The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC118-SW-65

Windows 2000/XP Device Driver

6 Channel Motion Control

Version 1.0.x

User Manual

Issue 1.0

August 2004

TEWS TECHNOLOGIES GmbH
Am Bahnhof 7
25469 Halstenbek / Germany
Phone: +49-(0)4101-4058-0
Fax: +49-(0)4101-4058-19
e-mail: info@tews.com
www.tews.com

TEWS TECHNOLOGIES LLC
1 E. Liberty Street, Sixth Floor
Reno, Nevada 89504 / USA
Phone: +1 (775) 686 6077
Fax: +1 (775) 686 6024
e-mail: usasales@tews.com
www.tews.com

TPMC118-SW-65

6 Channel Motion Control

Windows 2000/XP Device Driver

Issue	Description	Date
1.0	First Issue	August 5, 2004

Table of Content

1 Introduction

The TPMC118-SW-65 Windows WDM (Windows Driver Model) device driver is a kernel mode driver which allows the operation of the TPMC118 on an Intel or Intel-compatible x86 Windows 2000 or Windows XP operating system.

The standard file and device (I/O) functions (CreateFile, CloseHandle, and DeviceIoControl) provide the basic interface for opening and closing a resource handle and for performing device I/O control operations.

The TPMC118 device driver supports the following features:

- read digital input values
- wait for transition on digital input
- write DAC output value
- read encoder value from a specified channel
- read synchronized encoder values
- configure encoder interfaces (resolution, mode, preload value)

2 Installation

The software is delivered on a 3½" HD diskette.

Following files are located on the diskette:

TPMC118.sys	Windows driver binary
TPMC118.h	Header-file with IOCTL code definitions
TPMC118.inf	Windows installation script
TPMC118-SW-65.pdf	This document
\Example\Example.c	Microsoft Visual C example application

2.1 Software Installation

2.1.1 Windows 2000 / XP

This section describes how to install the TPMC118 Device Driver on a Windows 2000 / XP operating system.

After installing the TPMC118 card(s) and boot-up your system, Windows 2000 / XP setup will show a "***New hardware found***" dialog box.

1. The "***Upgrade Device Driver Wizard***" dialog box will appear on your screen.
   Click "***Next***" button to continue.
2. In the following dialog box, choose "***Search for a suitable driver for my device***".
   Click "***Next***" button to continue.
3. In Drive A, insert the TPMC118 driver disk; select "***Disk Drive***" in the dialog box.
   Click "***Next***" button to continue.
4. Now the driver wizard should find a suitable device driver on the diskette.
   Click "***Next***" button to continue.
5. Complete the upgrade device driver and click "***Finish***" to take all the changes effect.
6. Now copy all needed files (tpmc118.h, TPMC118-SW-65.pdf) to the desired target directories.

After successful installation the TPMC118 device driver will start immediately and creates devices (TPMC118_1, TPMC118_2 ...) for all recognized TPMC118 modules.

2.1.2 Confirming Windows 2000 / XP Installation

To confirm that the driver has been properly loaded in Windows 2000 / XP, perform the following steps:

1. From Windows 2000 / XP, open the "***Control Panel***" from "***My Computer***".
2. Click the "***System***" icon and choose the "***Hardware***" tab, and then click the "***Device Manager***" button.
3. Click the "**+**" in front of "***Other Devices***".
   The driver "***TPMC118 6 Channel Motion Control***" should appear.

3 TPMC118 Device Driver Programming

The TPMC118-SW-65 Windows WDM device driver is a kernel mode device driver.

The standard file and device (I/O) functions (CreateFile, CloseHandle, and DeviceIoControl) provide the basic interface for opening and closing a resource handle and for performing device I/O control operations.

All of these standard Win32 functions are described in detail in the Windows Platform SDK Documentation (Windows base services / Hardware / Device Input and Output).

For details refer to the Win32 Programmers Reference of your used programming tools (C++, Visual Basic etc.)

3.1 TPMC118 Files and I/O Functions

The following section does not contain a full description of the Win32 functions for interaction with the TPMC118 device driver. Only the required parameters are described in detail.

