PHPoC

Программный гид по устройствам P20

Версия 1.2

Sollae Systems Co., Ltd.

Сайт PHPoC: http://www.phpoc.com

Сайт ezTCP http://www.eztcp.com

Содержание

1 Обзор

1.1 Устройство

«Устройством» называется физическая плата и программные функции, которые предоставляет PHPoC. Каждое устройство предоставляет специальный файл, который может быть использован как общий (чтение/запись файлов).

Структура файла PHPoC выглядит следующим образом:

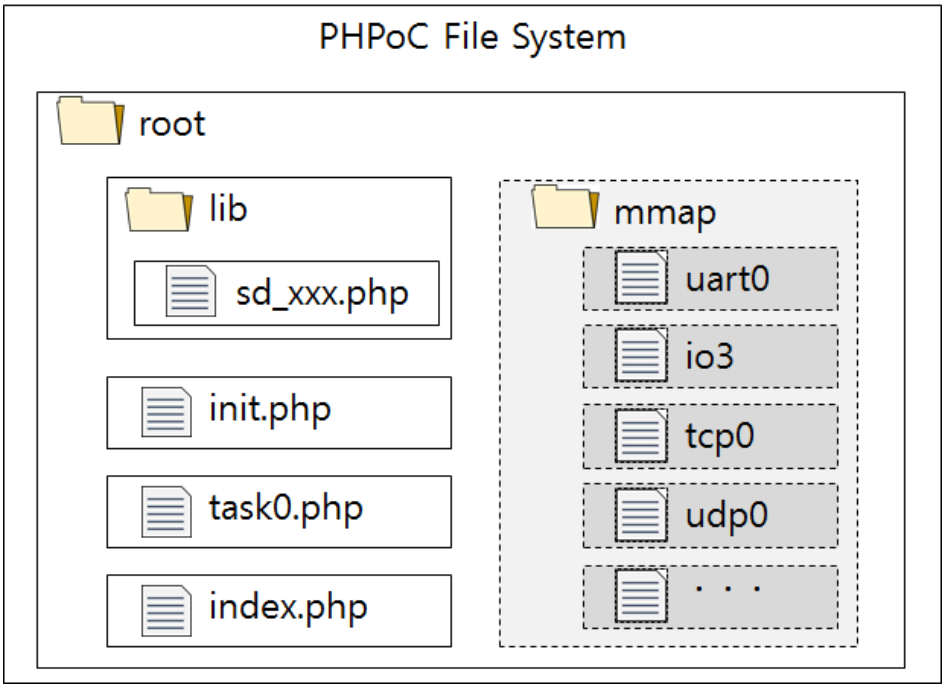


Рисунок 1-1 Файловая система PHPoC

1.2 Путь к файлам устройств

Все файлы устройств PHPoC находятся в каталоге mmap (memory map) в корневой папке. Чтобы получить доступ к определенному устройству, вы должны использовать путь, указанный в примере ниже.

```
/mmap/DEVICE NAME
```

☞ ***Все файлы, которые вы загружаете в PHPoC, находятся в корневом каталоге. Вы можете получить доступ только к корневому каталогу и каталогу /mmap в данной файловой системе. Кроме того, пользователям не разрешается создавать или удалять каталоги.***

1.3 Типы устройств

PHPoC предоставляет следующие типы устройств.

Раздел	Имя устройства
Hardware	Цифровые I/O(вводи вывод), UART(последовательный), NET(сеть)
Software	TCP, UDP, ST(программный таймер)

Таблица 1-1 Типы устройств

Типы могут отличаться в зависимости от продукта и версии прошивки.

☞ ***Обратитесь к Приложению для получения подробной информации об устройствах в зависимости от типа продукта.***

1.4 Этапы использования устройств

Основные этапы использования устройств следующие:

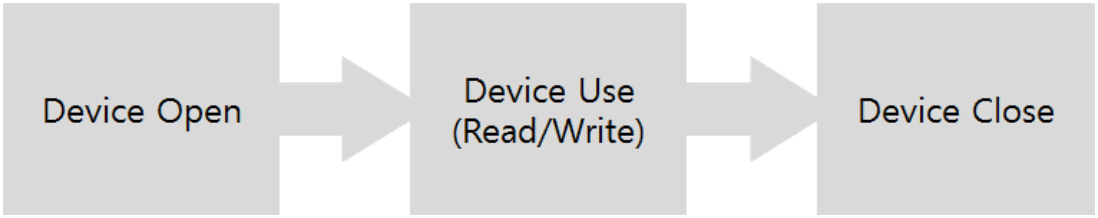


Рисунок 1-2 Этапы использования устройств

1.4.1 Открытие устройства

Функция pid_open предназначена для открытия устройств. Данная функция возвращает pid (Peripheral ID), который представляет собой целочисленное значение, и эот значение используется для доступа к устройствам в качестве уникального номера.

1.4.2 Использование устройства

После успешного открытия устройства (которое вернуло pid) оно готово к использованию.

1.4.3 Закрытие устройства

Когда устройство больше не используется, его необходио закрыть при помощи функции pid_close.

☞ ***Внимание: Невозможно использовать только что использованный физический порт новым устройствам, если порт не был перезагружен (даже если устройство было закрыто). Другими словами, физический порт не может использовтаься более чем двумя устройствами до его инициализации.***

2 Цифровые I/O

2.1 Обзор

Цифровые I/O могут использоваться для контроля или управления цифровым вводом/выводом. Данное устройство также используется для подключения светоидных индикаторов состояния системы (LED).

- Структура цифровых I/0

Каждый цифровой порт ввода/вывода I/O может иметь два разных состояния: High (или 1) и Low (или 0). В следствие чего, каждый порт соответствует двоичной цифре, как вы можете увидеть на схеме ниже.

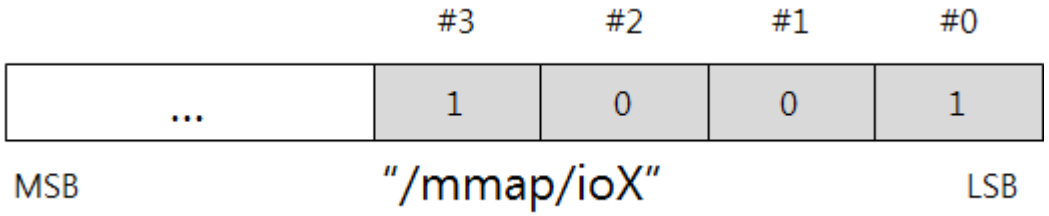


Схема 2-1 map-пример цифровых I/O

2.2 Этапы использования I/O

Основные этапы использования цифровых I/O портов выглядят следующим образом:

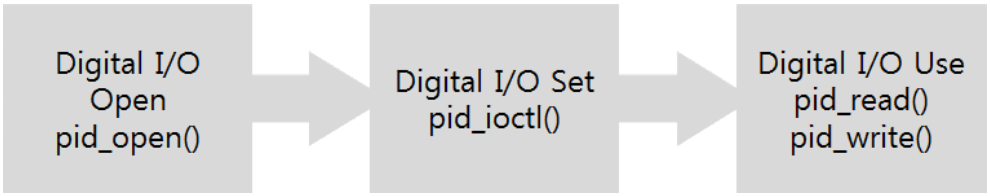


Схема 2-2 Этапы использования I/O

2.3 Открытие цифровых I/O

Для открытия цифровых I/O необходима функция pid_open.

```
$pid = pid_open("/mmap/io3");          // opening digital I/O 3
```

☞ ***Обратитесь к Приложению для получения подробной информации о цифровых I/O (включая LED) в зависимости от типа продукта.***

2.4 Настройка цифровых I/O

Перед использованием цифрового ввода/вывода (I/O) необходимо настроить каждый порт. Для настройки используется функция pid_ioctl.

```
pid_ioctl($pid, "set N1[-N2] mode TYPE");
```

N1 и N2 указывают диапазон нескольких портов. В случае настройки только одного порта, возможно использовать только N1. Доступны следующие типы:

2.4.1 Доступные типы цифровых I/O

ТИП		Описание
in		Input
out	-	Output
	low	Output: default LOW
	high	Output: default HIGH
led_sys		System Status LED
led_net0_act / led_net1_act		Activation of NET(net0 - wired, net1 - wireless) link LED: - successfully established network link - LOW - at the moment sending or receiving network data - HIGH
led_net0_link / led_net1_link		Network Link LED: connected to network - LOW
led_net0_rx / led_net1_rx		Network Receive LED: at the moment receiving data - LOW
led_net0_tx / led_net1_tx		Network Send LED: at the moment sending data - LOW

Таблица 2-1 Доступные цифровые типы I/O

☞ ***С продуктами, которые оснащены LED, каждый порт цифрового ввода/вывода может быть сопоставлен с физичесиким LED. LED включается, когда состояние порта LOW и выключается, кода состояние порта HIGH. Если вам требуется более подробная информация, обратитесь к схеме/диаграмме вашего продукта..***

- Пример настройки цифровых I/O

```
$pid = pid_open("/mmap/io3");          // open digital I/O 3
pid_ioctl($pid, "set 0 mode out");     // set port 0 to output
pid_ioctl($pid, "set 1-2 mode in");    // set port 1 ~ 2 to input
// setting port 3 to network link LED
pid_ioctl($pid, "set 3 mode led_net0_link");
// setting port 12 to network receive LED
pid_ioctl($pid, "set 12 mode led_net0_rx");
// setting port 13 to network send LED
pid_ioctl($pid, "set 13 mode led_net0_tx");
```

2.5 Использование цифровых I/O

2.5.1 Чтение состояний цифрового ввода/вывода

При считывании состояния цифровых портов I/O, вы можете получить множество состояний при помощи функции pid_read или одиночное состояние при помощи функции pid_ioctl.

```
pid_read($pid, VALUE);            // read all state(in 16bits unit)
pid_ioctl($pid, "get N ITEM");    // read a single state(in a bit unit)
```

При чтении одиночного состояния доступны следующие пункты (ITEM):

ITEM	Описание	
Mode	Return the port status in string type	I/O pin: "in", "out", "led_xxx" or etc.
		Designated to output pin of ST: "st_out"
Input	Return the input port status in integer (0: LOW, 1: HIGH)	
output	Return the output port status in integer (0: LOW, 1: HIGH)	

Таблица 2-2 Доступные ITME при считывании одиночного состояния

- Пример считывания всех цифровых состояний I/0

  В примере ниже показано состояние порта от 0 до 3 после команды портам показать их состояние:

```
$value = 0;
$pid = pid_open("/mmap/io3");          // open digital I/O 3
pid_ioctl($pid, "set 0-3 mode in");    // set port 0 ~ 3 to input
pid_read($pid, $value);                // read digital I/O status(16bits unit)
printf("0x%x\r\n", $value);            // output example: 0xf00f
```

- Пример считывания одиночного состояния I/0

  Пример ниже показывает вывод состояния порта 0 номер 3 после настройки его на цифровой вывод и получения сведений о его режиме и состоянии.

```
$pid = pid_open("/mmap/io3");              // open digital I/O 3
pid_ioctl($pid, "set 0 mode out high");    // set port 0 to output
$mode = pid_ioctl($pid, "get 0 mode");     // read a digital I/O mode
$output = pid_ioctl($pid, "get 0 output"); // read a digital I/O state
printf("%s, %d\r\n", $mode, $output);      // output example: out, 1
```

☞ ***При чтении состояния порта с функцией pid_ioctl вы должны использовтаь "get N input", если он настроен на входной порт и использовать "get N output" , если порт установлен на выходной порт.***

2.5.2 Запись значений в цифровых I/O

При записи значений в цифровые I/O порты, вы можете настроить значение для нескольких портов при помощи функции pid_write или для одиночного порта при помощи pid_ioctl.

```
pid_write($pid, VALUE);                  // write to all ports(16 bits unit)
pid_ioctl($pid, "set N output TYPE");    // write to a single port(a bit unit)
```

- Пример записи значений во все порта

  Приведенный ниже пример показывает состояние цифрового I/O порта после установки 0 ~ 7 пинов 3 цифорового I/O порта в выходной порт и записи заданного значения.

```
$value = 0;
$pid = pid_open("/mmap/io3");                  // open digital I/O 3
pid_ioctl($pid, "set 0-7 mode out");           // set port 0 ~ 7 to output
pid_read($pid, $value);                        // read status
pid_write($pid, ($value & 0xff00) | 0x0055); // write 0x55
pid_read($pid, $value);                        // read status
printf("0x%0x\r\n", $value);                   // output example: 0x0055
```

- Пример записи значений в одиночный порт

  Приведенный ниже пример показывает состояние 3-го цифрового I/O порта 0 после настройки его на цифровой вывод со значением по умолчанию LOW и записи HIGH.

```
$pid = pid_open("/mmap/io3");              // open digital I/O 3
pid_ioctl($pid, "set 0 mode out low");     // set port 0 to output(LOW)
pid_ioctl($pid, "set 0 output high");      // write HIGH
$output = pid_ioctl($pid, "get 0 output"); // read state of port 0
printf("%d\r\n", $output);                 // output: 1
```

- Пример настройки ограничения вывода

  В примере ниже сравнивается разница между заблокированным и разблокированным состоянием порта 0.

```
$pid = pid_open("/mmap/io3");              // open digital I/O 3
pid_ioctl($pid, "set 0 mode out low");     // set port 0 to output(LOW)
pid_ioctl($pid, "set 0 lock");             // set port 0 to output restriction
pid_ioctl($pid, "set 0 output high");      // write HIGH to port 0
$output1 = pid_ioctl($pid, "get 0 output"); // read state of port 0
pid_ioctl($pid, "set 0 unlock");           // release the output restriction
pid_ioctl($pid, "set 0 output high");      // write HIGH to port 0 again
$output2 = pid_ioctl($pid, "get 0 output"); // read state of port 0
printf("%d, %d\r\n", $output1, $output2); // output: 0, 1
```

2.5.3 Управление реле

Некоторые из внешних устройств имеют релейные порты, которые подключены к цифровому выходу.

- Пример

Пример ниже демонстрирует одновременное включение и выключение всех релейных выходов каждую секунду после последовательного включения порта реле в порядке от 0 к 3.

```
$pid = pid_open("/mmap/io4");          // open digital I/O 4 (relay port)
pid_ioctl($pid, "set 7 mode out");     // Set port 7(Relay OE) to output
// Setting port 8 ~ 11 to output (relay port 0 ~ 3)
pid_ioctl($pid, "set 8-11 mode out");
pid_write($pid, 0x0100);               // turn relay port 0 on
sleep(1);
pid_write($pid, 0x0200);               // turn relay port 1 on
sleep(1);
pid_write($pid, 0x0400);               // turn relay port 2 on
sleep(1);
pid_write($pid, 0x0800);               // turn relay port 3 on
sleep(1);
pid_write($pid, 0x0f00);               // turn relay port 0 to 3 on
sleep(1);
pid_write($pid, 0x0000);               // turn relay port 0 to 3 off
```

☞ ***Данный пример недоступен для продуктов, которые не имеют релейного порта.***

3 UART

UART (Универсальный асинхронный приемопередатчик) - это наиболее широко используемая связь в мире.

