**Лабораторная работа 1**

**Работа с GPIO: создание первого проекта**

Мощевикин А.П., ПетрГУ

***Общие сведения о портах ввода-вывода в Комдив-МК***

Микроконтроллер Комдив-МК содержит контроллер ввода-вывода, управляющий работой 6 портов, в каждом по 8 бит: GP\_A [7:0], GP\_B [7:0], GP\_C [7:0], GP\_D [7:0], GP\_E [7:0] и GP\_F [7:0], всего 48 линий. Под регистры каждого из 6 портов выделено адресное пространство в 128 (0x80) адресов.

Файл ./psp\_mc/arch/kmk/kmk.uhe определяет базовые (начальные) адреса каждого порта, как:

#define GPIOA\_BASE 0xbb400200

#define GPIOB\_BASE 0xbb400280

#define GPIOC\_BASE 0xbb400300

#define GPIOD\_BASE 0xbb400380

#define GPIOE\_BASE 0xbb400400

#define GPIOF\_BASE 0xbb400480

Как указано в [1], по любой линии любого порта можно выставить и считать цифровой сигнал «0» или «1», настроить прерывания по уровню или изменению состояния входного сигнала, а также зафиксировать время этого события.

Многие линии портов ввода-вывода имеют дополнительное (альтернативное) назначение.

Перед работой (в начале программы) необходимо сконфигурировать используемые линии.

По таблице 2.7.1 Мультиплицирование выводов на кристалле [1] необходимо выбрать назначение линии, и внести номер столбика в биты 4..2 регистра конфигурации этой линии conf0-conf7 (смещение в диапазоне 0x08..0x0f относительно базового адреса выбранного порта, по регистру на каждую линию порта, 0x08 для разряда 0, 0x0f для разряда 7). Если выбрано назначение линии, как порта ввода-вывода (GPIO, General Purpose Input Output), то значения всех трех битов должны быть равны нулю.

Также в режиме GPIO необходимо в битах 1..0 этого же регистра указать режим работы линии. Она может работать (1) на выход, (2) на вход и (3) на вход с подтяжкой напряжения наверх к питанию (pullup). Режим (2) или режим (3) выбирается в зависимости от примененных схемотехнических решений и типов электронных компонентов на плате.

Например, на плате БАГЕТ-ПЛК1-01 размещена пользовательская кнопка SW2. Она в момент нажатия замыкает вывод микроконтроллера GPIO\_D6 на землю. Если необходимо считывать нажатие, то линию необходимо конфигурировать, как вход с подтяжкой наверх. Указание использовать принудительную программную подтяжку наверх подключает встроенный в Комдив-МК резистор между линией порта ввода-вывода и питанием. Таким образом, после отпускания кнопки резистор быстро вернет высокий уровень напряжения на линии. Если подтяжку не включать, то напряжение будет подниматься наверх медленно, и в какой-то момент при считывании напряжения на линии микроконтроллер не будет понимать, «0» на входе или «1».

Для работы с портами ввода-вывода в Комдив-МК есть 6 основных регистров:

ODR (output data register, смещение 0x00) – регистр выходных данных,

BSR (bit set register, смещение 0x01) – регистр выставления индивидуальных битов,

BRR (bit reset register, смещение 0x02) – регистр снятия индивидуальных битов,

BIR (bit inverse register, смещение 0x03) – регистр инверсии индивидуальных битов,

IDR (input data register, смещение 0x04) – регистр входных данных,

ISR (interrupt status register, смещение 0x05) – регистр состояния прерываний.

Если линии микроконтроллера сконфигурированы «на выход», то выставить значения логических нулей или единиц можно двумя основными способами.

