

Jetbox 8210 DIO Programming
Application Note

Version 0.1, Feb 2007

www.korenix.com

Copyright Notice

Copyright© 2007 Korenix Technology Co., Ltd.

All rights reserved.

Reproduction without permission is prohibited.

Information provided in this manual is intended to be accurate and reliable. However, the original manufacturer assumes no responsibility for its use, or for any infringements upon the rights of third parties that may result from its use.

The material in this document is for product information only and is subject to change without notice. While reasonable efforts have been made in the preparation of this document to assure its accuracy, Korenix assumes no liabilities resulting from errors or omissions in this document, or from the use of the information contained herein. Korenix reserves the right to make changes in the product design without notice to its users.

Acknowledgments

Korenix is a registered trademark of Korenix Technology Co., Ltd.

All other trademarks or registered marks in the manual belong to their respective manufacturers.

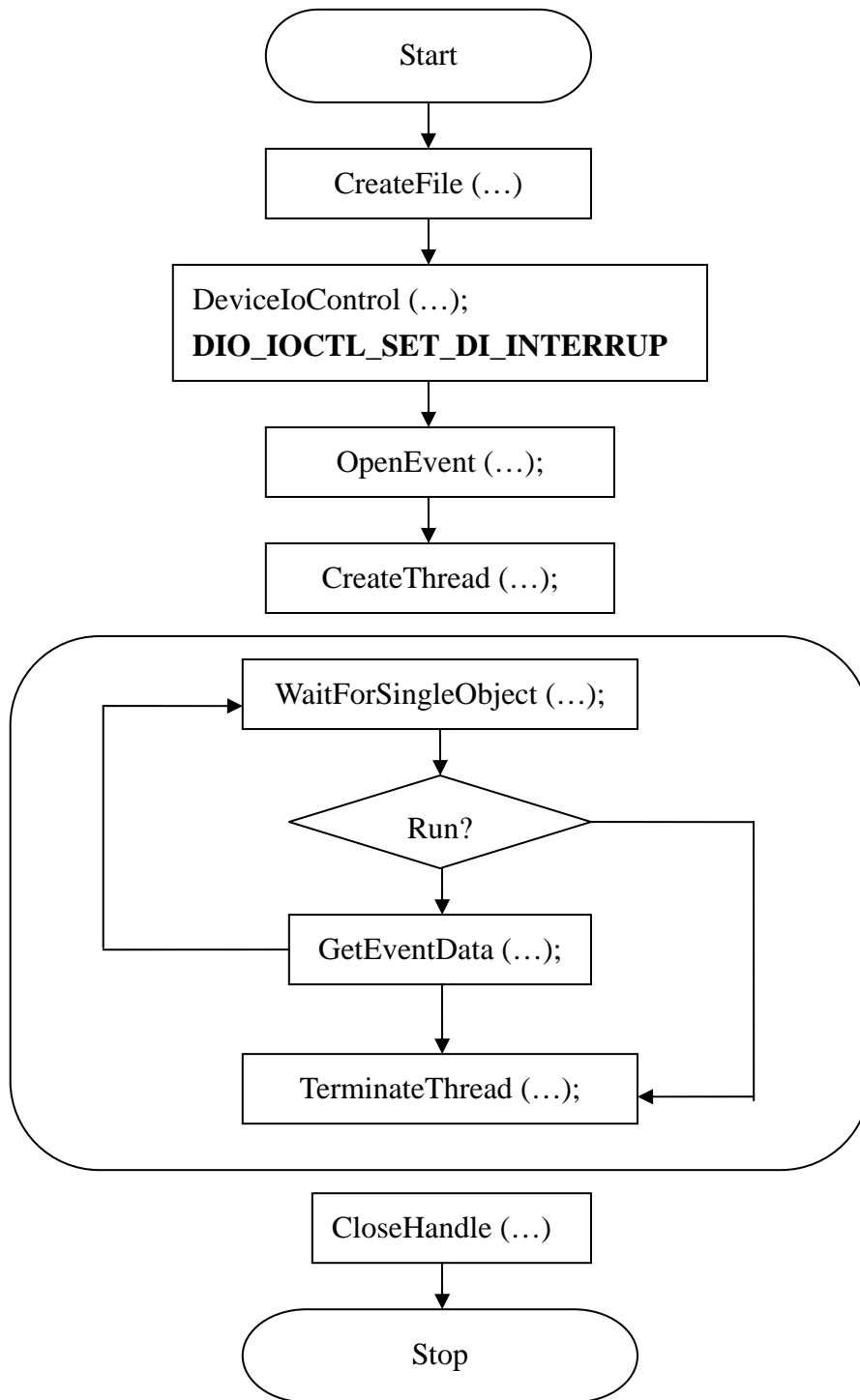
Programming with DIO Driver

Introduction

The DIO is operated as a GPIO device by the driver “DIO.DLL” on JetBox. This device supports 16 bits output and 16 bits input.

