**Лабораторная работа 4**

**Контроллер прерываний INTC, обработка прерываний от портов ввода-вывода**

Мощевикин А.П., ПетрГУ

***Общие сведения о контроллере прерываний в микроконтроллере Комдив-МК***

Микроконтроллер Комдив-МК содержит достаточно большой набор периферии (порты ввода-вывода, таймеры, контроллеры интерфейсов I2C, UART, пр.). Существует два способа работы с ними.

1. Программный опрос. По своей внутренней шине микроконтроллер опрашивает регистры состояний периферийных устройств и выясняет, которое из них готово к общению или требует принятия решений.
2. По прерываниям. Этот способ реализуется с помощью встроенного контроллера прерываний (INTC – Interrupt Controller). Внутри микроконтроллера все периферийные устройства «заведены» на входы контроллера прерываний. Как только какое-то из устройств требует внимания, оно активирует свой вход. Контроллер прерываний анализирует, разрешено ли прерывание, формирует очередь обработки прерываний и выдает информацию о прерывании процессорному ядру. Процессор прерывает ход основной программы, выполняет код обработчика прерываний, затем возвращается к основной программе.

Контроллер прерываний в микроконтроллере Комдив-МК описан в [1]. Его упрощенная схема представлена на рисунке 1.

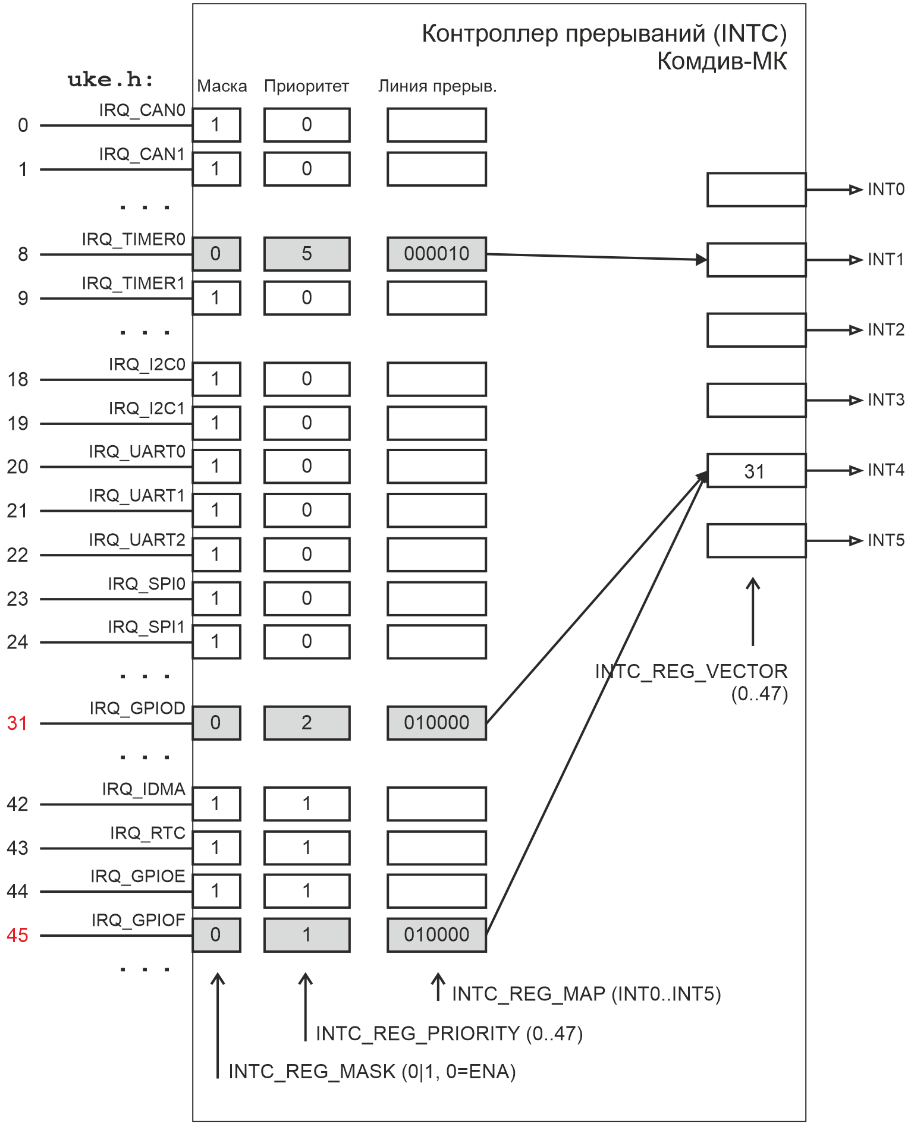


Рис. 1 – Упрощенная схема контроллера прерываний Комдив-МК. Слева – набор входных запросов на прерывание IRQ, справа – набор выходных линий прерывания процессора INT.

У контроллера – 48 входных линий запросов на прерывание IRQ (Interrupt Requests) от периферийных модулей и 6 выходных линий прерывания процессора INT. Контроллер прерываний выступает как мультиплексор (маршрутизатор), который соединяет определенный вход с определенным выходом.

48 входных линий описаны в файле ./psp\_mc/arch/kmk/uke.h. На рисунке 1 часть линий не указана.

6 выходных линий (INT0..INT5) отображены в регистре 13 Cause управляющего сопроцессора CP0 на биты IP7..IP2 [2, раздел 7.12]. Считав эти биты, процессор узнает, по какой из линий INT0..INT5 пришло прерывание.

***Регистры контроллера прерываний***

Базовый адрес контроллера прерываний определен в файле ./psp\_mc/arch/kmk/uke.h.

#define INTC\_BASE 0xbb502000