1. Запись целым байтом. Для этого следует использовать регистр ODR.
2. Запись побитно. В случае необходимости выставить или сбросить только некоторые биты, их маска записывается в регистры BSR или BRR, соответственно. Например, если необходимо принудительно обнулить линии GPIO\_PD0 и GPIO\_PD6, в регистр BRR порта D необходимо записать значение 65 (0b01000001, 0x41). Если необходимо инвертировать значение бита(ов), то маска записывается в регистр BIR соответствующего порта. При записи в регистры BSR, BRR, BIR автоматически происходит изменение содержимого регистра ODR.

Если линия микроконтроллера сконфигурирована «на вход», то узнать состояние линии можно, считав значение регистра IDR. Далее, применив побитную маску, можно узнать значение конкретного бита порта.

Смещения для регистров указаны относительно базового адреса порта. Например, для порта D адрес регистра BIR в общем линейном пространстве будет 0xbb400383.

Все операции с портами ввода-вывода (равно как и с другой периферией в Комдив-МК) ведутся с помощью обращения по адресам в общем линейном пространстве. Таким образом, чтобы инвертировать значение старшего бита порта D, необходимо по адресу 0xbb400383 (адрес регистра IDR порта D) записать число 128 (0b10000000, 0x80).

Для создания встроенного программного обеспечения для платы БАГЕТ-ПЛК1-01 используют два альтернативных подхода: разработка программ в стиле Arduino на Си-подобном языке с компоновкой программы в блоках setup() и loop() и разработка программ на стандартизованном языке Си.

Разработчики платы БАГЕТ-ПЛК1-01 рекомендуют использовать среду разработки и структуру проекта, описанные в [2] и [3, раздел 3].

Проекты Arduino следует создавать в папке ./arduino/apps/ (внутри пакета поддержки программирования микроконтроллера, ППП МК), а проекты на языке Cи – в папке ./psp\_mc/apps/ .

***Arduino-проекты***

Для Arduino-проектов перечень режимов работы линии порта ввода-вывода указан в файле ./arduino/arduino\_core/Arduino.h

#define INPUT 0x0

#define OUTPUT 0x1

#define INPUT\_PULLUP 0x2

Исходный код функций работы с портами приведен в файле ./arduino/arduino\_core/wiring\_dgital.c.

Функция pinMode() используется для конфигурирования линии, digitalWrite() – для записи значения, digitalRead() – для чтения.

Например, чтобы заставить светиться зеленый светодиод VD2 на плате БАГЕТ-ПЛК1-01, необходимо сначала установить линию 4 порта D на выход, затем выставить на этой линии «1».

pinMode(PIN\_PD4, OUTPUT);

digitalWrite(PIN\_PD4, 1);

Поскольку на плате БАГЕТ-ПЛК1-01 установлены разъемы форм-фактора Arduino (см. рисунки 1 и 2), разработчики ППП МК постарались сохранить преемственность именования линий ввода-вывода в Arduino-стиле (см. файл ./arduino/arduino\_core/pins\_arduino.h).



Рисунок 1 – Плата БАГЕТ-ПЛК1-01 с разъемами J1-J4 интерфейса Arduino UNO R3 [4].



Рисунок 2 – Плата Arduino UNO R3 с разъемами для подключения цифровых и аналоговых линий.

Так, в ППП МК цифровые линии на разъемах J1-J4 интерфейса Arduino UNO R3 (Digital Pins) также имеют численные значения от 0 до 13. Численные значения от 14 до 19 – это идентификаторы линий Analog Pins A0..A6.

Линии порта А (PIN\_PA0.. PIN\_PA7) – значения в диапазоне 32..37.

Линии порта B (PIN\_PB0.. PIN\_PB7) – значения в диапазоне 64..71.

Линии порта C (PIN\_PC0.. PIN\_PC7) – значения в диапазоне 96..103.

Линии порта D (PIN\_PD0.. PIN\_PD7) – значения в диапазоне 128..135.

Линии порта E (PIN\_PE0.. PIN\_PE7) – значения в диапазоне 160..167.

