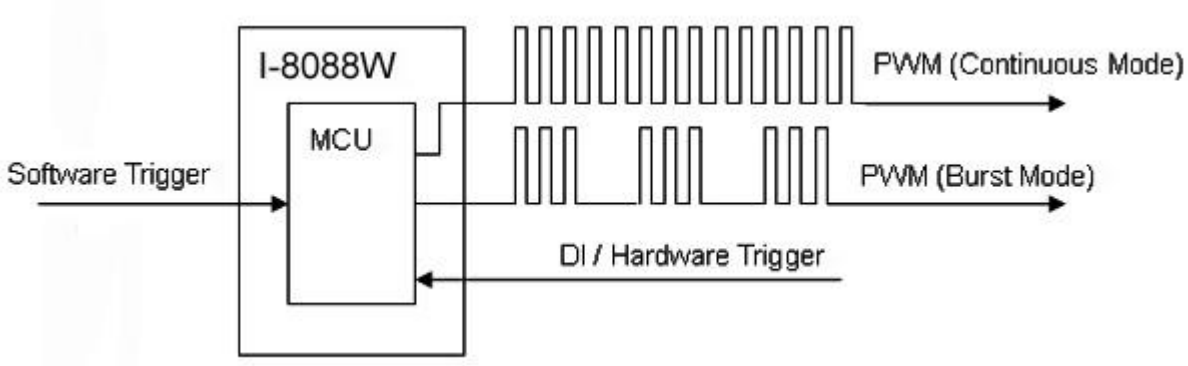
I-8088W
API Reference Manual

Version 1.0.0, Aug. 2010

Service and usage information for
MiniOS7, CE and XPE Platform



Written by Martin Hsu
Edited by Janice Hong

General Information

Warranty

All products manufactured by ICP DAS are under warranty regarding defective materials for a period of one year, beginning from the date of delivery to the original purchaser.

Warning

ICP DAS assumes no liability for any damage resulting from the use of this product.ICP DAS reserves the right to change this manual at any time without notice.The information furnished by ICP DAS is believed to be accurate and reliable.However, no responsibility is assumed by ICP DAS for its use, not for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

The names used for identification only may be registered trademarks of their respective companies.

TABLE OF CONTENTS

1. INTRODUCTION

PWM (Pulse width modulation) is a powerful technique for controlling analog circuits. It uses digital outputs to generate a waveform with variant duty cycle and frequency to control analog circuits. I-8088W has 8 PWM output channels and 8 digital inputs. It can be used to develop powerful and cost effective analog control system.

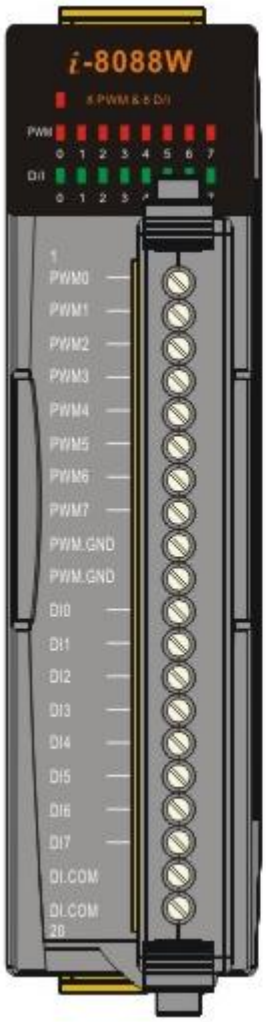
Features:

- ✓ Automatic generation of PWM outputs by hardware, without software intervention.
- ✓ 10 Hz ~ 500 kHz (non-continuous) PWM output frequency with 0.1% ~ 99.9% duty cycle
- ✓ Software and hardware trigger mode for PWM output
- ✓ Individual and synchronous PWM output
  Using software trigger mode, you can set configuration for all PWM channels then trigger them one by one or all of them at the same time.
- ✓ Burst mode PWM operation for standby
- ✓ DI channel can be configured as simple digital input channel or hardware trigger source of the PWM output.

1.1. Specification

PWM Output	
Channels	8-ch
Scaling Resolution	16-bit (1 ~128 µs for each step)
Frequency Range	10 Hz ~ 500 kHz (non-continuous)
Duty Cycle	0.1% ~ 99.9%
PWM Mode	Burst Counting, Continuous mode
Burst Counter	1 ~ 65535
Hardware Trigger Mode	Trigger Start & Trigger Stop
Output Type	Source
Max Load Current	1 mA
Intra-module Isolation, Field to Logic	3750 Vrms
ESD Protection	4 kV Contact for each channel
Digital Input	
Input Channels	8 (Sink/Source)
Input Type	One Common for All Digital Input
On Voltage Level	+5 ~ +30 V
Off Voltage Level	< 0.8V
Input Impedance	4.7 kΩ, 1/4 W
Intra-module Isolation, Field to Logic	3750 Vrms
ESD Protection	4 kV Contact for each channel
LED Display	
1 LED as Power Indicator /16 LED as PWM and Digital Input Indicator	
Power	
Power Consumption	40 mA @ 5 V, 2 W ±5%
Environment	
Operating Temperature	-25 ~ 75°C
Storage Temperature	-30 ~ 85°C
Humidity	5 to 95% RH, Non-condensing
Dimensions	
30mm x 102mm x 115mm (W x L x H) Detail	

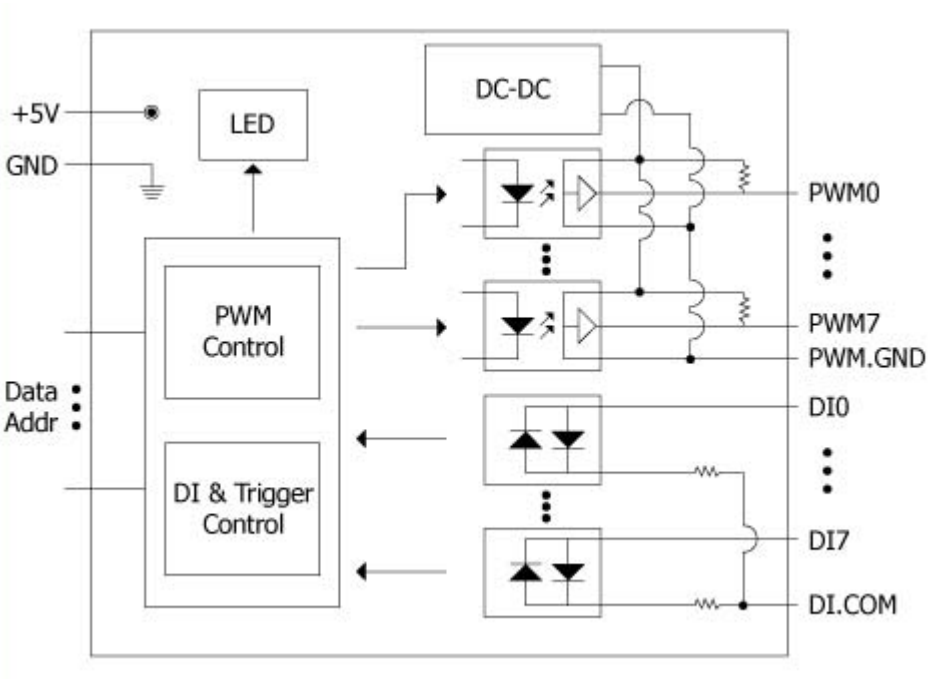
1.2. Pin Assignment



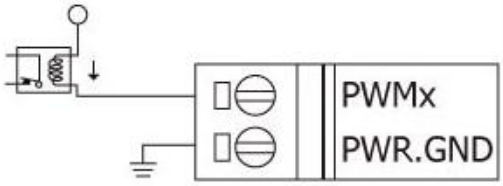
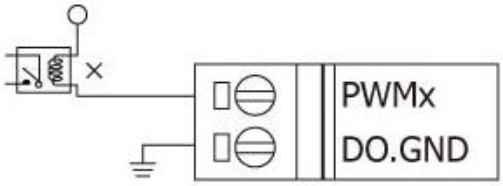
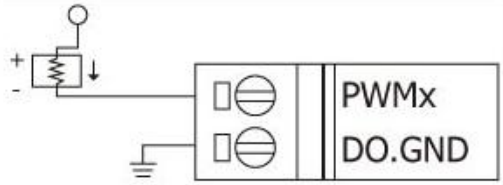
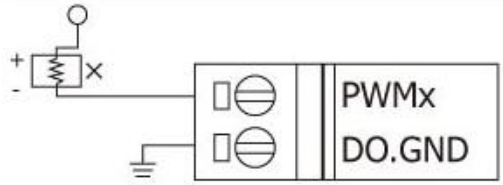
Terminal No.	Pin Assignment
01	PWM0
02	PWM1
03	PWM2
04	PWM3
05	PWM4
06	PWM5
07	PWM6
08	PWM7
09	PWM.GND
10	PWM.GND
11	DI0
12	DI1
13	DI2
14	DI3
15	DI4
16	DI5
17	DI6
18	DI7
19	DI.COM
20	DI.COM

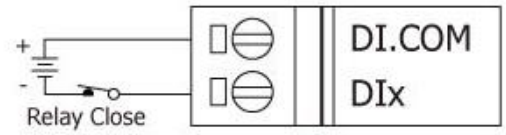
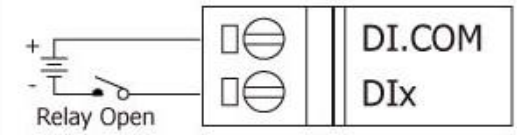
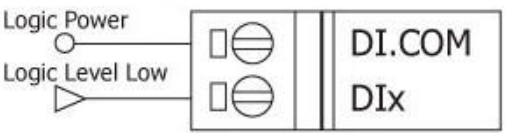
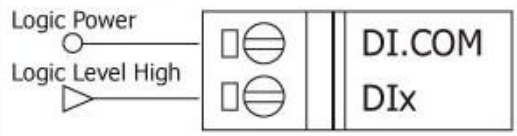
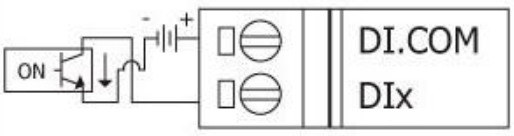
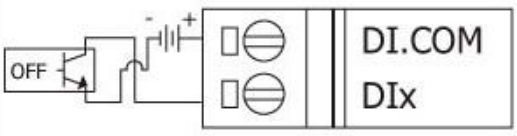
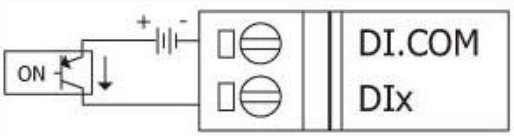
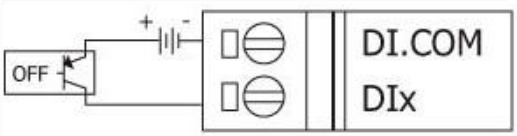
- ✓ Pin 1 ~ 8: PWM0 ~ PWM7, are designed for PWM output
- ✓ Pin 9 ~ 10: PWM.GND is isolated ground.
- ✓ Pin 11 ~ 18: DI0 ~ DI7 are designed for digital input that also capable of setting as an external trigger signal to start or stop its PWM pulse.
- ✓ Pin 19 ~ 20: DI.COM is isolated ground.

