



MediaTek

Driver Customization Tool

User Manual

Preliminary Information

Revision: 0.1

Release Date: Jun. 14, 2006

Legal Disclaimer

BY OPENING OR USING THIS FILE, BUYER HEREBY UNEQUIVOCALLY ACKNOWLEDGES AND AGREES THAT THE SOFTWARE/FIRMWARE AND ITS DOCUMENTATIONS ("MEDIATEK SOFTWARE") RECEIVED FROM MEDIATEK AND/OR ITS REPRESENTATIVES ARE PROVIDED TO BUYER ON AN "AS-IS" BASIS ONLY. MEDIATEK EXPRESSLY DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. NEITHER DOES MEDIATEK PROVIDE ANY WARRANTY WHATSOEVER WITH RESPECT TO THE SOFTWARE OF ANY THIRD PARTY WHICH MAY BE USED BY, INCORPORATED IN, OR SUPPLIED WITH THE MEDIATEK SOFTWARE, AND BUYER AGREES TO LOOK ONLY TO SUCH THIRD PARTY FOR ANY WARRANTY CLAIM RELATING THERETO. MEDIATEK SHALL ALSO NOT BE RESPONSIBLE FOR ANY MEDIATEK SOFTWARE RELEASES MADE TO BUYER'S SPECIFICATION OR TO CONFORM TO A PARTICULAR STANDARD OR OPEN FORUM.

BUYER'S SOLE AND EXCLUSIVE REMEDY AND MEDIATEK'S ENTIRE AND CUMULATIVE LIABILITY WITH RESPECT TO THE MEDIATEK SOFTWARE RELEASED HEREUNDER WILL BE, AT MEDIATEK'S OPTION, TO REVISE OR REPLACE THE MEDIATEK SOFTWARE AT ISSUE, OR REFUND ANY SOFTWARE LICENSE FEES OR SERVICE CHARGE PAID BY BUYER TO MEDIATEK FOR SUCH MEDIATEK SOFTWARE AT ISSUE.

THE TRANSACTION CONTEMPLATED HEREUNDER SHALL BE CONSTRUED IN ACCORDANCE WITH THE LAWS OF THE STATE OF CALIFORNIA, USA, EXCLUDING ITS CONFLICT OF LAWS PRINCIPLES.



Revision History

Revision	Date	Author	Comments
0.1	5/13/05	Simon Shih	Draft version with content to demonstrate how to operate UI.

Table of Contents

Legal Disclaimer	2
Revision History	3
Table of Contents	4
1 Introduction	6
1.1 Introduction	6
2 Operation of User Interface	7
2.1 Main Menu	7
2.1.1 [New]/[Open]	7
2.1.2 [Save]/[Save as]	8
2.1.3 [About]	9
2.1.4 [Exit]	9
2.1.5 Current Data Workspace File Path	9
2.1.6 Chipset selection	9
2.1.7 [Edit]	9
2.1.8 [CompareEdit]	10
2.1.9 [GenCode]	11
2.2 Component: GPIO (General Purpose IO)/GPO(General Purpose Output)	11
2.2.1 In MT6xxx.fig	11
2.2.2 In GPIO.cmp	11
2.2.3 GPIO Editing Window	12
2.2.4 GPO Editing Window	15
2.2.5 Generated Source and Header files	16
2.2.6 Rules for GPIO	18
2.3 Component: EINT (External Interrupt)	19
2.3.1 In MT6xxx.fig	19
2.3.2 In EINT.cmp	19
2.3.3 EINT Editing Window	20
2.3.4 Generated Source and Header files	22
2.3.5 Rules for EINT	23
2.4 Component: ADC (Analog to Digital Converter)	23
2.4.1 In MT6xxx.fig	23
2.4.2 In ADC.cmp	23
2.4.3 ADC Editing Window	24
2.4.4 Generated Source and Header files	25
2.4.5 Rules for ADC	25
2.5 Component: KEYPAD	26
2.5.1 In MT6xxx.fig	26
2.5.2 In KEYPAD.cmp	26
2.5.3 KEYPAD Editing Window	27
2.5.4 Generated Source and Header files	30
2.6 Component: UEM (User Equipment Management)	30
2.6.1 In MT6xxx.fig	30
2.6.2 In UEM.cmp	30
2.6.3 UEM Editing Window	31
2.6.4 Generated Source and Header files	33