3.1 Этапы испольлзования UART

Основные этапы использования UART выглядят следующим образом:

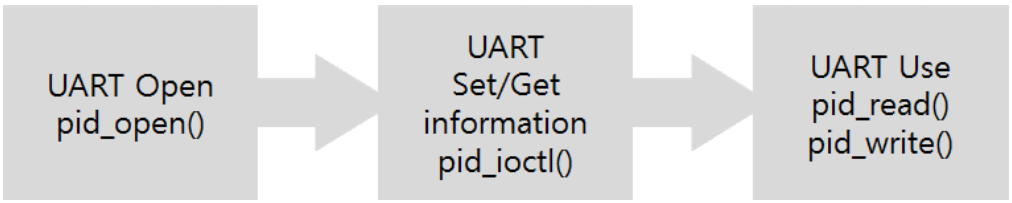


Рисунок 3-1 Этапы использования UART

3.2 Открытие UART

Для открытия UART необходима функция pid_open.

```
$pid = pid_open("/mmap/uart0");                    // opening UART 0
```

☞ ***Обратитесь к приложению для подробной информации по UART, исходя из типа вашего продукта.***

3.3 Настройка UART

Перед использованием UART необходима настройка параметров. Для настройки необходима функция pid_ioctl.

```
pid_ioctl($pid, "set ITEM VALUE");
```

ITEM обозначает установку позиции, а VALUE означает возможное значение элемента.

3.3.1 Доступные элементы UART

ITEM	VALUE	Описание
baud	e.g. 9600	baud rate[bps], 2400 ~ 230400
parity	0	no parity
	1	EVEN parity
	2	ODD parity
	3	MARK parity (always 1)
	4	SPACE parity (always 0)
data	8	8 data bit
	7	7 data bit(it can be only used with parity bit)
stop	1	1 stop bit
	2	2 stop bit
flowctrl	0	no flow control
	1	RTS/CTS hardware flow control
	2	Xon/Xoff software flow control
	3	TxDE flow control for RS485

Таблица 3-1 Доступные элементы UART

- Пример настройки UART

```
$pid = pid_open("/mmap/uart0");      // open UART 0
pid_ioctl($pid, "set baud 9600");    // baud rate: 9600 bps
pid_ioctl($pid, "set parity 0");     // no parity
pid_ioctl($pid, "set data 8");       // data bit length: 8
pid_ioctl($pid, "set stop 1");       // stop bit length: 1
pid_ioctl($pid, "set flowctrl 0");   // no flow control
```

3.3.2 Настройка типа связи UART

UART может быть сконфигурирован с RS422, RS485 или с RS232, в зависимости от типа устройства. Для этого требуется настройка TxDE и настройка соответствующих цифровых ввода/вывода.

Пример ниже покажет, как установить последовательные типы связяи UART 0.

```
$pid = pid_open("/mmap/uart0");       // open UART 0
pid_ioctl($pid, "set baud 9600");     // baud rate: 9600 bps
pid_ioctl($pid, "set parity 0");      // parity: none
pid_ioctl($pid, "set data 8");        // data bit: 8
pid_ioctl($pid, "set stop 1");        // stop bit: 1
pid_ioctl($pid, "set flowctrl 3");    // flow control: TxDE
$pid_mode = pid_open("/mmap/io4"); // open Digital I/O 4 for UART mode
pid_ioctl($pid_mode, "set 0-3 mode out");   // set port 0~3 to output
pid_write($pid_mode, 0x05);           // RS232
//pid_write($pid_mode, 0x02);         // (remove comment sign for RS422)
//pid_write($pid_mode, 0x0c);         // (remove comment sign for RS485)
pid_close($pid_mode);
```

☞ ***Данные примерах предполагается, что пин 0~3 цифрового I/O под номером 4 предназначен для конфигурации типов UART. Проверьте информацию об устройстве, поскольку назначение цифрового I\|O может отличаться в зависимости от типа продукта.***

☞ ***Обратитесь к Приложению для подробной информации о цифровых I/O в зависимости от типа продукта.***

3.4 Получение статуса UART

Для получения разнообразных состояний/статусов UART необходима функция pid_ioctl.

```
$return = pid_ioctl($pid, "get ITEM");
```

3.4.1 Доступные состояния UART

ITEM	Описание	Возврат значений	Возвращаемый тип
baud	baud rate[bps]	e.g. 9600	Целое число
parity	Parity	0 / 1 / 2 / 3 / 4	Целое число
data	data bit[bit]	8 / 7	Целое число
stop	stop bit[bit]	1 / 2	Целое число
flowctrl	Flowctrl	0 / 1 / 2 / 3	Целое число
txbuf	size of send buffer[Byte]	e.g. 1024	Целое число
txfree	remaining send buffer size[Byte]	e.g. 1024	Целое число
rxbuf	size of receive buffer[Byte]	e.g. 1024	Целое число
rxlen	received data size[Byte]	e.g. 10	Целое число

Таблица 3-2 доступные состояния UART

- Пример получения состояний UART

Получение текущей информации по UART выглядит следующим образом:

```
$pid = pid_open("/mmap/uart0");                  // open UART 0
$baud = pid_ioctl($pid, "get baud");             // get baud rate
$parity = pid_ioctl($pid, "get parity");         // get parity
$data = pid_ioctl($pid, "get data");             // get data bit
$stop = pid_ioctl($pid, "get stop");             // get stop bit
$flowctrl = pid_ioctl($pid, "get flowctrl");     // get flow control mode
echo "baud = $baud\r\n";                         // output e.g.: baud = 9600
echo "parity = $parity\r\n";                     // output e.g.: parity = 0
echo "data = $data\r\n";                         // output e.g.: data = 8
echo "stop = $stop\r\n";                         // output e.g.: stop = 1
echo "flowctrl = $flowctrl\r\n";                 // output e.g.: flowctrl = 0
```

3.4.2 Оставшийся размер данных в буфере отправки

Оставшийся размер данных в буфере отправки может быт рассичтан следующим образом:

```
Оставшийся размер данных в буфере = размер буфера – оставшийся размер буфера
```

- Пример:

В примере ниже показано, как проверить размер оставшихся данных буфера отправки.

```
$txlen = -1;
$data = "0123456789";
$pid = pid_open("/mmap/uart0");       // open UART 0
pid_ioctl($pid, "set baud 9600");     // baud rate: 9600 bps
pid_ioctl($pid, "set parity 0");      // parity: none
pid_ioctl($pid, "set data 8");        // data bit: 8
pid_ioctl($pid, "set stop 1");        // stop bit: 1
pid_ioctl($pid, "set flowctrl 0");    // flow control: none
pid_write($pid, $data);               // write data to UART
while($txlen)
{
        $txbuf = pid_ioctl($pid, "get txbuf"); // get size of send buffer
        // get remaining size of send buffer
        $txfree = pid_ioctl($pid, "get txfree");
        $txlen = $txbuf - $txfree;    // calculate remaining data size
        echo "tx len = $txlen\r\n";   // prints the size
        usleep(1000);
}
pid_close($pid);
```

3.4.3 Размер полученных данных

Ниже показано, как получить размер данных UART.

```
$rxlen = pid_ioctl($pid, "get rxlen[ $string]");
```

- Получение размера полученных данных со строкой

Если после команды rxlen item указана строка, функция pid_ioctl возвращает 0 до тех пор пор, пока строка не войдет в UART. Если указанная строка приходит, то она возвращает весь размер данных, включая строку.

3.4.4 Оставшийся размер буфера приема

Оставшийся размер буфера приема может быть рассчитан следующим образом:

```
Оставшийся размер буфера приема = размер буфера – размер полученных данных
```

- Пример

Пример показывает, как получить оставшийся размер буфера приема.

```
$rdata = "";
$pid = pid_open("/mmap/uart0");       // open UART 0
pid_ioctl($pid, "set baud 9600");     // baud rate: 9600 bps
pid_ioctl($pid, "set parity 0");      // parity: none
pid_ioctl($pid, "set data 8");        // data bit: 8
pid_ioctl($pid, "set stop 1");        // stop bit: 1
pid_ioctl($pid, "set flowctrl 0");    // flow control: none
$rxbuf = pid_ioctl($pid, "get rxbuf"); // get size of receive buffer
$rxlen = pid_ioctl($pid, "get rxlen"); // get received data size
$rxfree = $rxbuf - $rxlen;            // get remaining size of receive buffer
echo "rxfree = $rxfree\r\n";          // print the size
pid_close($pid);
```

3.5 Использование UART

3.5.1 Чтение данных

Данные, полученные с UART,сохраняются в буфере приема. Для чтения данных необходима функция pid_read.

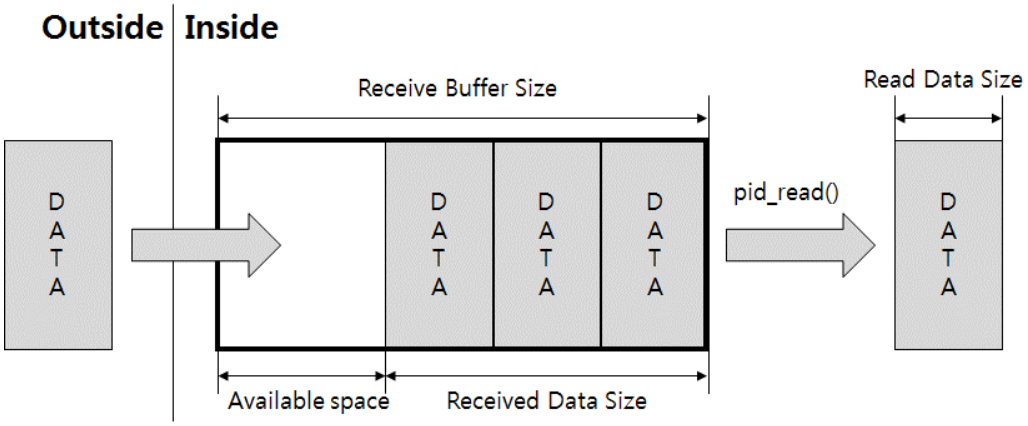


Схема 3-2 Чтение данных с UART

Ниже показано, как использовать функцию pid_read.

```
pid_read($pid, $var[, $len]);
```

Аргумент $var - это переменная для сохранения прочитанных данных, а $len – это размер прочитанных данных.

- Пример

Данный пример проверяет и выводит полученные данные в UART каждую секунду.

```
$rdata = "";
$pid = pid_open("/mmap/uart0");                 // open UART 0
pid_ioctl($pid, "set baud 9600");               // baud rate: 9600bps
pid_ioctl($pid, "set parity 0");                // parity: none
pid_ioctl($pid, "set data 8");                  // data bit: 8
pid_ioctl($pid, "set stop 1");                  // stop bit: 1
$rxbuf = pid_ioctl($pid, "get rxbuf");          // get size of receive buffer
while(1)
{
        $rxlen = pid_ioctl($pid, "get rxlen");  // get size of received data
        $rx_free = $rxbuf - $rxlen;             // get remaining size
        echo "$rx_free / $rxbuf\r\n";           // print remaining size
        $len = pid_read($pid, $rdata, $rxlen); // read data
        echo "len = $len / ";                   // print size of read data
        echo "rdata = $rdata\r\n";              // print read data
        sleep(1);
}
pid_close($pid);
```

3.5.2 Отправка данных

Данные, записанные функцией pid_write, хранятся в буфере отправки и передаются внешне через UART.

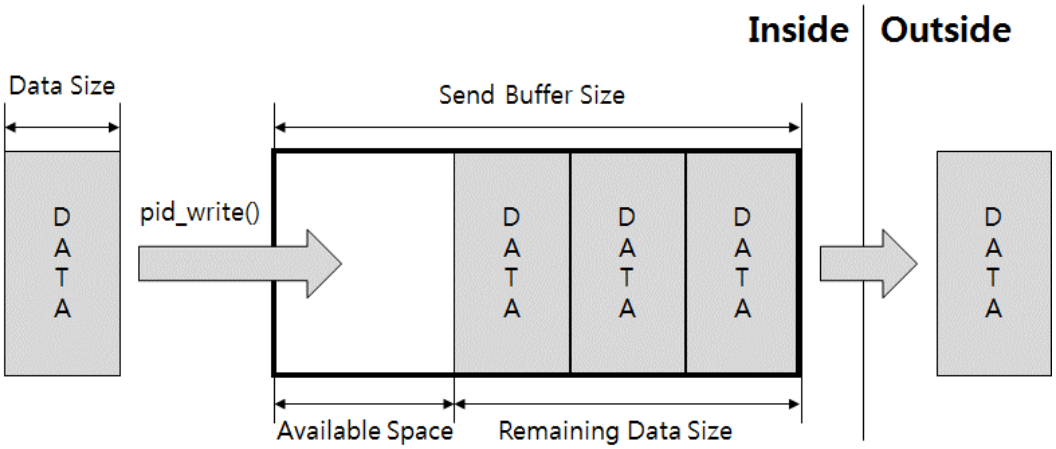


Схема 3-3 тправка данных в UART

Нижеприведенный пример показывает, как использовать функцию pid_write.

```
pid_write($pid, $var[, $wlen]);
```

Аргумент $var – это переменная, содержащая данные для отправки, а $wlen – это размер отправляемых данных.

- Пример

В данном примере каждую секунду выводится оставшийся размер буфера отправки и длина отправляемых данных.

```
$sdata = "0123456789";
$pid = pid_open("/mmap/uart0");                 // open UART 0
pid_ioctl($pid, "set baud 9600");               // baud rate: 9600bps
$txbuf = pid_ioctl($pid, "get txbuf");          // get size of send buffer
while(1)
{
        $txfree = pid_ioctl($pid, "get txfree"); // get remaining size
        echo "txfree = $txfree\r\n";            // print remaining size
        $len = pid_write($pid, $sdata, $txfree);// write data
        echo "len = $len\r\n";                  // print length of data sent
        sleep(1);
}
pid_close($pid);
```

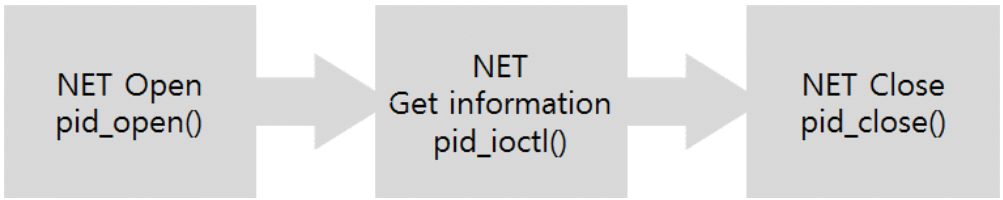
Третий аргумент функции pid_write означает длину записи данных. Длина записываемых данных должна быть меньше, чем оставшийся размер буфера отправки (во избежаение потери данных). Настоятельно рекомендуется проверить оставшийся размер буфера отправки перед отправкой данных.

4 NET (Network)

NET означает физический интерфейс проводной и беспроводной сети.

4.1 Этапы использования NET

Основные этапы использования NET следующие:



Смеха 4-1 Этапы использования NET

4.2 Открытие NET

Для открытия NET необходима функция pid_open.

```
$pid = pid_open("/mmap/net0");                    // opening NET 0
```

☞ ***Обратитесь Приложению для подробной информации о NET в зависимости от типа продукта.***

4.3 Получение статуса/состояния NET

Для получения состояния NET порта необходима функция pid_ioctl.

```
$return = pid_ioctl($pid, "get ITEM");
```

ITEM – это название досутпных состояний/статусов.

