The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC917-SW-65

Windows 2000/XP Device Driver

4 MB SRAM with Battery Backup and

4 Channel Serial Interface PMC

Version 1.0.x

User Manual

Issue 1.0.0

May 2005

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

TPMC917-SW-65

4 MB SRAM with Battery Backup and
4 Channel Serial Interface PMC Module

Windows WDM Device Driver

Issue	Description	Date
1.0.0	First Issue	May 12, 2005

Table of Content

1 Introduction

The TPMC917-SW-65 Windows WDM (Windows Driver Model) device driver is a kernel mode driver which allows the operation of the TPMC917 on an Intel or Intel-compatible x86 Windows 2000 or Windows XP operating system.

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

Because the TPMC917 is a multifunctional device, the different functionalities are provided by different device drivers.

The TPMC917 device drivers support the following features:

- use the serial channels
- read/write access to SRAM
- monitoring state of backup battery

The TPMC917 serial device driver is fully compatible to the standard Windows WDM serial device driver (*serial.sys*).

2 Installation

Following files are located on the distribution media:

tp917Bus.sys	Windows WDM driver binary for function enumerator (busdriver)
tp917Bus.inf	Windows WDM installation script for function enumerator (busdriver)
tpmc917com.sys	Windows WDM driver binary for a serial channel (com driver)
tpmc917com.inf	Windows WDM installation script for a serial channel (com driver)
tpmc917ram.sys	Windows WDM driver binary for SRAM access (ram driver)
tpmc917ram.inf	Windows WDM installation script for SRAM access (ram driver)
tpmc917ram.h	Header file with IOCTL code and structure definitions (ram driver)
TPMC917-SW-65.pdf	This document
\Example\Example.c	Microsoft Visual C example application
Release.txt	Information about the Device Driver Release

2.1 Software Installation

2.1.1 Windows 2000 / XP

This section describes how to install the TPMC917 Device Driver on a Windows 2000 / XP operating system.

After installing the TPMC917 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 TPMC917 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. Repeat the steps above for each serial and ram device which is created by the busdriver.
7. Copy needed files (tpmc917ram.h, TPMC917-SW-65.pdf) to desired target directory.

After successful installation a device is created for each function (refer to next chapter).

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 "***Ports (COM & LPT)***".
   4 serial devices "***TPMC917 (Serial Channel)(COMxx)***" should appear (only TPMC917-10).
4. Click the "**+**" in front of "***Other Devices***".
   The driver "***TPMC917 (SRAM)***" should appear.

3 TPMC917 Serial Device Driver

3.1 Device Driver Programming

The TPMC917 serial device driver is fully compatible to the standard Windows serial device driver (*serial.sys*) and therefore programming of the TPMC917 serial driver is the same as the standard Windows serial driver.

The standard file input and output (I/O) functions (CreateFile, CloseHandle, ReadFile, ReadFileEx, WriteFile, and WriteFileEx) can be used for opening and closing a communications resource handle and for performing read and write operations.

The Microsoft® Win32® application programming interface (API) also includes a set of functions that provide special communication services like reading and setting communication parameter, transmitting immediate characters, setting timeouts and so on.

All of these standard Win32 communication functions are described in detail in the Windows Platform SDK Documentation (Windows base services / Communication).

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

The windows name of the first port is *\Device\tpmc917com0,* of the second port *\Device\ tpmc917com1* and so on.

The DOS device name for TPMC917 serial devices are COM1, COM2, COM3 and so on. If there are other serial devices in the system, the prefix starts with a higher number (see windows name).

The mapping between windows device names and DOS device names for TPMC917 serial devices can be retrieved from the Registry.

Refer to:

HKEY_LOCAL_MACHINE\Hardware\DEVICEMAP\SERIALCOMM

3.2 FIFO Configuration

After Installation of the TPMC917 Device Driver the trigger level for transmit and receive FIFO are set to their default values.

Default values are:

Receive FIFO	Transmit FIFO
56	8

If the default values are not suitable the configuration can be changed by modifying the registry, for instance with regedt32.