Для настройки контроллера прерываний используются следующие регистры (в скобках указано смещение относительно базового адреса):

Mask (смещение INTC\_MASK\_REG\_OFFS = 0х00) – регистр маски запросов для 48 входов,

Priority (смещение INTC\_PRIOR\_REG\_OFFS = 0х10) – регистры приоритетов для всех 48 входов,

Map (смещение INTC\_MAP\_REG\_OFFS = 0х50) – регистры маршрутизатора,

Vector (смещение INTC\_VECTOR\_REG\_OFFS = 0х08) – регистр номеров входных прерываний.

В регистр Mask необходимо выставить «0» на тех входных линиях IRQ, запросы от которых разрешены (см. серые прямоугольники со значением «0» внутри на рисунке 1).

Для каждой линии разрешенных запросов нужно установить приоритет Priority: значения от 0 до 47, 47 – наивысший приоритет. Значения приоритетов не должны повторяться для разрешенных запросов на прерывание.

Каждую входную линию запроса IRQ нужно соотнести с выходной линией INT, задав значение в соответствующем регистре Map с помощью позиционного кода.

На рисунке 1 запрос от нулевого таймера-счетчика IRQ\_TIMER0 (8) заведен на линию INT1 (значение 000010 в регистре Map, выставлен второй бит), а запросы на прерывания от портов ввода-вывода D и F (IRQ\_GPIOD и IRQ\_GPIOF) – на линию INT4 (значение 010000 в регистре Map).

В случае возникновения прерывания в регистр Vector на соответствующей выходной линии INTx будет помещено значение номера входной линии IRQ, вызвавшей прерывание.

Если запросы на прерывание поступают одновременно по нескольким входным линиям, заведенным на одну выходную, то контроллер прерываний использует значение Priority. Рисунок 1 иллюстрирует этот момент. Запросы по линиям 31 и 45 возникли одновременно. Контроллер прерываний создал очередь: первым будет обработано прерывание IRQ\_GPIOD, поскольку приоритет у него выше. В регистр Vector на линии INT4 помещено значение 31 – номер входной линии IRQ\_GPIOD.

***Программная организация контроллера прерываний, его настройка и функционирование***

Разработчики платы БАГЕТ-ПЛК1-01 [3] рекомендуют использовать среду разработки и структуру проекта, описанные в [4] и [5, раздел 3]. Проекты на языке Cи следует создавать в папке ./psp\_mc/apps/ .

Описанные ниже части программного кода взяты из пакета поддержки программирования микроконтроллера (ППП МК) [5-6].