4.3.1 Доступные состояния/статусы NET

ITEM	Описание	Возврат значений	Тпи возврата
hwaddr	MAC Address	e.g. 00:30:f9:00:00:01	Строка
ipaddr	IP Address	e.g. 10.1.0.1	Строка
netmask	Subnet Mask	e.g. 255.0.0.0	Строка
gwaddr	Gateway Address	e.g. 10.1.0.254	Строка
nsaddr	Name Server Address	e.g. 10.1.0.254	Строка
mode	10M Ethernet	10BASET	Строка
	100M Ethernet	100BASET	Строка
	WLAN Unavailable	""(an Empty String)	Строка
	WLAN Infrastructure	INFRA	Строка
	WLAN Ad-hoc	IBSS	Строка
	WLAN Soft AP	AP	Строка
speed	Ethernet Speed[Mbps]	0 / 10 / 100	Целое число
	WLAN Speed[100Kbps]	0 / 10 / 20 / 55 / 110 / 60 / 90 / 120 / 180 / 240 / 360 / 480 / 540	Целое число

Таблица 4-1 Доступные состояния NET

- Пример получения состояния NET

Данный примре проверяет и выводит различные состояния/статусы NET.

```
$pid = pid_open("/mmap/net0");                  // open NET 0
echo pid_ioctl($pid, "get hwaddr"), "\r\n";     // get MAC address
echo pid_ioctl($pid, "get ipaddr"), "\r\n";     // get IP address
echo pid_ioctl($pid, "get netmask"), "\r\n";    // get subnet mask
echo pid_ioctl($pid, "get gwaddr"), "\r\n";     // get gateway address
echo pid_ioctl($pid, "get nsaddr"), "\r\n";     // get name server address
echo pid_ioctl($pid, "get mode"), "\r\n";       // get Ethernet mode
echo pid_ioctl($pid, "get speed"), "\r\n";      // get link speed
pid_close($pid);                                // close NET 0
```

5 TCP

TCP, отвечающий за передачу по TCP/IP, яввяется одним из самых фундаментальных протоколов наряду с широко используемым протоколом UDP. Этот протокол содержит фазу соединения и обеспечивает целостность данных.

5.1 Этапы использования TCP

Основные этапы использования TCP выглядят следующим образом:

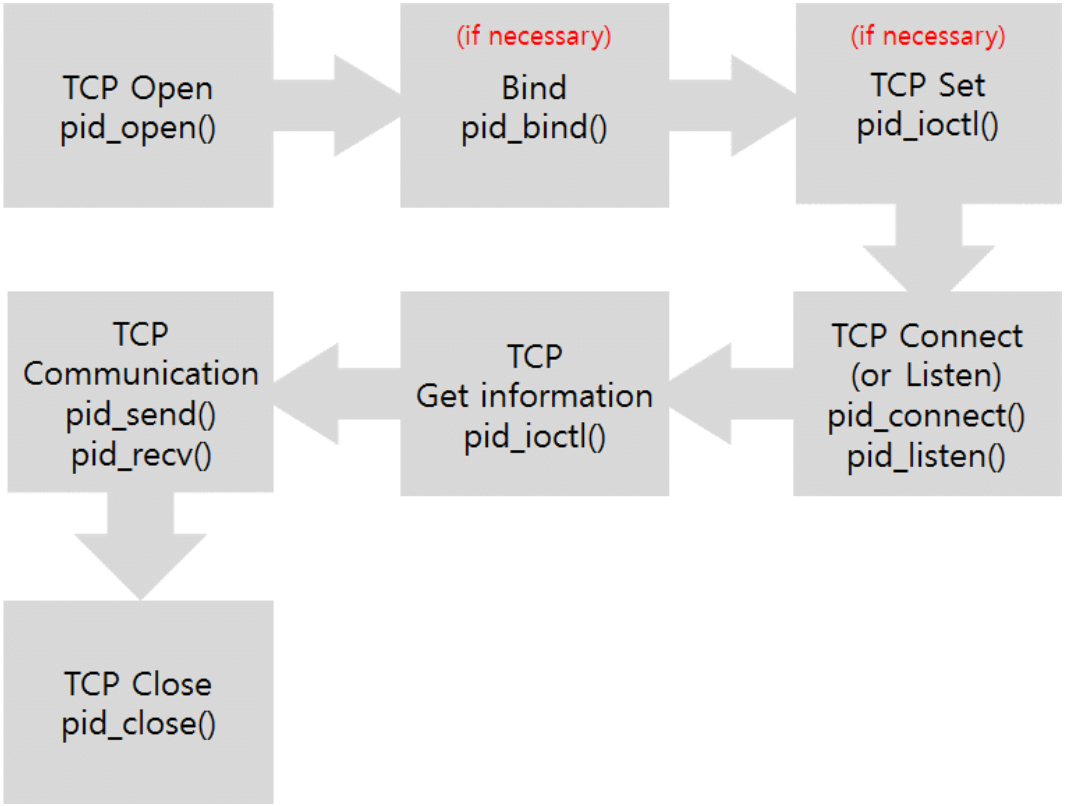


Рисунок 5-1 Этапы использования TCP

☞ ***В случае установки устройства на сервер TCP этап «привязки» не может быть опущена.***

5.2 Открытие TCP

Для открытия сессии TCP необходима функция pid_open.

```
$pid = pid_open("/mmap/tcp0");                    // open TCP 0
```

☞ ***Обратитесь к Приложению для подробной информации по TCP в зависимости от типа продукта.***

5.3 Настройка TCP

Перед использованием TCP может потребоваться настройка некоторых параметров. В протоколах SSL, SSH, TELNET или в особенности веб-сокета, перед подключением требуется настройка SSL при помощи команды pid_ioctl.

```
pid_ioctl($pid, "set ITEM VALUE");
```

ITEM (элемент) означает установку позиций и VALUE – возможное значение элемента.

5.3.1 Доступные элементы TCP

ITEM	VALUE		Описание
nodelay	0		Enable Nagle algorithm
	1		Disable Nagle algorithm
api	ssl		Use SSL
	ssh		Use SSH server
	telnet		Use TELNET server
	ws		Use Web Socket server
ssl method	ssl3_client		SSL client (SSL 3.0)
	tls1_client		SSL client (TLS 1.0)
	ssl3_server		SSL server (SSL 3.0)
	tls1_server		SSL server (TLS 1.0)
ssh auth	accept		Accept SSH authorization
	reject		Reject SSH authorization
ws	path	PATH	Set path of web socket URI
	mode	0	Set data type of web socket: text
		1	Set data type of web socket: binary
	proto	PROTOCOL	Set protocol of web socket
	origin	ADDR	Specify a host to allow connection

Таблица 5-1 Доступные элементы TCP

Алгоритм TCP Nagle предназначен для улучшения эффективности передачи данных за счет сокращения числа сегментов. Таким образом, это может сопровождаться небольшой задержкой.

☞ ***Примечание: Элементы команды "set api" доступны только для TCP от 0 до 3. Кроме того, невозможно установить другой режим api mode после установки TCP-устройства в один из режимов до перезагрузкой продукта.***

5.3.2 Как использовать SSL

PHPoC может быть SSL-сервером или клиентом благодаря команде "set api ssl". В примере ниже показано, как использовать его в качестве SSL-сервера.

- Пример SSL-сервера

```
$port = 1470;                                   // port number
$pid = pid_open("/mmap/tcp0");                  // open TCP 0
pid_ioctl($pid, "set api ssl");                 // set api to SSL
pid_ioctl($pid, "set ssl method tls1_server");  // set SSL server mode
pid_bind($pid, "", $port);                      // binding
pid_listen($pid);                               // listen TCP connection
do
        $state = pid_ioctl($pid, "get state");
while(($state != SSL_CLOSED) && ($state != SSL_CONNECTED));

if($state == SSL_CONNECTED)
{
        echo "Connection has been established!\r\n";
        pid_close($pid);                        // close TCP connection
}
```

☞ ***Перед тем, как использовать SSL-сервер, необходимо создать сертификат в PHPoC. Создайте или сохраните сертификат для вашего продукта при помощи Отладчика PHPoC.***

В следующем примере показано, как использовать PHPoC в качестве клиента SSL.

- Пример SSL-клиента

```
$addr = "10.1.0.2";                             // server's IP address
$port = 1470;                                   // server's port number
$pid = pid_open("/mmap/tcp0");                  // open TCP 0
pid_ioctl($pid, "set api ssl");                 // set api to SSL
pid_ioctl($pid, "set ssl method tls1_client");  // set SSL client mode
pid_connect($pid, $addr, $port);                // connect to TCP server
do
        $state = pid_ioctl($pid, "get state");
while(($state != SSL_CLOSED) && ($state != SSL_CONNECTED));

if($state == SSL_CONNECTED)
{
        echo "Connection has been established!\r\n";
        pid_close($pid);                        // close TCP connection
}
```

☞ ***Связь SSL не может быть выполнена в случае нехватки памяти, вызванной увеличением памяти PHPoC.***

5.3.3 Как использовать TELNET

PHPoC можно установить как TELNET-сервер при помощи команды "set api telnet". Ниже приведен пример TELNET-сервера.

- Пример TELNET-сервера

```
$port = 23;                                     // port number
$pid = pid_open("/mmap/tcp0");                  // open TCP 0
pid_ioctl($pid, "set api telnet");              // set api to TELNET
pid_bind($pid, "", $port);                      // binding
pid_listen($pid);                               // listen TCP connection
do
        $state = pid_ioctl($pid, "get state");
while(($state != TCP_CLOSED) && ($state != TCP_CONNECTED));

if($state == TCP_CONNECTED)
{
        pid_send($pid, "Welcome to PHPoC TELNET server\r\n");
        echo "Connection has been established!\r\n";
        pid_close($pid);                        // close TCP connection
}
```

В приведенном выше примере PHPoC прослушивает TELNET-соединения с клиентов. После установки соединения выводится приветственное сообщение, и сообщение закрывается.

5.3.4 Как использовать SSH-сервер

PHPoC может быть установлен как SSH-сервер путем использования команды "set api ssh". В следующем примере показано, как создать SSH-сервер.

- Пример SSH-сервера

```
$port = 22;                                     // port number
$pid = pid_open("/mmap/tcp0");                  // open TCP 0
pid_ioctl($pid, "set api ssh");                 // set api to SSH
pid_bind($pid, "", $port);                      // binding
pid_listen($pid);                               // listen TCP connection
while(1)
{
        $state = pid_ioctl($pid, "get state");
        if($state == SSH_AUTH)
        {
                $username = pid_ioctl($pid, "get ssh username");
                $password = pid_ioctl($pid, "get ssh password");
                echo "$username / $password\r\n";
                pid_ioctl($pid, "set ssh auth accept");
        }
        if($state == SSH_CONNECTED)
        {
                pid_send($pid, "Welcome to PHPoC SSH server\r\n");
                echo "Connection has been established!\r\n";
                pid_close($pid);
                break;
        }
}
```

В примере выше PHPoC прослушивает SSH-соединения от клиентов. После того, как соединение установлено, печатается имя пользователя и пароль клиента. После этого выводится приветственное сообщение и соединение закрывается.

☞ ***Процесс аутенфикации, отвечающий за идентификацию пользователя, должен быть реализован в пользовательском скрипте.***

5.3.5 Как использовать сервер веб-сокета

PHPoC может быть сервером веб-сокета при помощи команды "set api ws". В следующем примере показано, как использовать сервер веб-сокета.

- Пример сервера веб-сокета

В данном примере выполняется прослушивание TCP-соединения от клиентов. После установления соединения PHPoC выводит данные, полученные от клиентов, включая количество полученных данных.

```
$pid = pid_open("/mmap/tcp0");              // open TCP 0
pid_ioctl($pid, "set api ws");              // set api to web socket
pid_ioctl($pid, "set ws path test");        // set URI path: /test
pid_ioctl($pid, "set ws mode 0");           // set transmission mode: text
pid_ioctl($pid, "set ws origin 10.1.0.1");  // specify a host to allow connection
pid_ioctl($pid, "set ws proto myproto");    // set protocol: myproto

pid_bind($pid, "", 0);                      // binding: default(80)

$rwbuf = "";
$count = 1;

while(1)
{
    if(pid_ioctl($pid, "get state") == TCP_CLOSED)
        pid_listen($pid);                                   // listen TCP connection

    $rlen = pid_ioctl($pid, "get rxlen");
    if($rlen)
    {
        pid_recv($pid, $rwbuf);
        echo "$rwbuf\r\n";
        pid_send($pid, "echo reply $count");            // send data back
        $count++;
    }
}
pid_close($pid);
```

☞ ***Вы можете создать более мощный веб-интерфейс, изменив приведенный выше сценарий клиентского веб-сокета (index.php) и сценарий веб-сервера.***

☞ ***Требуется использовать браузер, который поддерживает веб-сокет.***

5.4 Соединение TCP

5.4.1 TCP-клиент (Активное соединение)

Активное соединение означает отправку TCP-соединения на TCP-сервер и этот хост называется TCP-клиентом. Для выполнения TCP-клиента используется функция pid_connect.

```
pid_connect($pid, $addr, $port);
```

Аргумент $addr является IP-адресом TCP-сервера, а $port – номером порта.

- Пример TCP-клиента

```
$pid = pid_open("/mmap/tcp0");      // open TCP
$addr = "10.1.0.2";                 // IP address of TCP server
$port = 1470;                       // TCP port
pid_connect($pid, $addr, $port);    // active TCP connection
sleep(25);
pid_close($pid);
```

5.4.2 TCP-сервер (Пассивное соединение)

Пассивное соединение означает прослушивание пакета запроса TCP-соединения от TCP-клиента и называется TCP-сервером. Для выполения TCP-сервера требуются функции pid_bind и pid_listen.

```
pid_bind($pid, "", $port);
pid_listen($pid[, $backlog]);
```

Аргумент $port – это номер TCP-порта.

- Пример TCP-сервера

```
$pid = pid_open("/mmap/tcp0");      // open TCP
$port = 1470;                       // TCP port number
pid_bind($pid, "", $port);          // bind with the port number
pid_listen($pid);                   // passive TCP connection
sleep(25);
pid_close($pid);
```

5.5 Получение статуса/состояния TCP

Для получения состояния TCP необходима команда функции pid_ioctl.

```
$return = pid_ioctl($pid, "get ITEM");
```

5.5.1 Доступные состояния/статусы TCP

ITEM	Описание	Возврат значений	Тип возврата
state	TCP session is closed	TCP_CLOSED	integer
	TCP session is connected	TCP_CONNECTED	integer
	TCP session waits for connection	TCP_LISTEN	integer
	SSL session is closed	SSL_CLOSED	integer
	SSL session is connected	SSL_CONNECTED	integer
	SSL session waits for connection	SSL_LISTEN	integer
	SSL session is closed	SSH_CLOSED	integer
	SSL session is connected	SSH_CONNETED	integer
	SSL session waits for connection	SSH_LISTEN	integer
	SSL authentication is completed	SSH_AUTH	integer
srcaddr	local IP address	e.g. 192.168.0.1	string
srcport	local port number	e.g. 1470	integer
dstaddr	peer IP address	e.g. 192.168.0.2	string
dstport	peer TCP number	e.g. 1470	integer
txbuf	size of send buffer[Byte]	e.g. 1152	integer
txfree	remaining send buffer size[Byte]	e.g. 1152	integer
rxbuf	size of receive buffer[Byte]	e.g. 1068	integer
rxlen	received data size[Byte]	e.g. 200	integer
ssh username	SSH user name	e.g. user	string
ssh password	SSH password	e.g. password	string

Таблица 5-2 Доступные состояния TCP

5.5.2 Состояние TCP-сессии

Проверка состояния соединения по TCP очень важна, потому что передача данных TCP осуществляется после этапа соединения/подключения. Существует три состояния сеанса: TCP_CLOSED, когда сеанс не подключен; TCP_CONNECTED, когда сеанс подключенн и TCP_LISETN, когда TCP-сервер прослушивает соединение. SSL и SSH также имеют три статуса/состояния: SSL_CLOSED, SSL_CONNECTED и SSL_LISTEN, а SSH имеет также дополнитеотное состояние об аутентификации (SSH_AUTH). Ниже показано, как получить состояние сеанса.

```
$state = pid_ioctl($pid, "get state");
```

☞ ***Неизвестное значение, которое не указано в таблице выше, может быть возвращено, если вы пытаетесь получить состояние, при котором PHPoC подключается или закрывает соединение. Обратите внимание, что не рекомендуется использовать эти значения в вашем сценарии, так как он может быть изменен в будущем.***

5.5.3 Оставшийся размер данных в буфере отправки

Оставшийся размер данных в буфере отправки может быть вычислен следующим образом:

```
Оставшийся размер данных в буфере отправки = размер буфера – оставшийся размер буфера
```

- Пример

В данном примере, PHPoC отправляет 8 байтов данных на сервер сразу после TCP-соединения. При отправке PHPoC выводит размер оставшихся данных в буфере отправки.

```
$tx_len = -1;
$pid = pid_open("/mmap/tcp0");                  // open TCP 0
do
{
        pid_connect($pid, "10.1.0.2", 1470);    // TCP active connection
        usleep(500000);
}
while(pid_ioctl($pid, "get state") != TCP_CONNECTED);
pid_send($pid, "01234567");                     // send 8 bytes
while($tx_len && (pid_ioctl($pid, "get state") == TCP_CONNECTED))
{
        $txbuf = pid_ioctl($pid, "get txbuf");  // get the size of send buffer
        // get the empty size of send buffer
        $txfree = pid_ioctl($pid, "get txfree");
        // calculate the size of remaining data in send buffer
        $tx_len = $txbuf - $txfree;
        echo "tx len = $tx_len\r\n";            // print the result
        usleep(10000);
}
pid_close($pid);                                // close TCP
```

5.5.4 Размер полученных данных

Ниже показано, как выяснить размер полученных данных из сокета TCP.

```
$rxlen = pid_ioctl($pid, "get rxlen[ $string]");
```

- Узнать размер полученных данных со строкой

Если после команды rxlen item указана строка, то функция pid_ioctl возвращает 0 до тех пор, пока строка не войдет в сокет TCP. После того, как строка войдет, функция возвращает весь размер данных, включая строку.