1.3. Block Diagram



1.4. Wiring Connection

Output Type	ON State LED ON Readback as 1	OFF State LED OFF Readback as 0
Drive Relay	Relay ON	Relay Off
		
Resistance Load		
		

Input Type	ON State LED ON Readback as 0	OFF State LED OFF Readback as 1
Relay Contact	Relay ON	Relay Off
		
TTL/CMOS Logic	Voltage > 5V	Voltage < 0.8V
		
NPN Output	Open Collector On	Open Collector Off
		
PNP Output	Open Collector On	Open Collector Off
		

2. Software and GETTING STARTED

2.1. Software

In this section, we will introduce you one simple program (8080demo.exe) which have three setup modes – Normal, Hardware Trigger and Synchronize. We need to check the following steps before running the program.

1. First, check the power cable, the communication cable between controller and PC has been connected well, and then check the i-80088W has been plugged in the controller. Refer to “ipac_8000_user_manual” - section 2.1:
   http://ftp.icpdas.com/pub/cd/8000cd/napdos/ipac8000/document/

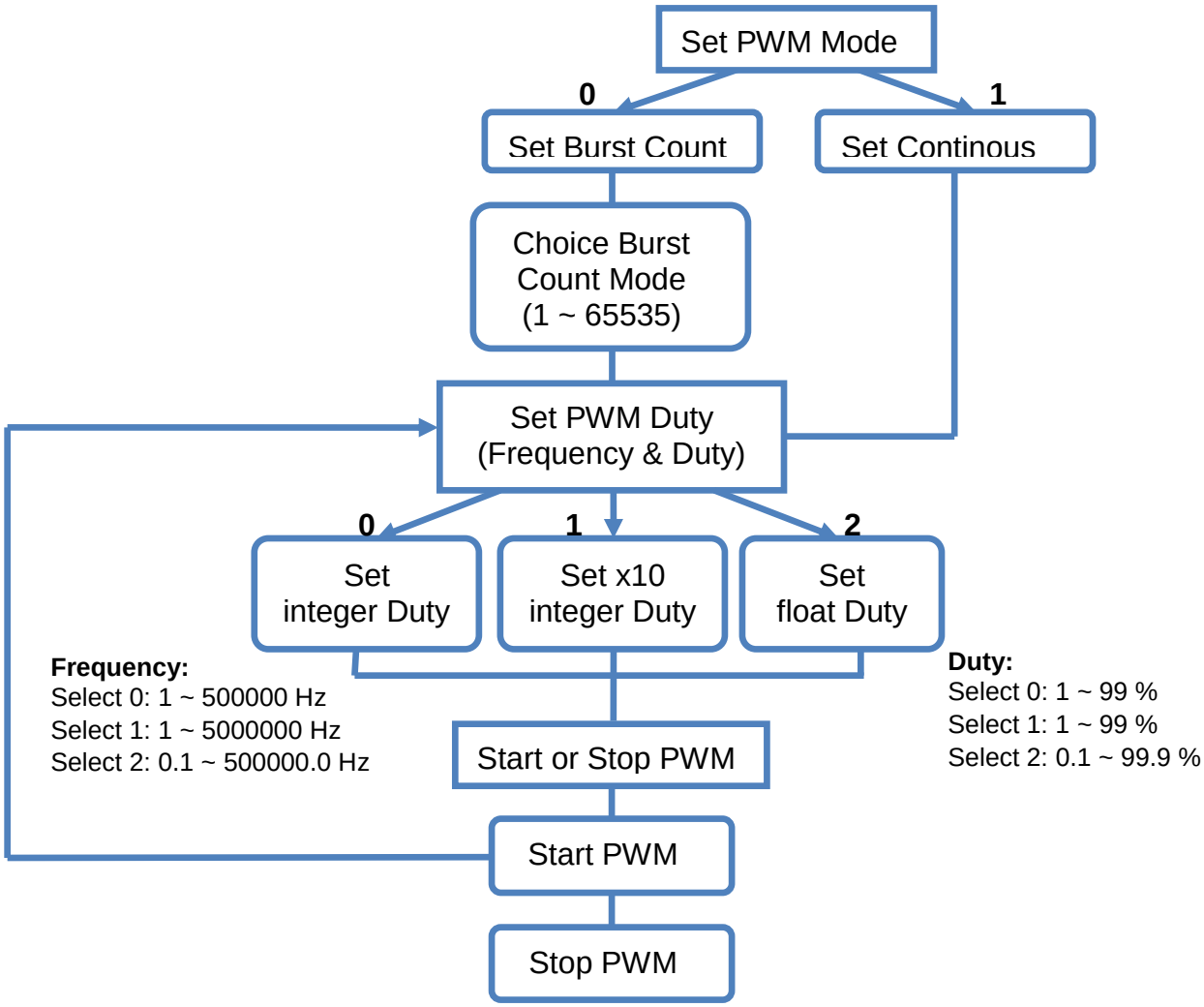
2. Next, check the communication between controller and PC is fine, and the demo program files have been downloaded to the controller. Refer to “ipac_8000_user_manual” - section 2.2 ~ 2.4:
   http://ftp.icpdas.com/pub/cd/8000cd/napdos/ipac8000/document/

You can find the related files in the product CD or below website:

WinPAC:
Document: ftp://ftp.icpdas.com/pub/cd/winpac/napdos/wp-8x4x_ce50/document/sdk_document
eVC SDK: ftp://ftp.icpdas.com/pub/cd/winpac/napdos/wp-8x4x_ce50/sdk/io_modules/evc/
.Net SDK: ftp://ftp.icpdas.com/pub/cd/winpac/napdos/wp-8x4x_ce50/sdk/io_modules/dotnet/
eVC Demo: ftp://ftp.icpdas.com/pub/cd/winpac/napdos/wp-8x4x_ce50/demo/winpac/evc/pac_io/local/
.Net Demo: ftp://ftp.icpdas.com/pub/cd/winpac/napdos/wp-8x4x_ce50/demo/winpac/dotnet/c%23.net/pac_io/local/
iPAC-8000
Document: ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/minios7/document/
Libray: ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/ipac8000/demo/basic/lib/
Demo: ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/ipac8000/demo/basic/io_in_slot/8088w/
I-8000
Document: ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/minios7/document/
Library: ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/8000/841x881x/demo/lib/
Demo: ftp://ftp.icpdas.com/pub/cd/8000cd/napdos/8000/841x881x/demo/io_in_slot/8088w/

2.2. Simple PWM operation

2.2.1. Flow Chart



Note: Each time you change the settings of “PWM Duty", you have to re-send the “Start PWM” command to ensure the operation properly.

2.2.2. How to Setup the Standard PWM

Please make sure you have completed the steps in section 2.1 before operating the following steps.

Description of the demo:

In this example, we will use the demo to set I-8088W as “Continuous” mode and its frequency is 10 Hz, PWM duty is 50%. When we send the “Start Normal PWM” commend, the DI0 will blinking per 0.5s.

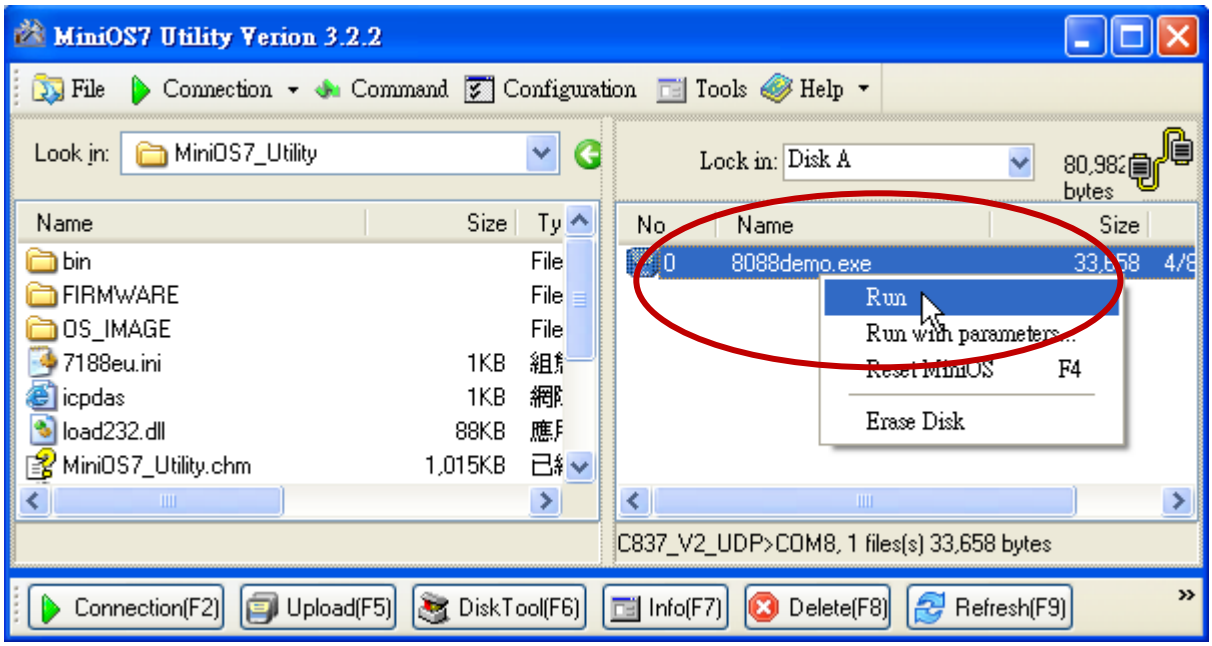
Wire connection of I-8088W:

To do this, you need to wire PW0 to DI0, DI.COM to External 5V, PWM.GND to External GND. (Please refer to section 1.2 Pin Assignment)