3	Customization Procedure for a New Project	35
3.1	Step 1: Turn on Feature.....	35
3.2	Step 2: Create a data workspace file	35
3.3	Step 3: Edit settings on Tool.....	36
3.4	Step 4: Settings beyond the Tool.....	36
3.4.1	Debug Mode.....	36
3.4.2	GPIO output initial value.....	36
3.4.3	Settings for unused GPIO pins.....	36
3.5	Step 5: Build the load.....	36
3.6	Step 6: Misc. and Debug	37
3.6.1	Case 1	37
3.6.2	Case 2.....	37
	[References].....	38

1 Introduction

1.1 Introduction

The document is to illustrate the whole procedure for user to customize driver settings with tool DrvGen.exe. The purpose of this procedure is to provide an easy and better way to manage driver settings that can be customized by different project. In other words, the entire related driver settings will be centralized. And this will reduce the complexities and bugs due to wrong settings. Moreover, the tool has been integrated into the project building procedure.

Now there are 5 driver component settings included as below:

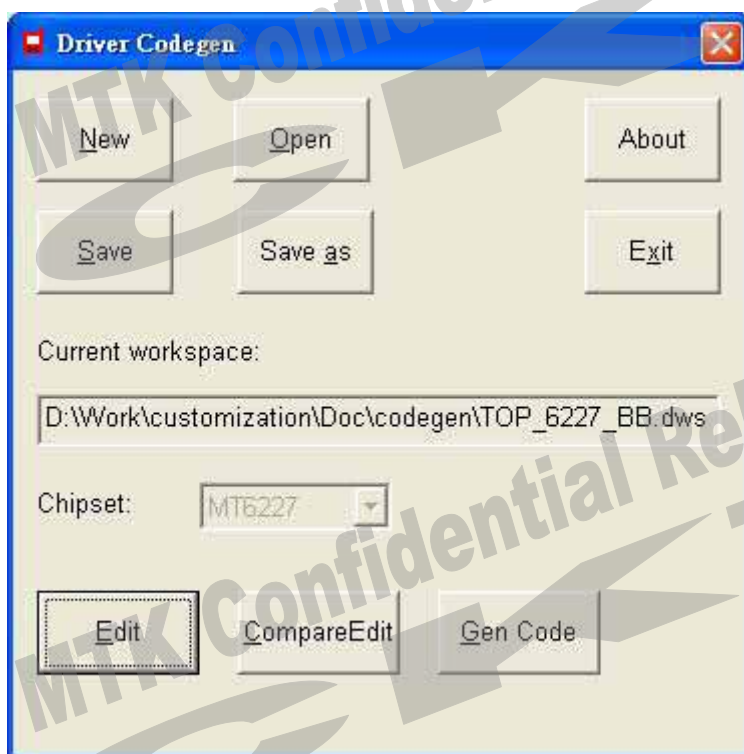
1. GPIO and GPO
2. ADC (Analog to Digital Converter)
3. EINT (External interrupt)
4. UEM (User Equipment Management)
5. Keypad

Each component's settings will be demonstrated in the following sections.

2 Operation of User Interface

User Interface of Driver Customization Tool includes Main Menu and pages for GPIO, GPO, ADC, EINT, Keypad and UEM component. In this chapter, we will see how to use it in detail.