To change the transmit trigger level the following value must be modified.

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\TPMC917\TxFIFO

Valid trigger levels are 8, 16, 32 and 56.

To change the receiver trigger level the following value must be modified.

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\TPMC917\RxFIFO

Valid trigger levels are 8, 16, 56 and 60

The configuration of the FIFO trigger level is used for all TPMC917 devices in common. To make the changes current the system must be restarted.

4 TPMC917 SRAM Device Driver

The TPMC917 SRAM Windows WDM device driver is a kernel mode device driver using Direct I/O.

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.)

4.1 TPMC917 SRAM Files and I/O Functions

The following section doesn't contain a full description of the Win32 functions for interaction with the TPMC917 SRAM device driver. Only the required parameters are described in detail.

4.1.1 Opening a TPMC917 SRAM Device

Before you can perform any I/O the TPMC917 RAM device must be opened by invoking the **CreateFile** function. **CreateFile** returns a handle that can be used to access the TPMC917 SRAM device.

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

Parameters

LPCTSTR lpFileName

Points to a null-terminated string, which specifies the name of the TPMC917 SRAM device to open. The *lpFileName* string should be of the form **\\.\TPMC917Ram_x** to open the device *x*. The ending x is a one-based number. The first TPMC917 SRAM device found by the driver is \\.\TPMC917Ram_1, for a second TPMC917 module the created SRAM device name will be \\.\TPMC917Ram_2 and so.

DWORD dwDesiredAccess

Specifies the type of access to the TPMC917 SRAM Device.
For the TPMC917 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

Pointer to a security structure. Set to NULL for TPMC917 devices.

DWORD dwCreationDistribution

Specifies which action to take on files that exist, and which action to take when files do not exist. TPMC917 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 (no overlapped I/O).

HANDLE hTemplateFile

This value must be NULL for TPMC917 SRAM devices.

Return Value

If the function succeeds, the return value is an open handle to the specified TPMC917 SRAM device. If the function fails, the return value is INVALID_HANDLE_VALUE. To get extended error information, call ***GetLastError***.

Example

```
HANDLE    hDevice;

hDevice = CreateFile(
     “\\\\.\\TPMC917Ram_1”,
     GENERIC_READ | GENERIC_WRITE,
     0,
     NULL,                    // no security attrs
     OPEN_EXISTING,           // TPMC917 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()

4.1.2 Closing a TPMC917 SRAM Device

The **CloseHandle** function closes an open TPMC917 SRAM device handle.

BOOL CloseHandle(
HANDLE *hDevice;*
);

Parameters

HANDLE hDevice

Identifies an open TPMC917 SRAM device 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 ()

4.1.3 TPMC917 SRAM 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,
    DWORD        dwIoControlCode,
    LPVOID       lpInBuffer,
    DWORD        nInBufferSize,
    LPVOID       lpOutBuffer,
    DWORD        nOutBufferSize,
    LPDWORD      lpBytesReturned,
    LPOVERLAPPED lpOverlapped
);
```

Parameters

HANDLE hDevice

Handle to the TPMC917 SRAM device that is to perform the operation.

DWORD dwIoControlCode

Specifies the control code for the operation. This value identifies the specific operation to be performed. The following values are defined in *tpmc917ram.h*:

Value	Meaning
IOCTL_TP917RAM_GET_MEMSIZE	return the available onboard memory size
IOCTL_TP917RAM_WRITE	write data into the onboard SRAM
IOCTL_TP917RAM_READ	read data from the onboard SRAM

See behind for more detailed information on each control code.

LPVOID lpInBuffer

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

DWORD nInBufferSize

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

LPVOID lpOutBuffer

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

DWORD nOutBufferSize

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

LPDWORD 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 lpOverlapped

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

To use these TPMC917 SRAM specific control codes, the header file tpmc917ram.h must be included.

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***.

Note: The TPMC917 SRAM device driver always returns standard Win32 error codes on failure. Please refer to the Windows Platform SDK Documentation for a detailed description of returned error codes.