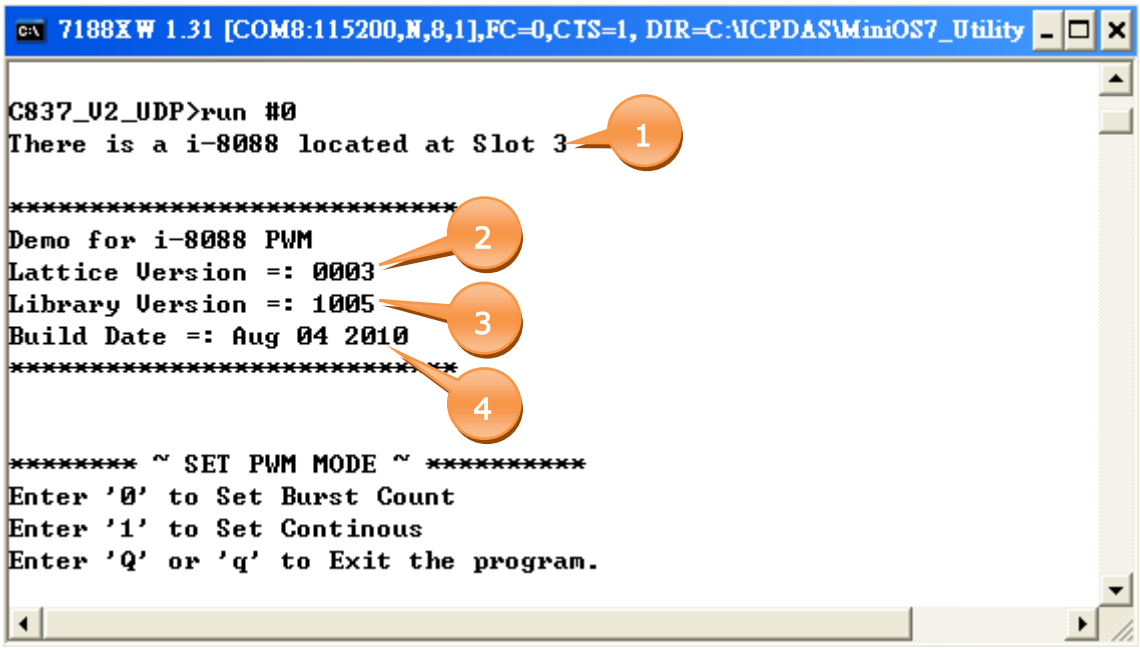
Now, follow the steps to configure the related parameter:

Step1: Run “8080demo.exe”

Right click on the “8088demo.exe” and click “Run”. You can also double-click on the file name to run.



When you run the program, it will initialize the i-8088W module and obtain the related information as shown below.



I In below table, we will list some instructions associated with above information.

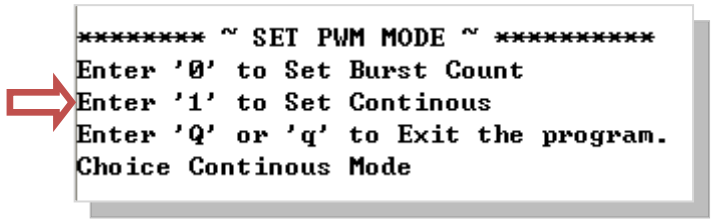
You can also refer to chapter 3 (for miniOS7 PAC) or chapter 4 (for WinCE PAC) to learn how to use these instrctions.

NO.	Info.	API
1	Initialize i-8088W	(Refer to section 3.1 i8088W_Init) or section 4.1 pac_i8088W_Init)
2	Lattice Version	The firmware version of I-8088W. (Refer to section 3.2 or 4.2)
3	Library Version	The version of library file. (Refer to section 3.3 or 4.3)
4	Build Date	The built date of library file. (Refer to section 3.4 or 4.4)

Step2: Set PWM Mode

This setting includes two modes – “Burst Count Mode” and “Continous Mode”. “Burst Count Mode” means it can output mutiple fixed pluse in a period time and then stop output. “Continous Mode” means it can output one fixed pluse in a period time and continue output.

In this example, we enter "1" to set it as "Continous mode".



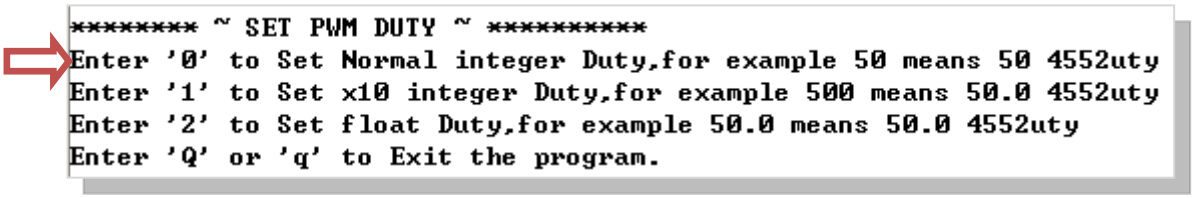
API : Refer to section 3.9 or section 4.9

Step3: Set PWM Duty

There are three options in this setting, if you choose:

- "0" means you can enter an integer value (ex. input 50 to set it as 50%)
- “1” means you can enter an integer value to set it as one decimal place value. (ex. input 999 to set it as 99.9%)
- “2” means you can enter a one decimal place value. (ex. input 99.9 to set it as 99.9%)

In this example, please enter "0" to set it as “Normal integer Duty”



Then, we will set its frequency as 10 Hz, PWM duty as 50%.

```
******** ~ SET PWM DUTY ~ **********
Enter '0' to Set Normal integer Duty,for example 50 means 50 4868uty
Enter '1' to Set x10 integer Duty,for example 500 means 50.0 4868uty
Enter '2' to Set float Duty,for example 50.0 means 50.0 4868uty
Enter 'Q' or 'q' to Exit the program.
Choice Set Normal integer Duty
First Input Hz 1~500000
10
Then Input duty 1~99
50
The PWM will be 10 Hz with 50 % Duty
```

API : Refer to section 3.5, section 3.6, section 3.7 or
section 4.5, section 4.6, section 4.7

Step4: Start PWM

You will see three modes in below picture, please enter “0” to set it as “Normal PWM”

```
******** ~ START OR STOP PWM ~ **********
Enter '0' to Set Normal PWM
Enter '1' to Set Hardware Trigger PWM
Enter '2' to Set Synchronous PWM
Enter '3' to Set PWM Mode
Enter 'Q' or 'q' to Exit the program.
```

Then enter “0” to start the PWM.

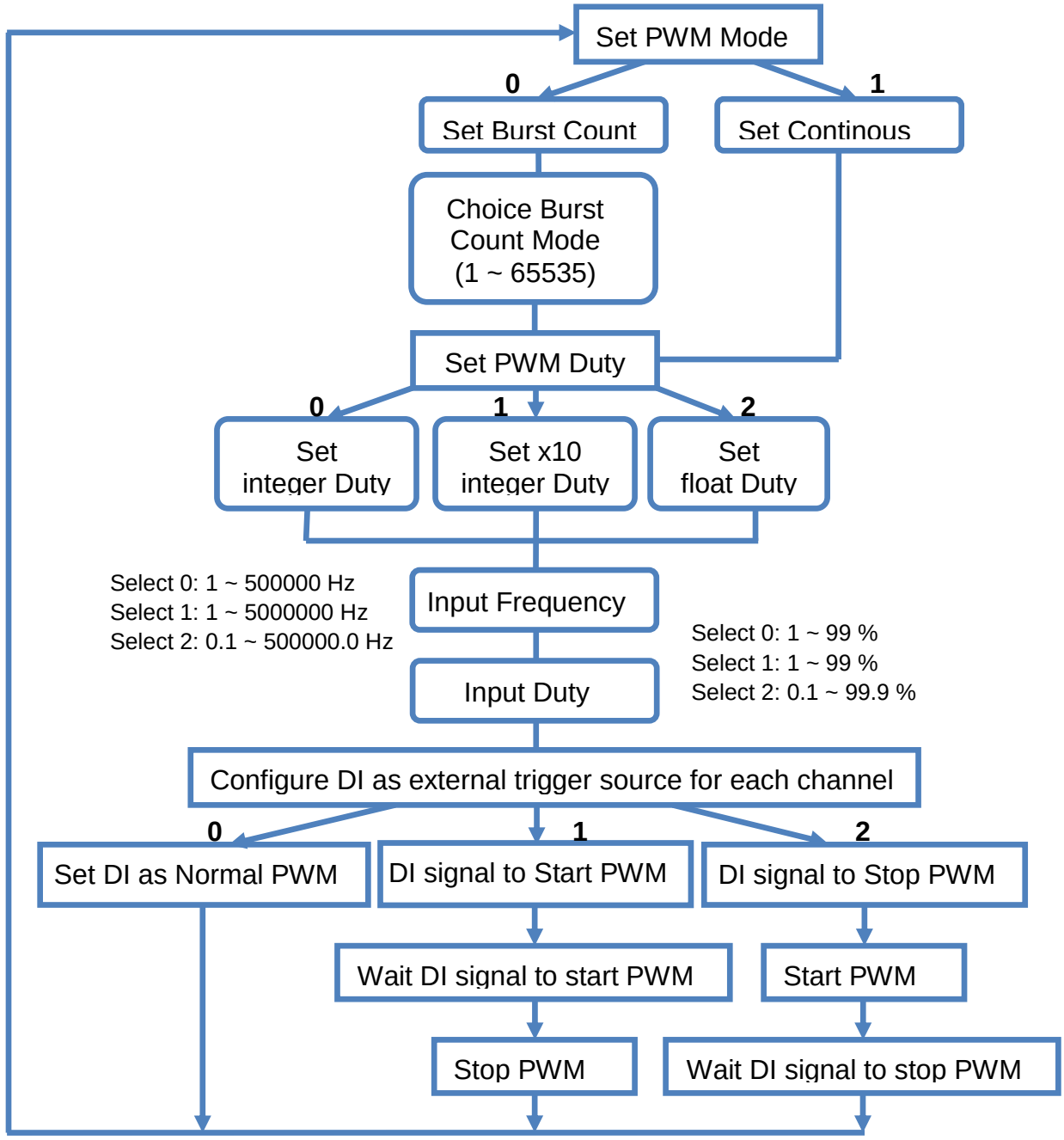
```
******** ~ START OR STOP Normal PWM ~ **********
Enter '0' to Stop Normal PWM
Enter '1' to Start Normal PWM
```

API : Refer to section 3.11, section 3.12,
section 4.11, section 4.12

If you have completed the correct setup, you will see below picture. In this example, it will send the “start PWM“ command to channel 0 ~ 7, the condition is 10 Hz with 50% duty, and we has connected the PWM0 to DI0, so the DI0 will blink per 0.5 seconds.