3.1.1 Opening a TPMC118 Device

Before you can perform any I/O the *TPMC118* device must be opened by invoking the **CreateFile** function. **CreateFile** returns a handle that can be used to access the *TPMC118* device.

```
HANDLE CreateFile(
      LPCTSTR   lpFileName,
      DWORD     dwDesiredAccess,
      DWORD     dwShareMode,
      LPSECURITY_ATTRIBUTES lpSecurityAttributes,
      DWORD     dwCreationDistribution,
      DWORD     dwFlagsAndAttributes,
      HANDLE    hTemplateFile
);
```

Parameters

LPCTSTR lpFileName

This parameter points to a null-terminated string, which specifies the name of the TPMC118 to open. The *lpFileName* string should be of the form **\\.\TPMC118_*x*** to open the device *x*. The ending x is a one-based number. The first device found by the driver is \\.\TPMC118_1, the second \\.\TPMC118_2 and so on.

DWORD dwDesiredAccess

This parameter specifies the type of access to the TPMC118.
For the TPMC118 this parameter must be set to read-write access (GENERIC_READ | GENERIC_WRITE)

DWORD dwShareMode

Set of bit flags that specify how the object can be shared. Set to 0.

LPSECURITY_ATTRIBUTES lpSecurityAttributes

This argument is a pointer to a security structure. Set to NULL for TPMC118 devices.

DWORD dwCreationDistribution

Specifies the action to take on existing files, and which action to take when files do not exist. TPMC118 devices must be always opened **OPEN_EXISTING**.

DWORD dwFlagsAndAttributes

Specifies the file attributes and flags for the file. This value must be set to 0 for TPMC118 devices (no Overlapped I/O).

HANDLE hTemplateFile

This value must be NULL for TPMC118 devices.

Return Value

If the function succeeds, the return value is an open handle to the specified TPMC118 device. If the function fails, the return value is INVALID_HANDLE_VALUE. To get extended error information, call ***GetLastError***.

Example

```
HANDLE    hDevice;

hDevice = CreateFile(
     "\\\\.\\TPMC118_1",
     GENERIC_READ | GENERIC_WRITE,
     0,
     NULL,                  // no security attrs
     OPEN_EXISTING,         // TPMC118 device always open existing
     0,                     // no overlapped I/O
     NULL
);

if (hDevice == INVALID_HANDLE_VALUE) {
     ErrorHandler( "Could not open device" ); // process error
}
```

See Also

CloseHandle(), Win32 documentation CreateFile()

3.1.2 Closing a TPMC118 Device

The **CloseHandle** function closes an open TPMC118 handle.

BOOL CloseHandle(
HANDLE *hDevice;*
);

Parameters

HANDLE hDevice

Identifies an open TPMC118 handle.

Return Value

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call ***GetLastError*.**

Example

```
HANDLE  hDevice;

if( CloseHandle( hDevice ) ) {
     ErrorHandler( "Could not close device" ); // process error
}
```

See Also

CreateFile (), Win32 documentation CloseHandle ()

3.1.3 TPMC118 Device I/O Control Functions

The **DeviceIoControl** function sends a control code directly to a specified device driver, causing the corresponding device to perform the specified operation.

```
BOOL DeviceIoControl(
    HANDLE          hDevice,          // handle to device of interest
    DWORD           dwIoControlCode,  // control code of operation to perform
    LPVOID          lpInBuffer,       // pointer to buffer to supply input data
    DWORD           nInBufferSize,    // size of input buffer
    LPVOID          lpOutBuffer,      // pointer to buffer to receive output data
    DWORD           nOutBufferSize,   // size of output buffer
    LPDWORD         lpBytesReturned,  // pointer to variable to receive output byte count
    LPOVERLAPPED    lpOverlapped      // pointer to overlapped structure for asynchronous
                                      // operation
);
```

Parameters

hDevice

Handle to the TPMC118 that is to perform the operation.

dwIoControlCode

Specifies the control code for the operation. This value identifies the specific operation to be performed. The following values are defined in *TPMC118.h* :

Value	Meaning
IOCTL_TP118_READ	Read value from digital input lines
IOCTL_TP118_WRITE	Set DAC output value
IOCTL_TP118_SETMODE	Configure the encoder channel
IOCTL_TP118_SETPRLD	Set new preload value
IOCTL_TP118_READENC	Read encoder actual value
IOCTL_TP118_SYNCON	Add new channel(s) to the synchronized read encoder channel set
IOCTL_TP118_SYNCREAD	Read encoder values of the synchronized channels
IOCTL_TP118_WAIT	Wait for specified transition on digital input

See below for more detailed information on each control code.