5.5.5 Оставшийся размер буфера приема

Оставшийся размер буфера приема может быть рассчитан следующим образом:

```
Оставшийся размер буфера приема = размер буфера – размер полученных данных
```

- Пример

В данном примере показано, как получить оставшийся размер буфера приема.

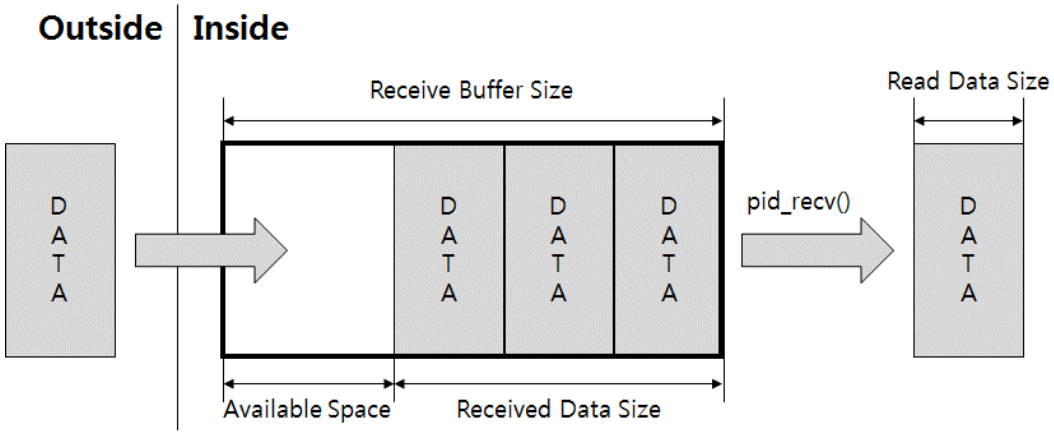
```
$rx_free = 1068;
$pid = pid_open("/mmap/tcp0");                  // open TCP 0
do
{
        pid_connect($pid, "10.1.0.2", 1470);    // TCP active connection
        usleep(500000);
}
while(pid_ioctl($pid, "get state") != TCP_CONNECTED);

while(($rx_free > 500) && (pid_ioctl($pid, "get state") == TCP_CONNECTED))
{
        $rxbuf = pid_ioctl($pid, "get rxbuf");    // get the size of receive buffer
        $rxlen = pid_ioctl($pid, "get rxlen");    // get the size of received data
        // calculate the available space of receive buffer
        $rx_free = $rxbuf - $rxlen;
        echo "rx free = $rx_free\r\n";            // print the result
        sleep(1);
}
pid_close($pid);                                  // close TCP
```

5.6 Связь TCP

5.6.1 Получение данных TCP

Данные, полученные из сети по протоколу TCP, сохранятюся в буфере приема. Функция pid_recv необходима для чтения данных.



Смеха 5-2 Получение данных TCP

Ниже показано, как использовать функцию pid_recv.

```
pid_recv($pid, $value[, $len]);
```

- Пример

В данном примере выполняется проверка и вывод полученных TCP-данных каждую секунду.

```
$rdata = "";
$pid = pid_open("/mmap/tcp0");                  // open TCP 0
pid_connect($pid, "10.1.0.2", 1470);            // TCP active connection
do
{
        sleep(1);
        $state = pid_ioctl($pid, "get state");    // get TCP session state
        $rxlen = pid_ioctl($pid, "get rxlen");    // get received data size
        $rlen = pid_recv($pid, $rdata, $rxlen); // receive data
        echo "rlen = $rlen / ";                   // print received data size
        echo "rdata = $rdata\r\n";                // print received data
        if($rlen)
                $rdata = "";                    // flush receive buffer
}
while($state == TCP_CONNECTED);
pid_close($pid);
```

5.6.2 Отправка TCP-данных

Данные, отправленные функцией pid_send, сохранятются в буфере отправки и передаются в сеть по протоколу TCP.

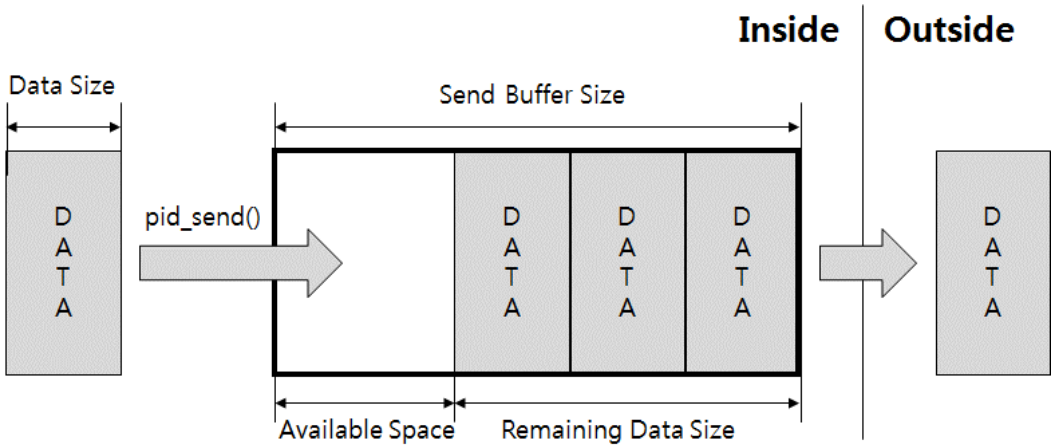


Схема 5-3 Отправка TCP-данных

Ниже показано, как использовать функцию pid_send.

```
pid_send($pid, $value[, $len]);
```

- Пример

Пример ниже показывает отправку данных в сеть по протоколу TCP, проверяя доступное пространство буфера отправки каждую секунду.

```
$sdata = "0123456789";
$pid = pid_open("/mmap/tcp0");                  // open TCP 0
pid_connect($pid, "10.1.0.2", 1470);            // TCP active connection
do
{
        sleep(1);
        $state = pid_ioctl($pid, "get state");    // get session state
        // get available space of send buffer
        $txfree = pid_ioctl($pid, "get txfree");
        $tx_len = pid_send($pid, $sdata, $txfree);    // send data
        echo "tx len = $tx_len\r\n";              // print size of send data
}
while($state == TCP_CONNECTED);
pid_close($pid);
```

Третий аргумент функции pid_send означает длину отправляемых данных. Во избежание потери данных, длина отправляемых данных должна быть меньше, чем оставшийся размер данных буфера. Настоятельно рекомендуется проверить оставшийся размер буфера отправки перед отправкой данных.

6 UDP

Хотя UDP не обеспечивает целостность данных и фазу соединения, он обладает хорошими функциями, такими как простая передача заголовка и подсказки.

6.1 Этапы использования UDP

Основные этапы использования UDP выглядят следующим образом:

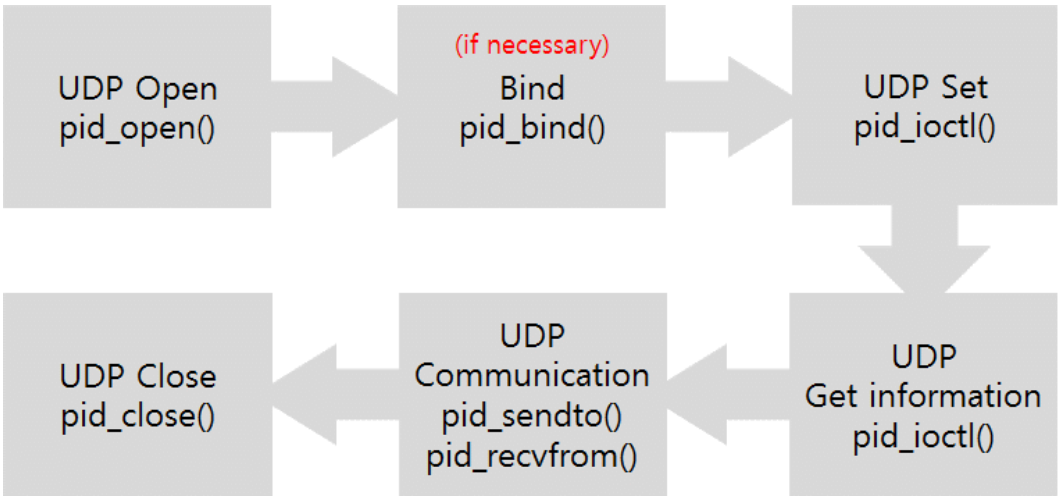


Схема 6-1 Этапы использования UDP

☞ ***Соединительный сокет (Binding socket) может быть пропущен, если нет требований к предварительной настройке или передачи данных.***

6.2 Открытие UDP

Для открытия UDP необходима функция pid_open.

```
$pid = pid_open("/mmap/udp0");        // open UDP 0
```

☞ ***Обратитесь к Приложению для подробной информации о UDP в зависимости от типа продукта.***

6.3 Связывание

Для связывания, которое использует функцию pid_bind, необходимо указать IP-адрес назначения для получения данных из сети через UDP.

```
$pid = pid_bind($pid, $addr, $port);
```

Аргумент $addr является IP-адресом, а $port – номером порта для привязки. Когда для IP-адреса указывается пустая строка (""), PHPoC считает, что это значение является текущим локальным IP-адресом..

☞ ***Знакчение пустой строки ("") является единственным параметром для аргумента $addr функции связывания (bind).***

- Пример связывания

```
$pid = pid_open("/mmap/udp0");      // open UDP
$port = 1470;                       // UDP port number
pid_bind($pid, "", $port);          // binding
```

6.4 Настройка UDP

Перед отправкой данных UDP нужно указать IP-адрес и номер порта получателя. Если все указано, то эти параметры могут быть опущены в 4-м и 5-м аргументах функции pid_sendto.

Настройка команды зшв_шщсед необходима для настройки UDP.

```
pid_ioctl($pid, "set ITEM VALUE");
```

ITEM (элемент) означает установку позиций, а VALUE – возможное значение элемента.

6.4.1 Доступные элементы UDP

Элемент	Значение	Описание
dstaddr	e.g. 10.1.0.2	destination IP address
dstport	e.g. 1470	destination port number

Таблица 6-1 Доступные элементы UDP

- Пример настройки UDP

```
$pid = pid_open("/mmap/udp0");              // open UDP 0
pid_bind($pid, "", 1470);                   // binding
pid_ioctl($pid, "set dstaddr 10.1.0.2");    // destination IP address
pid_ioctl($pid, "set dstport 1470");        // destination port number
```

6.5 Получение состояния UDP

Для получения состояния UDP команда функции pid_ioctl.

```
$return = pid_ioctl($pid, "get ITEM");
```

6.5.1 Доступные состояния UDP

Элемент	Описание	Возврат значений	Тип возврата
srcaddr	source IP address	e.g. 192.168.0.1	String
srcport	source port number	e.g. 1470	Integer
dstaddr	destination IP address	e.g. 192.168.0.2	String
dstport	destination port number	e.g. 1470	Integer
rxlen	received data size[Byte]	e.g. 200	Integer

Таблица 6-2 Доступные состояния UDP

6.5.2 Размер полученных данных

Чтобы получить размер полученных данных, требуется коменда "get rxlen" функции pid_ioctl.

```
$rxlen = pid_ioctl($pid, "get rxlen");
```

- Пример

Данный пример закрывается после вывода информации о полученном размере данных, если данные поступают из сети. Переодически проверяется, есть ли полученные данные.

```
$rbuf = "";
$pid = pid_open("/mmap/udp0");                  // open UDP 0
pid_bind($pid, "", 1470);                       // binding
do
{
        $rxlen = pid_ioctl($pid, "get rxlen");  // get received data size
        if($rxlen)
        {
                pid_recvfrom($pid, $rbuf, $rxlen);      // receive data
                echo "$rxlen bytes\r\n";         // print size of received data
        }
        usleep(100000);
}while($rxlen == 0);                            // while receiving no data
pid_close($pid);
```

6.6 Связь UDP

6.6.1 Получение данных UDP

Для получения данных из сети посредством UDP требуется функция pid_recvfrom. Есть 2 буфера приема UDP, ниже показан их принцип работы.

☞ ***Обратитесь к Приложению для подробной информации о размере буфера приема UDP в зависимости от типа продукта.***

- Прием данных UDP из сети

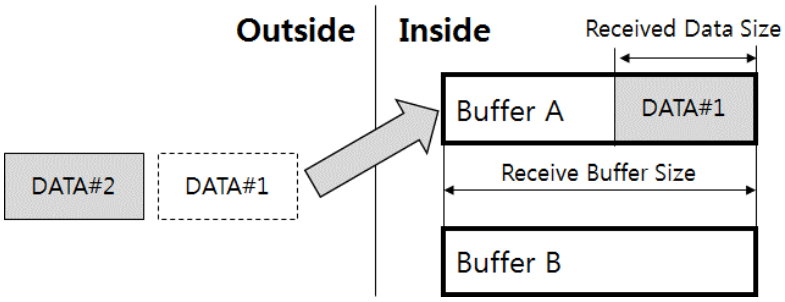


Схема 6-2 Получение данных UDP

- Чтение данных UDP из буфера приема

После прочтения данных из буфера приема, PHPoC очищает буфер путем вызова функции pid_recvfrom.