2.3. Use DI to trigger PWM

2.3.1. Flow Chart

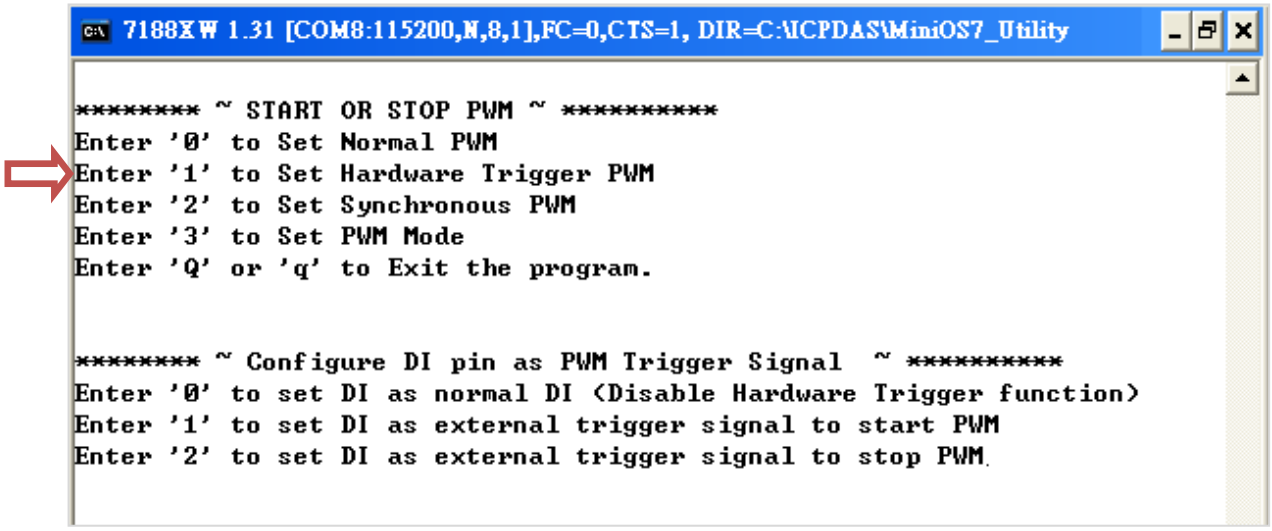


2.3.2. How to Setup the Trigger PWM

There are 8 DI pins on I-8088W, normally these DI pin just acts as digital input channels. We can also configure them as external trigger signal pins to start or stop the PWM output.

Step 1: Follow the same way in section 2.2.2 to configure PWM output mode and set PWM duty and frequency.

Step 2: Enter “1” to set “Hardware Trigger PWM”



There are three options to set the DI channels as PWM trigger signal:

- “0” : Normal DI.
  If you set it as "Normal DI" mode, it will be unrelated to the PWM function.

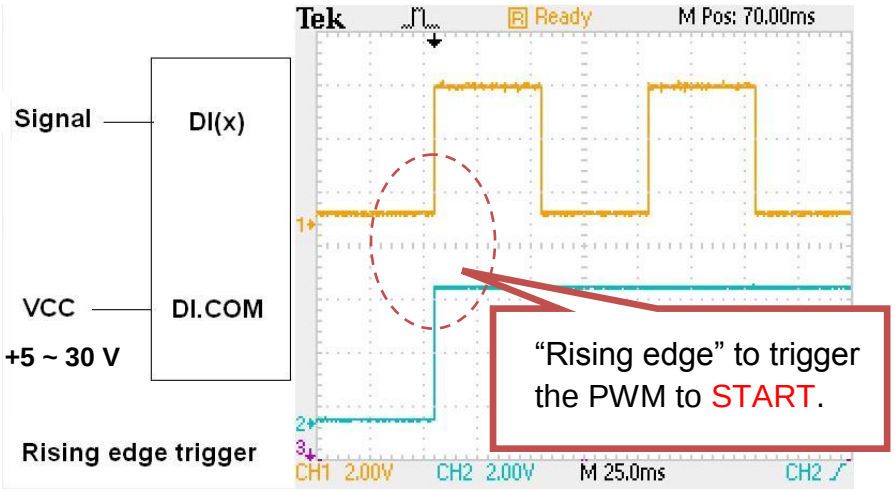
- “1”: Accept DI signal to start the PWM output.

- “2”: Accept DI signal to stop the PWM output.

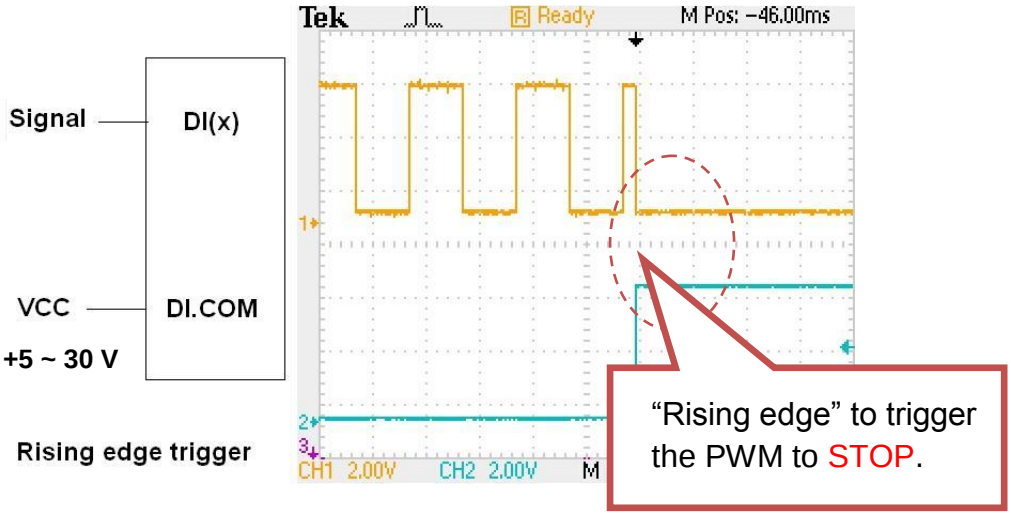
You can use i8088W_SetHardwareTrigChannel (section 3.17) or pac_i8088W_SetHardwareTrigChannel (section 4.17) function to configure each DI pin.

There are two ways to configure the DI signal to start or stop the PWM output. One is rising edge to trigger the PWM to start or stop, the wiring like below two pictures.

Rising edge to Start PWM

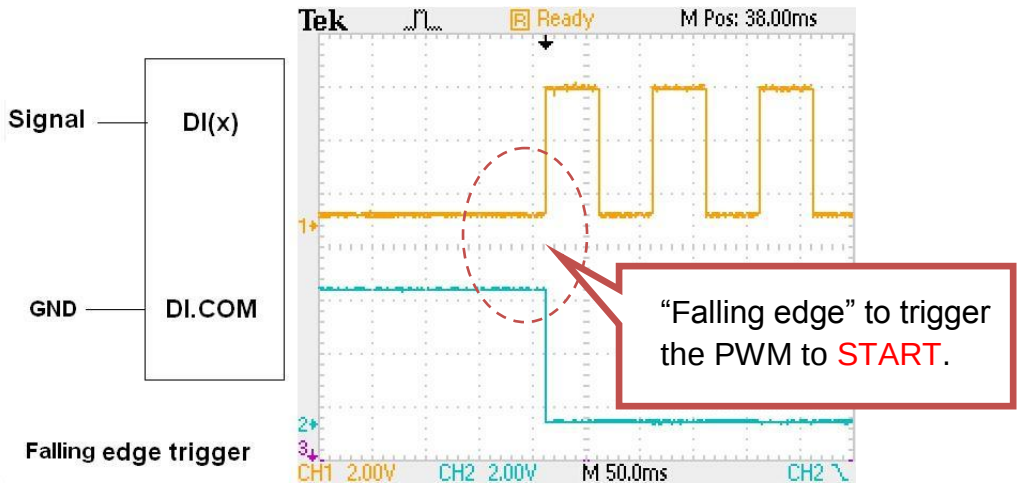


Rising edge to Stop PWM

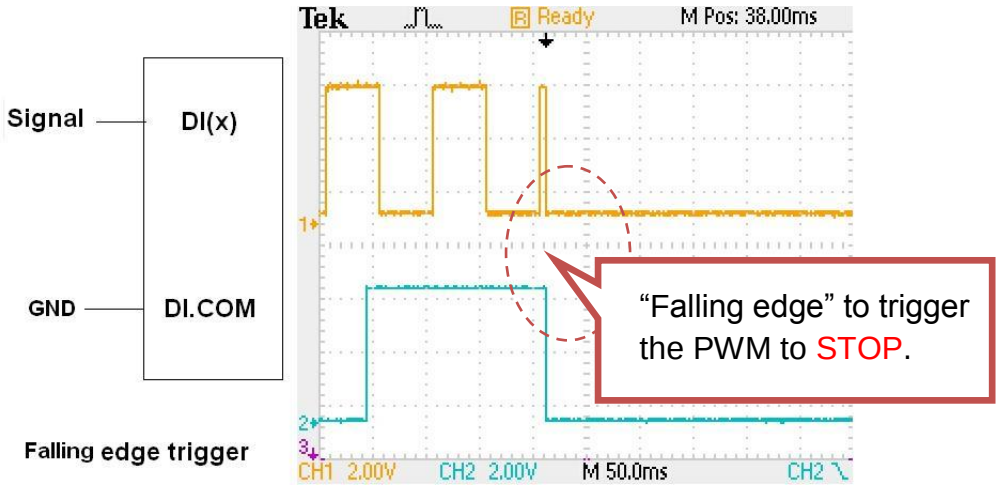


The other is falling edge to trigger the PWM to start or stop, the wiring like below two pictures.

