WinSystems

Watchdog Timer
Windows XP
Driver Documentation

Table of Contents

1 Introduction

Many of WinSystems' full-featured, high-performance single board computers offer an integrated watchdog timer, which can be used to guard against software lockups. The watchdog can be configured from the CMOS setup utility or directly from software. The selection in the CMOS setting serves as the default timeout value as the processor boots. The BIOS option is for enabling the watchdog only during boot. The provided driver allows control of the watchdog timer under the Windows XP operating system and has been tested using the Windows XP Operating System Service Pack 2.

1.1 Hardware Requirements

This driver has been designed to work with the following WinSystems' products.

- EBC-Z5xx
- EBC-855
- EPX-855
- PPM-LX800-G
- LPM-LX800-G

1.2 Host System Requirements

Your development machine should contain the following:

- Microsoft Windows XP.

1.3 Driver Deliverables

The following file(s) should be downloaded from the WinSystems website

- WSWDTXP.zip. This zip file contains the WinSystems WSWDTXP Driver and support files for the Windows XP Operating System.

2 Driver Integration

2.1 Driver Installation

Step 1 Unzip the WSWDTXP.zip file into a folder on the computer containing the Microsoft Windows XP OS.

Step 2 Installation is accomplished via the "Add New Hardware" applet in the Windows control panel. Select "Have Disk" and navigate to the directory containing the driver files. Once selected, the Windows installer will copy the PCMMIO.SYS file to the appropriate place in the Windows installation.

Step 3 The default hardware configuration for installation is I/O port 565H. The driver I/O port setting cannot be changed using the Device Manager.

2.2 Driver Files

The following files are included in WSWDTXP.zip:

- WSWDTXP.inf

  This is the WinSystems WDT Driver Windows XP installation file.

- WSWDTXP.cat

  This is a dummy catalog file. You may submit this driver to the Microsoft Windows Hardware Quality Lab (WHQL) for WHQL certification.

- WSWDTXP.sys

  This is the retail build of the WinSystems WDT Driver.

- Ioctls.h

  This file contains structure definitions and IOCTL definitions to be used by an application to communicate with the WinSystems WDT Driver. This file is included in this document in Appendix A: Ioctls.h.

3 Using the Driver

The IOCTLS.H file included in the driver distribution files contains structure definitions and IOCTL definitions to be used by an application to communicate with the WinSystems WSWDTXP Driver.

An application uses the Windows API function *CreateFile* to open the driver. The following example opens the first instance of the WinSystems WDT Driver.

```
HANDLE hWIOHandle = CreateFile(  _T("\\\\.\\WSWDT"),
                                 GENERIC_READ | GENERIC_WRITE,
                                 0,
                                 NULL,
                                 OPEN_EXISTING,
                                 FILE_FLAG_OVERLAPPED,
                                 NULL);
```

Note: You must use the FILE_FLAG_OVERLAPPED file attribute flag in the call to CreateFile.

Once a handle to the driver has been obtained by calling *CreateFile*, the handle is used in calls to the Windows API function *DeviceIoControl* to access the driver's features.

```
BYTE write_data;
DWORD back;
DeviceIoControl(hWDTHandle,
              IOCTL_WRITE_WDT,
              &write_data,
              sizeof(BYTE),
              NULL,
              0,
              &back,
              NULL);
```

When finished using the WinSystems WDT Digital I/O Driver a call should be made to the Windows API function *CloseHandle*.

```
CloseHandle(hWDTHandle);
```

The following is a description of each of the driver IOCTL functions that can be called via the Windows API function *DeviceIoControl*. Information as to the *DeviceIoControl* parameter definitions can be found in the IOCTLS.H file. This file is included in this document in Appendix A: Ioctls.h.

3.1 IOCTL_WRITE_WDT

Writes an 8-bit value to the watchdog timer register found at I/O port 566H. This function is used to both set and pet the timer. The watchdog is enabled by writing a timeout value other than zero. Writing 00h will disable the watchdog. The watchdog timer is serviced by writing the timeout to I/O port 566H. If the watchdog has not been serviced within the allotted time, the circuit resets the CPU.

3.2 IOCTL_READ_WDT

Reads an 8-bit value from the watchdog timer register found at I/O port 566H.

3.3 IOCTL_ENBL_SEC

Sets the MSB found at I/O port 565H. This results in the time-out value being measured in seconds.

3.4 IOCTL_ENBL_MIN

Clears the MSB found at I/O port 565H. This results in the time-out value being measured in minutes.

4 Appendix A: Ioctls.h

```
// IOCTLS.H -- IOCTL code definitions for WSWDTXP driver
// Copyright (c) WinSystems.  All rights reserved.

#ifndef IOCTLS_H
    #define IOCTLS_H

#ifndef CTL_CODE
    #pragma message("CTL_CODE undefined. Include winioctl.h or wdm.h")
#endif

typedef unsigned char BYTE;
typedef unsigned char* PBYTE;
typedef unsigned short WORD;
typedef unsigned short* PWORD;
typedef unsigned long DWORD;
typedef unsigned long* PDWORD;

// Define the type of device. This parameter can be no bigger than a WORD
// value. The values used by Microsoft are in the range of 0-32767 and the
// values between 32768-65535 are reserved for use by OEMs and IHVs
#define WDT_DEV_TYPE         32768

//-----------------------------------------------------------------------------
// IOCTL_READ_WDT
//
// code             IOCTL_READ_WDT
// cbin             Ignored
// cbout            [IN] Size of output data buffer, should be sizeof(BYTE)
// pCopyBuffer      [OUT] Pointer to a BYTE containing the data read
// info             [OUT] Number of output bytes put into the copy buffer
//-----------------------------------------------------------------------------
#define IOCTL_READ_WDT   CTL_CODE(WDT_DEV_TYPE, 0x800, METHOD_BUFFERED,
                                  FILE_ANY_ACCESS)

//-----------------------------------------------------------------------------
// IOCTL_WRITE_WDT
//
// code             IOCTL_WRITE_WDT
// cbin             [IN] Size of input data buffer, should be sizeof(BYTE)
// cbout            Ignored
// pCopyBuffer      [IN] Pointer to a BYTE containing the data to write
// info             Ignored
//-----------------------------------------------------------------------------
#define IOCTL_WRITE_WDT   CTL_CODE(WDT_DEV_TYPE, 0x801, METHOD_BUFFERED,
                                   FILE_ANY_ACCESS)

//-----------------------------------------------------------------------------
// IOCTL_ENABL_SEC
//
// code             IOCTL_ENABL_SEC
// cbin             Ignored
// cbout            Ignored
// pCopyBuffer      Ignored
// info             Ignored
//-----------------------------------------------------------------------------
#define IOCTL_ENABL_SEC   CTL_CODE(WDT_DEV_TYPE, 0x802, METHOD_BUFFERED,
                                   FILE_ANY_ACCESS)
```

```
//-----------------------------------------------------------------------------
// IOCTL_ENABL_MIN
//
// code             IOCTL_ENABL_MIN
// cbin             Ignored
// cbout            Ignored
// pCopyBuffer      Ignored
// info             Ignored
//-----------------------------------------------------------------------------
#define IOCTL_ENABL_MIN   CTL_CODE(WDT_DEV_TYPE, 0x803, METHOD_BUFFERED,
                                   FILE_ANY_ACCESS)

#endif
```