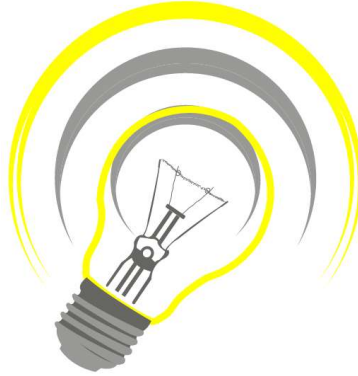


BCS Ideas Corporation



RP2005.dll Function List

Table of Contents

RP_ListDIO.....	3
RP_OpenDIO.....	4
RP_CloseDIO.....	5
RP_ReadPort.....	6
RP_ReadAll.....	7
RP_WritePort.....	9
RP_WriteAll.....	10
RP_SetWDT_Outputs.....	12
RP_GetVer.....	13

RP_ListDIO

Get information concerning the devices currently connected. This function returns the number of devices connected, and each device's serial number and product description.

Compatibility

RP Series : All

DIO Series : All

unsigned long **RP_ListDIO** (int * *iNumBrds*, char * *SN*, char * *Desc*)

Parameters

<i>iNumDev</i>	The number of RP2005 devices currently attached to USB
<i>SN</i>	Comma delimited string containing the serial number of each RP2005 device currently attached to USB
<i>Desc</i>	Comma delimited string containing the device description of each RP2005 device currently attached to USB

Return Value

0 if successful, otherwise the return value is an error code.

Remarks

This function is used to return each device's serial number. The serial number is then used by **RP_Open** to obtain a handle for subsequent reading and writing of DIO.

Examples

Sample code shows how to get the number of devices, each serial number and each description.

```
unsigned long ulErrCode;
int iNumDevs;
char SN[256];
char Desc[256];

ulErrCode = RP_ListDIO( &iNumDevs, SN, Desc );
if (ulErrCode == 0)
{
    // Do something
}
else
{
    // Handle error
}
```

RP_OpenDIO

Open the device and return a handle which will be used for subsequent reading and writing of DIO.

Compatibility

RP Series : All

DIO Series : All

unsigned long **RP_OpenDIO** (char * *SN*, unsigned long **hDIO*)

Parameters

<i>SN</i>	The serial number for the device.
<i>hDIO</i>	Pointer to a variable of type long where the handle will be stored. This handle must be used to access the device.

Return Value

0 if successful, otherwise the return value is an error code.

Remarks

Example

This sample shows how to open a device.

```
unsigned long hDIO; // handle of an open device
unsigned long ulErrCode;
char SN[256];
char Desc[256];

ulErrCode = RP_ListDIO( &numDevs, SN, Desc );
if (( ulErrCode == 0 ) && ( numDevs == 1 ))
{
    ulErrCode = RP_OpenDIO( SN, & hDIO );
    if ( ulErrCode == 0 )
    {
        // Do something
    }
    else
    {
        // Handle error
    }
}
else
{
    // Handle error
}
```

RP_CloseDIO

Close an open device.

Compatibility

RP Series : All

DIO Series : All

unsigned long **RP_CloseDIO**(unsigned long *hDIO*)

Parameters

hDIO Handle of the device to close.

Return Value

0 if successful, otherwise the return value is an error code.

Example

This sample shows how to close a device.

```
unsigned long hDIO;
unsigned long ulErrCode;
char SN[256];
char Desc[256];

ulErrCode = RP_ListDIO( &numDevs, SN, Desc );
if ( ( ulErrCode == 0 ) && ( numDevs == 1 ) )
{
    ulErrCode = RP_OpenDIO( SN, & hDIO );
    if ( ulErrCode == 0 )
    {
        // Do something
        ulErrCode = RP_CloseDIO( hDIO );
    }
    else
    {
        // Handle error
    }
}
else
{
    // Handle error
}
```

RP_ReadPort

Read data from a particular input port.

Compatibility

RP Series : All

DIO Series : All

unsigned long *RP_ReadPort*(unsigned long *hDIO*, unsigned char *ucPort*, unsigned char * *ucPVal*, char * *ErrMsg*)

Parameters

<i>hDIO</i>	Handle of the device to read.
<i>ucPort</i>	The number of the port to be read.
<i>ucPVal</i>	Pointer to a variable of type unsigned char which receives the value of the port.
<i>ErrMsg</i>	String containing any error messages.

Return Value

0 if successful, otherwise the return value is an error code.

Remarks

The function does not return until the requested port has been read or read timeout occurs. The read timeout is set to 1 second.