Falling edge to Start PWM

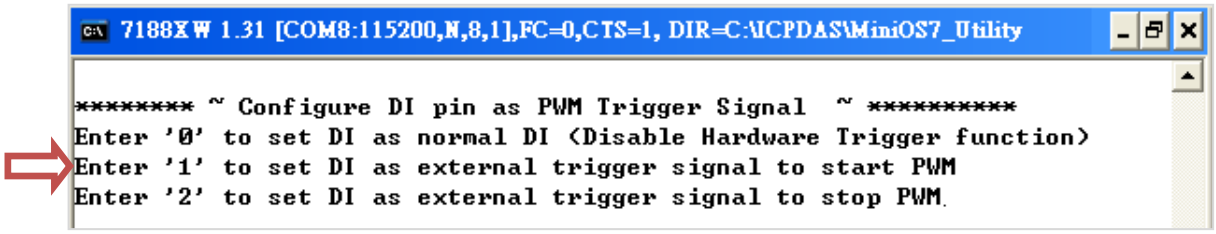


Falling edge to Stop PWM



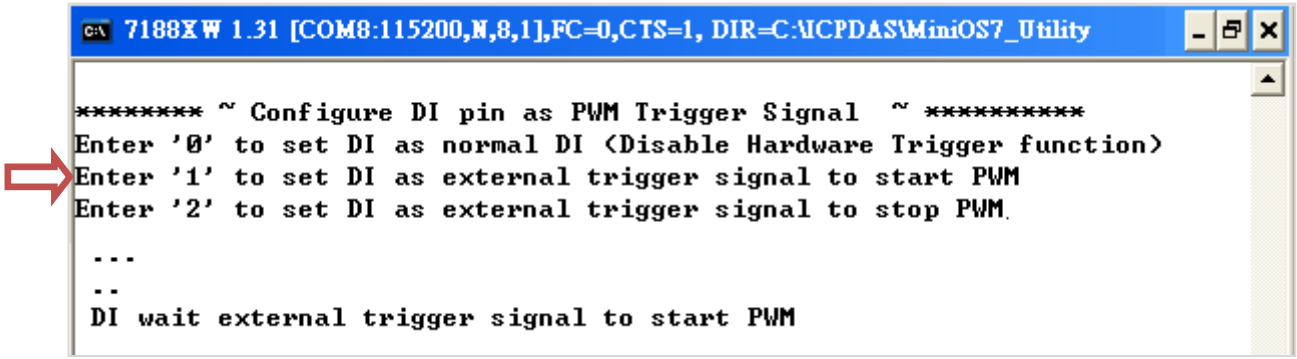
Step 3: Start the PWM

You can enter “1” to start the PWM output. It will start PWM when received an external trigger signal.



Start the PWM:

Using the i8088W_PWM_Start (section 3.11) or
pac_ i8088W_PWM_Start (section 4.11)
function.



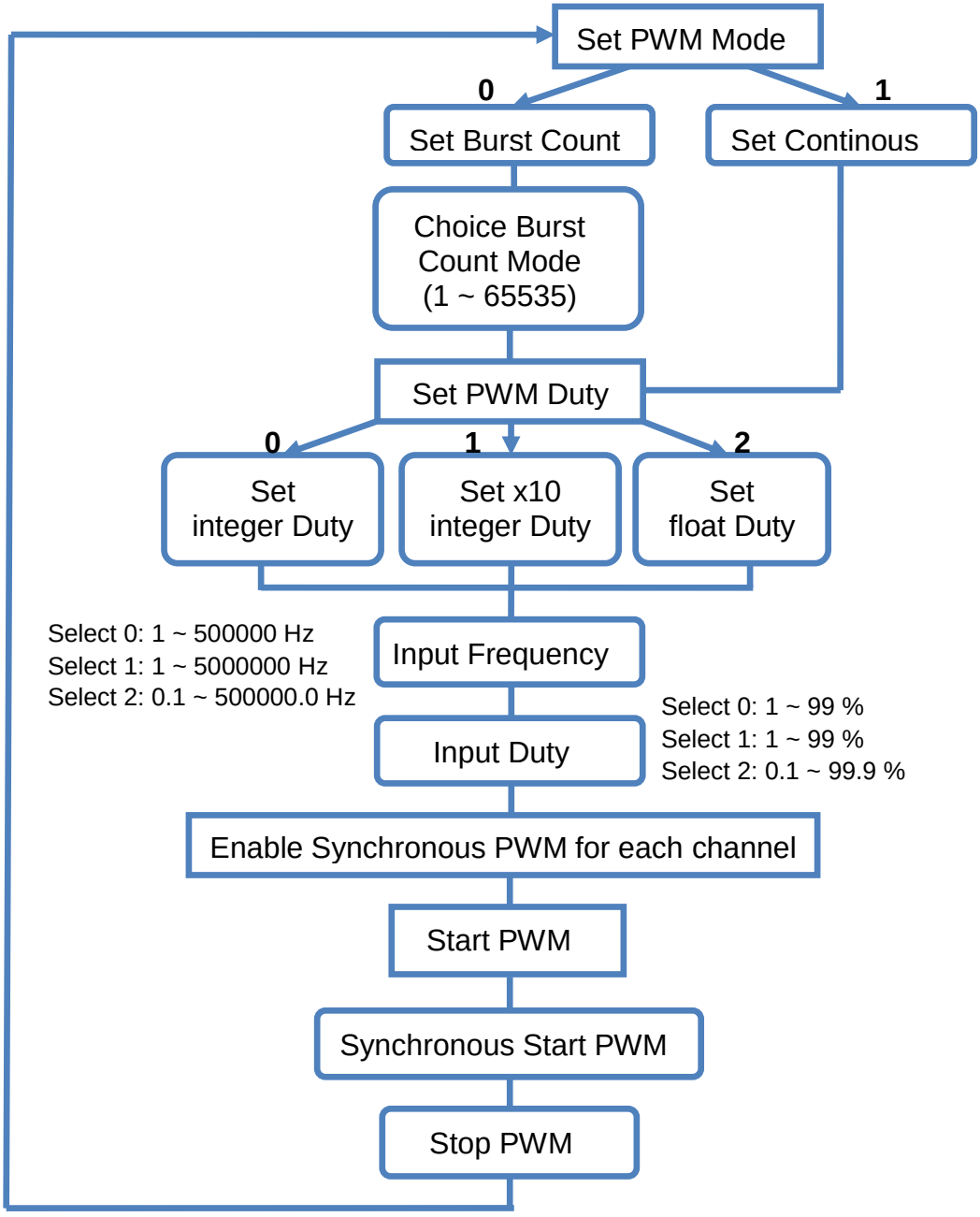
Stop the PWM:

Using the i8088W_PWM_Stop (section 3.12) or
pac_ i8088W_PWM_Stop (section 4.12)
function.

In Normal PWM mode, the signal will continue transfer until you go back to enter “2” to stop the PWM output.

2.4. Synchronize PWM

2.4.1. Flow Chart



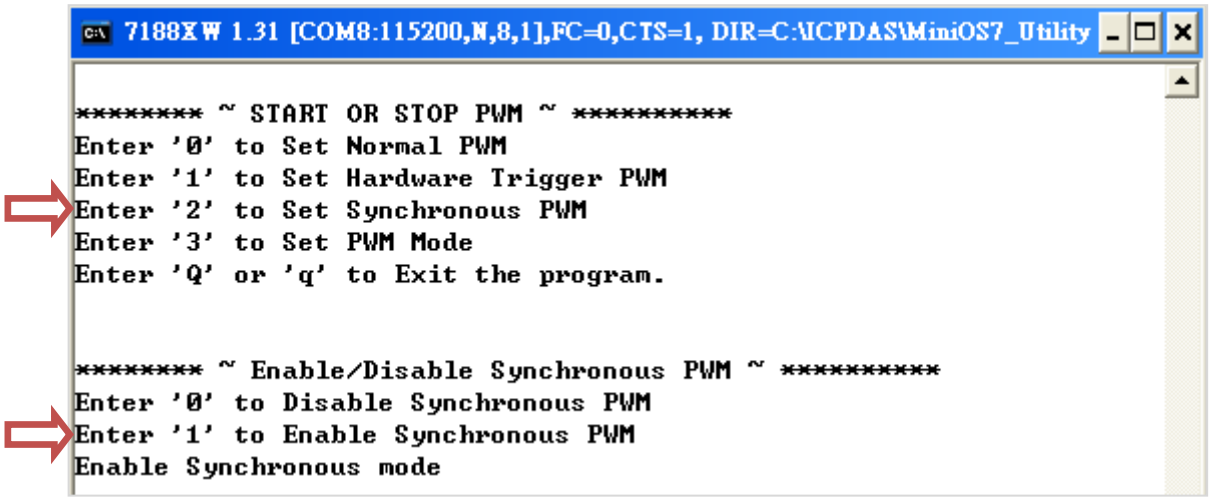
2.4.2. How to Setup the Synchronous PWM

I-8088W can configure each PWM output channel as synchronous mode.

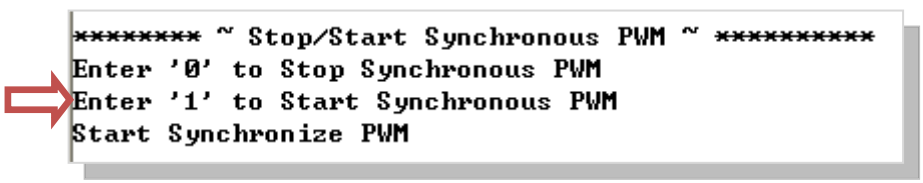
Step 1: Follow the same way in section 2.2.2 to configure PWM output mode and set PWM duty and frequency.

Step 2: Enter “2” to set “Synchronous PWM”, and then enter “1” to enable it.

Note: You can use i8088W_SetSyncChannel (section 3.13) or pac_ i8088W_SetSyncChannel (section 4.13) function to set the specific channel as synchronous mode.



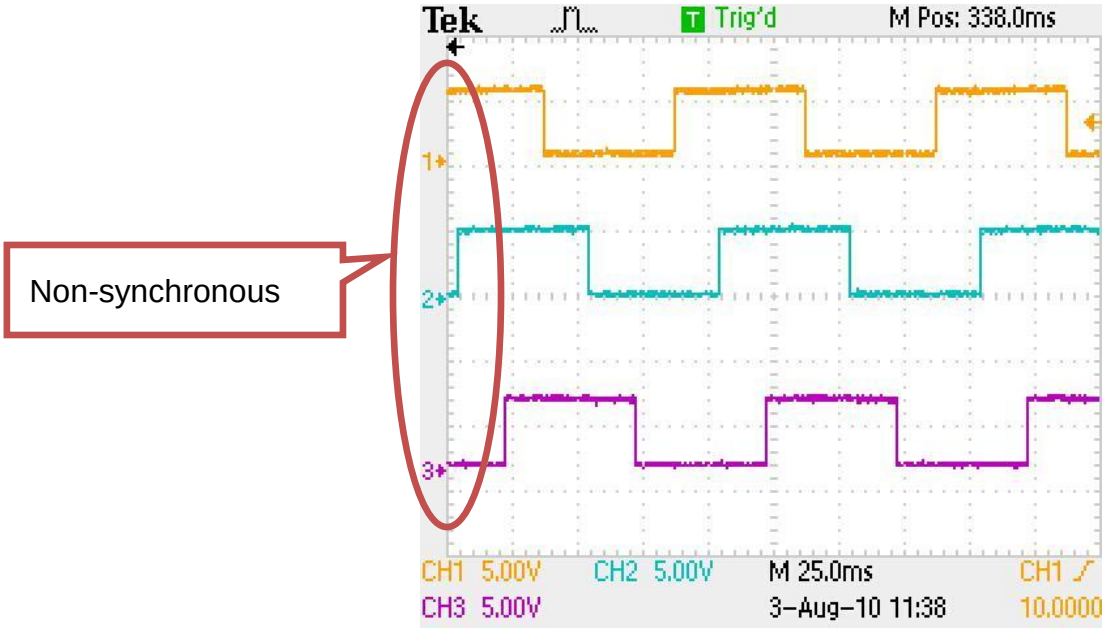
When you enable PWM output channels as synchronous mode, it will call i8088W_Sync_Start (section 3.15) or pac_i8088W_Sync_Start (section 4.15) function to start synchronous PWM output and synchronize the rising edge for each synchronous PWM pulse.