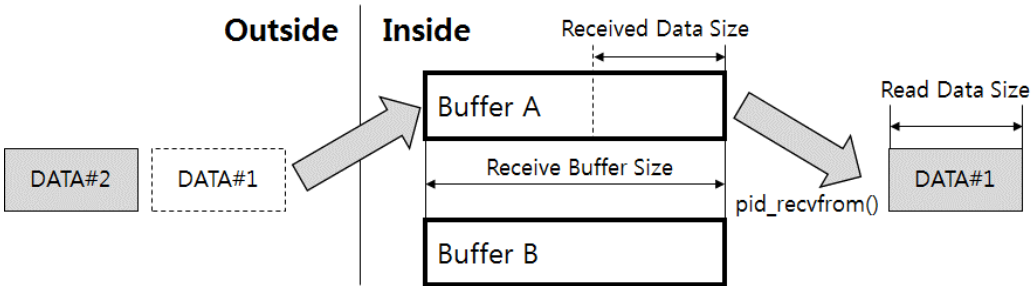


Схема 6-3 Чтение данных UDP из буфера приема

- Чтение данных размером меньше, чем полученные.

Отсавшиеся данные после считывания (непрочитанные) будут потеряны после приема.

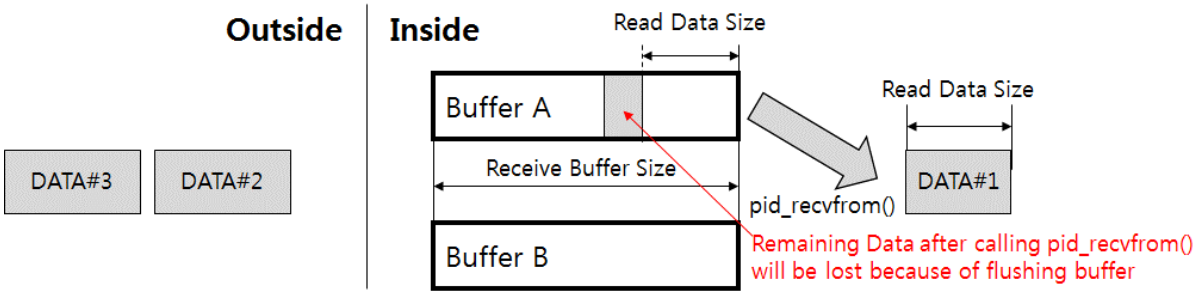


Схема 6-4 Чтение данных размером меньше, чем полученные

- **Потеря данных по причине недоступного буфера приема**

  Хотя каждый из двух буферов приема имеет данные, которые, в свою очередь, содержат непрочитанные данные, последующие данные из сети не смогут быть получены. Поэтому рекомендуется сразу же после считывания полученных данных просмотреть данные в принятом буфере как можно скорее.

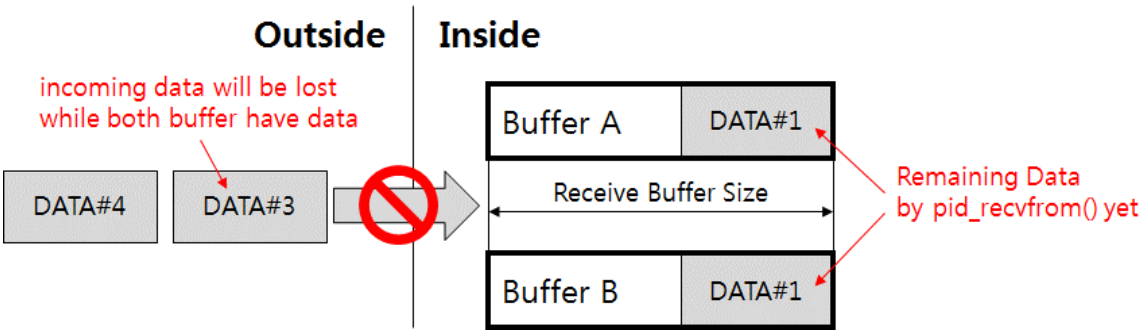


Схема 6-5 Потеря данных по причине недоступного буфера приема

- **Пример**

Пример ниже выводит полученные данные UDP, каждую секунду проверяя, поступают ли данные из сети.

```
$rbuf = "";
$pid = pid_open("/mmap/udp0");                    // open UDP 0
pid_bind($pid, "", 1470);                         // binding
do
{
        $rxlen = pid_ioctl($pid, "get rxlen");    // get received data size
        if($rxlen)
        {
                pid_recvfrom($pid, $rbuf, $rxlen); // receive data
                echo "$rbuf\r\n";                 // print received data
        }
        usleep(100000);
}while(1)                                         // infinite loop
```

6.6.2 Отправка данных UDP

Для отправки данных UDP необходима функция pid_sendto.

```
$sdata = "01234567";
$pid = pid_open("/mmap/udp0");                              // open UDP 0
$slen = pid_sendto($pid, $sdata, 8, 0, "10.1.0.2", 1470);   // send data
echo "slen = $slen\r\n";                                    // print size of send data
pid_close($pid);
```

7 ST

PHPoC снабжен устройством ST (программного таймера).

7.1 Этапы использования ST

Основыне этапы использования ST выглядят следующим образом:

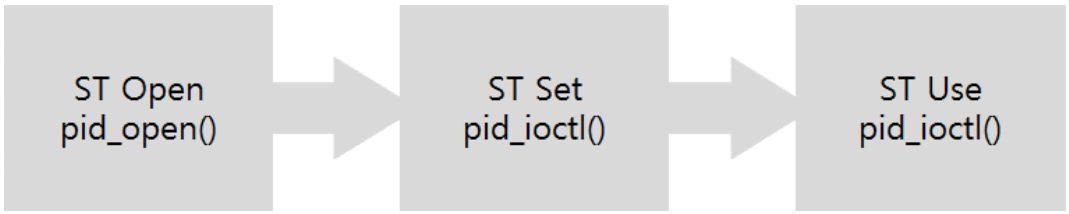


Схема 7-1 Этапы использования ST

7.2 Открытие ST

Для открытия ST необходима функция pid_open.

```
$pid = pid_open("/mmap/st0");          // open ST 0
```

☞ ***Обратитесь к Приложению для подробной информации о ST в зависимости от типа продукта.***

7.3 Установка и использование ST

Для использвоания ST необходима функция pid_ioctl. Имеются 4 режима.

Режим	Описание
Free mode	Нормальный режим счетчика
Output Pulse mode	Режим вывода импульсного сигнала через указанный вывод
Output Toggle mode	Режим вывода сигнала переключения через указанный вывод
Output PWM mode	Режим вывода бесконечного импульса через указанный вывод

Таблица 7-1 Режимы ST

7.3.1 Общие команды

Команды, перечисленные в таблице ниже, используются во всех режимах ST.

Ком.	Подкоманды			Описание
set	mode	free		set mode: free
		output	pulse	set mode: output Pulse
			toggle	set mode: output Toggle
			pwm	set mode: output infinite pulse
	div	sec		set unit: second
		ms		set unit: millisecond
		us		set unit: microsecond
reset	-			reset
get	state			get current state
start	-			start
stop	-			stop

Таблица 7-2 Общие команды

- Настройка ST режима

ST обеспечивает как нормальный режим счетчика (он же свободный), так и режим выходного сигнала. Существуют 3 режимы выхода: импульсный (pulse), режим переключателя (toggle) и режим pwm. Pwm – это бесконечный импульсный режим. Значением по умолчанию для режима ST является свободный(нормальный) режим. В таблице ниже показано, как установить ST для каждого режима.

Режим	Синтаксис
free	pid_ioctl($pid, "set mode free");
pulse	pid_ioctl($pid, "set mode output pulse");
toggle	pid_ioctl($pid, "set mode output toggle");
pwm	pid_ioctl($pid, "set mode output pwm");

Таблица 7-3 Настройка ST режима

- Настройка ST-юнита

Ниже можно увидеть 3 юнита, которые предоставляет ST. Значение по умолчанию – миллисекунда.

Юнит	Синтаксис
Секунда	pid_ioctl($pid, "set div sec");
Миллисекунда	pid_ioctl($pid, "set div ms");
Микросекунда	pid_ioctl($pid, "set div us");

Таблица 7-4 Настройка ST-югита

- Сброс

Данная команда немедленно останавливает работу ST и выполняет сброс.

Команда	Синтаксис
reset	pid_ioctl($pid, "reset");

Таблица 7-5 Сброс

- Получение состояния

Данная команда получает текущее состояние ST.

Команда	Синтаксис
get state	pid_ioctl($pid, "get state");

Таблица 7-6 Получение состояния

Возвращаемые значения для данной команды следующие:

Возврат значений	Описание
0	Stopped
1 ~ 5	Running

Таблица 7-7 Возвращаемые значения получения состояния

- Запуск

Данная команда запускает ST.

Команда	Синтаксис
start	pid_ioctl($pid, "start");

Таблица 7-8 Запуск

- Остановка

Данная команда немедленно прекразает работу ST. В выходных режимах состояние выходного пин сохраняет текущее состояние.

Команда	Синтаксис
stop	pid_ioctl($pid, "stop");

Таблица 7-9 Остановка

7.3.2 Свободный режим

Свободным режимом называют обычный режим счетчика ST.

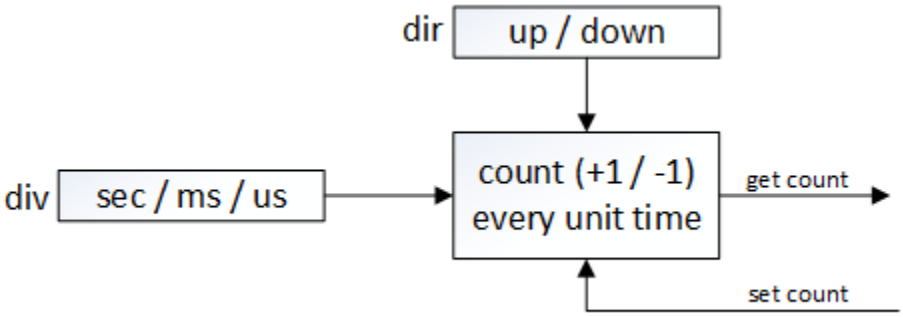


Схема 7-2 Диаграмма свободного режима

В свободном режиме доступны следующие функции pid_ioctl:

Ком.	Подкоманды		Описание
Set	Mode	free	set mode: free mode
	Div	sec	set unit: second
		ms	set unit: millisecond
		us	set unit: microsecond
	Dir	up	set counter direction: up counter
		down	set counter direction: down counter
	Count	[T]	set default count value in down counter mode
reset	-		reset
Get	count		get count value
	state		get current state
start	-		start
stop	-		stop

Таблица 7-10 Команды свободного режима

- Настройка направления счетчиком

ST может использовать как счетчик вверх, так и счетчик вниз. Значением по умолчанию для этого элемента является счетчик вверх.

Направление	Синтаксис
Счетчик вверх	pid_ioctl($pid, "set dir up");
Счетчик вниз	pid_ioctl($pid, "set dir down");

Таблица 7-11 Настройка направления счетчика

- Установка счетчика

Когда счетчик ST в режиме вниз, то можно установить значение счетчика по умолчанию. Ниже показано, установить значение счетчика:

Команда	Синтаксис
set count	pid_ioctl($pid, "set count T");

Таблица 7-12 Установка счетчика

Хотя вы устанавливаете значение счета T, оно не применяется, поа ST работает в режиме счетчика. Значение счетчика начинается с нуля в режиме счетчика. Доступные значения счетчика в нижнем режиме выглядят следующим образом:

Команда	Доступные значения счетчика
set count	0 ~ (2 to the power of 64 - 1)

Таблица 7-13 Доступные значения счетчика

- Получение значения счетчика

Команда "get count" возвращает текущее значение счетчика.

Команда	Синтаксис
get count	pid_ioctl($pid, "get count");

Таблица 7-14 Получение значения счетчика

7.3.3 Примеры свободного режима

Команда "get count" позволяет получить текущее значенеи счетчика ST. Данный пример показывает прошедшее время после работы таймера.

```
$tick = pid_ioctl($pid, "get count");
```

- Пример счетчика вверх

  Данный пример ST показывает установку счетчика вврех и выводит значение счетчика каждую секунду.

```
$pid = pid_open("/mmap/st0");                  // open ST 0
pid_ioctl($pid, "set mode free");              // set mode: free
pid_ioctl($pid, "set div sec");                // set unit: second
pid_ioctl($pid, "set dir up");                 // set direction: up counter
pid_ioctl($pid, "start");                      // start ST
for($i=0; $i<10; $i++)
{
        $value = pid_ioctl($pid, "get count"); // read the count value
        echo "$value\r\n";                     // print the count value
        sleep(1);
}
pid_close($pid);
```

- Пример счетчика вниз

  В данном примере ST устанавливает счетчик вниз с начальным значением счета и выводит значение счетчика каждую секунду.

```
$pid = pid_open("/mmap/st0");                  // open ST 0
pid_ioctl($pid, "set mode free");              // set mode: free
pid_ioctl($pid, "set div sec");                // set unit: second
pid_ioctl($pid, "set dir down");               // set direction: down counter
pid_ioctl($pid, "set count 10");               // set count value: 10
pid_ioctl($pid, "start");                      // start ST
for($i = 0; $i < 10; $i++)
{
        $value = pid_ioctl($pid, "get count"); // read the count value
        echo "$value\r\n";                     // print the count value
        sleep(1);
}
pid_close($pid);
```

7.3.4 Режим переключения (Toggle Mode)

Данный режим переключает состояние заданного выходного контакта ST.

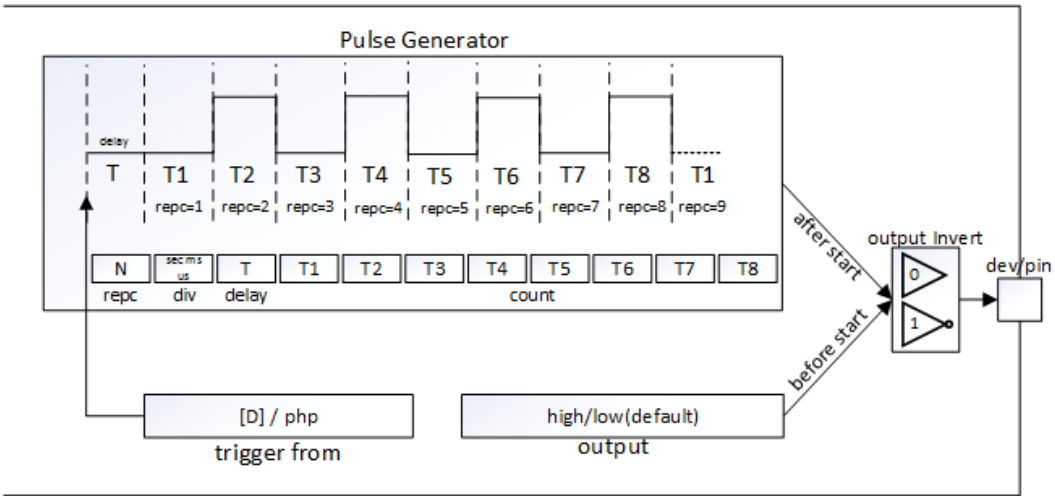


Схема 7-3 Диаграмма режима переключения

Доступные команды в режиме переключения выглядят следующим образом:

Ком.	Подкоманда				Описание
set	mode	output	toggle		set mode: toggle
	div	sec			set unit: second
		ms			set unit: millisecond
		us			set unit: microsecond
	output	low			output LOW
		high			output HIGH
		dev	io3/io4	#pin	set output device and pin
		invert	0		not invert output
			1		invert output
	count	[T1] … [T8]			set output timing parameters
	delay	[D]			set delay before output signal
	repc	[N]			set repeat count
	trigger	from	st#		set trigger target: st0 ~ st7
			php		set trigger target: none
reset	-				reset
get	state				get state
	repc				get remaining repeat count
start	-				start
stop	-				stop

Таблица 7-15 Команды режима переключения

- Установка вывода

Подкоманды команды "set output" в режиме переключения выглядят следующим образом:

Подкоманда	Синтаксис
set output pin	pid_ioctl($pid, "set output dev io3 0"); // pin 0 of io3
output HIGH	pid_ioctl($pid, "set output high");
output LOW	pid_ioctl($pid, "set output low");

invert output	pid_ioctl($pid, "set output invert 0"); // normal output
	pid_ioctl($pid, "set output invert 1"); // inverted output

Таблица 7-16 Установка вывода

Все подкоманды выполяются сразу после выполнения каждой командной строки.

