Winmate® API Specifications

Project Name: PPC Digital I/O SDK

Revision: 2.0
March 31, 2014

Contents

Revision History

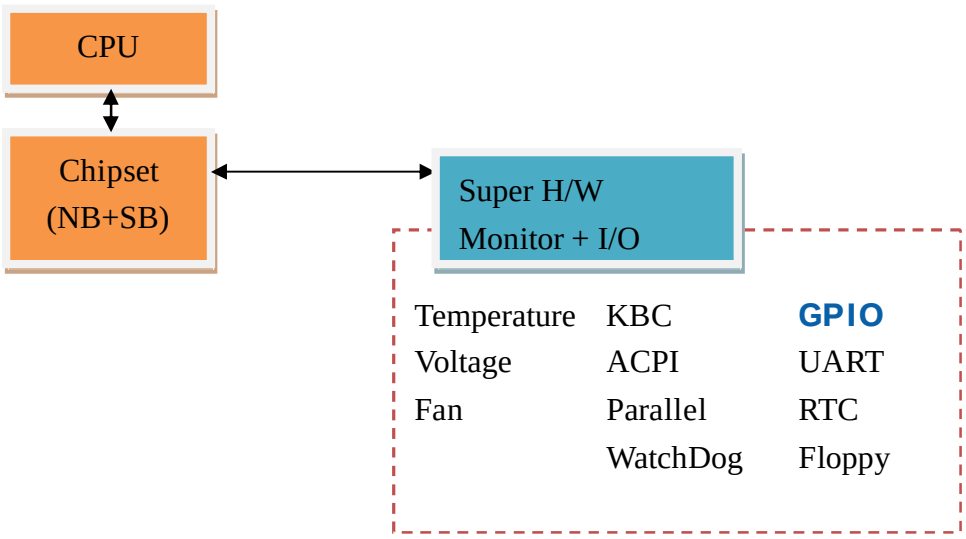
Revision	Date	Description
2.0	2014/03/31	1. Initial draft.

1. General Description

1.1 Introduction

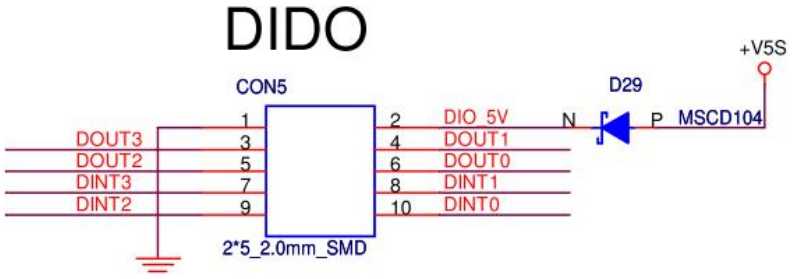
The document describes how to program the Digital I/O. There are 8 or 12 programmable digital I/O pins, each pin can be set for output data control or get input status.

1.2 Block Diagram



1.3 I/O Pin Define

GPIO define 8 pins:



GPIO define 12 pins:

Pin Definition:

Pin of DIO	Pin name of DIO	BIOS Initial Value (S5 -> S0) User define
1	Ground	
2	VCC	
3	DOUT3	PIN3
4	DOUT1	PIN4
5	DOUT2	PIN5
6	DOUT0	PIN6
7	DINT3	PIN7
8	DINT1	PIN8
9	DINT2	PIN9
10	DINT0	PIN10
11	GPIO53_IN0	PIN11
12	GPIO56_OUT0	PIN12
13	GPIO54_IN1	PIN13
14	GPIO57_OUT1	PIN14

※Pin 11 ~ 14 only for GPIO define 12 pins use.

I/O Address:

BIT	Read/Write	Binary Address	Hex Address	PIN
0	Read/Write	0000 0000 0001	0x001	PIN3
1	Read/Write	0000 0000 0010	0x002	PIN4
2	Read/Write	0000 0000 0100	0x004	PIN5

3	**Read/Write**	**0000 0000 1000**	**0x008**	**PIN6**
4	**Read/Write**	**0000 0001 0000**	**0x010**	**PIN7**
5	**Read/Write**	**0000 0010 0000**	**0x020**	**PIN8**
6	**Read/Write**	**0000 0100 0000**	**0x040**	**PIN9**
7	**Read/Write**	**0000 1000 0000**	**0x080**	**PIN10**
8	**Read/Write**	**0001 0000 0000**	**0x100**	**PIN11**
9	**Read/Write**	**0010 0000 0000**	**0x200**	**PIN12**
10	**Read/Write**	**0100 0000 0000**	**0x400**	**PIN13**
11	**Read/Write**	**1000 0000 0000**	**0x800**	**PIN14**

※Pin 8 ~ 11 only for GPIO define 12 pins use.