Линии порта F (PIN\_PF0.. PIN\_PF7) – значения в диапазоне 192..197.

Таким образом, три старших бита в байте номера линии отвечают за номер порта (когда все три бита равны нулю – это Arduino UNO), а младшие биты – за номер линии. На всех портах, кроме Arduino – по 8 линий, на Arduino-порту – 20 линий (14 цифровых и 6 аналоговых).

При создании проектов в Arduino-стиле к линиям интерфейса Arduino UNO R3 следует обращаться, используя число. Например, 15 – это линия A1. К другим портам на плате БАГЕТ-ПЛК1-01 необходимо обращаться, используя шаблон PIN\_PXY, где X – это буква порта (A..F), а Y – номер бита. Например, PIN\_PD4 – линия 4 порта D.

***Проекты на языке Си***

Для проектов на языке Cи макросы определяют сразу и назначение, и режим работы линии в файле ./psp\_mc/arch/kmk/kmk\_common.h

#define GPIO\_MODE\_INPUT 0

#define GPIO\_MODE\_INPUT\_PULLUP 1

#define GPIO\_MODE\_OUTPUT 2

#define GPIO\_MODE\_ALT1 7

#define GPIO\_MODE\_ALT2 0xB

#define GPIO\_MODE\_ALT3 0xF

В файле ./psp\_mc/arch/kmk/kmk\_common.h также определены следующие макросы. GPIO\_SetPinMode() – для конфигурирования линии, GPIO\_WritePin() – для записи, GPIO\_ReadPin() – для чтения.

Тот же самый пример. Чтобы заставить светиться зеленый светодиод VD2, необходимо сначала установить линию 4 порта D на выход, затем выставить на этой линии «1».

GPIO\_SetPinMode(GPIOD\_BASE, 4, GPIO\_MODE\_OUTPUT);

GPIO\_WritePin(GPIOD\_BASE, 4, 1);

Макрос GPIO\_SetPinMode определен как:

#define GPIO\_SetPinMode(GPIO, pin, mode) \

(\*(((volatile unsigned char \*)(GPIO)) + GPIO\_MODE\_REG\_OFFS + pin) = (mode))

Для примера выше он работает следующим образом. Перед компиляцией проекта идет обработка текста программ препроцессором. Вызов GPIO\_SetPinMode(GPIOD\_BASE, 4, GPIO\_MODE\_OUTPUT) будет преобразован в …

\*(((volatile unsigned char \*)(0xbb400380)) + 8 + 4) = 2

… поскольку GPIO\_MODE\_OUTPUT – это 2, GPIO\_MODE\_REG\_OFFS определено, как

#define GPIO\_MODE\_REG\_OFFS 8

Следуя самым внутренним скобкам в этом выражении, сначала указывается, что по адресу памяти будет храниться целочисленное однобайтовое беззнаковое значение. Вставляя модификатор volatile, программист явно указывает компилятору не оптимизировать размещение этой переменной в памяти, не пытаться, например, делать из нее регистровую переменную. Процессорное ядро обязательно должно обратиться к памяти по этому адресу: только так дело дойдет до аппаратуры (контроллера портов-ввода вывода).

Далее к адресу 0xbb400380 прибавляется смещение 12. Результат – 0xbb40038С. А это адрес регистра конфигурации conf0-conf7 порта D. В регистр конфигурации порта D будет записано 2, т.е. указание настроить линию на выход.

В последней строчке примера выше макрос GPIO\_WritePin записывает «1» в разряд 4 регистра BSR порта D, тем самым на линии PD4 выставляется высокий уровень потенциала, что заставляет светодиод VD2 светиться.

***Создание Arduino-проекта***

В качестве примера создадим простой Arduino-проект (заготовку для более сложных проектов), в котором после инициализации микроконтроллера в терминале появляется текст «Init OK» и начинает мигать зеленый светодиод VD2 с частотой 1 Гц.

Создание и редактирование проекта удобно проводить в редакторе VS Code.