2.1 Main Menu

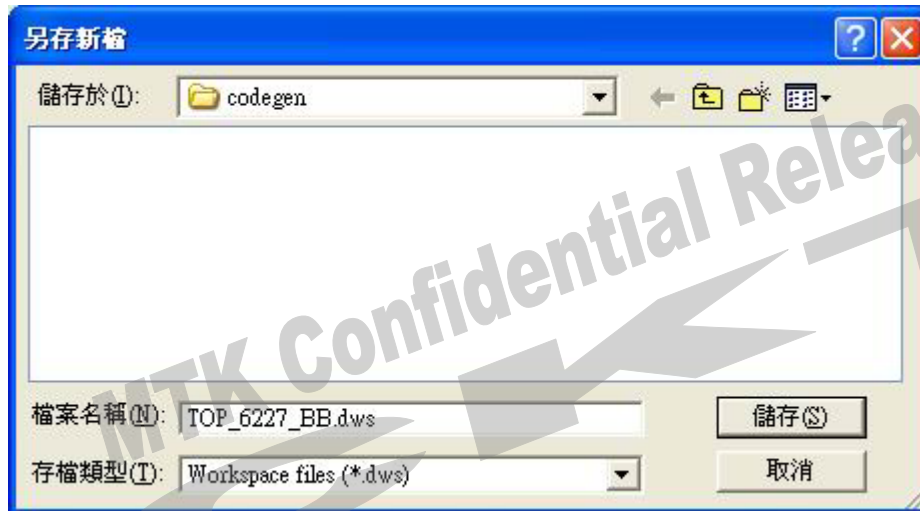


The main menu of this tool includes:

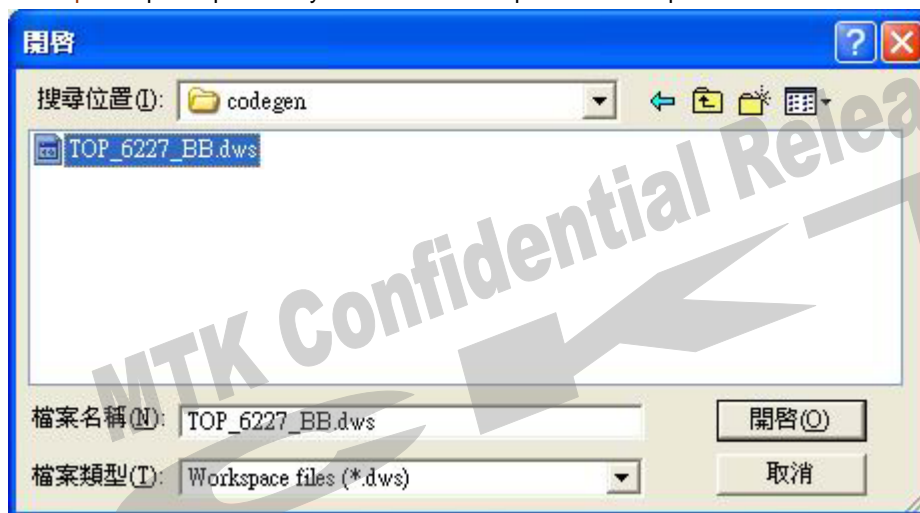
1. [New]/[Open] button
2. [Save]/[Save as] button
3. [About] button
4. [Exit] button
5. Current data workspace path
6. Chipset selection
7. [Edit] button
8. [CompareEdit] button
9. [Gen Code] button

2.1.1 [New]/[Open]

1. **New:** Start a new data workspace file (.dws) in a fold which user can specify.

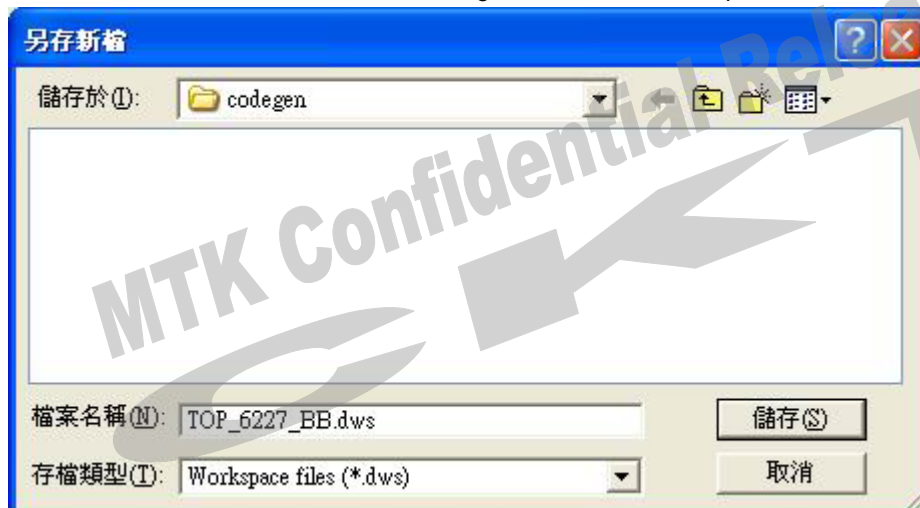


2. **Open:** Open a previously saved data workspace file in a specified folder.



