The Embedded I/O Company

TEWS TECHNOLOGIES

TDRV007-SW-65

Windows 2000/XP Device Driver

ARCNET Controller

Version 1.0.x

User Manual

Issue 1.0.1

June 2008

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

TEWS TECHNOLOGIES LLC
9190 Double Diamond Parkway,
Suite 127, Reno, NV 89521, USA
www.tews.com
Phone: +1 (775) 850 5830
Fax: +1 (775) 201 0347
e-mail: usasales@tews.com

TDRV007-SW-65

Windows 2000/XP Device Driver

ARCNET Controller

Supported Modules:
TPMC815
THP815

Issue	Description	Date
1.0.0	First Issue	January 07, 2008
1.0.1	Files moved to subdirectory	June 23, 2008

Table of Contents

1 Introduction

The TDRV007-SW-65 Windows WDM (Windows Driver Model) device driver is a kernel mode driver which allows the operation of the TPMC815 product family on an Intel or Intel-compatible x86 Windows 2000, Windows XP, Windows XP embedded 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 TDRV007-SW-65 device driver supports the following features:

- Configure ARCNET node and set node online
- Remove node from ARCNET (set offline)
- Send messages
- Receive messages
- Read ARCNET map
- Get diagnostic information (reconfiguration cycles)

The TDRV007-SW-65 device driver supports the modules listed below:

TPMC815	1 Channel ARCNET	(PMC)
THP815	1 Channel ARCNET	(PC/104-Plus)

In this document all supported modules and devices will be called TDRV007. Specials for certain devices will be advised.

To get more information about the features and use of supported devices it is recommended to read the manuals listed below.

TPMC815 Product Family User manual

TPMC815 Product Family Engineering Manual

COM20020-5 ULANC Reference Manual

2 Installation

Following files are located in directory TDRV007-SW-65 on the distribution media:

tdrv007.sys	Windows 2000/XP driver binary
tdrv007.inf	Windows 2000/XP installation script
tdrv007.h	Header file with IOCTL code and structure definitions
TDRV007-SW-65-1.0.1.pdf	This document
example/tdrv007exa.c	Example application
Release.txt	Release information
ChangeLog.txt	Release history

2.1 Software Installation

2.1.1 Windows 2000 / XP

This section describes how to install the TDRV007 Device Driver on a Windows 2000 / XP operating system.

After installing the TDRV007 module(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 TDRV007 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 found device.

After successful installation the TDRV007 device driver will start immediately and create devices (TDRV007_1, TDRV007_2 ...) for all recognized TDRV007 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 "***TEWS TECHNOLOGIES – TDRV007 ARCNET (<ModuleName>)***" should appear.

3 Device Driver Programming

The TDRV007-SW-65 Windows WDM device driver is a kernel mode device driver using Direct I/O.

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 Files and I/O Functions

The following section doesn't contain a full description of the Win32 functions for interaction with the TDRV007 device driver. Only the required parameters are described in detail.

3.1.1 Opening a Device

Before you can perform any I/O the TDRV007 device must be opened by invoking the CreateFile function. CreateFile returns a handle that can be used to access the TDRV007 device.

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

Parameters

lpFileName

Points to a null-terminated string, which specifies the name of the TDRV007 device to open. The lpFileName string should be of the form \\.\TDRV007_x to open the device x. The ending x is a one-based number. The first device found by the driver is \\.\TDRV007_1, the second \\.\TDRV007_2 and so on.

dwDesiredAccess

Specifies the type of access to the TDRV007. For the TDRV007 this parameter must be set to read-write access (GENERIC_READ | GENERIC_WRITE)

dwShareMode

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

lpSecurityAttributes

Pointer to a security structure. Set to NULL for TDRV007 devices.

dwCreationDistribution

Specifies which action to take on files that exist, and which action to take when files do not exist. TDRV007 devices must be always opened **OPEN_EXISTING**.

dwFlagsAndAttributes

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

hTemplateFile

This value must be NULL for TDRV007 devices.