See Also

Win32 documentation DeviceIoControl ()

4.1.3.1 IOCTL_TP917RAM_GETMEMSIZE

This TPMC917 SRAM Ioctl function returns the SRAM's memory size located on the TPMC917 module. The parameter *lpOutBuffer* passes a pointer to an *ULONG* value to the device driver.

Example

```
#include "tpmc917ram.h"

HANDLE                  hDevice;
BOOLEAN                 success;
ULONG                   NumBytes;
ULONG                   MemorySize;

success = DeviceIoControl (
     hDevice,                          // TPMC917 SRAM handle
     IOCTL_TP917RAM_GETMEMSIZE,        // control code
     NULL,
     0,
     &MemorySize,                      // buffer which receives the data
     sizeof(ULONG),                    // size of buffer
     &NumBytes,                        // number of bytes transferred
     NULL
);
if( success ) {
     printf( "Memory available: %d bytes\n", MemorySize );
}
else {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INVALID_USER_BUFFER	*The size of the supplied output buffer is too small.*

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

4.1.3.2 IOCTL_TP917RAM_WRITE

This TPMC917 SRAM Ioctl function writes a dynamically adjustable data buffer into the onboard SRAM. No data verification is performed. The parameter *lpInBuffer* passes a pointer to a TP917RAM_MEMIO_BUF structure to the device driver.

The TP917RAM_MEMIO_BUF structure has the following layout:

```
typedef struct
{
   ULONG      offset;
   ULONG      size;
   UCHAR      pData[1];        /* dynamically expandable */
} TP917RAM_MEMIO_BUF;
```

Members

offset

Specifies the offset relative to the beginning of the SRAM, where the data is to be written. The SRAM is treated like a linear contiguous memory block

size

Specifies the number of bytes to be written. The data buffer must be large enough to hold the amount of data.

pData

Specifies the data buffer, which can be dynamically enlarged. Due to restrictions of the Windows I/O manager, the data buffer together with the data structure must be available as a single contiguous block of memory.

Example

```
#include “tpmc917ram.h”

HANDLE                  hDevice;
BOOLEAN                 success;
ULONG                   NumBytes;
ULONG                   totalSize;
TP917RAM_MEMIO_BUF      *pMemIoBuf;

//
// allocate enough memory and write 13 bytes into SRAM, Offset=0
//
totalSize = sizeof(TP917RAM_MEMIO_BUF) + 13*sizeof(UCHAR);
pMemIoBuf = (TP917RAM_MEMIO_BUF*)malloc( totalSize );
memset( pMemIoBuf, 0, totalSize );
pMemIoBuf->size = 13;
pMemIoBuf->offset = 0;
```

```
sprintf( pMemIoBuf->pData, “Hello World!” );

success = DeviceIoControl (
     hDevice,                          // TPMC917 SRAM handle
     IOCTL_TP917RAM_WRITE,             // control code
     pMemIoBuf,                        // input buffer
     totalSize,                        // size of input buffer
     NULL,
     0,
     &NumBytes,                        // number of bytes transferred
     NULL
);
if( success ) {
     // data successfully written
}
else {
     // Process DeviceIoControl() error
}
free( pMemIoBuf );
```

Error Codes

ERROR_INVALID_USER_BUFFER	*The size of the supplied input buffer is too small.*
ERROR_NOT_ENOUGH_MEMORY	*Offset + size is too large for the available SRAM.*

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

4.1.3.3 IOCTL_TP917RAM_READ

This TPMC917 SRAM Ioctl function reads a dynamically adjustable data buffer from the onboard SRAM. The parameter *lpOutBuffer* passes a pointer to a TP917RAM_MEMIO_BUF structure to the device driver.

The TP917RAM_MEMIO_BUF structure has the following layout:

```
typedef struct
{
   ULONG      offset;
   ULONG      size;
   UCHAR      pData[1];          /* dynamically expandable */
} TP917RAM_MEMIO_BUF;
```

Members

offset

Specifies the offset relative to the beginning of the SRAM, from where the data is read. The SRAM is treated like a linear contiguous memory block.

size

Specifies the number of bytes to read. The data buffer must be large enough to hold the amount of data.

pData

Specifies the data buffer, which can be dynamically enlarged. Due to restrictions of the Windows I/O manager, the data buffer together with the data structure must be available as a single contiguous block of memory.

Example

