototypes -----------------------------------------------\*/  static void Delay( uint32\_t delay );  static void PeriphInit( void );  static int Button( int state);  /\* Private functions ---------------------------------------------------------\*/  int main()  {    int delay = DELAY\_MAX;  PeriphInit();  while(1)  {  uint8\_t flag = PORT\_ReadInputDataBit(MDR\_PORTC,PORT\_Pin\_2);  uint8\_t up = PORT\_ReadInputDataBit(MDR\_PORTB,PORT\_Pin\_5);  uint8\_t down = PORT\_ReadInputDataBit(MDR\_PORTE,PORT\_Pin\_1);  if (flag == 0) {  delay = DELAY\_MIN;  }  if (up == 0 & delay>step)  delay -= step;  if (down == 0 & delay<DELAY\_MAX\*2)  delay += step;  Delay( delay );  PORT\_SetBits( MDR\_PORTC, PORT\_Pin\_1 );  PORT\_ResetBits( MDR\_PORTC, PORT\_Pin\_0 );  Delay( delay );  PORT\_SetBits( MDR\_PORTC, PORT\_Pin\_0 );  PORT\_ResetBits( MDR\_PORTC, PORT\_Pin\_1 );  }  }  static int upButton (int delay) {  }  static void Delay( uint32\_t delay )  {  while( --delay )  {  \_\_NOP();  }  }  static void PeriphInit( void )  {  PORT\_InitTypeDef PORT\_InitStruct;  /\* Âêëþ÷åíèå òàêòèðîâàíèÿ ïîðòà C \*/  RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTC | RST\_CLK\_PCLK\_PORTB | RST\_CLK\_PCLK\_PORTE, ENABLE);  /\* Íàñòðîéêà ïîðòà C.0 äëÿ âûâîäà â äèñêðåòíîì ðåæèìå. \*/  PORT\_InitStruct.PORT\_Pin = PORT\_Pin\_1 | PORT\_Pin\_0;  PORT\_InitStruct.PORT\_OE = PORT\_OE\_OUT;  PORT\_InitStruct.PORT\_FUNC = PORT\_FUNC\_PORT;  PORT\_InitStruct.PORT\_MODE = PORT\_MODE\_DIGITAL;  PORT\_InitStruct.PORT\_SPEED = PORT\_SPEED\_SLOW;  PORT\_InitStruct.PORT\_PULL\_UP = PORT\_PULL\_UP\_OFF;  PORT\_InitStruct.PORT\_PULL\_DOWN = PORT\_PULL\_DOWN\_OFF;  PORT\_InitStruct.PORT\_PD\_SHM = PORT\_PD\_SHM\_OFF;  PORT\_InitStruct.PORT\_PD = PORT\_PD\_DRIVER;  PORT\_InitStruct.PORT\_GFEN = PORT\_GFEN\_OFF;  PORT\_Init(MDR\_PORTC, &PORT\_InitStruct);    PORT\_InitStruct.PORT\_Pin = PORT\_Pin\_2;  PORT\_InitStruct.PORT\_OE = PORT\_OE\_IN;  PORT\_Init(MDR\_PORTC, &PORT\_InitStruct);  PORT\_InitStruct.PORT\_Pin = PORT\_Pin\_1;  PORT\_Init(MDR\_PORTE, &PORT\_InitStruct);  PORT\_InitStruct.PORT\_Pin = PORT\_Pin\_5;  PORT\_Init(MDR\_PORTB, &PORT\_InitStruct);    }  #if ( USE\_ASSERT\_INFO == 1 )  void assert\_failed(uint32\_t file\_id, uint32\_t line)  {  while (1)  {  }  }  #elif ( USE\_ASSERT\_INFO == 2 )  void assert\_failed(uint32\_t file\_id, uint32\_t line, const uint8\_t\* expr)  {  while (1)  {  }  }  #endif /\* USE\_ASSERT\_INFO \*/  /\* END OF FILE main.c \*/ |

## **Выводы**

По итогам лабораторной работы было произведено ознакомление с интегрированной средой разработки IAR Embedded Workbench for ARM, а также функциями CMSIS и MDRSPL. Также были получены навыки создания и отладки программного обеспечения для целевой платформы на примере разработки программ, взаимодействующих с портами ввода-вывода.