Для того, чтобы настроить маршрутизацию в контроллере прерываний, т.е. связать входные линии запросов IRQ и выходные линии прерываний INT, используется функция

void kmk\_init\_int\_line(int IRQ, int INT, int priority),

объявленная в ./psp\_mc/arch/kmk/kmk\_common.h. Исходный код приведен в файле ./psp\_mc/arch/kmk\_init/kmk\_init.S.

Она вносит позиционный номер линии выходного прерывания INT (0..5) в ячейку определенного входного запроса IRQ в регистре контроллера прерывания Map. Кроме этого, она устанавливает поле приоритета в регистре Priority для этого запроса IRQ. А также в регистре Mask взводит разрешение прерывания по этому запросу.

Как уже было сказано, когда возникает прерывание, в одном из битов IP7..IP2 регистре 13 Cause управляющего сопроцессора CP0 выставляется значение «1».

В файле ./psp\_mc/arch/kmk\_init/kmk\_init.S содержится код общего обработчика любых прерываний kmk\_int\_handler на Ассемблере.

kmk\_int\_handler:

/\* k0 contains CP0\_CAUSE \*/

mfc0 k1, CP0\_STATUS

nop

and k0, k0, k1

srl k0, k0, 10

andi k0, k0, 0x2f /\* k0[5:0] = IM[7:2] & IP[7:2] \*/

beqz k0, 1f /\* nothing to do if no interrupt bits set \*/

// nop /\* execute next clz in any case \*/

clz k0, k0 /\* result should be from 31 (INT0) to 26 (INT5) \*/

li k1, 31

sub k0, k1, k0 /\* k0 = INTn \*/

sll k0, k0, 2

la k1, vectors

add k1, k1, k0 /\* address in vector table \*/

lw k0, (k1) /\* vector itself \*/

beqz k0, 1f /\* nothing to do if vector is not set \*/

nop

jr k0

nop

1:

eret

nop

.align 4

vectors: .word 0, 0, 0, 0, 0, 0

В этой процедуре в ходе анализа содержимого регистра CP0\_CAUSE в регистр k0 попадает значение номера линии прерываний INTn (см. строку k0 = INTn листинга выше).

В оперативной памяти организован массив vectors из шести векторов прерываний. В каждую ячейку этой таблицы заранее заносится адрес подпрограммы обработки прерывания. В нулевую ячейку – адрес для обработки прерывания INT0, в первую – адрес для INT1 и т.д.

Складывая номер линии прерываний, умноженный на 4 (sll k0, k0, 2), с базовым адресом массива векторов прерываний vectors, определяется адрес конкретного вектора в этой таблице. Далее в инструкции jr k0 выполняется переход на выбранную подпрограмму обработчика прерываний.

Для того, чтобы заполнить таблицу векторов прерываний vectors, используется функция kmk\_set\_int\_vector(2, &kmk\_my\_int2).

После ее выполнения в третий элемент массива vectors будет записан адрес функции kmk\_my\_int2.

Таблица векторов прерываний будет выглядеть следующим образом.

vectors: .word 0, 0, <адрес kmk\_my\_int2>, 0, 0, 0

Функция kmk\_my\_int2 описана в файле kmk\_irqs.S, который обязательно должен находится внутри каталога проекта. Она вызывает функцию c\_handle\_irq\_saving\_regs(2), передавая ей номер линии прерывания (INT2 в текущей реализации ППП МК).

Код функции c\_handle\_irq\_saving\_regs() представлен в файле ./psp\_mc/arch/kmk/kmk\_irqs.h. Функция сохраняет регистры процессора и вызывает функцию маршрутизатора прерывания interrupt\_router (см. файл ./psp\_mc/arch/kmk\_init/kmk\_init.S), передавая ей номер выходной линии прерывания (INT2) через регистр a0 процессора. Завершается функция c\_handle\_irq\_saving\_regs() инструкцией eret, финализирующей обработчик прерывания.