Основой для простого проекта будет служить проект strtest, который содержится в папке ./arduino/apps/strtest/ пакета поддержки программирования микроконтроллера Комдив-МК.

Все дальнейшие инструкции подразумевают, что работа выполняется в ОС Windows с предустановленной виртуальной машиной WSL2 с Debian и ППП МК.

*Шаг 1 – Создание проекта*

В Windows запустите терминал PowerShell, откройте консоль Debian. Далее в командной строке консоли Debian запустите VS Code, выполнив команду:

code .

В редакторе VS Code в панели со списком файлов кликните ПКМ на папке apps и выберите «Создать папку». Введите имя, например, blank. Скопируйте оба файла (app.cpp и Makefile) из папки strtest в папку blank.

Кликните ПКМ на папке blank и выберите «Создать файл». Введите название kmk.sh и вставьте в этот файл следующее содержимое.

#!/bin/sh

BAREBOXIMG=$1

if [ ! -e "$BAREBOXIMG" ]; then

 echo "usage:"

 echo " $0 <barebox-image>"

 exit 1

fi

cat > /tmp/layout <<EOF

00000000:0007ffff barebox

EOF

dd if=/dev/zero ibs=1M count=4 | tr "\000" "\377" > /tmp/padded.img

dd if=$BAREBOXIMG of=/tmp/padded.img conv=notrunc

flashrom -p ft2232\_spi:type=kmk -c W25Q32BV/W25Q32CV/W25Q32DV -w /tmp/padded.img

Сохраните файл, нажав Ctrl-S.

Чтобы сделать файл kmk.sh исполняемым, в терминале PowerShell в консоли Debian необходимо с помощью команд cd <название каталога> и ls –la (вывести содержимое каталога) переместиться в папку ./arduino/apps/blank/. Далее выполнить команду

chmod +x kmk.sh

После этого выполнить ls –la и убедиться, что в каталоге blank содержится три файла, и атрибуты файла kmk.sh соответствуют представленным на рисунке 3, слева (-rwxr-xr-x). На рисунке 3 справа приведена структура проекта в VS Code.

 

Рисунок 3 – Список файлов c их атрибутами в каталоге blank (слева), структура проекта в редакторе VS Code (справа).

*Шаг 2 – Редактирование кода*

Файл app.cpp

|  |  |
| --- | --- |
| БЫЛО: | СТАЛО: |
| #include <Arduino.h>String testString, sARM, sMIPS;void setup(){ Serial.begin(115200); testString = "Test ARM is so ARM as ARM. Let's test this ARM"; sARM = "ARM"; sMIPS = "MIPS";}void loop(){ testString.replace(sARM, sMIPS); Serial.println(testString); delay(1000); testString.replace(sMIPS, sARM); Serial.println(testString); delay(1000);} | #include <Arduino.h>void setup(){ pinMode(PIN\_PD4, OUTPUT); // led VD2 Serial.begin(115200); Serial.println("\n\rInit OK");}void loop(){ digitalWrite(PIN\_PD4, 1); delay(500); // 500 ms digitalWrite(PIN\_PD4, 0); delay(500);} |