The parameter *ucPort* represents either input port 0-3 or output port 0-3. A value of 0 will access input port 0, a value of 1 will access input port 1, a value of 2 will access input port 2, a value of 3 will access input port 3, a value of 16 (0x10) will access output port 0, a value of 17 (0x11) will access output port 1, a value of 18 (0x12) will access output port 2, a value of 19 (0x13) will access output port 3. Any other value will return an error.

The parameter *ucPVal* represents the value of the requested port. A value of 0 (00000000 binary) means all 8 bits are active. A value of 255 (11111111 binary) means all 8 bits are off.

Example

This sample shows how to read bit 6 of input port 0.

```
unsigned long hDIO; // handle of an open device
unsigned long ulErrCode;
unsigned char ucPort = 0;
unsigned char ucPVal, ucBit6;
char ErrMsg[256];

    ulErrCode = RP_ReadPort(hDIO, ucPort, &ucPVal, ErrMsg);
if (ulErrCode == 0)
{
    ucBit6 = ucPVal & 0x40; // 0100 0000
    // Do something
}
else
{
    // Handle error
}
```

RP_ReadAll

Read all 8 ports of data from the device.

Compatibility

RP Series : firmware version of 2.00 or higher.

DIO Series : All

unsigned long **RP_ReadAll**(unsigned long *hDIO*, unsigned long * *uILP*, unsigned long * *uIHP*, char * *ErrMsg*)

Parameters

<i>hDIO</i>	Handle of the device to read.
<i>uILP</i>	Pointer to a variable of type unsigned long which receives the value of four of the ports.
<i>uIHP</i>	Pointer to a variable of type unsigned long which receives the value of four of the ports.
<i>ErrMsg</i>	String containing any error messages.

Return Value

0 if successful, otherwise the return value is an error code.

Remarks

The function does not return until the requested port has been read or read timeout occurs. The read timeout is set to 1 second.

The parameter *uILP* stores the port data as follows :

Bits	RP0000xxxxN	RP000000xN
24 - 31	Input Port 0	Output Port 0
16 - 23	Input Port 1	Output Port 1
8 - 15	Input Port 2	Output Port 2
0 - 7	Input Port 3	Output Port 3

The parameter *uIHP* stores the port data as follows :

Bits	RP0000xxxxN	RP000000xN
24 - 31	Output Port 0	Output Port 4
16 - 23	Output Port 1	Output Port 5
8 - 15	Output Port 2	Output Port 6
0 - 7	Output Port 3	Output Port 7

For any port a value of 0 (00000000 binary) means all 8 bits are active. A value of 255 (11111111 binary) means all 8 bits are off.

Example

This sample shows how to read bit 6 of input port 0.

```
unsigned long hDIO; // handle of an open device
unsigned long ulErrCode;
unsigned long ucLPort;
unsigned long ucHPort;
unsigned char ucBit6;
char ErrMsg[256];

ulErrCode = RP_ReadAll(hDIO, &ucLPort, &ucHPort, ErrMsg);
if (ulErrCode == 0)
{
    ucBit6 = ((ucLPort & 0x40000000) > 0) ? 1 : 0;
    // Do something
}
else
{
    // Handle error
}
```


RP_WritePort

Set bits for an output port.

Compatibility

RP Series : All.

DIO Series : All

unsigned long **RP_WritePort** (unsigned long *hDIO*, unsigned char *ucPort*, unsigned char *ucPVal*, char * *ErrMsg*)

Parameters

<i>hDIO</i>	Handle of the device to write.
<i>ucPort</i>	The number of the output port to be written.
<i>ucPVal</i>	The value to write to the output port.
<i>ErrMsg</i>	String containing any error messages.

Return Value

0 if successful, otherwise the return value is an error code.

Remarks

The function does not return until the requested port has been read or read timeout occurs. The read timeout is set to 1 second.

The parameter *ucPort* represents either output port 0, 1, 2 or 3. A value of 16 (0x10) will access port 0, a value of 17 (0x11) will access port 1, a value of 18 (0x12) will access port 2 and a value of 19 (0x13) will access port 3. Any other value will return an error.

The parameter *ucPVal* represents the value that the port will be set to. A value of 0 (00000000 binary) means all 8 bits are on (the outputs are active). A value of 255 (11111111 binary) means all 8 bits are off.

Example

This sample shows how to set bit 3 of output port 0.

```
unsigned long hDIO; // handle of an open device
unsigned long ulErrCode;
unsigned char ucPort = 0x10; // Output port 0
unsigned char ucPVal;
char ErrMsg[256];

    ulErrCode = RP_ReadPort(hDIO, ucPort, &ucPVal, ErrMsg);
if (ulErrCode == 0)
{
    ucPVal &= 0xf7; // 1111 0111
    ulErrCode = RP_WritePort(hDIO, ucPort, ucPVal, ErrMsg);
    if (ulErrCode == 0)
    {
        ulErrCode = RP_ReadPort(hDIO, ucPort, &ucPVal, ErrMsg);
    }
    else
    {
        // Handle error
    }
}
else
{
    // Handle error
}
```

RP_WriteAll

Set bits for all of the output ports.