/\* a0 = CPU external interrupt line number \*/

interrupt\_router:

li t0, INTC\_BASE+8 /\* INTC\_REG\_VECTOR address \*/

addu t0, t0, a0

lbu a0, (t0) /\* a0 = interrupt input number - IRQ\_<dev> \*/

la t1, ext\_vectors

sll t0, a0, 2

addu t1, t1, t0

lw t0, (t1) /\* handler address or 0 \*/

la t1, handler\_params

beq t0, zero, 1f

addu t1, t1, a0 /\* t1 = addr of param \*/

addiu sp, sp, -4

sw ra, (sp)

jalr t0

lbu a0, (t1) /\* a0 contains parameter (byte) \*/

lw ra, (sp)

addiu sp, sp, 4

1: jr ra

nop

ext\_vectors:

.word 0, 0, 0, 0, 0, 0, 0, 0 /\* 0 - 7 \*/

.word 0, 0, 0, 0, 0, 0, 0, 0 /\* 8 - 15 \*/

.word 0, 0, 0, 0, 0, 0, 0, 0 /\* 16 - 23 \*/

.word 0, 0, 0, 0, 0, 0, 0, 0 /\* 24 - 31 \*/

.word 0, 0, 0, 0, 0, 0, 0, 0 /\* 32 - 39 \*/

.word 0, 0, 0, 0, 0, 0, 0, 0 /\* 40 - 47 \*/

handler\_params:

.word 0, 0, 0, 0, 0, 0, 0, 0 /\* 0 - 31 \*/

.word 0, 0, 0, 0 /\* 32 - 47 \*/

Функция interrupt\_router выполняет роль маршрутизатора прерывания. Она вызывает обработчик прерывания именного того периферийного устройства, которое ранее выставило входной запрос на прерывание IRQ, транслированный в прерывание на выходной линии INT.

В первых трех инструкциях по номеру выходной линии (INT2) рассчитывается адрес байта в регистре Vector контроллера прерываний (для линии INT2 он равен INTC\_BASE+10). Значение байта по этому адресу попадает в регистр a0 процессора, оно равно номеру входной линии текущего запроса на прерывание (IRQ).

Для того, чтобы узнать, кому дальше обслуживать вызванное прерывание, необходимо воспользоваться маршрутизацией, т.е. считать ячейки в заранее заполненных таблицах ext\_vectors и handler\_params (они объявлены в листинге выше).

В первой указаны адреса функций обработчиков запросов на прерывание IRQ для всех входных линий (0..47), а во второй – параметр, передаваемый в функцию (через регистр a0 процессора).

Инструкция jalr t0 выполняет переход в выбранный обработчик прерывания (на момент перехода в регистре процессора t0 записан адрес функции обработчика прерываний, взятый из таблицы ext\_vectors).

Для того, чтобы заполнить таблицы ext\_vectors и handler\_params, т.е. настроить маршрутизацию, используется функция

void int\_connect(int irq, void(\*routine)(), char param ),

объявленная в ./psp\_mc/arch/kmk/kmk\_common.h. Ее исходный код приведен в файле ./psp\_mc/arch/kmk\_init/kmk\_init.S.

int\_connect:

la t0, handler\_params

addu t0, t0, a0

sb a2, (t0) /\* store parameter \*/

la t0, ext\_vectors

sll a0, a0, 2

addu t0, t0, a0

jr ra

sw a1, (t0) /\* store handler \*/

Номер элемента в обеих таблицах определяется первым параметром, передаваемым в функцию. В таблицу handler\_params записывается третий параметр param, в таблицу ext\_vectors – второй (адрес функции обработчика запроса на прерывание).

***Пример настройки линии порта ввода-вывода для работы с прерываниями***

Любую линию любого порта ввода-вывода (порты A..F) можно настроить в режим работы с прерываниями.

На плате БАГЕТ-ПЛК1-01 размещена пользовательская кнопка SW2. Она в момент нажатия замыкает вывод микроконтроллера GPIO\_D6 на землю. В этот момент напряжение на входе GPIO\_D6 изменяется с высокого на низкое. Этот переход можно отследить аппаратно с помощью прерываний.

Файл ./psp\_mc/arch/kmk/kmk.uhe определяет базовый адрес порта D.