Файл Makefile

|  |  |
| --- | --- |
| БЫЛО: | СТАЛО: |
| PSP\_MK\_ROOT = ../../../psp\_mcAPP\_ROOT = .BARE = ../../bareDUINO\_CORE = ../../arduino\_coreDUINO\_LIBS = ../../librariesLOCAL\_INCLUDE\_DIRS = $(BARE) \ $(BARE)/system/libc \ $(DUINO\_CORE) \ $(DUINO\_CORE)/compiler ~~\~~ ~~$(DUINO\_LIBS)/EtherCard/src \~~ ~~$(DUINO\_LIBS)/SPI/src~~ LOCAL\_LIB\_DIRS = $(DUINO\_CORE)CLEANFILES =CFLAGS = -DARDUINO\_KOMDIVDUINO=1ASMFLAGS = LDFLAGS = LDSCRIPT = $(BARE)/kmk.ldsSRC = app.cppSRC += $(BARE)/board.cSRC += $(BARE)/kmk\_irqs.SSRC += $(BARE)/random\_r.cSRC += $(BARE)/random.cSRC += $(DUINO\_CORE)/heap/malloc.cSRC += $(DUINO\_CORE)/heap/realloc.cinclude $(PSP\_MK\_ROOT)/arch/kmk/Makefile\_kmk\_application.mk | PSP\_MK\_ROOT = ../../../psp\_mcAPP\_ROOT = .BARE = ../../bareDUINO\_CORE = ../../arduino\_coreDUINO\_LIBS = ../../librariesLOCAL\_INCLUDE\_DIRS = $(BARE) \ $(BARE)/system/libc \ $(DUINO\_CORE) \ $(DUINO\_CORE)/compiler LOCAL\_LIB\_DIRS = $(DUINO\_CORE)CLEANFILES =CFLAGS = -DARDUINO\_KOMDIVDUINO=1ASMFLAGS = LDFLAGS = LDSCRIPT = $(BARE)/kmk.ldsSRC = app.cppSRC += $(BARE)/board.cSRC += $(BARE)/kmk\_irqs.SSRC += $(BARE)/random\_r.cSRC += $(BARE)/random.cSRC += $(DUINO\_CORE)/heap/malloc.cSRC += $(DUINO\_CORE)/heap/realloc.cinclude $(PSP\_MK\_ROOT)/arch/kmk/Makefile\_kmk\_application.mk |

Сохраните все файлы, нажав Ctrl-S в каждой вкладке.

*Шаг 3 – Компиляция, прошивка и запуск*

Для программирования платы БАГЕТ-ПЛК1-01 в режим baremetal перемычки SA4-SA7 должны быть установлены, как показано на рисунке 4 (перемычки SA5-7 устанавливаются в левое положение, перемычка SA4 – в правое положение INT RAM). Установку и снятие перемычек требуется выполнять в обесточенном состоянии.



Рисунок 4 – Положение перемычек SA4-SA7 для работы в режиме baremetal без встроенной ОС Debian (Ethernet, EXT ROM, INT RAM, LITTLE ENDIAN).

Выполнить «проброс» USB-порта в WSL2 Debian. Для этого воспользоваться командами usbipd list и usbipd attach --wsl --busid 3-9 (вместо цифр 3-9 подставить необходимое) [2]. Внимание: эти команды запускаются в командной строке Windows (а не в терминале PowerShell Debian).

Для удобства отладки можно создать два терминальных окна в VS Code: для ввода команд в Debian и для консоли платы БАГЕТ-ПЛК1-01 (см. рисунок 5).



Рисунок 5 – редактор VS Code c двумя окнами терминала: (1) – вкладка «Terminal», (2) – добавление окна терминала, (3) – список окон, (4) – окно с командной строкой Debian, (5) – окно для запуска minicom.

Нажать на вкладку Terminal (1). Добавить еще один терминал, кликнув на значок «+» (2). В окне (3), удерживая Ctrl, выбрать оба терминала. Нажать ПКМ на одном из выбранных терминалов. В контекстном меню выбрать «Join Terminals». После этого на экране будут доступны оба терминала одновременно, в окнах (4) и (5).

Подключить питание к плате БАГЕТ-ПЛК1-01 и соединить ее кабелем USB-C с компьютером, предварительно установив перемычки.

В терминале (4), находясь внутри папки проекта (./arduino/apps/blank/), провести сборку проекта и прошивку ПЗУ платы:

make && sudo ./kmk.sh ram.bin

В терминале (5) выполнить

sudo minicom -D /dev/ttyUSB1

Примечание: выход из терминала minicom – по нажатию Ctrl-A + Q.