2.1.2 [Save]/[Save as]

1. **Save:** Save the modification of setting in the original data workspace file.
2. **Save as:** Save the modification of setting in another data workspace file that can be specified.



2.1.3 [About]

The [About] button shows the version information for this tool.



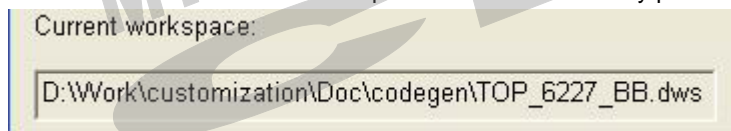
2.1.4 [Exit]

The [Exit] button is to exit the program. Before actually leaving the program, a pop-up message will show up to ask user if he/she wants to save the data workspace file.



2.1.5 Current Data Workspace File Path

This shows the current data workspace file with the directory path that user are working in.



2.1.6 Chipset selection

When a new data workspace file is created, user must select a chipset type via this field.



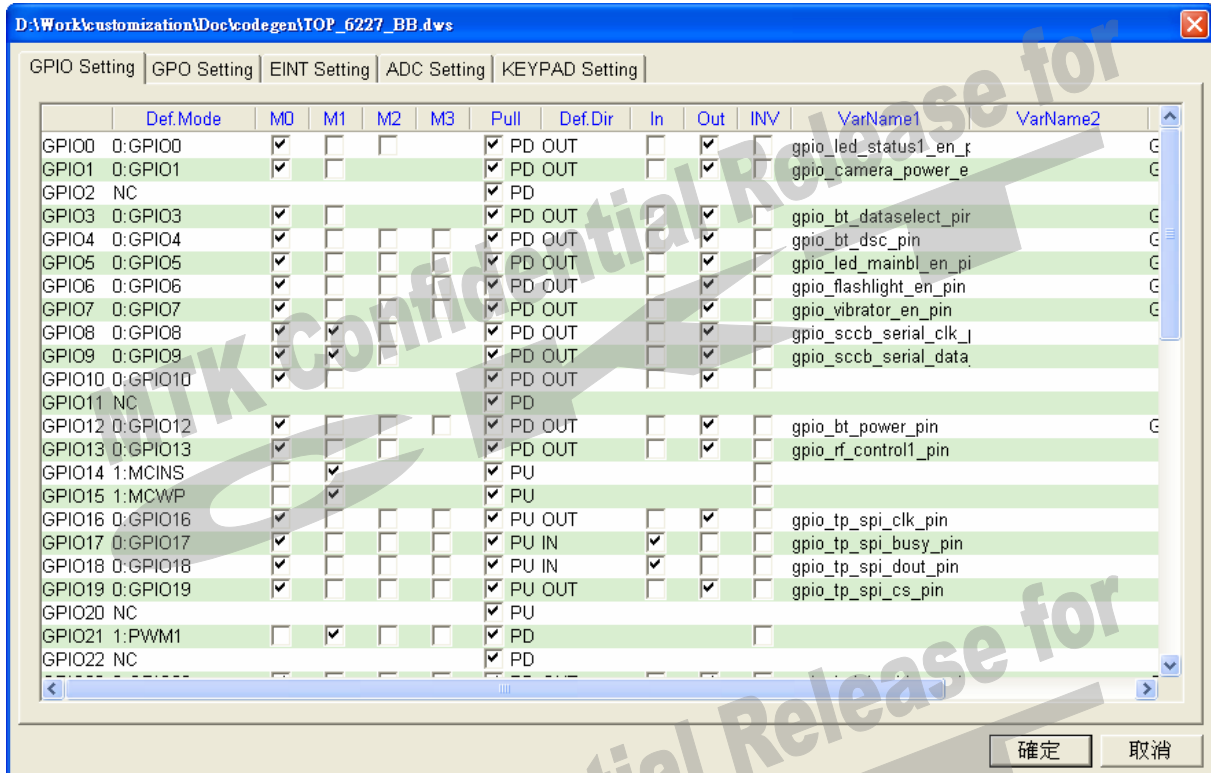
When opening a previous data workspace file, this field just shows the chipset type.