- Настройка задержки

Данная команда предназначена для задержки перед тем, как PHPoC выведет сигнал. Еденица задержки зависит от устройства, которое устанавливается командой "set div".

Команда	Синтаксис
set delay	pid_ioctl($pid, "set delay D");

Таблица 7-17 Настройка задержки

- Установка счетчика повтора

Данная команда предназначена для установки количества повторений на выходе. Вы можете установить любые значения от нуля до 1 миллиарда для счетчика повторений N. Если вы не укажете N, то по умолчанию он будет равен нулю. Установка этого значения на ноль означает максимальное количество повторений (1 миллиард).

Команда	Синтаксис
set repc	pid_ioctl($pid, "set repc N");

Таблица 7-18 Установка счетчика повторений

- Установка значений счетчика

Данная команда предназначена для определения времени выходного сигнала. В режиме переключения действительное число значений счета может колебаться от 1 до 8. Как использовать данную команду:

Команда	Синтаксис
set count	pid_ioctl($pid, "set count T1 T2 ... T8");

Таблица 7-19 Установка значений счетчика

Доступные для счетчиков значения в режиме переключения следующие:

Еденица	Доступные значения для счета (10μs ~ полчаса)
Микросекунда	10 ~ 1,800,000,000
Миллисекунда	1 ~ 1,800,000
Секунда	1 ~ 1,800

Таблица 7-20 Доступные значения для счета

На приведенной ниже схеме показана форма сигнала в случае установки только одного значения T1 с задержкой D.

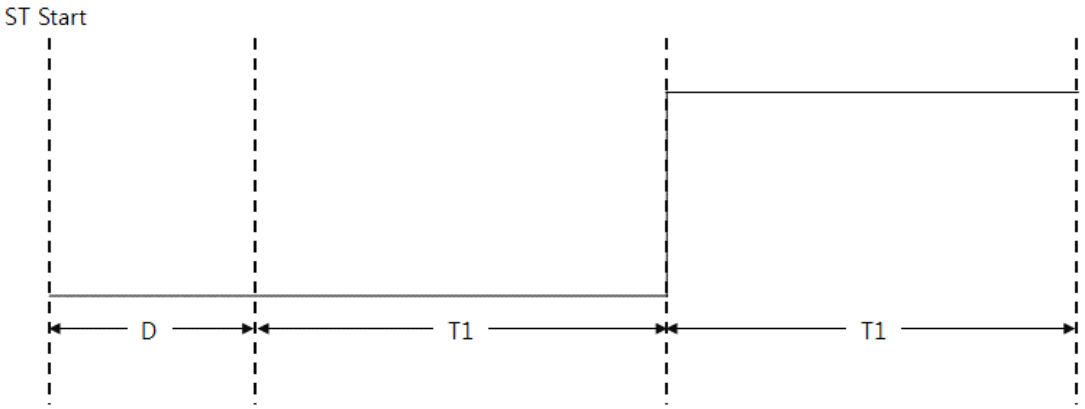


Схема 7-4 Форма сигнала в режиме переключения (LOW -> HIGH)

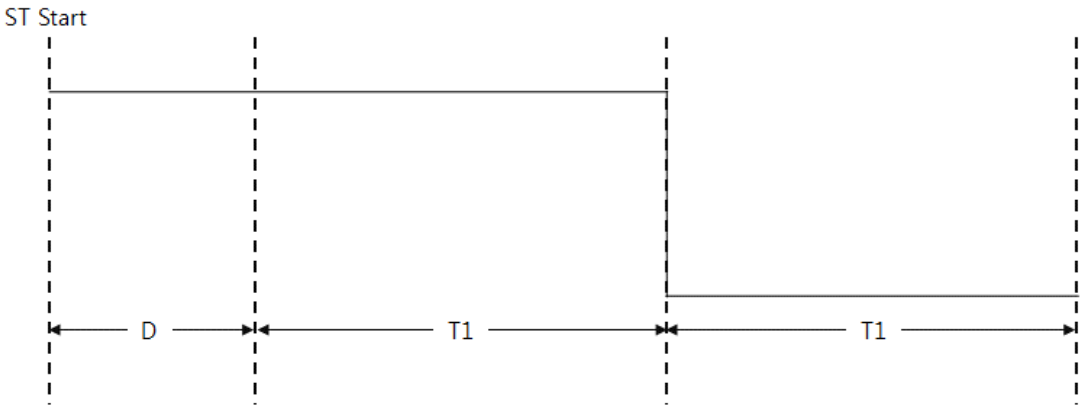


Схема 7-5 Форма сигнала в режиме переключения (HIGH -> LOW)

Если вы задали 2 значения или больше, каждое значение подсчета будет использоваться по порядку. Когда количество повторений больше числа отсчетов установок, то значения счетчика снова будет использованы из первого значения счета. Например, форма сигнала установки – это 3 значения счета (T1, T2 и T3) с 4-мя повторными отсчетами (включая задержку D), которые выглядят следующим образом:

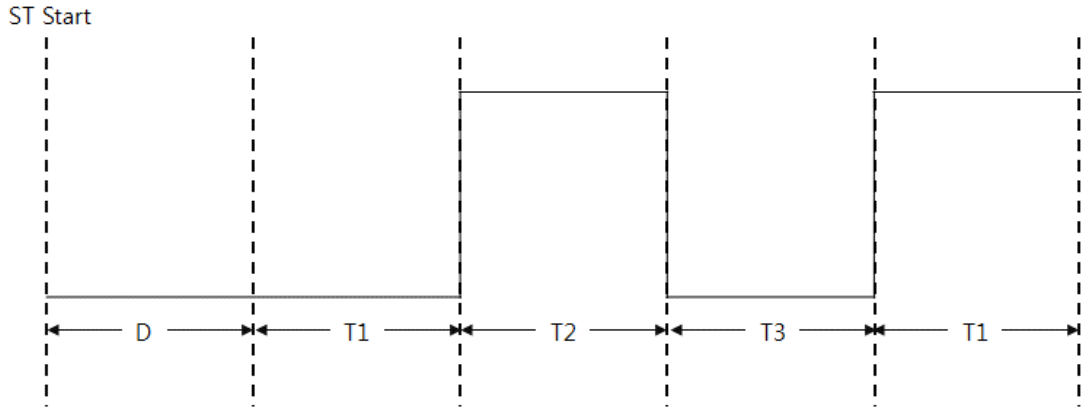


Схема 7-6 Форма сигнала режима переходника с множеством значений

- Настройка триггера

Данная команда используется, когда вы хотите синхронизировать время начала ST с другим ST. Целью триггера должно быть одно из устройств ST.

Цель	Синтаксис
ST(st0/1...)	pid_ioctl($pid, "set trigger from st0");
Php	pid_ioctl($pid, "set trigger from php");

Таблица 7-21 Настройка триггера

Значением по умолчанию для цели триггера является «php»(без цели).

- Получение счетчика повторов

Команда «get repc» предназначена для чтения оставшегося количества повторов, которые будут выполнены.

Команда	Синтаксис
get repc	pid_ioctl($pid, "get repc");

Таблица 7-22 Получение счетчика повторов

7.3.5 Пример режима переключения

Режим переключения переключает выходные сигналы.

- Пример режима переключения

```
$pid = pid_open("/mmap/st0");                 // open ST 0
pid_ioctl($pid, "set div sec");               // set unit: second
pid_ioctl($pid, "set mode output toggle");    // set mode: toggle
pid_ioctl($pid, "set output dev io3 0");      // set output device / pin: io3 / 0
pid_ioctl($pid, "set repc 1");                // set repeat count: 1
pid_ioctl($pid, "set count 1");               // set count: T1 only
pid_ioctl($pid, "start");                     // start ST
while(pid_ioctl($pid, "get state"));
pid_close($pid);
```

Значение «set count» - это количество времени между стартом ST и сигналом переключения выхода. На схеме ниже показана форма сигнала вышеприведенного примера.

Схема 7-7 Пример режима переключения

- Пример повторяющегося режима переключения

```
$pid = pid_open("/mmap/st0");                 // open ST 0
pid_ioctl($pid, "set div sec");               // set unit: second
pid_ioctl($pid, "set mode output toggle");    // set mode: toggle
pid_ioctl($pid, "set output dev io3 0");      // set output device / pin: io3 / 0
pid_ioctl($pid, "set repc 3");                // set repeat count: 3
pid_ioctl($pid, "set count 1 2 1");           // set count values: 1, 2 and 1
pid_ioctl($pid, "start");                     // start ST
while(pid_ioctl($pid, "get state"));
pid_close($pid);
```

В приведенном выше примере заданы 3 значения счетчика (T1, T2 и T3), а это 1, 2 и 1 секунды. Форма сигнала следующая:

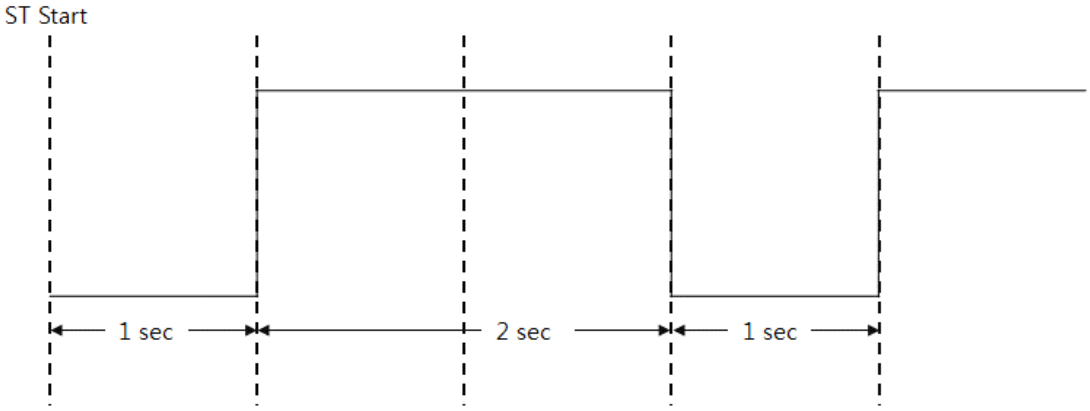


Схема 7-8 Пример повторяющегося режима переключения

7.3.6 Импульсный режим

Импульсный режим выводит прямоугольные волны.

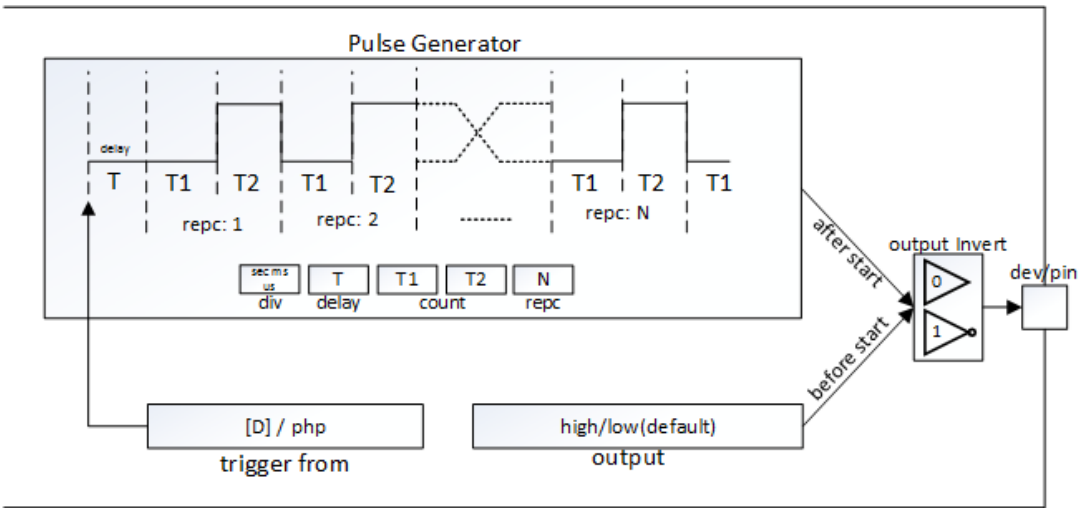


Схема 7-9 Диаграмма импульсного режима

В импульсном режиме доступны следующие команды:

Ком.	Подкоманды				Описание
set	mode	output	pulse		set mode: pulse
	div	sec			set unit: second
		ms			set unit: millisecond
		us			set unit: microsecond
	output	low			output: LOW
		high			output: HIGH
		dev	io3/4	#pin	set output device and pin
		invert	0		not invert output
			1		invert output
	count	[T1] [T2]			set output timing parameters
	delay	[D]			set delay
	repc	[N]			set repeat count
	trigger	from	st#		set trigger target: st0 ~ st7
			php		set trigger target: none
reset	-				reset
get	state				get state
	repc				get remaining repeat count
start	-				start
stop	-				stop

Таблица 7-23 Команды импульсного режима

- Настройка вывода

Подкоманды команды «set output» в импульсном режиме следующие:

Плдкоманды	Синтаксис
set output pin	pid_ioctl($pid, "set output dev io3 0"); // pin 0 of io3
output HIGH	pid_ioctl($pid, "set output high");
output LOW	pid_ioctl($pid, "set output low");

invert output	pid_ioctl($pid, "set output invert 1"); // normal output
	pid_ioctl($pid, "set output invert 0"); // inverted output

Таблица 7-24 Настройка вывода

Все подкоманды выполняются сразу после выполнения каждой командной строки.

- Настройка задержки

Данная команда предназначена для задержки перед тем, как PHPoC выведет сигнал. Еденица задержки зависит от устройства, которое устанавливается командой «set div».

Команда	Синтаксис
set delay	pid_ioctl($pid, "set delay D");

Таблица 7-25 Настройка задержки

- Настройка количества повторений

Данная команда предназначена для определения времени для выходного сигнала. Вы можете установить любые значения от нуля до 1-го миллиарда для счетчика N. Если N не указан, то автоматически будет установлен ноль, который является значением по умолчанию. Установка данного значения на ноль означает максимальное количество повторений (1 миллиард).

Команда	Синтаксис
set repc	pid_ioctl($pid, "set repc N");

Таблица 7-26 Настройка количества повторений

- **Задать значение счетчика**

Данная команда предназначена для определения времени выходного сигнала. В импульсном режиме требуются 2 значения счетчика (T1 и T2).

Команда	Синтаксис
set count	pid_ioctl($pid, "set count T1 T2");

Таблица 7-27 Задать значение счетчика

Доступные значения для счетчиков T1 и T2 в импульсном режиме следующие:

Юнит	Доступные значения счетчиков
Микросекунда	0, 10 ~ 1,800,000,000
Миллисекунда	0 ~ 1,800,000
Секунда	0 ~ 1,800

Таблица 7-28 Доступные значения счетчика

На схеме ниже показан сигнал в случае установки T1 и T2 с задержкой D в импульсном режиме.