```
#include “tpmc917ram.h”

HANDLE                  hDevice;
BOOLEAN                 success;
ULONG                   NumBytes;
ULONG                   totalSize;
TP917RAM_MEMIO_BUF      *pMemIoBuf;

//
// allocate enough memory and read 20 bytes from SRAM, Offset=0
//
totalSize = sizeof(TP917RAM_MEMIO_BUF) + 20*sizeof(UCHAR);
pMemIoBuf = (TP917RAM_MEMIO_BUF*)malloc( totalSize );
memset( pMemIoBuf, 0, totalSize );
pMemIoBuf->size = 20;
pMemIoBuf->offset = 0;
```

```
success = DeviceIoControl (
     hDevice,                          // TPMC917 RAM handle
     IOCTL_TP917RAM_READ,              // control code
     NULL,
     0,
     pMemIoBuf,                        // output buffer
     totalSize,                        // size of output buffer
     &NumBytes,                        // number of bytes transferred
     NULL
);
if( success ) {
     // data successfully read
     printf( “Data=’%s’\n”, pMemIoBuf->pData );
}
else {
     // Process DeviceIoControl() error
}
free( pMemIoBuf );
```

Error Codes

ERROR_INVALID_USER_BUFFER	*The size of the supplied output buffer is too small.*
ERROR_NOT_ENOUGH_MEMORY	*Offset + size is too large for the available SRAM.*

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

4.1.3.4 IOCTL_TP917RAM_GETBATSTAT

This TPMC917 SRAM Ioctl function returns the state of the SRAM's battery backup. The parameter *lpOutBuffer* passes a pointer to an *ULONG* value to the device driver. Possible returned values are *TP917_BATTERY_LOW* and *TP917_BATTERY_OK* (see tpmc917ram.h).

Example

```
#include "tpmc917ram.h"

HANDLE                  hDevice;
BOOLEAN                 success;
ULONG                   NumBytes;
ULONG                   BatteryState;

success = DeviceIoControl (
     hDevice,                          // TPMC917 SRAM handle
     IOCTL_TP917RAM_GETBATSTAT,        // control code
     NULL,
     0,
     &BatteryState,                    // buffer which receives the data
     sizeof(ULONG),                    // size of buffer
     &NumBytes,                        // number of bytes transferred
     NULL
);
if( success ) {
     if (BatteryState == TP917_BATTERY_LOW)
     {
          printf( "BATTERY LOW!!! \n" );
     } else {
          printf( "Battery OK.\n" );
     }
}
else {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INVALID_USER_BUFFER	*The size of the supplied output buffer is too small.*

All other returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

4.1.3.5 IOCTL_TP917RAM_WAITBATLOW

This TPMC917 SRAM Ioctl function waits indefinitely until the SRAM's battery backup state turns to low. No timeout can be specified, so this function should be called out of a dedicated battery monitoring thread. No additional parameter is necessary.

Example

```
#include "tpmc917ram.h"

HANDLE                  hDevice;
BOOLEAN                 success;
ULONG                   NumBytes;

success = DeviceIoControl (
     hDevice,                        // TPMC917 SRAM handle
     IOCTL_TP917RAM_WAITBATLOW,      // control code
     NULL,
     0,
     NULL,
     0,
     &NumBytes,                      // number of bytes transferred
     NULL
);
if( success ) {
     printf( "BATTERY LOW!!! \n" );
}
else {
     // Process DeviceIoControl() error
}
```

Error Codes

All returned error codes are system error conditions.

See Also

Win32 documentation DeviceIoControl()

5 Example Application

The provided example application demonstrates the usage of the above stated functions of the TPMC917 functionalities, except the serial channels. They can be used directly with a terminal application or the corresponding Windows API functions.

The example application shows the usage of the SRAM access functions as well as monitoring the state of the backup battery.

For further information on the example application please refer to the inline documentation.