Direction:

Bit	Read/Write	Description
11 ~ 0	**Read/Write**	GPIO I/O Value 0: The respective GPIO PIN is programmed as an Output (Write) port 1: The respective GPIO PIN is programmed as an Input (Read) port

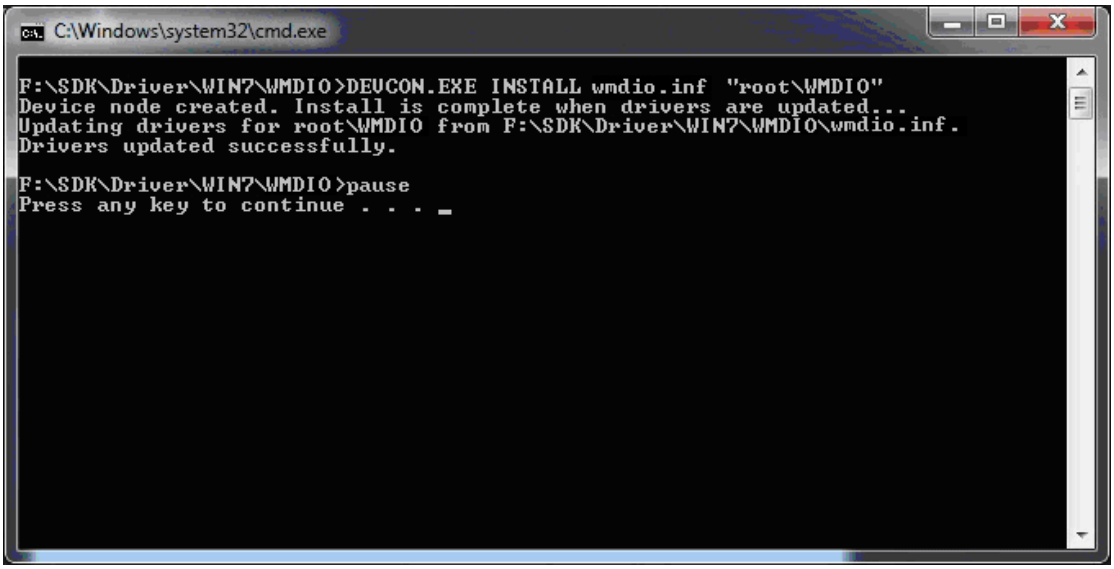
※Bit 8 ~ 11 only for GPIO define 12 pins use.

2.Driver

2.1 Install WMDIO Driver

The drivers are for Windows 7/8. Please double click install.bat in driver directories to start install.

When the driver is successfully installed, you can see "Drivers Updated successfully" message in the dos prompt.



And "WMDIO" devices are also added in the Device Manger under "System devices"

Warning: It is important that only one WMDIO devices can appear in Device Manager.

3.Programming environment

3.1 Project Setting

To control the device power state, brightness, led flash control and obtain SMBIOS information, dynamic file (dll), library (lib) and header (h) files are provided to develop the Application.

WMDIODLL.lib	**Library**
WMDIODLL.dll	**Dynamic Library**
WMDIODLL.h	**Header File**