The device driver provides IO Control functions to direct access the device or use the Windows named event to capture the change of the state.

The following flowchart shows the call flow of a demo application using DIO driver with event.



Application Programming Interfaces

The programming interfaces are described as following, and refer with the Windows CE 5.0 document for details.

CreateFile

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

This function creates, opens, or truncates a file, COM port, device, service, or console. It returns a handle that you can use to access the object.

Coding Examples:

```
// Open One DIO device  
HANDLE hDIODevice = CreateFile( _T("DIO1:"),  
                                GENERIC_READ | GENERIC_WRITE,  
                                0,  
                                NULL,  
                                OPEN_EXISTING,  
                                0,  
                                0);
```

DeviceIoControl

```
BOOL DeviceIoControl(  
  
    HANDLE hDevice,  
  
    DWORD dwIoControlCode,  
  
    LPVOID lpInBuffer,  
  
    DWORD nInBufferSize,  
  
    LPVOID lpOutBuffer,  
  
    DWORD nOutBufferSize,  
  
    LPDWORD lpBytesReturned,  
  
    LPOVERLAPPED lpOverlapped  
  
);
```

This function sends an IOCTL directly to a specified device driver, causing the corresponding device to perform the specified operation.

DIO_IOCTL_READ_DI

```
BOOL DeviceIoControl(  
  
    HANDLE hDevice,  
  
    DWORD dwIoControlCode,  
  
    LPVOID lpInBuffer,  
  
    DWORD nInBufferSize,  
  
    LPVOID lpOutBuffer,  
  
    DWORD nOutBufferSize,  
  
    LPDWORD lpBytesReturned,  
  
    LPOVERLAPPED lpOverlapped  
  
);
```

This IOCTL is used to get the DI states of all the 16 DI channels.

Coding Examples:

```
// Read the DI state of all the 16 channels  
DWORD dwActual = 0;  
UCHAR ucIoControlBuf[2] = {0x00,0x00};  
BOOL bReturn = DeviceIoControl(hDIODevice,  
                                DIO_IOCTL_READ_DI,  
                                NULL,  
                                0,  
                                ucIoControlBuf,  
                                sizeof(ucIoControlBuf),  
                                &dwActual,  
                                NULL  
                                );
```

DIO_IOCTL_READ_DO

```
BOOL DeviceIoControl(  
  
    HANDLE hDevice,  
  
    DWORD dwIoControlCode,  
  
    LPVOID lpInBuffer,  
  
    DWORD nInBufferSize,  
  
    LPVOID lpOutBuffer,  
  
    DWORD nOutBufferSize,  
  
    LPDWORD lpBytesReturned,  
  
    LPOVERLAPPED lpOverlapped  
  
);
```

This IOCTL is used to get the DO states of all the 16 DO channels.

Coding Examples:

```
// Read the DO states of all the 16 channels  
DWORD dwActual = 0;  
UCHAR ucIoControlBuf[2] = {0x00,0x00};  
BOOL bReturn = DeviceIoControl(hDIODevice,  
                                DIO_IOCTL_READ_DO,  
                                ucIoControlBuf,  
                                sizeof(ucIoControlBuf),  
                                NULL,  
                                0,  
                                &dwActual,  
                                NULL  
                                );
```

DIO_IOCTL_WRITE_DO

```
BOOL DeviceIoControl(  
  
    HANDLE hDevice,  
  
    DWORD dwIoControlCode,  
  
    LPVOID lpInBuffer,  
  
    DWORD nInBufferSize,  
  
    LPVOID lpOutBuffer,  
  
    DWORD nOutBufferSize,  
  
    LPDWORD lpBytesReturned,  
  
    LPOVERLAPPED lpOverlapped  
  
);
```

This IOCTL is used to set the DO states of all the 16 channels.

Coding Examples:

```
// Write the DO states of all the 16 channels  
DWORD dwActual = 0;  
UCHAR ucIoControlBuf[2] = {0xAA,0x55};  
BOOL bReturn = DeviceIoControl(hDIODevice,  
                                DIO_IOCTL_WRITE_DO,  
                                NULL,  
                                0,  
                                ucIoControlBuf,  
                                sizeof(ucIoControlBuf),  
                                &dwActual,  
                                NULL  
                                );
```

DIO_IOCTL_WRITE_CHANNEL

```

BOOL DeviceIoControl(

    HANDLE hDevice,

    DWORD dwIoControlCode,

    LPVOID lpInBuffer,

    DWORD nInBufferSize,

    LPVOID lpOutBuffer,

    DWORD nOutBufferSize,

    LPDWORD lpBytesReturned,

    LPOVERLAPPED lpOverlapped

);

```

This IOCTL is used to set the DO state of one channel, and the lpInBuf should be a pointer to a TDOChannelData struct.

```

typedef struct {
    DWORD dwDOChannel;
    BOOL fSetHigh;
} TDOChannelData;

```

Member

dwDOChannel DO channel index.

fSetHigh DO channel state, and the value “TRUE” set the channel to high and the value “FALSE” set the channel to low.