#define GPIOD\_BASE 0xbb400380

Для работы с прерываниями используются следующие регистры пространства порта ввода-вывода [7]:

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

event0-event7 (смещение 0x18-0x1f) – регистр настройки отслеживаемого события на линиях порта,

event\_time0-event\_time7 (смещение 0x20-0x5f) – регистр времени произошедшего события.

Значения битов 0..7 в регистре ISR каждого порта ввода-вывода показывают, выставлено ли прерывание по соответствующей линии этого порта ввода-вывода.

В соответствии с таблицей 2.9 [7], регистр event6 предназначен для настройки реакции на линии порта. Его адрес находится по смещению 0x1e относительно базового адреса.

| Разряд | Описание | Начальное состояние | Режим |
| --- | --- | --- | --- |
| 0 | Статус события | 0 | Чтение/сброс |
| 3:1 | Настройка события | 0 | Чтение/запись |
| 4 | Разрешение выставления прерывания по событию | 0 | Чтение/запись |
| 5 | Разрешение сохранения временной метки события | 0 | Чтение/запись |

Разряд 0 при чтении показывает статус события («1» – событие произошло). При записи «1» в него происходит сброс статуса события.

Значения разрядов [3..1] определяют тип события:

000 события не отслеживаются,

001 уровень 0 на входе,

010 уровень 1 на входе,

011 положительный перепад на входе,

100 отрицательный перепад на входе,

101 любой перепад на входе,

110-111 зарезервировано.

Для настройки отслеживаемого события на линии 6 используется регистр event6. Для того, чтобы при нажатии на кнопку SW2 происходило прерывание по отрицательному перепаду, в регистр event6 порта D необходимо записать значение 0b00111001=0x39=57.

Поскольку бит 5 разрешения сохранения временной метки выставлен, то в момент прерывания в 64-разрядный регистр event\_time6 по смещению 0x50 автоматически запишется значение времени события.

Таким образом, при создании программы с прерываниями от кнопки SW2 в функцию init\_board() необходимо поместить следующий код.

GPIO\_SetPinMode(GPIOD\_BASE, 6, GPIO\_MODE\_INPUT); // SW2 button, GPIO\_D6 as INPUT

\*(volatile unsigned char \*)(GPIOD\_BASE + 0x1e) = 0x39; // event6 register: falling edge, IRQ Enable, Timestamping Enable

***Действия программиста по настройке и использованию прерываний в программе***

Резюмируя вышесказанное, для использования прерываний в своей программе, программист должен выполнить следующее.

1. Создать функцию-обработчик запроса на прерывание от периферийного устройства, скажем, void my\_dev\_int\_handler() и связать ее с одной из 48 линий входных запросов IRQ (см. файл ./psp\_mc/arch/kmk/uke.h).

void my\_dev\_int\_handler(unsigned char param) {

. . .

}

int\_connect(IRQ\_NUMBER, my\_dev\_int\_handler, 0);

1. Настроить выбранное периферийное устройство и разрешить от него прерывания.
2. Объявить функцию обработки прерывания INT, привязать ее к одному из шести прерываний процессора INT0..INT5.

void kmk\_my\_int2(void);

kmk\_set\_int\_vector(INT\_NUMBER, &kmk\_my\_int2);

1. Настроить внутреннюю маршрутизацию контроллера прерываний (заполнить поля маски и приоритета, а также однозначно связать номер входного прерывания IRQ\_NUMBER с выходной линией INT\_NUMBER).

kmk\_init\_int\_line(IRQ\_NUMBER, INT\_NUMBER, 0);

***Задания для самостоятельной работы***

Разработчики платы БАГЕТ-ПЛК1-01 [2] рекомендуют использовать среду разработки и структуру проекта, описанные в [3] и [4, раздел 3].

Задания к лабораторной работе подразумевают, что работа выполняется в ОС Windows с предустановленной виртуальной машиной WSL2 с Debian [3] и ППП МК [4-5]. Проекты на языке Cи следует создавать в папке ./psp\_mc/apps/ .