Нажать кнопку сброса RESET на плате БАГЕТ-ПЛК1-01 и убедиться, что зеленый светодиод мигает, и что в окне (5) появилась надпись «Init OK».

***Создание проекта на языке Си***

В качестве примера создадим простой проект на языке Си (заготовку для более сложных проектов), в котором после инициализации микроконтроллера в терминале появляется текст «1 MHz oscillation, Init OK» и начинает мигать зеленый светодиод VD2 с максимально возможной частотой (ожидаемая частота выше 1 МГц; при частотах выше 24 Гц будет казаться, что светодиод горит непрерывно).

Создание и редактирование проекта удобно проводить в редакторе VS Code.

Часть файлов для простого проекта будет взята из проекта demo\_pca, который содержится в папке ./psp\_mc/apps/demo\_pca/ пакета поддержки программирования микроконтроллера Комдив-МК.

Все дальнейшие инструкции подразумевают, что работа выполняется в ОС Windows с предустановленной виртуальной машиной WSL2 с Debian и ППП МК.

*Шаг 1 – Создание проекта*

В Windows запустите терминал PowerShell, откройте консоль Debian. Далее в командной строке консоли Debian запустите VS Code, выполнив команду:

code .

В редакторе VS Code в панели со списком файлов кликните ПКМ на папке apps и выберите «Создать папку». Введите имя, например, blank.

По рекомендации разработчиков ППП МК в папке blank нужно создать две подпапки – komdiv\_board и src. И, соответственно, распределить код создаваемого приложения между ними. Поскольку наш проект достаточно маленький и очень платформенно-зависимый, папку src создавать излишне.

Итак, в папке blank создайте только одну подпапку komdiv\_board.

В папку komdiv\_board скопируйте следующие файлы из проекта demo\_pca:

config.h

kmk\_irqs.S

main.c

Makefile

К этим четырем файлам добавьте в эту же папку ранее созданный файл kmk.sh, с помощью которого передается скомпилированный файл образа ПЗУ в плату БАГЕТ-ПЛК1-01.

*Шаг 2 – Редактирование кода*

В файле Makefile удалите строчку

SRC += app/app.c

Эта строка лишняя, поскольку файла app.c в проекте нет.

В соответствии с инструкциями в [4, раздел 5.1] и [5, раздел 3.6.1] в файл config.h вставьте строчку

#define \_\_ARCH\_WITH\_SERIAL 1

Замените содержимое файла main.c на следующее.

#include "arch.h"

#include "kmk\_common.h"

#include <string.h>

void \_\_print\_console(const char \*pcStr);

void \_\_print\_console(const char \*pcStr)

{

 void \*pS = \_\_system\_serial\_get(0);

 if (pS) {

 unsigned uLen = strlen(pcStr);

 if (\_\_system\_serial\_write(pS, (unsigned char\*)pcStr, &uLen) == 1) {

 while(!\_\_system\_serial\_ready(pS, 2)) {}; //2=SERIAL\_OPERATION\_TX

 } else { // unfortutately, no chance to write, means serial is busy.

 }

 }

}

static inline int init\_board(void)

{

 GPIO\_SetPinMode(GPIOD\_BASE, 4, GPIO\_MODE\_OUTPUT); // RUN LED

 kmk\_serial\_init(0, KMK\_ADDR(UART0\_BASE), 0, 0, 0, 0, 0, 0, 0, 0);

 \_\_system\_serial\_configure(\_\_system\_serial\_get(0), 115200, 8, 0, 1);

 \_\_print\_console("\r\n1 MHz oscillation, Init OK");

 return 1;

}

int main(void)

{

 init\_board();

 while (1) {

 GPIO\_WritePin(GPIOD\_BASE, 4, 1);

 GPIO\_WritePin(GPIOD\_BASE, 4, 0);

 }

 return 1; // unreacheable

}

Функция \_\_print\_console() используется для вывода строк в консоль.