To use these TPMC118 specific control codes the header file TPMC118.h must be included in the application

lpInBuffer

Pointer to a buffer that contains the data required to perform the operation.

nInBufferSize

Specifies the size of the buffer pointed to by *lpInBuffer*.

lpOutBuffer

Pointer to a buffer that receives the operation's output data.

nOutBufferSize

Specifies the size of the buffer in bytes pointed to by *lpOutBuffer*.

lpBytesReturned

Pointer to a variable that receives the size, in bytes, of the data stored into the buffer pointed to by *lpOutBuffer*. A valid pointer is required.

lpOverlapped

Pointer to an *overlapped* structure. This parameter must be set to NULL(no overlapped I/O).

Return Value

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero. To get extended error information, call ***GetLastError***.

See Also

Win32 documentation DeviceIoControl()

3.1.3.1 IOCTL_TP118_READ

This TPMC118 control function reads the value of the digital input lines. A pointer to the callers buffer (*UCHAR*) is passed by the parameter *lpOutBuffer* to the driver. *lpInBuffer* is not used and must be set to NULL.

The value of the input lines will be returned in the specified UCHAR buffer. A '1' represents an active input level, a '0' a passive level. The bits are assigned as follows:

Bit	Channel
0	1
1	2
2	3
3	4
4	5
5	6
6	unused
7	unused

Example

```
#include "TPMC118.h"

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumWritten;
UCHAR     digIn;

//
// read digital input lines
//
success = DeviceIoControl (
     hDevice,                    // TPMC118 handle
     IOCTL_TP118_READ,           // control code
     NULL,                       // no buffer used
     0,
     &digIn,                     // buffer which receives input value
     sizeof(digIn),
     &NumBytes,
     NULL
);

if( success ) {
      /* Process data */;
}
else {
  ErrorHandler ( "Device I/O control error" ); // process error
}
```

Error Codes

ERROR_INVALID_BUFFER_SIZE	The size of the provided buffer is too small

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl(), TPMC118 Hardware User Manual

3.1.3.2 IOCTL_TP118_WRITE

This TPMC118 control function writes a specified value to a specified DAC channel. A pointer to the callers buffer (*TP118_WRITE_BUF*) is passed by the parameter *lpInBuffer* to the driver. *lpOutBuffer* is not used and must be set to NULL.

The TP118_WRITE_BUF structure has the following layout:

```
typedef struct {
   int      channel;        //  channel number (1..6)
   ULONG    value;          //  analog output value
} TP118_WRITE_BUF, *PTP118_WRITE_BUF;
```

Members

channel

Specifies the DAC channel number. Valid channel numbers are 1 up to 6.

value

Specifies the new output value that should be written to the specified channel. Valid values are between –0x8000 (-10V) and 0x7FFF (+10V).

Example

```
#include "TPMC118.h"

HANDLE              hDevice;
BOOLEAN             success;
ULONG               NumWritten;
TP118_WRITE_BUF     WrBuf;

//
// Write 0x1000 to DAC on channel 4
//
WrBuf.value = 0x1000;
WrBuf.channel = 4;
success = DeviceIoControl (
     hDevice,                    // TPMC118 handle
     IOCTL_TP118_WRITE,          // control code
     &WrBuf,                     // buffer specifies new DAC value
     sizeof(WrBuf),
     NULL,                       // no buffer used
     0,
     &NumBytes,
     NULL
);

if( success ) {
      /* Writing OK */;
```

```
}
else {
  ErrorHandler ( "Device I/O control error” ); // process error
}
```

Error Codes

ERROR_INVALID_BUFFER_SIZE	The size of the provided buffer is too small
STATUS_INVALID_PARAMETER	Invalid channel number specified

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl(), TPMC118 Hardware User Manual

3.1.3.3 IOCTL_TP118_SETMODE

This TPMC118 control function sets the encoder resolution and index mode of a specified channel. A pointer to the callers buffer (*TP118_MODE_BUF*) is passed by the parameter *lpInBuffer* to the driver. *lpOutBuffer* is not used and must be set to NULL.