Compatibility

RP Series : firmware version of 2.00 or higher.

DIO Series : All

unsigned long **RP_WriteAll** (unsigned long *hDIO*, , unsigned long *uLLP*, unsigned long *uIHP*, char * *ErrMsg*)

Parameters

<i>hDIO</i>	Handle of the device to write.
<i>uLLP</i>	The value to write to four of the output ports.
<i>uIHP</i>	The value to write to four of the output ports.
<i>ErrMsg</i>	String containing any error messages.

Return Value

0 if successful, otherwise the return value is an error code.

Remarks

The function does not return until the requested port has been written or write timeout occurs. The write timeout is set to 1 second.

The parameter *uLLP* stores the port data as follows :

Bits	RP0000xxxxN	RP000000xN
24 - 31	Output Port 0	Output Port 0
16 - 23	Output Port 1	Output Port 1
8 - 15	Output Port 2	Output Port 2
0 - 7	Output Port 3	Output Port 3

The parameter *uIHP* stores the port data as follows :

Bits	RP0000xxxxN	RP000000xN
24 - 31	Not Used	Output Port 4
16 - 23	Not Used	Output Port 5
8 - 15	Not Used	Output Port 6
0 - 7	Not Used	Output Port 7

For any port a value of 0 (00000000 binary) means all 8 bits are on (the outputs are active). A value of 255 (11111111 binary) means all 8 bits are off.

Example

This sample shows how to set bit 3 of output port 0.

```
unsigned long hDIO; // handle of an open device
unsigned long ulErrCode;
unsigned long ucLPort;
unsigned long ucHPort;
char ErrMsg[256];

ulErrCode = RP_ReadAll(hDIO, &ucLPort, &ucHPort, ErrMsg);
if (ulErrCode == 0)
{
    ucLPort &= 0xf7ffffff;
    ulErrCode = RP_WriteAll(hDIO, ucLPort, ucHPort, ErrMsg);
    if (ulErrCode == 0)
    {
        ulErrCode = RP_ReadAll(hDIO, &ucLPort, &ucHPort, ErrMsg);
    }
    else
    {
        // Handle error
    }
}
else
{
    // Handle error
}
```

RP_SetWDT_Outputs

Sets a timeout for the outputs.

Compatibility

RP Series : firmware version of 2.00 or higher.

DIO Series : All

unsigned long **RP_SetWDT_Outputs**(unsigned long hDIO, unsigned long ulTime, char *ErrMsg);

Parameters

<i>hDIO</i>	Handle of the device to write.
<i>ulTime</i>	Time out value in seconds. 0 disables the timer.
<i>ErrMsg</i>	String containing any error messages.

Return Value

0 if successful, otherwise the return value is an error code.

Remarks

The function does not return until the requested port has been written or write timeout occurs. The write timeout is set to 1 second.

A timer is loaded with the timeout value. The timer is reset every time a USB message is received. If the timer times out before the next USB message all outputs are turned off.

Example

This sample shows how to set the timeout to 10 seconds

```
unsigned long hDIO; // handle of an open device
unsigned long ulErrCode;
unsigned long ulTimeOut;
char ErrMsg[256];

ulTimeOut = 0x0000a;
ulErrCode = RP_SetWDT_Outputs(hDIO, ulTimeOut, ErrMsg);
if (ulErrCode == 0)
{
    // continue
}
else
{
    // Handle error
}
else
{
    // Handle error
}
```

RP_GetVer

Read software version from the device.

Compatibility

RP Series : All.

DIO Series : All

unsigned long **RP_GetVer** (unsigned long *hDIO*, char **SWVer*, char **ErrMsg*)

Parameters

<i>hDIO</i>	Handle of the device to write.
<i>SWVer</i>	String containing the software version of the device.
<i>ErrMsg</i>	String containing any error messages.

Return Value

0 if successful, otherwise the return value is an error code.

Remarks

The function does not return until the requested information has been returned or read timeout occurs. The read timeout is set to 1 second.

Examples

This sample shows how to read the software version.

```
unsigned long hDIO; // handle of an open device
unsigned long ulErrCode;
char ErrMsg[256];

ulErrCode = RP_GetVer( hDIO, SWVer, ErrMsg );
if (ulErrCode == 0)
{
    // Do something
}
else
{
    // Handle error
}
```