Return Value

If the function succeeds, the return value is an open handle to the specified TDRV007 device. If the function fails, the return value is INVALID_HANDLE_VALUE. To get extended error information, call *GetLastError*.

Example

```
HANDLE    hDevice;

hDevice = CreateFile(
     “\\\\.\\TDRV007_1”,
     GENERIC_READ | GENERIC_WRITE,
     0,
     NULL,              // no security attrs
     OPEN_EXISTING,     // TDRV007 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 Device

The CloseHandle function closes an open TDRV007 handle.

BOOL CloseHandle(
HANDLE *hDevice*
)

Parameters

hDevice

Identifies an open TDRV007 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 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

hDevice

Handle to the TDRV007 device 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 tdrv007.h:

Value	Meaning
IOCTL_TDRV007_READ	Read a packet
IOCTL_TDRV007_READ_NOWAIT	Same as IOCTL_TDRV007_READ but do not wait if no packet is available
IOCTL_TDRV007_WRITE	Transmit a packet
IOCTL_TDRV007_DIAG	Get number of node reconfigurations
IOCTL_TDRV007_MAP	Build a map of online nodes on the network
IOCTL_TDRV007_FLUSH	Flush FIFO of received packets
IOCTL_TDRV007_OFFLINE	Shutdown and remove the node from the network
IOCTL_TDRV007_ONLINE	Initialize the node and enter the network
IOCTL_TDRV007_SET_VARIANT	Setup the module variant if automatic recognition failed

See behind for more detailed information on each control code.

lpInBuffer

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

nInBufferSize

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

lpOutBuffer

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

nOutBufferSize

Specifies the size, in bytes, of the buffer 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 value must be set to NULL (no overlapped I/O).

To use these TDRV007 specific control codes the header file tdrv007.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.

Please note that the TDRV007 device driver returns always 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 ()

3.1.3.1 IOCTL_TDRV007_READ, IOCTL_TDRV007_READ_NOWAIT

This TDRV007 control function reads a data packet from the specified device. A pointer to the callers message buffer (*TP815_MSG*) is passed by the parameter *lpOutBuffer* to the driver.

After successful execution the message buffer (*lpOutBuffer)* receives the packet.

The control function *IOCTL_TDRV007_READ_NOWAIT* returns immediately with the error code *ERROR_NO_DATA* if no data are available.

```
typedef struct {
        UCHAR     SID;
        UCHAR     DID;
        USHORT    MsgLen;
        UCHAR     Data[TDRV007_MAX_LONG_MSG];
} TDRV007_MSG, *PTDRV007_MSG;
```

SID

Contains the source ID

DID

Contains the destination ID or 0 for broadcast messages

MsgLen

Contains the number of message data bytes.

Data

This buffer receives up to 253 bytes of data for short packets and up to 508 bytes for long packets.

Example

```
#include “tdrv007.h”

HANDLE              hDevice;
BOOLEAN             success;
ULONG               NumBytes;
TDRV007_MSG         MsgBuf;


success = DeviceIoControl (
     hDevice,                // device handle
     IOCTL_TDRV007_READ,     // control code
     NULL,                   // input buffer
     0,
     &MsgBuf,                // output buffer
     sizeof(MsgBuf),
     &NumBytes,              // number of bytes transferred
     NULL
);


if( success ) {
     // Process received data
}
else {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The message buffer is too small for a long or short packet.
ERROR_NOT_READY	The ARCNET controller is OFFLINE. Execute IOCTL_TDRV007_ONLINE to enter the network.
ERROR_SEM_TIMEOUT	No packet received within the specified time (DeviceReadTimeout).

All other returned error codes are system error conditions.

3.1.3.2 IOCTL_TDRV007_WRITE

This TDRV007 control function writes a packet to the specified device. A pointer to the callers message buffer (*TDRV007_MSG*) is passed by the parameter *lpInBuffer* to the driver.

```
typedef struct {
    UCHAR     SID;
    UCHAR     DID;
    USHORT    MsgLen;
    UCHAR     Data[TDRV007_MAX_LONG_MSG];
} TDRV007_MSG, *PTDRV007_MSG;
```

SID

Not used can be 0.