You will see the different between following two pictures.

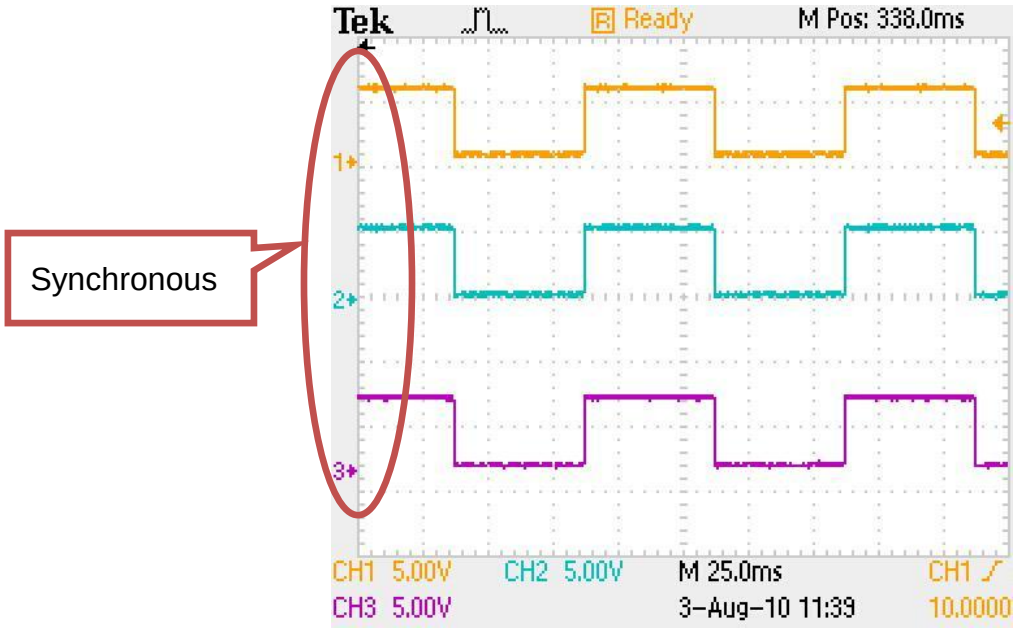
Start PWM:

In previous sections, we can start PWM by software command or DI trigger signal, but you can see the signal is non-synchronous in below picture.



Start Synchronous PWM:

So, we can call i8088W_Sync_Start or pac_i8088W_Sync_Start to start the PWM output and synchronize the rising edge for each synchronous PWM pulse.



3. API FOR IPAC-8000

3.1. i8088W_Init

The function can initialize the I-8088W and then check the hardware ID for each slot. If the return value is “0” that means there is an I-8088W module in that slot. If return “-1” that means there is no I-8088W module.

Syntax

```
short i8088W_Init(int slot);
```

Parameter

slot: 0 ~ 7

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slotIndex,err;
err=i8088W_Init(slotIndex);
if(err==0)
{
  Print(“There is an I-8088W at slot %d\n”,slotIndex);
}
else
{
  Print(“There is no I-8088W at slot %d\n”,slotIndex);
}
```

3.2. i8088W_GetFirmwareVersion

The function is used to get the firmware version of I-8088W.

Syntax

```
short i8088W_GetFirmwareVersion(int slot);
```

Parameter

slot: 0 ~ 7

Return Values

The firmware version of I-8088W hardware

Examples

[C]

```
short firmware_version;
firmware_version = i8088W_GetFirmwareVersion(int slot);
```

3.3. i8088W_GetLibVersion

The function is used to get the version of library file.

Syntax

```
short i8088W_GetLibVersion(void);
```

Parameter

None

Return Values

The versions of library file.

Examples

[C]

```
short version;
version = i8088W_GetLibVersion();
```

3.4. i8088W_GetLibDate

The function is used to get the built date of library file.

Syntax

```
short i8088W_GetLibDate(char *LibDate);
```

Parameter

LibDate the string buffer of library built date

Return Values

The built dates of library file.

Examples

[C]

```
char lib_date[32];
i8088W_GetLibDate(lib_date);
```

3.5. i8088W_SetPWMDuty

The function is used to set the related PWM parameters.

Syntax

```
short i8088W_SetPWMDuty(int slot,int ch,unsigned long hz,unsigned int duty) ;
```

Parameter

1. slot 0 ~ 7
2. ch 0 ~ 7
3. hz 0 ~ 450K
4. duty High part 0 ~ 99
5. Low part 100 - High part

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot,ch;
slot = 0;
for(ch=0;ch<8;ch++)
{
  i8088W_ SetPWMDuty (slot,ch,50);
}
```

3.6. i8088W_SetPWMDuty_Deci

The function is used to set the PWM parameters precisely.

Syntax

```
short i8088W_SetPWMDuty_Deci
(int slot,int ch,unsigned long deci_Hz,unsigned int deci_duty);
```

Note:

i8088W_SetPWMDuty_**Deci** is the same as i8088W_SetPWMDuty_**float** in usage, but the i8088W_SetPWMDuty_Deci will **run faster** than i8088W_SetPWMDuty_floa for the floating calculation reason

Parameter

1. slot 0 ~ 7
2. ch 0 ~ 7
3. deci_Hz example: 10 deci_Hz = 10 Hz
4. deci_duty 0~999, for example: 503 deci_duty = 50.3% High Part
5. Low Part=1000 - 503= 497 => 49.7% Low Duty cycle

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot,ch;
slot = 0;
for(ch=0;ch<8;ch++)
{
  i8088W_ SetPWMDuty_Deci(slot,ch,50.3);
}
```

3.7. i8088W_SetPWMDuty_float

The function is used to set the PWM parameters precisely.

Syntax

```
short i8088W_SetPWMDuty_Float(int slot,int ch,float f_Hz,float f_Duty);
```

Note:

i8088W_SetPWMDuty_**Deci** is the same as i8088W_SetPWMDuty_**float** in usage, but the i8088W_SetPWMDuty_Deci will **run faster** than i8088W_SetPWMDuty_floa for the floating calculation reason

Parameter

1. slot 0 ~ 7
2. ch 0 ~ 7
3. f_Hz 100 means f_Hz = 10 Hz
4. f_Duty 0.0~99.9, for example: 50.3 means 50.3% High Part
5. Low Part =100.0- f_Duty = 49.7 means = 49.7%

Return Values

Please refer to Error Code Table.

Examples

[C++]

```
int slot,ch;
slot = 0;
for(ch=0;ch<8;ch++)
{
  i8088W_ SetPWMDuty_Float (slot,ch,50.3);
}
```

3.8. i8088W_GetRealPWMDuty_Deci

The function will get real frequency and duty that can be produced by 8088W.

Syntax

```
short i8088W_GetRealPWMDuty_Deci
(int slot,int ch,unsigned long* deci_hz,unsigned int* deci_duty);
```

Note:

The duty and frequency of 8088W PWM is discrete not continuous,when we use i8088W_SetPWMDuty to configure the duty and frequency, we have to use this function to check the real duty and frequency which can be produced by 8088W normally, at low frequency 10K, the configued frequency will be closer to real frequcncy can be generated.

Parameter

1. slot 0 ~ 7
2. ch 0 ~ 7
3. deci_hz: the real frequency produced by 8088W unit (x10Hz)
4. deci_duty: the real duty produced by 8088W (x10 %)

Return Values

Please refer to Error Code Table.

Examples

[C]

```
unsigned long deci_hz;
unsigned int deci_duty
slot = 0;
for(ch=0;ch<8;ch++)
{
    i8088W_GetRealPWMDuty_Deci(slot,ch,& deci_hz, & deci_duty);
    Print("CH[%d] PWM Hz = %lu ; Duty = %u\n", ch, deci_hz, deci_duty);
}
```

3.9. i8088W_SetPWMCountMode

The function is used to set the count mode of I-8088W.

Syntax

```
short i8088W_SetPWMCountMode(int slot,int ch,unsigned char countMode);
```

Parameter

1. slot 0 ~ 7
2. ch 0 ~ 7
3. count Mode 1: Continuos ;
   0: Burst count

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot,ch;
slot = 0;
mode=0; //burst mode
for(ch=0;ch<8;ch++)
{
  i8088W_ SetPWMCountMode(slot,ch,mode);
}
```

3.10. i8088W_SetBurstCount

The function is used to set the BurstCount of I-8088W.

Syntax

```
short i8088W_SetBurstCount(int slot,int ch,unsigned int burstCount);
```

Parameter

1. slot 0 ~ 7
2. ch 0 ~ 7
3. burstCount 0~65536

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot,ch, burstCount;
slot = 0;
burstCount=10000;
for(ch=0;ch<8;ch++)
{
  i8088W_ SetBurstCount (slot,ch, burstCount);
}
```

3.11. i8088W_PWM_Start

The function is used to start the PWM pulse.

Syntax

```
short i8088W_PWM_Start(int slot,int ch);
```

Parameter

slot: 0 ~ 7

ch: 0 ~ 7

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot,ch;
slot = 0;
for(ch=0;ch<8;ch++)
{
  i8088W_PWM_Start (slot,ch);
}
```

3.12. i8088W_PWM_Stop

The function is used to stop the PWM pulse.

Syntax

```
short i8088W_PWM_Stop(int slot,int ch);
```

Parameter

slot: 0 ~ 7

ch: 0 ~ 7

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot,ch;
slot = 0;
for(ch=0;ch<8;ch++)
{
  i8088W_PWM_Stop (slot,ch);
}
```

3.13. i8088W_SetSyncChannel

The function is used to set the specific channel as synchronous.

Syntax

```
short i8088W_SetSyncChannel(int slot,int ch,int enBit);
```

Parameter

1. slot 0 ~ 7
2. ch 0 ~ 7
3. enBit 1: define channel as synchronous mode,
   0: not synchronous mode

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot,ch, enBit;
slot = 0;
enBit=1;
for(ch=0;ch<8;ch++)
{
  i8088W_ SetSyncChannel(slot,ch, enBit);
}
```

3.14. i8088W_GetSyncChannel

The function is used to get the synchronous channel by using array.