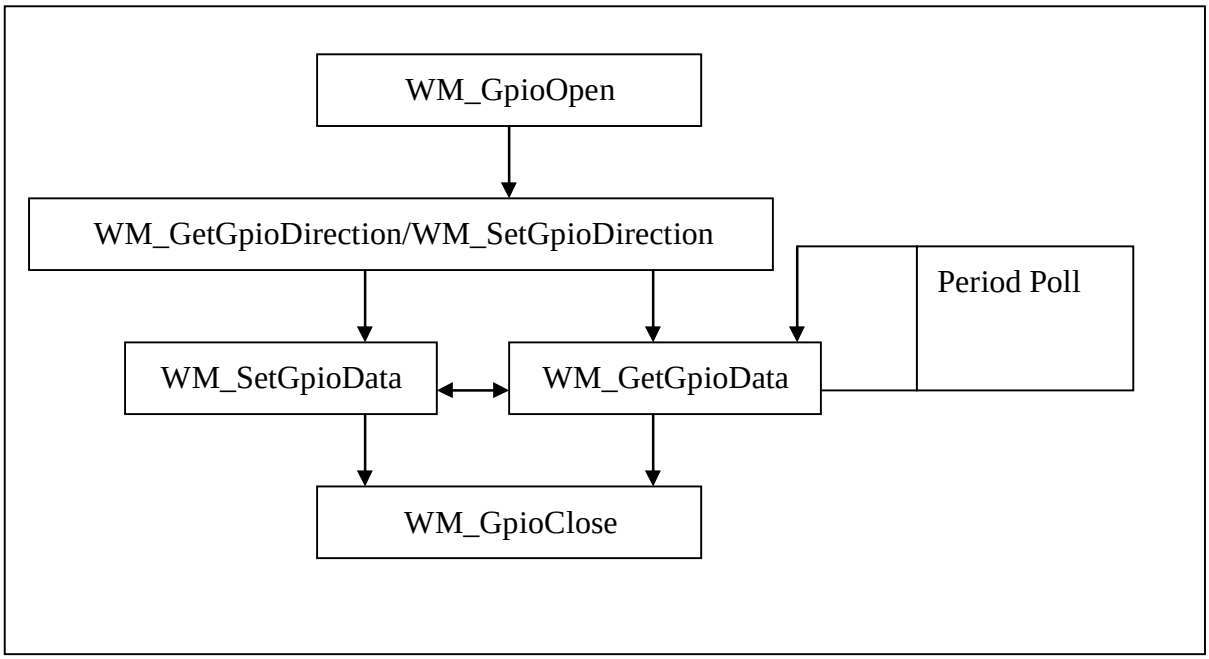
1. Include “WMDIODLL.h” in the project.
2. Add “WMDIODLL.lib” into project Link.
3. Put “WMDIODLL.dll” in the same path with application or into“windows “directory.
4. Check “WMDIODLL.dll” version is 2.0.0.0

3.2 WMDIODLL.h File Reference:

```
#ifdef WMDIODLL_EXPORTS
#define WMDIO_API extern "C"    __declspec(dllexport)
#else
#define WMDIO_API extern "C"    __declspec(dllimport)
#endif

WMDIO_API int WM_GpioOpen(void);
WMDIO_API int WM_GpioClose(void);
WMDIO_API int WM_SetGpioDirection(UINT16 uiData);//1:input 0:output
WMDIO_API int WM_GetGpioDirection(PUINT16 puiData);
WMDIO_API int WM_SetGpioData(UINT16 uiData);
WMDIO_API int WM_GetGpioData(PUINT16 puiData);
```

3.3 GPIO Function Block



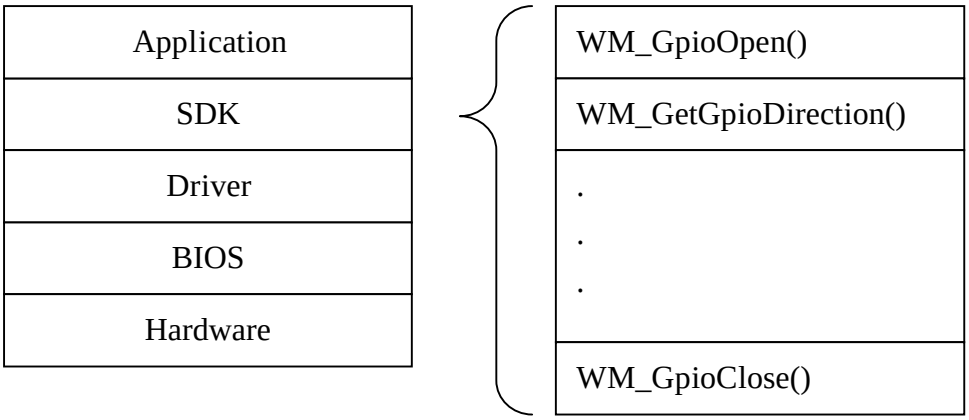
GPIO does not support interrupt.

To get the status of GPIO, polling is necessary.

4.API Definition

4.1 Function Procedure:

4.1.1 Function Block



4.2 Function Name:

4.2.1 WM_GpioOpen(void);

The WM_GpioOpen() Function open the device.

```
WMDIO_API int WM_GpioOpen(void);
```

Parameters

None

Return Value

The function returns TRUE if it is successful open the device, and FALSE otherwise.

