CC Pilot™ XS

I/O device interface description

CC SYSTEMS

Table of Contents

Introduction

Purpose

This document describes the software interface for accessing I/O.

References

-

History

Rev	Date	Author	Remarks
1.0	2006-01-09	Göran Nordin	First version.
1.1	2006-01-11	Göran Nordin	Name changed from TEXT(“IO1”) to TEXT (“IOP1”).
1.2	2008-04-08	Fredrik Lans	Revision

I/O device interface

IOCTLs are used to communicate with the I/0 device. The name of the I/0 device is “IOP1” in unicode character set.

Summary of IOCTLs

IOCTL_IO_READ	Reads the contents of an I/O register.
IOCTL_IO_WRITE	Writes data to an I/O register.

IOCTL_IO_READ

Description

Reads the contents of an I/O register. The windows DeviceIoControl function passes the IOCTL to the I/0 device.

Include files

#include "IoDrv.h"

Parameters supplied to DeviceIoControl

hDevice	Handle to the I/0 device. To obtain a device handle, call the CreateFile function with *lpFileName* parameter set to TEXT("IOP1:").
dwIoControlCode	Set to IOCTL_IO_READ.
lpInBuffer	Pointer to a unsigned short which contains the offset of the register to read.
nInBufferSize	Set to sizeof(unsigned short).
lpOutBuffer	Pointer to a unsigned short where the value of the register will be returned.
nOutBufferSize	Set to sizeof(unsigned short).
lpBytesReturned	Pointer to a variable that receives the size, in bytes, of the data stored into the buffer pointed to by *lpOutBuffer*.
lpOverlapped	Ignored; set to NULL

Return value

TRUE if operation succeeded otherwise FALSE. If operation failed then "GetLastError" can be used to get more information of the error.

Restrictions

-

Example

```
HANDLE hDevice;
```

```
if ((hDevice =
  CreateFile(
  TEXT("IOP1:"),
  GENERIC_READ | GENERIC_WRITE,
  FILE_SHARE_READ | FILE_SHARE_WRITE,
  NULL,
  OPEN_EXISTING,
  0,
  NULL)) != INVALID_HANDLE_VALUE)
{
  unsigned long bytesReturned;
  unsigned short offset = 0x0000;
  unsigned short value;

  if (DeviceIoControl(
    hDevice,
    IOCTL_IO_READ,
    &offset,
    sizeof(offset),
    &value,
    sizeof(value),
    &bytesReturned,
    NULL))
  {
    printf("Value at offset %hu is %u\n", offset, value);
  }
  else
  {
    printf(
      "!!ERROR, %lu when calling \"DeviceIoControl\"\n",
      GetLastError());
  }
}
else
{
  printf(
    "!!ERROR, %lu when calling \"CreateFile\"\n",
    GetLastError());
}
```

IOCTL_IO_WRITE

Description

Writes data to an I/O register. The windows DeviceIoControl function passes the IOCTL to the I/0 device.

Include files

#include "IoDrv.h"

Parameters supplied to DeviceIoControl

hDevice	Handle to the I/0 device. To obtain a device handle, call the CreateFile function with *lpFileName* parameter set to TEXT("IOP1:").
dwIoControlCode	Set to IOCTL_IO_WRITE.

lpInBuffer	Pointer to an IoctlIoWrite structure. The IoctlIoWrite structure is as follows: typedef struct _ IoctlIoWrite { unsigned short offset; unsigned short mask; unsigned short value; } IoctlIoWrite; *offset* is the offset of the I/O register. *mask* indicates which bits in value that are valid. A '1' means that bit is valid. *value* is the value to set.
nInBufferSize	Set to sizeof(IoctlIoWrite).
lpOutBuffer	Set to NULL.
nInBufferSize	Set to zero.
lpBytesReturned	Pointer to a variable that receives the size, in bytes, of the data stored into the buffer pointed to by *lpOutBuffer*.
lpOverlapped	Ignored; set to NULL

Return value

TRUE if operation succeeded otherwise FALSE. If operation failed then "GetLastError" can be used to get more information of the error.

Restrictions

-

Example

```
HANDLE hDevice;

if ((hDevice =
  CreateFile(
  TEXT("IOP1:"),
  GENERIC_READ | GENERIC_WRITE,
  FILE_SHARE_READ | FILE_SHARE_WRITE,
  NULL,
  OPEN_EXISTING,
  0,
  NULL)) != INVALID_HANDLE_VALUE)
{
  unsigned long bytesReturned;
  IoctlIoWrite ioctlIoWrite = { 0x0006, 0x0001, 0x0001};

  if (!DeviceIoControl(
    hDevice,
    IOCTL_IO_WRITE,
    &ioctlIoWrite,
    sizeof(ioctlIoWrite),
    NULL,
    0,
    &bytesReturned,
    NULL))
    {
      printf(
        "!!ERROR, %lu when calling \"DeviceIoControl\"\n",
        GetLastError());
    }
  }
  else
  {
    printf(
      "!!ERROR, %lu when calling \"CreateFile\"\n",
      GetLastError());
  }
```

I/O map

Offset 0

Bit:	Type:	Description:
0	Read Only	Digital input 0.
1	Read Only	Digital input 1.
2	Read Only	Digital input 2.
3	Read Only	Reserved.
4	Read Only	Reserved.
5	Read Only	Reserved.
6	Read Only	Reserved.
7	Read Only	Reserved.
8	Read Only	Reserved.
9	Read Only	Reserved.
10	Read Only	Reserved.
11	Read Only	Reserved.
12	Read Only	Reserved.
13	Read Only	Reserved.
14	Read Only	Reserved.
15	Read Only	Reserved.

Offset 6

Bit:	Type:	Description:
0	Read/Write	External 12 output on/off (‘1’ on).
1	Read/Write	External power output on/ off (‘1’ on).
2	Read Only	Reserved.
3	Read Only	Reserved.
4	Read Only	Reserved.
5	Read Only	Reserved.
6	Read Only	Reserved.
7	Read Only	Reserved.
8	Read Only	Reserved.
9	Read Only	Reserved.
10	Read Only	Reserved.
11	Read Only	Reserved.
12	Read Only	Reserved.
13	Read Only	Reserved.
14	Read Only	Reserved.
15	Read Only	Reserved.

Configuration

The configuration parameters are stored in the registry key:
“HKEY_CURRENT_USER\ControlPanel\Io”.

The named event TEXT(“IoCfgChangeEvent”) can be set to make the driver read configuration values without requiring a restart.

The configuration values are intended for automatic I/O operation for different power states.

If an application should control I/O, via IOCTL_IO_WRITE, then those I/O offsets should not be present in the configuration values.

Summary of configuration values

Name:	Type:	Description:
D0Io	REG_BINARY	Defines I/O values that should automatically be set when entering the D0 state. The format of the data is a number of IoctlIoWrite structures. A value of 00, 06, 00, 01, 00, 01, will set bit 0 of register at offset 0x0006 when entering the D0 state.
D1Io	REG_BINARY	Defines I/O values that should automatically be set when entering the D1 state. Format is identical to D0IoValue.
D2Io	REG_BINARY	Defines I/O values that should automatically be set when entering the D1 state. Format is identical to D0IoValue.
BatteryIo	REG_BINARY	Defines I/O values that should automatically be set when power is changed from AC to battery. This value overrides all “DxIo” values. Format is identical to D0IoValue.