The TP118_MODE_BUF structure has the following layout:

```
typedef struct {
   int      channel;        //  channel number (1..6)
   ULONG    mode;           //  new mode value
} TP118_MODE_BUF, *PTP118_MODE_BUF;
```

Members

channel

Specifies the encoder channel number. Valid channel numbers are 1 up to 6.

mode

Specifies the new encoder mode. This value is a combination of two values, the first specifies the encoder resolution and the second specifies the reference mode. The flags must be ORed binary. A detailed description of the modes is given in the TPMC118 User Manual.

flag	description
TP118_MODE_DISABLE	Disable encoder channel
TP118_MODE_1X	Encoder resolution – single
TP118_MODE_2X	Encoder resolution – double
TP118_MODE_4X	Encoder resolution – quadruple

flag	description
TP118_MODE_NONEREF	None reference mode
TP118_MODE_REF	Reference mode
TP118_MODE_AUTOREF	Autoreference mode
TP118_MODE_INDEX	Index mode

Example

```
#include “TPMC118.h”

HANDLE              hDevice;
BOOLEAN             success;
ULONG               NumWritten;
TP118_MODE_BUF      ModeBuf;

//
// Setup channel 4 for quadruple and for reference mode
//
ModeBuf.mode = TP118_MODE_4X | TP118_MODE_REF;
ModeBuf.channel = 4;
```

```
success = DeviceIoControl (
     hDevice,                    // TPMC118 handle
     IOCTL_TP118_SETMODE,        // control code
     &ModeBuf,                   // buffer specifies new encoder mode
     sizeof(ModeBuf),
     NULL,                       // no buffer used
     0,
     &NumBytes,
     NULL
);

if( success ) {
      /* Setting OK */;
}
else {
  ErrorHandler ( "Device I/O control error” ); // process error
}
```

Error Codes

ERROR_INVALID_BUFFER_SIZE	The size of the provided buffer is too small
STATUS_INVALID_PARAMETER	Invalid channel number or invalid mode specified

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl(), TPMC118 Hardware User Manual

3.1.3.4 IOCTL_TP118_SETPRLD

This TPMC118 control function sets the encoder preload value of a specified channel. A pointer to the callers buffer (*TP118_PRLD_BUF*) is passed by the parameter *lpInBuffer* to the driver. *lpOutBuffer* is not used and must be set to NULL.

The TP118_PRLD_BUF structure has the following layout:

```
typedef struct {
   int      channel;        //  channel number (1..6)
   ULONG    value;          //  new preload value
} TP118_PRLD_BUF, *PTP118_PRLD_BUF;
```

Members

channel

Specifies the encoder channel number. Valid channel numbers are 1 up to 6.

mode

Specifies the preload value. The preload value will be loaded into the encoder register. The point of loading the encoder register depends on the chosen reference mode. If none reference mode is chosen, the value will be used immediately. See also TPMC118 User Manual.

Example

```
#include "TPMC118.h"

HANDLE              hDevice;
BOOLEAN             success;
ULONG               NumWritten;
TP118_PRLD_BUF      PrldBuf;

//
// Set preload value for channel 4 to 0x12345678
//
PrldBuf.value = 0x12345678;
PrldBuf.channel = 4;
success = DeviceIoControl (
     hDevice,                    // TPMC118 handle
     IOCTL_TP118_SETPRLD,        // control code
     &PrldBuf,                   // buffer specifies new preload value
     sizeof(PrldBuf),
     NULL,                       // no buffer used
     0,
     &NumBytes,
     NULL
);
```

```
if( success ) {
      /* Setting OK */;
}
else {
  ErrorHandler ( "Device I/O control error” ); // process error
}
```

Error Codes

ERROR_INVALID_BUFFER_SIZE	The size of the provided buffer is too small
STATUS_INVALID_PARAMETER	Invalid channel number specified

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl(), TPMC118 Hardware User Manual

3.1.3.5 IOCTL_TP118_READENC

This TPMC118 control function reads the actual encoder value of a specified channel. A pointer to the callers buffer (*TP118_READENC_BUF*) are passed by the parameters *lpInBuffer* and *lpOutBuffer* to the driver.