Syntax

```
short i8088W_GetSyncChannel(int slot,int syncArr[]) ;
```

Parameter

1. slot 0 ~ 7
2. syncArr[] is an 8 bit array
3. syncArr[i]=1 means channel[i] is set as synchronous

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot, syncArr[];
slot = 0;
 i8088W_ GetSyncChannel (slot, syncArr);
```

3.15. i8088W_Sync_Start

The function is used to start the synchronization of PWM pulse.

Syntax

```
short i8088W_Sync_Start(int slot);
```

Parameter

1. slot 0 ~ 7

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot;
slot = 0;
i8088W_Sync_Start (slot);
```

3.16. i8088W_Sync_Stop

The function is used to stop the synchronization of PWM pulse.

Syntax

```
short i8088W_Sync_Stop(int slot);
```

Parameter

1. slot 0 ~ 7

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot;
slot = 0;
i8088W_ Sync_Stop (slot);
```

3.17. i8088W_SetHardwareTrigChannel

The "DI" pin of I-8088W can be set as hardware trigger pin or simply DI pin. The user can call this function to specify the status of channels.

Syntax

```
short i8088W_SetHardwareTrigChannel(int slot,int ch,int triggerState)
```

Parameter

1. slot 0 ~ 7
2. ch 0 ~ 7
3. triggerState =0, disabled hardware trigger (Normal DI pin)
4. triggerState=1, means this DI has been configured as external trigger signal for I-8088W to startup its PWM pulse.
5. triggerState=2, this DI has been configured as external trigger signal for I-8088W to stop its PWM pulse.

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot,ch, triggerState;
slot = 0;
triggerState=0;
for(ch=0;ch<8;ch++)
{
  i8088W_ SetHardwareTrigChannel (slot,ch, triggerState);
}
```

3.18. i8088W_GetHardwareTrigChannel

The "DI" pin of I-8088W can be set as hardware trigger pin or simply DI pin. The user can call this function to know the status of channels.

Syntax

```
i8088W_GetHardwareTrigChannel(int slot,int ch,int* triggerState);
```

Parameter

1. slot 0 ~ 7
2. ch 0 ~ 7
3. trigState
   - 0 : means the hardware trigger function is disable. (Normal DI pin)
   - 1: means this DI has been configured as external trigger signal for I-8088W to startup its PWM pulse.
   - 2: this DI has been configured as external trigger signal for I-8088W to stop its PWM pulse.

Return Values.

Please refer to Error Code Table.

Examples

[C]

```
int slot,ch, triggerState;
slot = 0;
triggerState=0;
for(ch=0;ch<8;ch++)
{
    i8088W_ GetHardwareTrigChannel(slot,ch,& triggerState);
}
```

3.19. i8088W_GetPWMActiveState

The function is used to get PWM status.

Syntax

```
short i8088W_GetPWMActiveState(int slot,unsigned int *State,int ActiveArr[]);
```

Parameter

1. slot 0 ~ 7
2. *State: the PWM output status of i-8088 value 0~0xff
3. ActiveArr[]: the *State value will be parse into Bit Array

   When PWM generate pulse output, the state will be 1, we can know which PWM channel is sending PWM output. If we configure PWM as Burst Count mode, the PWM output will be stop when it sends confgured count of PWM pulse (for example 1000 pulse) by using this function, we can know the actural PWM output state.

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot=0,ch,activatedBit[8];
unsigned int activatedState=0;
i8088W_GetPWMActiveState (slot,& activatedState, activatedBit);
for(ch=0;ch<8;ch++)
{
    if(activatedBit[ch])
        Print("PWM CH[%d] is activated\n",ch);
}
```

3.20. i8088W_GetDI

The function is used to get DI status. The user can view which channels have received the signals.

Syntax

```
short i8088W_GetDI(int slot,unsigned int *diVal,int diArr[]);
```

Parameter

1. slot 0 ~ 7
2. *diVal: the DI status of i-8088 value 0~0xff
3. diArr[]: the *DI value will be parse into Bit Array

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot,ch,enBit[8];
Slot=0;
unsigned int dival=0;

i8088W_GetDI(slot, &diVal,enBit);
Print ("DI Vaule = %02X\n",diVal);

for (ch=0; ch< 8; ch++)
{
    Print ("DI[%d]= %d\n",ch, enBit[ch]);
}
```

4. API FOR WINPAC-8000

4.1. pac_i8088W_Init

The function can initialize the I-8088W and then check the hardware ID for each slot. If the return value is “0” that means there is an I-8088W module in that slot. If return “-1” that means there is no I-8088W module.

Syntax

```
short pac_i8088W_Init(int slot);
```

Parameter

Slot: 0 ~ 7

Return Values

Please refer to Error Code Table.

Examples

[C++]

```
int slotIndex,err;
err=pac_i8088W_Init(slotIndex);
if(err==0)
{
  printf("There is an I-8088W at slot %d\n",slotIndex);
}
else
{
  printf("There is no I-8088W at slot %d\n",slotIndex);
}
```

[C#]

```
using pac8088WNet;
int slotIndex,err;
err= pac8088W.Init(slotIndex);
if(err==0)
{
       Console.WriteLine("There is an I-8088W at slot {0}",slotIndex);
}
else
{
      Console.WriteLine("There is no I-8088W at slot {0}",slotIndex);
}
```

4.2. pac_i8088W_GetFirmwareVersion

The function is used to get the firmware version of I-8088W.

Syntax

```
short pac_i8088W_GetFirmwareVersion(int slot);
```

Parameter

slot: 0 ~ 7

Return Values

The firmware version of I-8088W hardware

Examples

[C++]

```
short firmware_version;
firmware_version = pac_i8088W_GetFirmwareVersion (slot);
```

[C#]

```
using pac8088WNet;
short firmware_version;
firmware_version = pac8088W.GetFirmwareVersion (slot);
```

4.3. pac_i8088W_GetLibVersion

The function is used to get the version of library file.

Syntax

```
short pac_i8088W_GetLibVersion ();
```

Parameter

none

Return Values

The versions of library file (i8088W.dll).

Examples

[C++]

```
short version;
version = pac_i8088W_GetLibVersion ();
```

[C#]

```
using pac8088WNet;
short version;
version = pac8088W.GetLibVersion ( );
```

4.4. pac_i8088W_GetLibDate

The function is used to get the built date of library file.

Syntax

```
short pac_i8088W_GetLibDate(char *LibDate);
```

Parameter

LibDate the string buffer of library built date

Return Values

The built dates of library file.

Examples

[C++]

```
char lib_date[32];
pac_i8088W_GetLibDate(lib_date);
```

[C#]

```
using pac8088WNet;
string lib_date;
lib_date = pac8088W.GetLibDate( );
```

4.5. pac_i8088W_SetPWMDuty

The function is used to set the related PWM parameters.

Syntax

```
short pac_i8088W_SetPWMDuty
(int slot,int ch,unsigned long hz,unsigned int duty) ;
```

Parameter

1. slot 0 ~ 7
2. ch 0 ~ 7
3. hz 0 ~ 450K
4. duty High part 0 ~ 99
5. Low part 100 - High part

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot,ch;
slot = 0;
for(ch=0;ch<8;ch++)
{
    pac_i8088W_SetPWMDuty (slot,ch,50);
}
```

[C#]

```
using pac8088WNet;
int slot,ch;
slot = 0;
for(ch=0;ch<8;ch++)
{
    pac8088W.SetPWMDuty (slot,ch,50);
}
```

4.6. pac_i8088W_SetPWMDuty_Deci

The function is used to set the PWM parameters precisely.

Syntax

```
short pac_i8088W_SetPWMDuty_Deci (int slot,int ch,
unsigned long deci_Hz,unsigned int deci_duty);
```

Note:

pac_i8088W_SetPWMDuty_**Deci** is the same as
pac_i8088W_SetPWMDuty_**float** in usage, but the
pac_i8088W_SetPWMDuty_Deci will **run faster** than
pac_i8088W_SetPWMDuty_floa for the floating calculation reason

Parameter

1. slot 0 ~ 7
2. ch 0 ~ 7
3. deci_Hz, example: 100 means deci_Hz = 10 Hz
4. deci_duty 0~999, for example: 503 deci_duty = 50.3% High Part
5. Low Part=1000 - 503= 497 => 49.7% Low Duty cycle

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot,ch;
slot = 0;
for(ch=0;ch<8;ch++)
{
  pac_i8088W_ SetPWMDuty_Deci(slot,ch,50.3);
}
```

[C#]

```
using pac8088WNet;
int slot,ch;
slot = 0;
for(ch=0;ch<8;ch++)
{
  pac8088W.SetPWMDuty_Deci (slot,ch,50.3);
}
```

4.7. pac_i8088W_SetPWMDuty_float

The function is used to set the PWM parameters precisely.

Syntax

```
short pac_i8088W_SetPWMDuty_Float(int slot,int ch,float f_Hz,float f_Duty);
```

Note:

pac_i8088W_SetPWMDuty_**Deci** is the same as
pac_i8088W_SetPWMDuty_**float** in usage, but the
pac_i8088W_SetPWMDuty_Deci will **run faster** than
pac_i8088W_SetPWMDuty_floa for the floating calculation reason

Parameter

1. slot 0 ~ 7
2. ch 0 ~ 7
3. f_Hz 100 means f_Hz = 10 Hz
4. f_Duty 0.0~99.9, for example: 50.3 means 50.3% High Part
5. Low Part =100.0- f_Duty = 49.7 means = 49.7%

Return Values

Please refer to Error Code Table.

Examples

[C++]

```
int slot,ch;
slot = 0;
for(ch=0;ch<8;ch++)
{
   pac_i8088W_ SetPWMDuty_Float (slot,ch,50.3);
}
```

[C#]

```
using pac8088WNet;
int slot,ch;
slot = 0;
for(ch=0;ch<8;ch++)
{
   pac8088W.SetPWMDuty_Float (slot,ch,50.3);
}
```

4.8. pac_i8088W_GetRealPWMDuty_Deci

The function will get real frequency and duty that can be produced by 8088W.

Syntax

```
short pac_i8088W_GetRealPWMDuty_Deci
(int slot,int ch,unsigned long* deci_hz,unsigned int* deci_duty);
```

Note:

8088W frequency and duty is non-continuous range, we have to use this function to get the real frequency and duty which 8088W can produce.

Parameter

1. slot 0 ~ 7
2. ch 0 ~ 7
3. deci_hz: 10 ~ 5000000
4. deci_duty: 1~999

Return Values

Please refer to Error Code Table.

