



— The bright side of industry automation

# **PWM Linux Library Documentation**

© 2000–2007 OLTOM Engineering AB

Email: [support@oltom.com](mailto:support@oltom.com)

Revision 1.2

# Introduction

The PWM Linux driver is provided as a dynamically linked library for 32-bit Intel x86 compatible systems. It is compatible with GLIBC 2 and later, and can be called from most programming environments that call C functions.

## Installation

1. Start a command shell.
2. Mount the PWM Software CD for example:

```
mount /dev/cdrom /mnt/cdrom
```

3. Change to the directory where you want the software to be installed, for example

```
cd /opt
```

4. Extract the libraries and header files from the cdrom. If your CD is mounted in some other location, use that instead

```
tar xzf /mnt/cdrom/pwm-linux.tgz
```

The software will be installed in the subdirectory `pwm` under the current directory, e.g. `/opt/pwm` in this example.

## Compiling

Include the PWM header in your program code:

```
#include "pwm.h"
```

Compile by

```
gcc -I/path-to-pwm-installation pwm-program.c -L/path-topwm-installation
```

# Function Calls

## `pwmSystem pwmInitSerial (const char * serialport, const int baud)`

This starts the connection to the PWM Driver software and also opens a communication port. Always start with this.

Return value: a `pwmSystem` handle to be used in for addressing the PWM context. -1 on failure to open serial port, `errno` contains the error code.

## `pwmSystem pwmInitI2C (const char * i2cport)`

This starts the connection to the PWM Driver software and also opens the appropriate I2C context. Note that you have to have hardware that is capable of multi-master mode. The SMBus driver as an example cannot handle multi-master mode.

## `pwmAddress pwmEnumerate (pwmSystem s)`

The Enumerate function returns a list of all the discoverable cards on the `pwmSystem` bus.

### `pwmStatus pwmSetOurAddress (pwmSystem s, const int ouraddress)`

`SetOurAddress` sets the address of the controlling computer. If you are using a serial port the address must match the address of the serial port on the PWM-card. Refer to the card documentation for more information on packet addressing. If you are using an I2C-connection you can use any legal 8-bit I2C-address.

## `pwmStatus pwmSetcurrent (pwmSystem h, pwmAddress addr, pwmint32 int)`

This sets the current to the motor that have been addressed in deciampère. Example: `PWMC0urrent(h, 100)` sets a current of 1.0 Ampere.

Return value: 0 if the command was sent successfully, -1 otherwise.

## `pwmStatus pwmMove (pwmSystem h, pwmAddress addr, pwmint32 steps)`

Use this to get the addressed motor to start moving. Negative values for one direction and positive values for the other, depending on the motor wiring. This is the only command that starts a movement. Use the Speed and Ramp commands to set up the movement, before starting the movement with `PWMmove`.

Return value: 0 if the command was sent successfully, -1 otherwise.

## `pwmStatus pwmSetSpeed (pwmSystem h, pwmAddress addr, pwmint32 speed)`

This sets the maximum speed of the addressed motor. The value should be in steps/second. Since this is a digital system with a discrete resolution all values are not exactly resolvable. However, since the resolution of the system is 51.2ms misalignment from the desired value will not be large. The system calculates the closest possible value of the desired value. To be able to ensure an exact speed calculation for the user the function will return with the used value in 51.2ms ticks. Example: If the function returns with a value of 100 the step period will be  $100 * 51.2\text{ms}$ . Note that this command should be used before the Move command.

Return value: 0 if the command was received and executed.

## `pwmStatus pwmSetRamp (pwmSystem h, pwmAddress addr, pwmint32 ramp)`

With this function the acceleration and retardation ramp will be set. The value is in steps/second<sup>2</sup>. Note that this command should be used before the Move command.

Return value: 0 if the command was received and executed without error.

### **pwmStatus pwmInquiry (pwmSystem h, pwmAddress addr)**

This function sends an inquire command to check if the addressed motor is busy moving.  
This function returns 0 if the motor is not busy.

### **pwmStatus pwmReset (pwmSystem h, pwmAddress addr)**

This will reset the addressed card to its power on state. Note that this means that the power will be set to the default power on value of 0.3 A.

### **pwmStatus pwmClose (pwmSystem h)**

This closes the COM port. Only necessary to use if you want to use the serial port for something else. If you close your program the port will automatically be available for other applications.

# Status and Error codes

The following codes are returned as pwmStatus.

```
pwmOk  
pwmErrCheckErr  
pwmErrReserved  
pwmErrReserved2  
pwmErrSequence  
pwmErrNAK  
pwmErrTimeout  
pwmErrSystem, System call error, check errno. Errno returned in byte 1.  
pwmErrArg, Invalid/illegal argument.
```

## Usage example

```
r= pwmInquiry (h, a);  
switch (r) {  
    case pwmErrCheckErr: fprintf (stderr, "Checksum error"); break;  
    ...  
}
```

# Disclaimer

We believe that the information contained herein was accurate in all respects at the time of printing. OLTOM Engineering AB cannot, however, assume any responsibility for errors or omissions in this text. Also note that the information in this document is subject to change without notice and should not be construed as a commitment by OLTOM Engineering AB.