The TP118_READENC_BUF structure has the following layout:

```
typedef struct {
   int      channel;        //  channel number (1..6)
   ULONG    value;          //  encoder value
} TP118_READENC_BUF, *PTP118_READENC_BUF;
```

Members

channel

Specifies the encoder channel number. Valid channel numbers are 1 up to 6.

mode

Returns the actual encoder value.

Example

```
#include "TPMC118.h"

HANDLE             hDevice;
BOOLEAN            success;
ULONG              NumWritten;
TP118_READENC_BUF  RdencBuf;

//
// Read the actual encoder value of channel 4
//
RdencBuf.channel = 4;
success = DeviceIoControl (
     hDevice,                    // TPMC118 handle
     IOCTL_TP118_READENC,        // control code
     &RdencBuf,                  // buffer specifies read channel
     sizeof(RdencBuf),
     &RdencBuf,                  // buffer returning the value
     sizeof(RdencBuf),
     &NumBytes,
     NULL
);

if( success ) {
      /* Process Data */;
}
else {
```

```
  ErrorHandler ( "Device I/O control error” ); // process error
}
```

Error Codes

ERROR_INVALID_BUFFER_SIZE	The size of a provided buffer is too small
STATUS_INVALID_PARAMETER	Invalid channel number specified

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl(), TPMC118 Hardware User Manual

3.1.3.6 IOCTL_TP118_SYNCON

This TPMC118 control function enables one or more channels to be read with the synchronous read command. A pointer to the callers buffer (*TP118_SYNCON_BUF*) is passed by the parameter *lpInBuffer* to the driver. *lpOutBuffer* is not used and must be set to NULL.

The TP118_SYNCON_BUF structure has the following layout:

```
typedef struct {
   ULONG    channels;          //  bitmask specifying channels
} TP118_SYNCON_BUF, *PTP118_SYNCON_BUF;
```

Members

channels

Specifies the channels to be enabled for synchronous read. A set bit enables a channel for synchronous read. Bit 0 specifies channel 1, Bit 1 specifies channel to and so on. Only Bit 0..5 are valid. A reset bit will not disable the channel for synchronous read.

Example

```
#include “TPMC118.h”

HANDLE              hDevice;
BOOLEAN             success;
ULONG               NumWritten;
TP118_SYNCON_BUF    SynconBuf;

//
// Enable channel 1,3 and 4
//
SynconBuf.channel = (1 << 0) | (1 << 2) | (1 << 3);
success = DeviceIoControl (
     hDevice,                    // TPMC118 handle
     IOCTL_TP118_SYNCON,         // control code
     &SynconBuf,                 // buffer specifying channels
     sizeof(SynconBuf),
     NULL,                       // no buffer used
     0,
     &NumBytes,
     NULL
);

if( success ) {
      /* Setting OK */;
}
else {
  ErrorHandler ( "Device I/O control error” ); // process error
}
```

Error Codes

ERROR_INVALID_BUFFER_SIZE	The size of the provided buffer is too small
STATUS_INVALID_PARAMETER	Invalid channel mask specified

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl(), TPMC118 Hardware User Manual

3.1.3.7 IOCTL_TP118_SYNCOFF

This TPMC118 control function disables all channels to be read with the synchronous read command. *lpOutBuffer* and *lpInBuffer* are not used and must be set to NULL.