В основном теле программы сначала вызывается функция init\_board(), инициализирующая линию GPIO\_PD4 на выход, затем выполняется инициализация и конфигурация консоли, потом в нее посылается текстовая строка.

Затем запускается бесконечный цикл, в котором на линию GPIO\_PD4 выставляется то «1», то «0».

Конфигурация и принципы работы с консолью описаны в [5], раздел 3.6.

Сохраните все файлы, нажав Ctrl-S в каждой вкладке.

*Шаг 3 – Компиляция, прошивка и запуск*

Для программирования платы БАГЕТ-ПЛК1-01 в режим baremetal перемычки SA4-SA7 должны быть установлены, как показано на рисунке 4 (перемычки SA5-7 устанавливаются в левое положение, перемычка SA4 – в правое положение INT RAM). Установку и снятие перемычек требуется выполнять в обесточенном состоянии.

Выполнить «проброс» USB-порта в WSL2 Debian и организовать два терминальных окна: для ввода команд в Debian и для консоли платы БАГЕТ-ПЛК1-01 (см. описание действий выше и рисунок 5).

В терминале (4), находясь внутри папки проекта (./psp\_mc/apps/blank/komdiv\_board/), провести сборку проекта и прошивку ПЗУ платы:

make && sudo ./kmk.sh ram.bin

В терминале (5) выполнить

sudo minicom -D /dev/ttyUSB1

Примечание: выход из терминала minicom – по нажатию Ctrl-A + Q.

Нажать кнопку сброса RESET на плате БАГЕТ-ПЛК1-01 и убедиться, что зеленый светодиод горит, и что в окне (5) появилась надпись «1 MHz oscillation, Init OK».

***Задания для самостоятельной работы***

1. Переработать Arduino-проект таким образом, что сразу после старта программа ожидает нажатия кнопки SW2 на плате БАГЕТ-ПЛК1-01, а затем переходит в бесконечный цикл мигания зеленым и желтым светодиодами. Когда горит один, другой – не горит.
2. Переработать проект на Си следующим образом. Необходимо измерить и вывести минимальное время, за которое микроконтроллер может сформировать 1 миллион импульсов. Импульс – это последовательное выставление на линии порта ввода-вывода сначала «1», потом «0». Для смены состояния линии с «1» в «0» и обратно рекомендуется использовать регистр инверсии бита BIR (необходимо учесть, что макрос для определения смещения этого регистра в файле kmk\_common.h не создан). Для замера времени можно воспользоваться макросом \_\_micros() (см. файл uke.h), а для вывода результата в консоль – функцией \_\_print\_console(). Необходимо также создать функцию перевода числа микросекунд в строку, либо воспользоваться сторонней библиотекой, содержащей функцию типа itoa().

***Список литературы***

[1] Микросхема интегральная К5500ВК018. Контроллер GPIO. ЮКСУ.431295.019Д4.19 // К5500ВК018\_d4\_19\_GPIO\_v.2.4.2.doc

[2] Мощевикин А.П., Голяков М.А. Установка среды разработки в Windows 10 для БАГЕТ-ПЛК1-01 // BAGET-PLK1-01\_getting\_started\_v\_2\_X.doc

[3] Пакет поддержки программирования микроконтроллера (ППП МК). Описание применения. ЮКСУ.91264-01 31 01 // ППП\_МК\_31\_Описание\_применения v12.docx

[4] ПЛК «БАГЕТ-ПЛК1-01» Руководство по эксплуатации. ЮКСУ.421457.002-01РЭ. https://www.niisi.ru/БАГЕТ-ПЛК1-01\_РЭ\_v3.3.pdf

[5] Пакет поддержки программирования микроконтроллера (ППП МК). Руководство программиста. ЮКСУ.91264-01 33 01 // ППП\_МК\_33\_Руководство\_программиста\_v5.docx