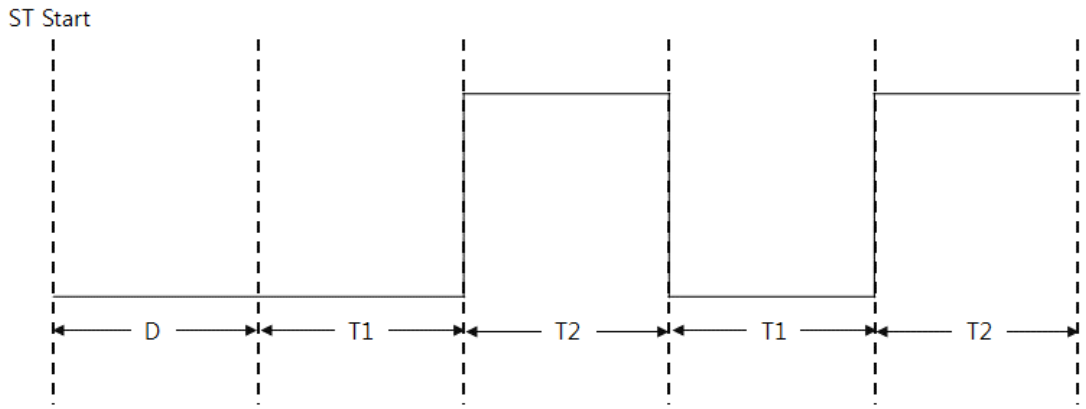


Схема 7-10 Форма сигнала импульсного режима

- **Получение повторов счетчика**

Команда «пуе кузс» предназначен для чтения оставшегося количества повторовю которые будут выполены.

Команда	Синтаксис
get repc	pid_ioctl($pid, "get repc");

Таблица 7-29 Получение повторов счетчика

7.3.7 Пример импульсного режима

- Пример импульсного режима (высокий пульс/HIGH)

```
$pid = pid_open("/mmap/st0");              // open ST 0
pid_ioctl($pid, "set div sec");            // set unit: second
pid_ioctl($pid, "set mode output pulse");  // set mode: pulse
pid_ioctl($pid, "set output dev io3 0");   // set output device / pin: io3 / 0
pid_ioctl($pid, "set count 1 2");          // set count values: 1 and 2
2pid_ioctl($pid, "set repc 1");            // set repeat count: 1
pid_ioctl($pid, "start");                  // start ST
while(pid_ioctl($pid, "get state"));
pid_close($pid);
```

Импульсный режим в основном изменяет уровень от низкого к высокому. Время изменения зависит от скорости деления и количества отсчетов (T1 и T2). На схеме ниже показана форма сигнаола приведенного выше примера.

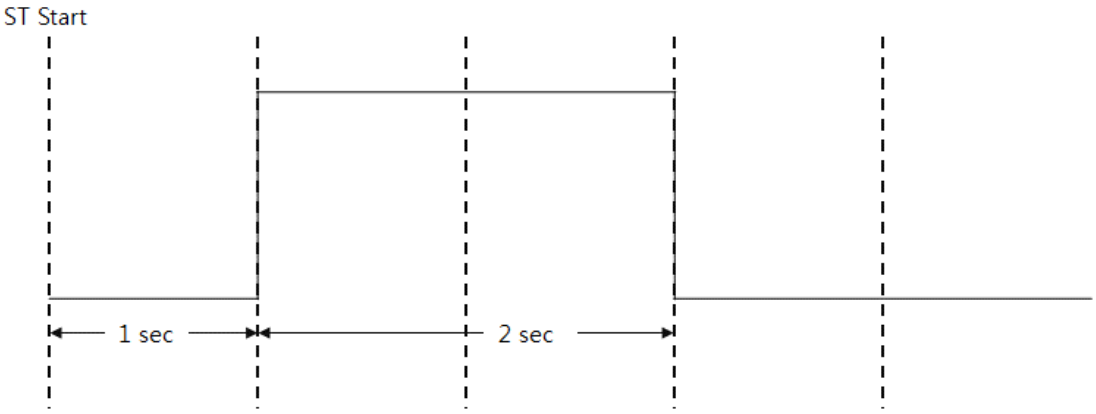


Схема 7-11 Пример импульсного режима (высокий импульс)

- Пример импульсного режима (нихкий пульс/LOW)

```
$pid = pid_open("/mmap/st0");              // open ST 0
pid_ioctl($pid, "set div sec");            // set unit: second
pid_ioctl($pid, "set mode output pulse");  // set mode: pulse
pid_ioctl($pid, "set output dev io3 0");   // set output device / pin: io3 / 0
pid_ioctl($pid, "set count 1 2");          // set count values: 1 and 2
pid_ioctl($pid, "set output invert 1");    // invert output
pid_ioctl($pid, "set repc 1");             // set repeat count: 1
pid_ioctl($pid, "start");                  // start ST
while(pid_ioctl($pid, "get state"));
pid_close($pid);
```

После выполнения командной строки «set output invert 1» все выходные уровни будут инвертированы, включая импульсный вывод. На схеме ниже показана форма сигнала, объясненная выше.

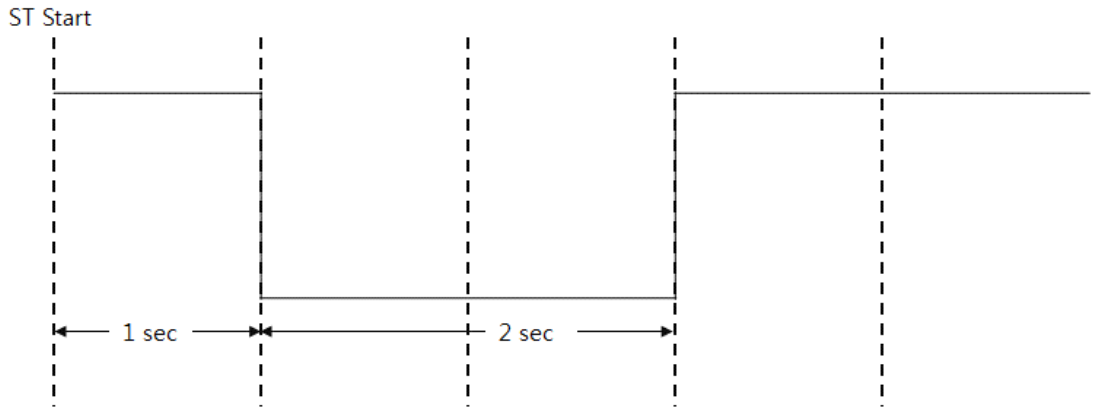


Схема 7-12 Пример импульсного режима (нихкий ипмульс)

7.3.8 Режим PWM

Режим PWM – это режим широто-импульсной модуляции, поэтому синтаксис практически совпадает с импульсным режимом, но с небольшой разницей.

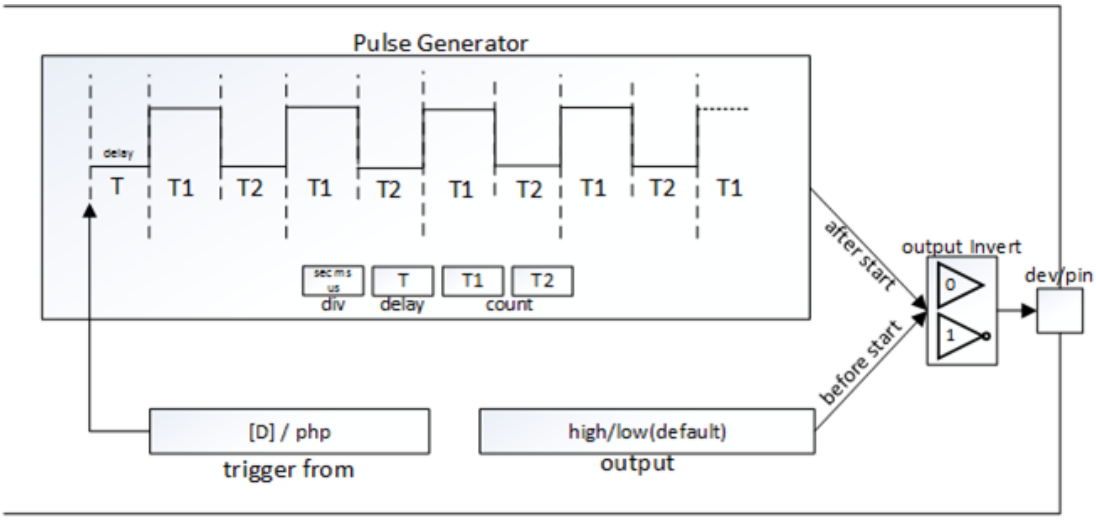


Схема 7-13 Диаграмма режима PWM

Доступные команды в режиме PWM выглядят следующим обращом:

Ком.	Подкоманды			Описание
set	mode	output	pwm	set mode: PWM
	div	sec		set unit: second
		ms		set unit: millisecond
		us		set unit: microsecond
	output	low		output LOW
		high		output HIGH
		dev	io3/4 #pin	set output device and pin
		invert	0	not invert output
			1	invert output
	count	[T1] [T2]		set output timing parameters
	delay	[D]		set delay
	trigger	from	st#	set trigger target: st0 ~ st7
			php	set trigger target: none
reset	-			reset
get	state			get current state
start	-			start
stop	-			stop

Таблица 7-30 Команды режима PWM

- Задать значения счетчика

Значения счетчика определяют время изменения уровней. В режиме PWM требуются 2 значения счетчика. Ниже показано, как установить значения:

Команда	Синтаксис
set count	pid_ioctl($pid, "set count T1 T2");

Таблица 7-31 Задать значения счетчика

Доступные значения счетчика в режиме PWM выглядят следующим обращом:

Юнит	Доступные значения счетчика (0 ~ полчаса)
Microsecond	0, 10 ~ 1,800,000,000
Millisecond	0 ~ 1,800,000
second	0 ~ 1,800

Таблица 7-32 Доступные значения счетчика

На схеме ниже показан сигнал в случае установки T1 и T2 с задержкой D в режиме PWM.

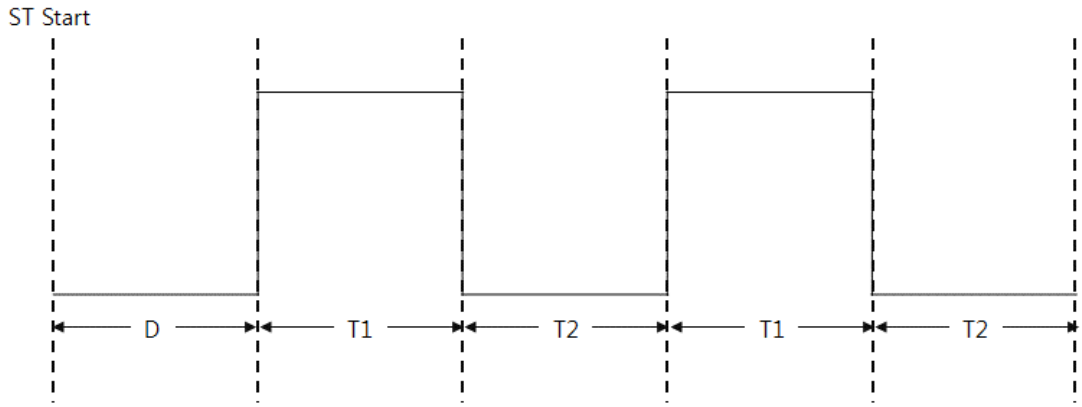


Схема 7-14 Форма сигнала режима PWM

7.3.9 Пример режима PWM

- Пример режима PWM

```
$pid = pid_open("/mmap/st0");              // open ST 0
pid_ioctl($pid, "set div sec");            // set unit: second
pid_ioctl($pid, "set mode output pwm");    // set mode: PWM
pid_ioctl($pid, "set output dev io3 0");   // set output dev / pin: io3 / 0
pid_ioctl($pid, "set count 1 1");          // set count values 1 and 1
pid_ioctl($pid, "start");                  // start ST
sleep(10);
pid_ioctl($pid, "stop");                   // stop ST
pid_close($pid);
```

Схема ниже показывает форму сигнала приведенного выше примера.

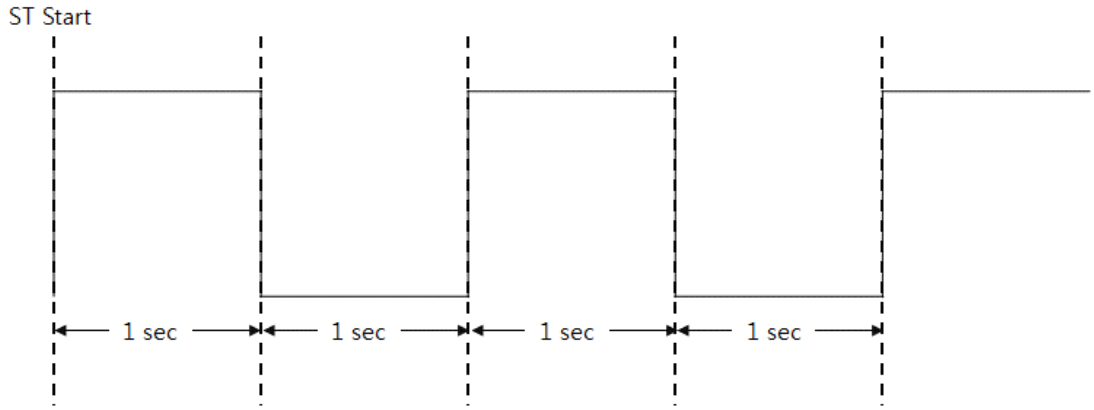


Схема 7-15 Пример режима PWM

7.3.10 Триггер

Команда Триггер(Trigger) используется, когда необходимо синхронизировать время начала ST с другим ST. Пример ниже показывает, как синхронизировать ST 1 с ST 0 при помощи триггера.

- Пример триггера

```
$pid0 = pid_open("/mmap/st0");              // open ST 0
pid_ioctl($pid0, "set div sec");            // set unit: second
pid_ioctl($pid0, "set mode output pulse");  // set mode: pulse
pid_ioctl($pid0, "set count 1 1");          // set count values: 1 and 1
pid_ioctl($pid0, "set repc 2");             // set repeat count: 2
pid_ioctl($pid0, "set output dev io3 0");   // set output dev / pin: io3 / 0

$pid1 = pid_open("/mmap/st1");              // open ST 1
pid_ioctl($pid1, "set div sec");            // set unit: second
pid_ioctl($pid1, "set mode output pulse");  // set mode: pulse
pid_ioctl($pid1, "set trigger from st0");   // set trigger target: st0
pid_ioctl($pid1, "set count 1 1");          // set count values: 1 and 1
pid_ioctl($pid1, "set repc 2");             // set repeat count: 2
pid_ioctl($pid1, "set output dev io3 1");   // set output dev / pin: io3 / 1

pid_ioctl($pid1, "start");                  // start ST 1
pid_ioctl($pid0, "start");                  // start ST 0

while(pid_ioctl($pid1, "get state"));
pid_close($pid0);
pid_close($pid1);
```

Как вы видите в примере выше, ST (который вы хотите синхронизировать с выходным временем), должен стартовать до запуска цели триггера.