В лабораторной работе №1 для считывания состояния цифрового входа использовалась функция GPIO\_ReadPin(), опрашивающая регистр входных данных IDR. В этой лабораторной работе показано, как тот же результат достигается с помощью аппаратных прерываний без использования зацикленного опроса состояния входной линии.

1. Используя исходный код лабораторной работы 1 для проекта на языке Си, создать программу в соответствии с предлагаемой структурой.

int main(void)

{

init\_board();

while (1) {

if (!GPIO\_ReadPin(GPIOD\_BASE, 6)) {

. . .

break;

}

}

return 1; // unreacheable

}

В функции init\_board() инициализировать линию GPIO\_D6 для работы с прерываниями. Создавать обработчик прерывания и выполнять настройку контроллера прерываний не требуется. Таким образом, самого прерывания происходить в разрабатываемой программе не будет. Будет возникать лишь запрос на прерывание от внешнего устройства (контроллера GPIO) и регистрации метки времени события.

В основной программе в функции main() реализовать считывание значений регистров ISR и event6 до нажатия кнопки SW2 и после (до цикла while(1) и внутри в теле if ()). Убедиться, что после нажатия кнопки SW2 значения обоих регистров изменилось. Проанализировать результат.

Вывести значения системного времени в момент нажатия кнопки SW2 двумя способами (выполнив макрос \_\_micros()) сразу после нажатия кнопки, а затем считав значение 64-разрядного регистра event\_time6). Сравнить результаты.

Примечание. Размер содержимого регистра event\_time6 – 8 байтов. Считать старшие 32 разряда 64-битного регистра можно следующим образом.

#define GPIO\_EVENT\_TIME6\_REG\_OFFS 0x50

unsigned long x;

x = \*((volatile unsigned long \*)(GPIOD\_BASE + GPIO\_EVENT\_TIME6\_REG\_OFFS + 4));

1. Создать новый проект (на основе предыдущего). Сделать так, что бы светодиод VD2 на плате БАГЕТ-ПЛК1-01 бесконечно мигал с периодичностью в 1 секунду (можно использовать функцию delay()). А в момент нажатия кнопки SW2 загорался светодиод VD5. Программа должна использовать запрос на прерывание IRQ от порта ввода-вывода GPIO\_D6, заведенный на линию INT2.

Основной цикл while (1) {} должен содержать только код управления светодиодом VD2. Управление светодиодом VD5 необходимо реализовать в обработчике прерывания. Внутри обработчика прерывания не забудьте сбросить статус события, выполнив:

\*(volatile unsigned char \*)(GPIOD\_BASE + 0x1e) = 0x39; // event6 register: falling edge, IRQ Enable, Timestamping Enable

Показать, что программа не реагирует на замыкания других линий порта D на землю (кроме линии GPIO\_D6).

1. Факультативно. Переработать программу из п.2: изменить номер используемого прерывания на INT3, светодиод VD5 при каждом нажатии кнопки SW2 должен изменять свое состояние: горит / не горит. Подумайте над защитой от дребезга кнопки SW2 (но не реализуйте). Учтите, что обработчик прерываний должен быть настолько коротким, насколько это возможно.
2. Факультативно. Придумать и реализовать программу, демонстрирующую работу контроллера прерываний при одновременном возникновении двух прерываний с разными уровнями приоритета. Например, при замыкании входных линий на землю (двух кнопок) на двух разных портах. Сначала должно обработаться прерывание с более высоким приоритетом, затем с более низким.

Для вывода результата в консоль можно воспользоваться функцией \_\_print\_console(). Для вывода целочисленных значений необходимо создать функцию перевода числа в строку, либо воспользоваться сторонней библиотекой, содержащей функцию типа itoa().

***Список литературы***

[1] Микросхема интегральная К5500ВК018. Указания по применению. Чаcть 6. Контроллер прерываний. ЮКСУ.431295.019Д4.5 // К5500ВК018\_d4\_05\_CON\_INT.v2.3.4.doc

[2] Микросхема интегральная К5500ВК018. Указания по применению. Часть 2. Ядро микропроцессора. 64-разрядный микропроцессор. ЮКСУ.431295.019Д4.1 // К5500ВК018\_d4\_01\_CORE.v2.doc

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

[4] 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

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

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

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