DID

Contains the destination ID or 0 for broadcast messages

MsgLen

Contains the number of message data bytes.

Data

This buffer receives up to 253 bytes of data for short packets and between 257 and 508 bytes for long packets. Data packets between 254 and 256 bytes must be filled with dummy data to make the packet fit into a long packet.

Example

```
#include “tdrv007.h”

HANDLE              hDevice;
BOOLEAN             success;
ULONG               NumBytes;
TDRV007_MSG         MsgBuf;

MsgBuf.DID = 1;
strcpy(MsgBuf.Data, “GET PARAMETER”);
MsgBuf.MsgLen = strlen(MsgBuf.Data);

success = DeviceIoControl (
     hDevice,                          // device handle
     IOCTL_TDRV007_WRITE,              // control code
     &MsgBuf,                          // input buffer
     sizeof(MsgBuf),
     NULL,                             // output buffer
     0,
     &NumBytes,                        // number of bytes transferred
     NULL
);

if( !success ) {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The data packet does not fit into the passed message buffer. Please check the field MsgLen of the message buffer.
ERROR_BAD_LENGTH	Illegal message length. A short may contain between 1 and 253 bytes, while a long packet contain between 257 and 508 bytes. NOTE. Data Packet length of 254, 255 and 256 must be filled with dummy bytes to make the packet fit into a long packet.
ERROR_NOT_READY	The ARCNET controller is OFFLINE. Execute IOCTL_TDRV007_ONLINE to enter the network.
ERROR_SEM_TIMEOUT	The data packet could not be sent within the specified time (DeviceWriteTimeout).
ERROR_REQ_NOT_ACCEP	No acknowledge received after transmission

All other returned error codes are system error conditions.

3.1.3.3 IOCTL_TDRV007_DIAG

This TDRV007 control function gets the number of node reconfigurations. The number of node reconfigurations reflects the quality of the network. If the number grows up soon there is something wrong on the network.

The internal reconfiguration counter will be reset after reading.

A pointer to a ULONG variable which receives the current counter value is passed by the parameters *lpOutBuffer* to the driver.

Example

```
#include “tdrv007.h”

HANDLE        hDevice;
BOOLEAN       success;
ULONG         NumBytes;
ULONG         NumRecons;

success = DeviceIoControl (
     hDevice,                    // device handle
     IOCTL_TDRV007_DIAG,         // control code
     NULL,                       // input buffer
     0,
     &NumRecons,                 // output buffer
     sizeof(ULONG),
     &NumBytes,                  // number of bytes transferred
     NULL
);

if( success ) {
     printf( "\n%ld recons since last query\n", NumRecons );
}
else {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The reconfiguration counter (ULONG) does not fit into the passed buffer. Please check the parameter nOutBufferSize of the DeviceIoControl () function call.

All other returned error codes are system error conditions.

3.1.3.4 IOCTL_TDRV007_MAP

This TDRV007 control function builds a map of nodes contained on the network. The driver puts the map into a user defined buffer with 32 bytes in length. Every bit (256) in this buffer represents a node on the network. Bits in this buffer are set (present) or reset (not present) depending on if a node is present or not. The first byte in the buffer presents nodes 0 through 7; the second byte presents 8 through 15, and so on. The least significant bit of a byte presents the smallest node number.

A pointer to the buffer which receives the node map must be passed by the parameters *lpOutBuffer* to the driver.