Examples

[C++]

```
int slot,ch;
unsigned long deci_hz;
unsigned int deci_duty

slot = 0;

for(ch=0;ch<8;ch++)

{

    pac_i8088W_GetRealPWMDuty_Deci (slot,ch,& deci_hz, & deci_duty);
    printf("CH[%d] PWM Hz = %lu ; Duty = %u\n", ch, deci_hz, deci_duty);

}
```

[C#]

```
using pac8088WNet;

int slot,ch;
UInt32 deci_hz=0;
Uint16 deci_duty=0

slot = 0;

for(ch=0;ch<8;ch++)

{

    pac8088W .GetRealPWMDuty_Deci (slot,ch,ref deci_hz, ref deci_duty);
    Console.WriteLine("CH[{0}] PWM Hz = {1} ; Duty = {2}", ch, deci_hz,
                                    deci_duty);

}
```

4.9. pac_i8088W_SetPWMCountMode

The function is used to set the count mode of I-8088W.

Syntax

```
short pac_i8088W_SetPWMCountMode (int slot,int ch,
unsigned char countMode);
```

Parameter

1. slot 0 ~ 7
2. ch 0 ~ 7
3. count Mode 1: Continuos ;
   0: Burst count

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot,ch;
slot = 0;
mode=0; //burst mode
for(ch=0;ch<8;ch++)
{
    pac_i8088W_ SetPWMCountMode(slot,ch,mode);
}
```

[C#]

```
using pac8088WNet;
int slot,ch;
slot = 0;
mode=0; //burst mode
for(ch=0;ch<8;ch++)
{
    pac8088W.SetPWMCountMode(slot,ch,mode);
}
```

4.10. pac_i8088W_SetBurstCount

The function is used to set the BurstCount of I-8088W.

Syntax

```
short pac_i8088W_SetBurstCount(int slot,int ch,unsigned int burstCount);
```

Parameter

1. slot 0 ~ 7
2. ch 0 ~ 7
3. burstCount 0~65536

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot,ch, burstCount;
slot = 0;
burstCount=10000;
for(ch=0;ch<8;ch++)
{
    pac_i8088W_SetBurstCount (slot,ch, burstCount);
}
```

[C#]

```
using pac8088WNet;
int slot,ch, burstCount;
slot = 0;
burstCount=10000;
for(ch=0;ch<8;ch++)
{
    pac8088W.SetBurstCount (slot,ch, burstCount);
}
```

4.11. pac_i8088W_PWM_Start

The function is used to start the PWM pulse.

Syntax

```
short pac_i8088W_PWM_Start(int slot,int ch);
```

Parameter

slot: 0 ~ 7

ch: 0 ~ 7

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot,ch;
slot = 0;
for(ch=0;ch<8;ch++)
{
    pac_i8088W_PWM_Start(slot,ch);
}
```

[C#]

```
using pac8088WNet;
int slot,ch;
slot = 0;
for(ch=0;ch<8;ch++)
{
    pac8088W.PWM_Start(slot,ch);
}
```

4.12. pac_i8088W_PWM_Stop

The function is used to stop the PWM pulse.

Syntax

```
short pac_i8088W_PWM_Stop(int slot,int ch);
```

Parameter

slot: 0 ~ 7

ch: 0 ~ 7

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot,ch;
slot = 0;
for(ch=0;ch<8;ch++)
{
    pac_i8088W_PWM_Stop (slot,ch);
}
```

[C#]

```
using pac8088WNet;
int slot,ch;
slot = 0;
for(ch=0;ch<8;ch++)
{
    pac8088W.PWM_Stop (slot,ch);
}
```

4.13. pac_i8088W_SetSyncChannel

The function is used to set the specific channel as synchronous.

Syntax

```
short pac_i8088W_SetSyncChannel(int slot,int ch,int enBit);
```

Parameter

1. slot 0 ~ 7
2. ch 0 ~ 7
3. enBit 1: define channel as synchronous mode,
   0: not synchronous mode

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot,ch, enBit;
slot = 0;
enBit=1;
for(ch=0;ch<8;ch++)
{
    pac_i8088W_ SetSyncChannel(slot,ch, enBit);
}
```

[C#]

```
using pac8088WNet;
int slot,ch, enBit;
slot = 0;
enBit=1;
for(ch=0;ch<8;ch++)
{
    pac8088W.SetSyncChannel (slot,ch, enBit);
}
```

4.14. pac_i8088W_GetSyncChannel

The function is used to get the synchronous channel by using array.

Syntax

```
short pac_i8088W_GetSyncChannel(int slot,int syncArr[]) ;
```

Parameter

1. slot 0 ~ 7
2. syncArr[] is an 8 bit array
3. syncArr[i]=1 means channel[i] is set as synchronous

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot, syncArr[];
slot = 0;
  pac_i8088W_ GetSyncChannel (slot, syncArr);
```

[C#]

```
using pac8088WNet;
int slot, syncArr[];
slot = 0;
pac8088W.GetSyncChannel (slot, syncArr);
```

4.15. pac_i8088W_Sync_Start

The function is used to start the synchronization of PWM pulse.

Syntax

```
short pac_i8088W_Sync_Start(int slot);
```

Parameter

1. slot 0 ~ 7

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot;
slot = 0;
pac_i8088W_Sync_Start (slot);
```

[C#]

```
using pac8088WNet;
int slot;
slot = 0;
pac8088W.Sync_Start(slot);
```

4.16. pac_i8088W_Sync_Stop

The function is used to stop the synchronization of PWM pulse.

Syntax

```
short pac_i8088W_Sync_Stop(int slot);
```

Parameter

1. slot 0 ~ 7

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot;
slot = 0;
pac_i8088W_Sync_Stop(slot);
```

[C#]

```
using pac8088WNet;
int slot;
slot = 0;
pac8088W.Sync_Stop(slot);
```

4.17. pac_i8088W_SetHardwareTrigChannel

The "DI" pin of I-8088W can be set as hardware trigger pin or simply DI pin. The user can call this function to specify the status of channels.

Syntax

```
short pac_i8088W_SetHardwareTrigChannel(int slot,int ch,int triggerState)
```

Parameter

1. slot 0 ~ 7
2. ch 0 ~ 7
3. triggerState =0, disabled hardware trigger (Normal DI pin)
4. triggerState=1, means this DI has been configured as external trigger signal for I-8088W to startup its PWM pulse.
5. triggerState=2, this DI has been configured as external trigger signal for I-8088W to stop its PWM pulse.

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot,ch, triggerState;
slot = 0;
triggerState=0;
for(ch=0;ch<8;ch++)
{
    pac_i8088W_SetHardwareTrigChannel (slot,ch, triggerState);
}
```

[C#]

```
using pac8088WNet;
int slot,ch, triggerState;
slot = 0;
triggerState=0;
for(ch=0;ch<8;ch++)
{
    pac8088W.SetHardwareTrigChannel(slot,ch, triggerState);
}
```

4.18. pac_i8088W_GetHardwareTrigChannel

The "DI" pin of I-8088W can be set as hardware trigger pin or simply DI pin. The user can call this function to know the status of channels.

Syntax

```
short pac_i8088W_GetHardwareTrigChannel(int slot,int ch,int* triggerState);
```

Parameter

4. slot 0 ~ 7
5. ch 0 ~ 7
6. trigState
   - 0 : means the hardware trigger function is disable. (Normal DI pin)
   - 1: means this DI has been configured as external trigger signal for I-8088W to startup its PWM pulse.
   - 2: this DI has been configured as external trigger signal for I-8088W to stop its PWM pulse.

Return Values.

Please refer to Error Code Table.

Examples

[C]

```
int slot,ch, triggerState;
slot = 0;
triggerState=0;
for(ch=0;ch<8;ch++)
{
  pac_i8088W_GetHardwareTrigChannel(slot,ch, &triggerState);
}
```

[C#]

```
using pac8088WNet;
int slot,ch, triggerState;
slot = 0;
triggerState=0;
for(ch=0;ch<8;ch++)
{
      pac8088W.GetHardwareTrigChannel(slot,ch,ref triggerState);
}
```

4.19. pac_i8088W_GetPWMActiveState

The function is used to get PWM status.

Syntax

```
short pac_i8088W_ GetPWMActiveState
(int slot,unsigned int *State,int ActiveArr[]);
```

Parameter

1. slot 0 ~ 7
2. *State: the PWM output status of i-8088 value 0~0xff
3. ActiveArr[]: the *State value will be parse into Bit Array

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot=0,ch,activatedBit[8];
unsigned int activatedState=0;
pac_i8088W_GetPWMActiveState (slot,& activatedState, activatedBit);
for(ch=0;ch<8;ch++)
{
    if(activatedBit[ch])
        printf("PWM CH[%d] is activated\n",ch);
}
```

[C#]

```
using pac8088WNet;
int slot=0,ch,activatedBit[8];
Uint16 activatedState=0;

pac8088W. GetPWMActiveState (slot, ref activatedState, enBit);
Console.WriteLine("PWM output state ={0:X2}", activatedState);

for (ch=0; ch< 8; ch++)
{
    if(activatedBit[ch]==1)
        Console.WriteLine("PWM CH [{0}] is activated ",ch);
}
```

4.20. pac_i8088W_GetDI

The function is used to get DI status. The user can view which channels have received the signals.

Syntax

```
short pac_i8088W_GetDI(int slot,unsigned int *diVal,int diArr[]);
```

Parameter

1. slot 0 ~ 7
2. *diVal: the DI status of i-8088 value 0~0xff
3. diArr[]: the *DI value will be parse into Bit Array

Return Values

Please refer to Error Code Table.

Examples

[C]

```
int slot,ch,enBit[8];
Slot=0;
unsigned int dival=0;

pac_i8088W_GetDI(slot, &diVal,enBit);
Print ("DI Vaule = %02X\n",diVal);

for (ch=0; ch< 8; ch++)
{
    Print ("DI[%d]= %d\n",ch, enBit[ch]);
}
```

[C#]

```
using pac8088WNet;
int slot=0,ch,enBit[8];
Uint16 dival=0;

pac8088W.GetDI(slot, &diVal,enBit);
Console.WriteLine("DI Vaule = {0:X2}",diVal);

for (ch=0; ch< 8; ch++)
{
    Console.WriteLine("DI[{0}]= {1}",ch, enBit[ch]);
}
```

APPENDIX A. ERROR CODE

0	OK
-1	ID_ERROR
-2	SLOT_OUT_RANGE
-3	CHANNEL_OUT_RANGE
-4	SELECT_CHANNEL_ERROR
-5	HI_DUTY_OUT_RANGE
-6	LO_DUTY_OUT_RANGE