Coding Examples:

```
// Set the DO channel 0 to be high.
```

```
TDOChannelData tDoChannelData;
tDoChannelData.dwDOChannel = 0;
tDoChannelData.FSetHigh = TRUE;
BOOL bReturn = DeviceIoControl(hDIODDevice,
                               DIO_IOCTL_WRITE_DO_CHANNEL,
                               &tDoChannelData,
                               sizeof(tDoChannelData),
                               NULL,
                               0,
                               &dwActual,
                               NULL
                               );
```

DIO_IOCTL_SET_DI_INTERRUPT

```
BOOL DeviceIoControl(

    HANDLE hDevice,

    DWORD dwIoControlCode,

    LPVOID lpInBuffer,

    DWORD nInBufferSize,

    LPVOID lpOutBuffer,

    DWORD nOutBufferSize,

    LPDWORD lpBytesReturned,

    LPOVERLAPPED lpOverlapped

);
```

This IOCTL is used to set the DI interrupt condition of one channel, and the lpInBuf should be a pointer to a **TDIInterruptData** structure.

Interrupt Condition List : (enum **TDIIInterruptType**)

DI_IntType_Disable	Disable the channel interrupt. (Default)
DI_IntType_BothEdge	When the channel state of change.
DI_IntType_RiseEdge	When state Low change to Hight.
DI_IntType_FallEdge	When state Hight change to Low.
DI_IntType_HighLevel	When channel state is Hight.
DI_IntType_LowLevel	When channel state is Low.
DI_IntType_Last	Reserve.

```
typedef enum {
    DI_IntType_Disable,
    DI_IntType_BothEdge,
    DI_IntType_RiseEdge,
    DI_IntType_FallEdge,
    DI_IntType_HighLevel,
    DI_IntType_LowLevel,
    DI_IntType_Last
} TDIIInterruptType;
```

```
typedef struct {
    DWORD dwDIChannel;
    TDIIInterruptType InterruptType;
} TDIIInterruptData;
```

Member

- dwDIChannel** DI channel index.
- TDIIInterruptType** DI Interrupt Type.

Coding Examples:

```
// Set the DI Interrupt conditional.
DOWRD dwBytesReturned = 0;
TDIIInterruptData DIInterruptData;
DIInterruptData.dwDIChannel = 0;
DIInterruptData.InterruptType = DI_IntType_BothEdge;
BOOL bReturn = DeviceIoControl(hDIIODevice,
                               DIO_IOCTL_SET_DI_INTERRUPT,
```

```

        &DIInterruptData,
        sizeof(DIInterruptData),
        NULL,
        0,
        &dwBytesReturned,
        NULL);
    
```

OpenEvent

```

HANDLE OpenEvent(

    DWORD dwDesiredAccess,

    BOOL bInheritHandle,

    LPCTSTR lpName

);
    
```

This function opens an existing named event object.

Predefined Event Name

The predefined event name is shown as following,

Prefix	Index	Name
DICH	0	DICH0
DICH	1	DICH1
DICH	2	DICH2
...
DICH	15	DICH15

Coding Examples:

```

Handle hCH0Event = OpenEvent(EVENT_ALL_ACCESS,
                             FALSE,
                             "DICH0"
                             );
    
```

WaitForSingleObject

```
DWORD WaitForSingleObject(  
  
    HANDLE hHandle,  
  
    DWORD dwMilliseconds  
  
);
```

This function returns when the specified object is in the signaled state or when the time-out interval elapses.

Coding Examples:

```
DWORD dwReturn = WaitForSingleObject(hCH0Event, INFINITE);  
if (dwReturn == WAIT_OBJECT_0)  
{  
    // Do something when DI channel change of state event was inserted.  
}
```

GetEventData

```
DWORD GetEventData(  
  
    HANDLE hEvent  
  
);
```

This function retrieves data associated with an event. Note: the return value zero doesn't mean any error happens.

Coding Examples:

```
DWORD dwState = GetEventData(hCH0Event);  
if (dwState == 1)  
{
```

```
    // Do something when the channel 1 was high.  
} else {  
    // Do something when the channel 0 was low.  
}
```

CloseHandle

```
BOOL CloseHandle(  
  
    HANDLE hObject  
  
);
```

This function closes an opened device object handle or an opened event handle.

Coding Examples:

```
// Close the opened event  
BOOL bReturn = CloseHandle(hCH0Event  
    );
```

```
// Close the opened DIO device  
BOOL bReturn = CloseHandle(hDIODevice  
    );
```