Example

```
#include “tdrv007.h”

HANDLE        hDevice;
BOOLEAN       success;
ULONG         NumBytes;
UCHAR         map[TDRV007_MAX_MAP_SIZE];

success = DeviceIoControl (
     hDevice,                    // device handle
     IOCTL_TDRV007_MAP,          // control code
     NULL,                       // input buffer
     0,
     map,                        // output buffer
     TDRV007_MAX_MAP_SIZE,
     &NumBytes,                  // number of bytes transferred
     NULL
);

if( success ) {
     printf("Network Map:\n\t");

     for (i = 0; i < TDRV007_MAX_MAP_SIZE; i++) {
          if ((i % 16) == 0) printf("\n\t");
          printf("%02X ",map[i]);
     }
}
else {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The node map does not fit into the passed buffer. Please check the parameter nOutBufferSize of the DeviceIoControl () function call.
ERROR_NOT_READY	The ARCNET controller is OFFLINE. Execute IOCTL_TDRV007_ONLINE to enter the network.
ERROR_NETWORK_UNREACHABLE	No token was seen. The network seems to be down.

All other returned error codes are system error conditions.

3.1.3.5 IOCTL_TDRV007_FLUSH

This TDRV007 control function flushes the driver's internal message FIFO.

No additional parameters are required for this call.

Example

```
#include "tdrv007.h"

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;

success = DeviceIoControl (
     hDevice,                    // device handle
     IOCTL_TDRV007_FLUSH,        // control code
     NULL,                       // input buffer
     0,
     NULL,                       // output buffer
     0,
     &NumBytes,                  // number of bytes transferred
     NULL
);

if( !success ) {
     // Process DeviceIoControl() error
}
```

3.1.3.6 IOCTL_TDRV007_OFFLINE

This TDRV007 control function shuts down the node hardware and removes the node from the network.

No additional parameters are required for this call.

Example

```
#include “tdrv007.h”

HANDLE                  hDevice;
BOOLEAN                 success;
ULONG                   NumBytes;

success = DeviceIoControl (
     hDevice,                    // device handle
     IOCTL_TDRV007_OFFLINE,      // control code
     NULL,                       // input buffer
     0,
     NULL,                       // output buffer
     0,
     &NumBytes,                  // number of bytes transferred
     &Overlapped
);


if( !success ) {
     // Process DeviceIoControl() error and free allocated memory
}
```

3.1.3.7 IOCTL_TDRV007_ONLINE

This TDRV007 control function initializes the ARCNET controller and sets the controller online to enter the network.

After successful execution of this control function the ARCNET controller is connected to the network and packets can be received from and transmitted to the network.

A pointer to the caller's configuration parameter (*TDRV007_CONFIG*) is passed by the parameter *lpInBuffer* to the driver.

```
typedef struct {
    BOOLEAN   Broadcast;
    BOOLEAN   LongPacket;
    USHORT    NodeID;
    USHORT    NetworkTimeout;
    USHORT    Speed;
    ULONG     DeviceWriteTimeout;
    ULONG     DeviceReadTimeout;
} TDRV007_CONFIG, *PTDRV007_CONFIG;
```

Broadcast

If this parameter is TRUE (1) the controller will accept broadcast messages from the network.

LongPacket

If this parameter is TRUE (1) the controller will receive both short and long packets, otherwise only short packets will be accepted.

NodeID

Specifies the node ID used by this controller in range between 1 and 255. If this parameter is set to 0, the node ID will be determined by reading the hardware DIP switch.

NetworkTimeout

Specifies the response, idle and recon timing of the ARCNET controller. The resulting timing depends on the configured network speed (see below) and the value of the bits ET1 and ET2 in the configuration register of the controller. All nodes should be configured with the same timeout value for proper network operation.
Valid values are:

Value	ET2	ET1	Response (µs)	Idle Time (µs)	Reconfig (µs)
0	0	0	596.8	656	840
1	0	1	298.4	328	840
2	1	0	74.7	82	840
3	1	1	37.35	41	420

For slower data rates, an internal clock divider scales down the clock frequency. Thus all timeout values are scaled up as shown in the following table:

Data Rate	Timeout scaling factor (multiply by)
5 Mbps	1
2.5 Mbps	2
1.25 Mbps	4
625 Kbps	8
312.5 Kbps	16

Speed

This parameter determines the network speed by setting the clock prescaler. The following speed values are predefined:

Symbol	Network Speed
TDRV007_5000KBPS	5 Mbps
TDRV007_2500KBPS	2.5 Mbps
TDRV007_1250KBPS	1.25 Mbps
TDRV007_625KBPS	625 Kbps
TDRV007_312KBPS	312.5 Kbps

DeviceWriteTimeout

Specifies the timeout in seconds for all following *IOCTL_TDRV007_WRITE* control functions.

DeviceReadTimeout

Specifies the timeout in seconds for all following *IOCTL_TDRV007_READ* control functions.

Example