- Диапазон ошибок ST

ST имеет следующий диапазон ошибок:

Случай	Диапазон ошибок
Одновременное использование 2-х ST	Приблизительно 1μs
Одновременное использование 8-ми ST	Приблизительно 4μs

Таблица 7-33 Диапазон ошибок ST

8 Приложение: Функции, связанные с устройством

PHPoC предоставляет множество встроенных функций при использовании устройства:

Функции	Формат использования
pid_bind	pid_bind(PID[, IP, PORT]);
pid_close	pid_close(PID);
pid_connect	pid_connect(PID, IP, PORT);
pid_ioctl	pid_ioctl(PID, COMMAND);
pid_listen	pid_listen(PID, [BACKLOG]);
pid_open	pid_open(PID[, FLAG]);
pid_read	pid_read(PID, BUF[, LEN]);
pid_recv	pid_recv(PID, BUF[, LEN, FLAG]);
pid_recvfrom	pid_recvfrom(PID, BUF[, LEN, FLAG, IP, PORT]);
pid_send	pid_send(PID, BUF[, LEN, FLAG]);
pid_sendto	pid_sendto(PID, BUF[, LEN, FLAG, IP, PORT]);
pid_write	pid_write(PID, BUF[, LEN]);

Таблица 8-1 Функции, связанные с устройством

☞ ***Пожалуйста, обратитесь к документу "Встроенные функции PHPoC" для получения подробной информации о встроенных функциях.***

9 Приложение: Информация об устройстве

9.1 Количество устройств в зависимости от типа продукта

Устройство		PBH-101	PBH-104	PBH-204
UART		1	4	1
NET		2	2	2
TCP		5	5	5
UDP		5	5	5
I/O	Digital Input	0	0	4
	Digital Output	0	0	4
	Digital Output(LED)	8	8	8
ST		8	8	8

Таблица 9-1 Количество устройтв в зависимости от типа продукта

9.2 Путь к файлу устройства в зависимости от типа продукта

9.2.1 UART

Продукт	Путь
PBH-101, PBH-204	/mmap/uart0
PBH-104	/mmap/uart0
	/mmap/uart1
	/mmap/uart2
	/mmap/uart3

Таблица 9-2 UART

9.2.2 NET

Продукт	Путь	Примечание
PBH-101, PBH-104, PBH-204	/mmap/net0	Wired LAN
	/mmap/net1	Wireless LAN

Таблица 9-3 NET

9.2.3 TCP

Продукт	Путь
PBH-101, PBH-104, PBH-204	/mmap/tcp0
	/mmap/tcp1
	/mmap/tcp2
	/mmap/tcp3
	/mmap/tcp4

Таблица 9-4 TCP

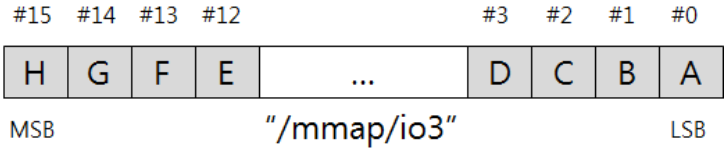
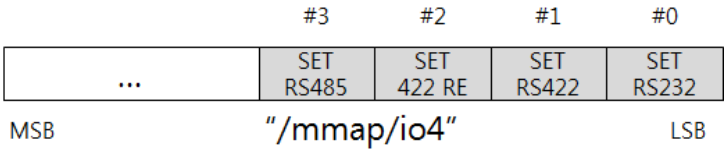
9.2.4 UDP

Продукт	Путь
PBH-101, PBH-104, PBH-204	/mmap/udp0
	/mmap/udp1
	/mmap/udp2
	/mmap/udp3
	/mmap/udp4

Таблица 9-5 UDP

9.2.5 I/O

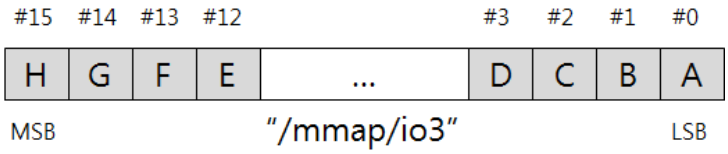
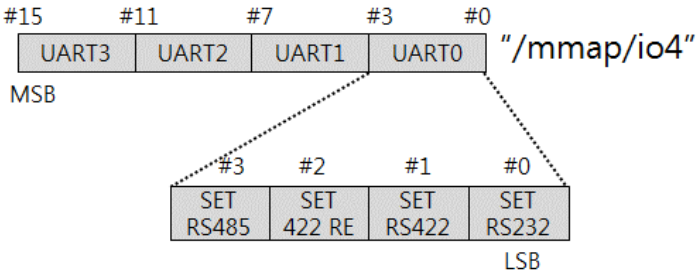
- PBH-101

<table>
<tr><th colspan="2">Продукт/раздел</th><th>Путь и информация о картировании</th></tr>
<tr><td rowspan="2">PBH-101</td><td>LED</td><td>/mmap/io3
 
</td></tr>
<tr><td>UART Mode
(Serial Type)</td><td>/mmap/io4
 

● example of setting UART mode
<table>
<tr><th>Mode</th><th>Value</th><th>SET RS485</th><th>SET 422 RE</th><th>SET RS422</th><th>SET RS232</th></tr>
<tr><td>RS232</td><td>0x05</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>RS422</td><td>0x02</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>RS485</td><td>0x0c</td><td>1</td><td>1</td><td>0</td><td>0</td></tr>
</table></td></tr>
</table>

Таблица 9-6 Цифровые порта входа/выхода PBH-101

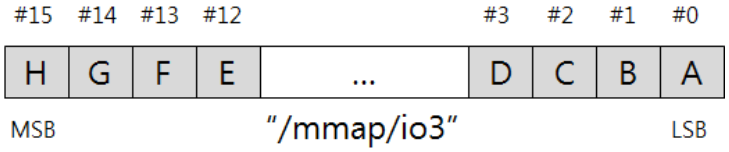
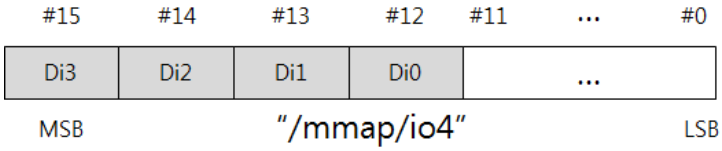
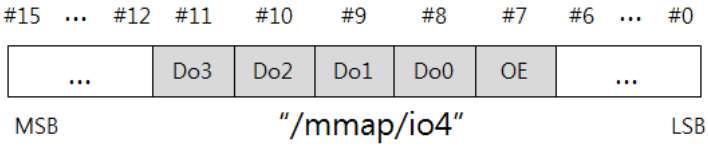
- PBH-104

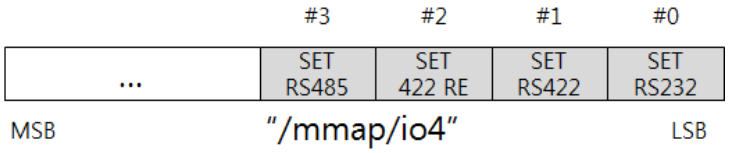
<table>
<tr><th colspan="2">Продукт/раздел</th><th>Путь и информация о картировании</th></tr>
<tr><td rowspan="2">PBH-104</td><td>LED</td><td>/mmap/io3
 
</td></tr>
<tr><td>UART Mode
(Serial Type)</td><td>/mmap/io4
 

● example of setting UART mode
<table>
<tr><th>Mode</th><th>Value</th><th>SET RS485</th><th>SET 422 RE</th><th>SET RS422</th><th>SET RS232</th></tr>
<tr><td>RS232</td><td>0x05</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>RS422</td><td>0x02</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>RS485</td><td>0x0c</td><td>1</td><td>1</td><td>0</td><td>0</td></tr>
</table>
● example of setting multiple UART
<table>
<tr><th>Mode</th><th>Value</th><th>UART0</th><th>UART1</th><th>UART2</th><th>UART3</th></tr>
<tr><td>RS232</td><td>0x5555</td><td>0x0005</td><td>0x0050</td><td>0x0500</td><td>0x5000</td></tr>
<tr><td>RS422</td><td>0x2222</td><td>0x0002</td><td>0x0020</td><td>0x0200</td><td>0x2000</td></tr>
<tr><td>RS485</td><td>0xCCCC</td><td>0x000C</td><td>0x00C0</td><td>0x0C00</td><td>0xC000</td></tr>
</table>
</td></tr>
</table>

Таблица 9-7 Цифровые порта входа/выхода PBH-104

- PBH-204

<table>
<tr><th colspan="2">Продукт/раздел</th><th>Путь и информация о картировании</th></tr>
<tr><td rowspan="4">PBH-204</td><td>LED</td><td>/mmap/io3
 
</td></tr>
<tr><td>Digital Input (photo-coupler)</td><td>/mmap/io4
 
</td></tr>
<tr><td>Digital Output (relay)</td><td>/mmap/io4
 

※ OE: bit for enabling or disabling output relay
- Enable: LOW(0), Disable: HIGH(1)</td></tr>
<tr><td>UART Mode</td><td>/mmap/io4
 

● example of setting UART mode
<table>
<tr><th>Mode</th><th>Value</th><th>SET RS485</th><th>SET 422 RE</th><th>SET RS422</th><th>SET RS232</th></tr>
<tr><td>RS232</td><td>0x05</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>RS422</td><td>0x02</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>RS485</td><td>0x0c</td><td>1</td><td>1</td><td>0</td><td>0</td></tr>
</table></td></tr>
</table>

Таблица 9-8 Цифровые порта входа/выхода PBH-204

9.2.6 ST

Продукт	Путь
PBH-101, PBH-104, PBH-204	/mmap/st0
	/mmap/st1
	/mmap/st2
	/mmap/st3
	/mmap/st4
	/mmap/st5
	/mmap/st6
	/mmap/st7

Таблица 9-9 ST

9.2.7 ENV и пользовательская память

Продукт/раздел		Путь	Размер (Байты)
PBH-101, PBH-104, PBH-204	System ENV	/mmap/envs	1536
	User ENV	/mmap/envu	1536
	User Memory	/mmap/um0	64
		/mmap/um1	64
		/mmap/um2	64
		/mmap/um3	64

Таблица 9-10 ENV и пользовательская память

10 Приложение: Технические характеристики и ограничения F/W

10.1 Прошивка

Прошивка	Продукт
P20	PBH-101, PBH-104, PBH-204

Таблица 10-1 Прошивка

10.2 Спецификация

Наименование	p20	Описание
ENVS	1,536	Size of System ENV, byte
ENVU	1,536	Size of User ENV, byte
WLAN	1	Wireless LAN
EMAC	1	Ethernet
UART	4	The number of UART
FLOAT	Support	Floating Point Numbers
SSL	Support	SSL communication
PHP_MAX_NAME_SPACE	16	The number of Namespace
PHP_NAME_LEN	32	Size of User Identifier
PHP_MAX_USER_DEF_NAME	480	The number of User Identifier
PHP_LLSTR_BLK_SIZE	64	Size of String Block, byte
PHP_MAX_LLSTR_BLK	192	The number of String Blocks
string buffer size	12K	Size of string buffer, byte
PHP_MAX_STRING_LEN	1,536	Size of string variable, byte
PHP_INT_MAX	≒ $9.2*10^{18}$	Max value of integer type
EZFS_MAX_NAME_LEN	64	Size of EZFS filename, byte
TASK	2	The number of Task
TCP	5	The number of TCP
UDP	5	The number of UDP
TCP_RXBUF_SIZE	1,068	TCP receive buffer size
TCP_TXBUF_SIZE	1,152	TCP send buffer size
PDB_TXBUF_SIZE	2,048	PHPoCD send buffer size
HTTP_TXBUF_SIZE	1,536	HTTP send buffer size
UART_RXBUF_SIZE	1,024	UART send/receive buffer size
UDP_RXBUF_SIZE	512	UDP receive buffer size
ST	8	Software Timer

Таблица 10-2 Прошивка и спецификация

10.3 Ограничения

Наименование	Ограничение
Level of Namespace	PHP_MAX_NAME_SPACE - 1
Level of Function Call	PHP_MAX_NAME_SPACE - 2
Size of User Identifier	PHP_NAME_LEN - 1
Size of String Variable	PHP_MAX_STRING_LEN - 2
Size of Array Offset	string length - 2
Size of Filename	EZFS_MAX_NAME_LEN - 1
Size of arguments for system function	PHP_LLSTR_BLK_SIZE - 1
Size of arguments for pid_ioctl function	PHP_LLSTR_BLK_SIZE - 1
Size of $address of function sendto	PHP_LLSTR_BLK_SIZE - 1
Size of $needle & $replace of function str_replace	PHP_LLSTR_BLK_SIZE - 1
Size of $address of function inet_pton	PHP_LLSTR_BLK_SIZE - 1
Size of $address of function inet_ntop	PHP_LLSTR_BLK_SIZE - 1
Size of $delimiter of function explode	PHP_LLSTR_BLK_SIZE - 1
Maximum size of UDP data for receiving	UDP receive buffer size - 2

Таблица 10-3 Ограничения

11 Индекс команды pid_ioctl

Device	Cmd.	Argument / Value	Page
NET	get	mode	- 20 -
	get	speed	- 20 -
	get	hwaddr	- 20 -
	get	ipaddr	- 20 -
	get	netmask	- 20 -
	get	gwaddr	- 20 -
	get	nsaddr	- 20 -
ST	get	count	- 41 -
	get	repc	- 44 -
	get	state	- 39 -
	set	div us/ms/sec	- 39 -
	set	mode free	- 39 -
	set	mode output toggle	- 39 -
	set	mode output pulse	- 39 -
	set	mode output pwm	- 39 -
	set	count (int)	- 41 -
	set	count (int1) [(int2) … (int8)]	- 44 -
	set	count (int1) (int2)	- 50 -
	set	dir up/down	- 41 -
	set	repc (int)	- 44 -
	set	delay (int)	- 44 -
	set	output low/high	- 44 -
	set	output invert [0/1]	- 44 -
	set	output dev io3/4 (int)	- 44 -
	set	trigger from php/st0/st1…/st7	- 44 -
	reset		- 39 -
	start		- 39 -
	stop		- 39 -
TCP	get	state	- 28 -
	get	rxlen	- 28 -
	get	rxbuf	- 28 -
	get	txfree	- 28 -
	get	txbuf	- 28 -
	get	dstport	- 28 -
	get	srcport	- 28 -

	get	dstaddr	- 28 -
	get	srcaddr	- 28 -
	get	ssh username	- 28 -
	get	ssh password	- 28 -
	set	nodelay 0/1	- 22 -
	set	api telnet/ssl/ssh/ws	- 22 -
	set	ssl method ssl3_client/ssl3_server/tls1_client/tls1/server	- 22 -
	set	ssh auth accept/reject	- 22 -
	set	ws path/mode/proto/origin	- 22 -
UART	get	rxlen	- 14 -
	get	txfree	- 14 -
	get	rxbuf	- 14 -
	get	txbuf	- 14 -
	get	flowctrl	- 14 -
	get	baud	- 14 -
	get	parity	- 14 -
	get	data	- 14 -
	get	stop	- 14 -
	set	baud (int)	- 12 -
	set	parity 0/1/2/3/4	- 12 -
	set	data 7/8	- 12 -
	set	stop 1/2	- 12 -
	set	flowctrl 0/1/2/3	- 12 -
UDP	get	rxlen	- 35 -
	get	dstport	- 35 -
	get	srcport	- 35 -
	get	dstaddr	- 35 -
	get	srcaddr	- 35 -
	set	dstaddr (string)	- 34 -
	set	dstport (int)	- 34 -
IO3/4	get	n mode	- 8 -
	get	n input/output	- 8 -
	set	n mode in/out/led_xx [low/high]	- 7 -
	set	n output low/high/toggle	- 7 -
	set	n lock/unlock	- 7 -

Таблица 11-1 Индекс команды pid_ioctl

12 История изменений

Дата	Версия	Примечание	Автор
2014.09.23	1.0	○ Первый релиз	Roy LEE
2015.08.07	1.1	○ Изменено название документа: добавлено "для P20" ○ Добавлены TCP API: TELNET, SSH, Веб-сокет ○ Добавлен режим вывода ST ○ Улучшено приложение ○ Исправлены некоторые ошибки и выражения	Roy LEE
2015.11.03	1.2	○ Исправлены некоторые ошибки и выражения	Roy LEE