Example

```
#include “TPMC118.h”

HANDLE              hDevice;
BOOLEAN             success;
ULONG               NumWritten;

//
// Disable synchronous read
//
success = DeviceIoControl (
     hDevice,                    // TPMC118 handle
     IOCTL_TP118_SYNCOFF,        // control code
     NULL,                       // no buffer used
     0,
     NULL,                       // no buffer used
     0,
     &NumBytes,
     NULL
);

if( success ) {
      /* disabling OK */;
}
else {
  ErrorHandler ( "Device I/O control error” ); // process error
}
```

Error Codes

All returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl(), TPMC118 Hardware User Manual

3.1.3.8 IOCTL_TP118_SYNCREAD

This TPMC118 control function reads the value of the digital input lines. A pointer to the callers buffer (*TP118_SYNCREAD_BUF*) is passed by the parameter *lpOutBuffer* to the driver. *lpInBuffer* is not used and must be set to NULL.

The TP118_SYNCREAD_BUF structure has the following layout:

```
typedef struct {
   ULONG   channels;                    // bitmask of active channels
   ULONG   values[TP118_MAX_CHAN];      // encoder values
} TP118_SYNCREAD_BUF, *PTP118_SYNCREAD_BUF;
```

Members

channels

Returns a bitmask specifying the synchronized channels. Only the values of the synchronized channels are valid, all others will be returned as 0. The bits are assigned as follows:

Bit	Channel
0	1
1	2
2	3
3	4
4	5
5	6

values

Returns a bitmask specifying the synchronized channels. Only the values of the synchronized channels are valid, all others will be returned as 0. The bits are assigned as follows:

Index	Channel
0	1
1	2
2	3
3	4
4	5
5	6

Example

```
#include “TPMC118.h”

HANDLE             hDevice;
BOOLEAN            success;
ULONG              NumWritten;
TP118_SYNCREAD_BUF syncBuf;

//
// read synchronous channel values
//
```

```
success = DeviceIoControl (
     hDevice,                    // TPMC118 handle
     IOCTL_TP118_SYNCREAD,       // control code
     NULL,                       // no buffer used
     0,
     &syncBuf,                   // buffer which receives in values
     sizeof(syncBuf),
     &NumBytes,
     NULL
);

if( success ) {
      /* Process Data */;
}
else {
  ErrorHandler ( "Device I/O control error” ); // process error
}
```

Error Codes

ERROR_INVALID_BUFFER_SIZE	The size of the provided buffer is too small

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl(), TPMC118 Hardware User Manual

3.1.3.9 IOCTL_TP118_WAIT

This TPMC118 control function will wait, not returning from the call, until a specified digital input channel receives a specified transition. A pointer to the callers buffer (*TP118_WAIT_BUF*) is passed by the parameter *lpInBuffer* to the driver. *lpOutBuffer* is not used and must be set to NULL.

The TP118_WAIT_BUF structure has the following layout:

```
typedef struct {
   int      channel;        //  channel number (1..6)
   ULONG    mode;           //  transition
} TP118_WAIT_BUF, *PTP118_WAIT_BUF;
```

Members

channel

Specifies the digital channel number. Valid channel numbers are 1 up to 6.

mode

Specifies the edge to wait for.

value	description
TP118_RISING_EDGE	wait for rising edge
TP118_FALLING_EDGE	wait for falling edge

Example

```
#include "TPMC118.h"

HANDLE              hDevice;
BOOLEAN             success;
ULONG               NumWritten;
TP118_WAIT_BUF      WaitBuf;

//
// Wait for Fallling edge on channel 4
//
WaitBuf.channel = 4;
WaitBuf.mode = TP118_FALLING_EDGE;
success = DeviceIoControl (
     hDevice,                    // TPMC118 handle
     IOCTL_TP118_WAIT,           // control code
     &WaitBuf,                   // buffer specifying channels
     sizeof(WaitBuf),
     NULL,                       // no buffer used
     0,
     &NumBytes,
     NULL
);
```

```
if( success ) {
      /* Process Data */;
}
else {
  ErrorHandler ( "Device I/O control error” ); // process error
}
```

Error Codes

ERROR_INVALID_BUFFER_SIZE	The size of the provided buffer is too small
STATUS_INVALID_PARAMETER	Invalid channel or invalid transition specified
STATUS_INSUFFICIENT_RESOURCES	A wait command is already busy on this event.

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl(), TPMC118 Hardware User Manual