```
#include “tdrv007.h”

HANDLE              hDevice;
BOOLEAN             success;
ULONG               NumBytes;
TDRV007_CONFIG      Config;

//
//   Setup node 123 to accept broadcast messages and long
//   packets with a network speed of 2.5 Mbps
//
Config.Broadcast            = TRUE;
Config.LongPacket           = TRUE;
Config.NodeID               = 123;
Config.NetworkTimeout       = 0;
Config.Speed                = TDRV007_2500KBPS;
Config.DeviceWriteTimeout   = 2;
Config.DeviceReadTimeout    = 2;

success = DeviceIoControl (
     hDevice,                    // device handle
     IOCTL_TDRV007_ONLINE,       // control code
     &Config,                    // input buffer
     sizeof(TDRV007_CONFIG),
     NULL,                       // output buffer
     0,
     &NumBytes,                  // number of bytes transferred
     NULL
);

if( !success ) {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The input buffer is too small for the configuration structure. Please check the parameter nInBufferSize of the DeviceIoControl () function call.
ERROR_INVALID_PARAMETER	Some configuration parameter (NodeID, NetworkTimeout, Speed) are out of range.
ERROR_NETWORK_UNREACHABLE	Unable to enter the network
STATUS_DUPLICATE_OBJECTID	Illegal or duplicate node ID
ERROR_NOT_READY	The device driver can't recognize the module variant automatically. This could occur due to an error of the PLX9050 PCI interface chip for certain memory configurations. Please use the special device I/O control function *IOCTL_TDRV007_SET_VARIANT* to setup module variant manually.

3.1.3.8 IOCTL_TDRV007_SET_VARIANT

This TDRV007 control function setup the TPCM815 variant code, if automatic recognition doesn't work (*IOCTL_TDRV007_ONLINE* returns *ERROR_NOT_READY*). This could occur due to an error of the PLX9050 PCI interface chip for certain memory configurations.

This device I/O control function is only relevant for TPMC815 V1.0 modules. Newer TPMC815 modules (Revision 2.0 or higher) or THP815 modules contain the variant information in the always readable PCI header. For these modules this I/O control function can be ignored.

A pointer to an unsigned char variable, which contains the variant code, is passed by the parameter *lpInBuffer* to the driver. Possible variant codes are defined in tdrv007.h. Use TPMC815_11 for a TPMC815-11 with traditional isolated hybrid interface or TPMC815_21 for a TPCM815-21 with isolated RS485 differential driver interface.

Example

```
#include "tdrv007.h"

HANDLE    hDevice;
BOOLEAN   success;
ULONG     NumBytes;
UCHAR  variant;

variant = TPMC815_21;

success = DeviceIoControl (
     hDevice,                          // device handle
     IOCTL_TDRV007_SET_VARIANT,        // control code
     &variant,                         // input buffer
     sizeof(UCHAR),
     NULL,                             // output buffer
     0,
     &NumBytes,                        // number of bytes transferred
     NULL
);
if( !success ) {
     // Process DeviceIoControl() error
}
```

Error Codes

ERROR_INSUFFICIENT_BUFFER	The node map does not fit into the passed buffer. Please check the parameter nOutBufferSize of the DeviceIoControl () function call.
ERROR_INVALID_PARAMETER	Invalid module variant code. Valid are 11 for a TPMC815-11 and 21 for a TPCM815-21.
ERROR_ACCESS_DENIED	The module variant was detected automatically, so it cannot be changed.

All other returned error codes are system error conditions.