Requirements

Header: Declared in WMDIODLL.h

4.2.2 WM_GpioClose(void);

The WM_GpioClose() Function Close the Device.

```
WMDIO_API int WM_GpioClose(void);
```

Parameters

None

Return Value

The function returns TRUE if it is successful close the device, and FALSE otherwise.

Requirements

Header: Declared in WMDIODLL.h

4.2.3 WM_SetGpioDirection(UINT16 uiData);

The WM_SetGpioDirection() Function set the direction of each GPIO pins.

```
WMDIO_API int WM_SetGpioDirection(UINT16 uiData);
```

Parameters

UINT16 uiData

Bit 0 – 11:

0: The respective GPIO PIN is programmed as an Output (Write) port

1: The respective GPIO PIN is programmed as an Input (Read) port

Example: 0xFF3

Input: PIN14,PIN13,PIN12,PIN11,PIN10,PIN9,PIN8,PIN7,PIN4,PIN3

Output: PIN6,PIN5

Pin	14	13	12	11	10	9	8	7	6	5	4	3
Bit	11	10	9	8	7	6	5	4	3	2	1	0
I/O	I	I	I	I	I	I	I	I	O	O	I	I
uiData	F				F				3			

WM_SetGpioDirection(0xFF3);

Return Value

The function returns TRUE if it is successful set the direction of device

Requirements

Header: Declared in WMDIODLL.h

4.2.4 WM_GetGpioDirection(PUINT16 puiData);

The WM_GetGpioDirection() Function get the direction of each GPIO pins.

```
WMDIO_API int WM_GetGpioDirection(PUINT16 puiData);
```

Parameters

OUT PUINT16 puiData:

Result :

True : It is successful get the data.

False: otherwise.

Return Value

Direction of gpio pins

0: The respective GPIO PIN is programmed as an Output (Write) port

1: The respective GPIO PIN is programmed as an Input (Read) port

Example: 0xF11

Input: PIN14,PIN13,PIN12,PIN11,PIN7,PIN3

Output: PIN10,PIN9,PIN8,PIN6,PIN5,PIN4

Pin	14	13	12	11	10	9	8	7	6	5	4	3
Bit	11	10	9	8	7	6	5	4	3	2	1	0
I/O	I	I	I	I	O	O	O	I	O	O	O	I
uiData	F				1				1			

Requirements

Header: Declared in WMDIODLL.h

4.2.5 WM_SetGpioData(UINT16 uiData);

The WM_SetGpioData() Function set the High/Low of each GPIO pins.

```
WMDIO_API int WM_SetGpioData(UINT16 uiData);
```

Parameters

UINT16 uiData

Bit 0 – 11:

0: The respective GPIO PIN is programmed as Low (0V).

1: The respective GPIO PIN is programmed as High (5V)

Example: 0x011

Set PIN3(Bit 0), PIN7(Bit 4) to High, others are Low.

Pin	14	13	12	11	10	9	8	7	6	5	4	3
Bit	11	10	9	8	7	6	5	4	3	2	1	0
H/L	L	L	L	L	L	L	L	H	L	L	L	H
uiData	0				1				1			

WM_SetGpioData(0x011);

Return Value

The function returns TRUE if it is successful set the data of GPIO, and FALSE otherwise.

Requirements

Header: Declared in WMDIODLL.h

4.2.6 WM_GetGpioData(PUINT16 puiData);

The WM_GetGpioData() Function get the High/Low of each GPIO pins.

```
WMDIO_API int WM_GetGpioData(PUINT16 puiData);
```

Parameters

OUT PUINT16 puiData:

Result :

True : It is successful get the data.

False: otherwise.

Return Value

The function returns the data of each GPIO Pins.

0: The respective GPIO PIN is programmed as Low (0V).

1: The respective GPIO PIN is programmed as High (5V)

Example: 0xFFC

Get PIN4(Bit 1),PIN3(Bit 0) to Low, others are High.

Pin	14	13	12	11	10	9	8	7	6	5	4	3
Bit	11	10	9	8	7	6	5	4	3	2	1	0
H/L	H	H	H	H	H	H	H	H	H	H	L	L
uiData	F				F				C			

Requirements

Header: Declared in WMDIODLL.h

4.3 Program Flow:

1. Open : WM_GpioOpen()
2. Set DIO direction : WM_SetGpioDirection()
3. Operation : WM_GetGpioData() / WM_SetGpioData()
4. Close: WM_GpioClose()