2.1.7 [Edit]

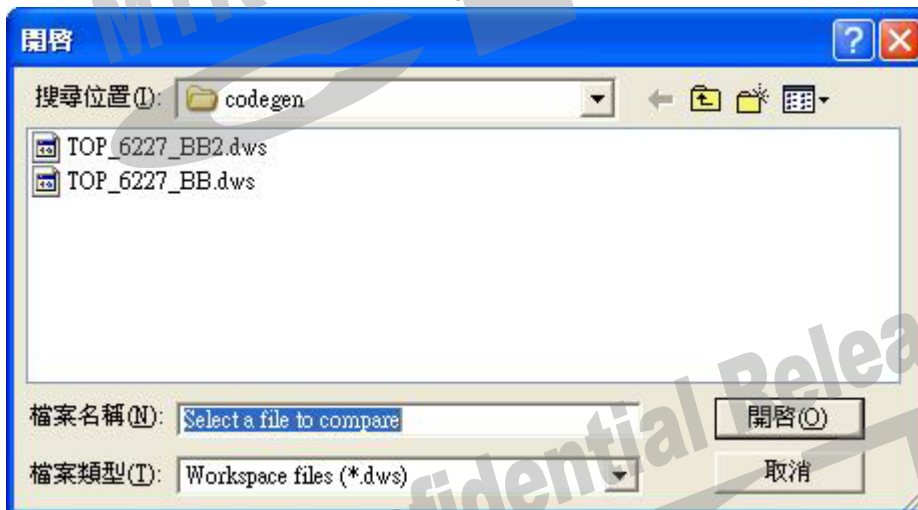
When the Edit button is pressed, an editing window will appear. It includes 5 components' settings: GPIO, GPO, EINT, ADC and KEYPAD. The detailed descriptions for this window will be in section 2.2 to section 2.6.

Preliminary Information

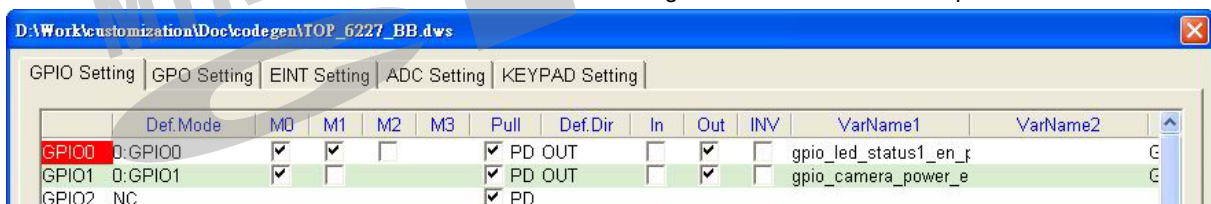


2.1.8 [CompareEdit]

When the **CompareEdit** button is pressed, a dialog box will show to indicate user to select another data workspace file in order to do the comparing between 2 different data workspace files.



After selecting another data workspace file, an editing window will appear just like pressing **Edit** button except that a red color will show on the head of the row in which the settings on these two data workspace are different.



Preliminary Information

2.1.9 [GenCode]

When the **GenCode** button is pressed, the corresponding source files and header files will be generated in the same folder as the data workspace file is in. What source files and header files will be generated is illustrated in section 2.2 to section 2.6.



Note that the tool also supports **command mode** to generate the source files and header files like below:

```
DrvGen.exe .\path\workspace.dws
```

For example:

```
D:\Work\customization\Doc\Drv_Tool>DrvGen.exe ..\codegen\TOP_6227_BB.dws
```

2.2 Component: GPIO (General Purpose IO)/GPO(General Purpose Output)

2.2.1 In MT6xxx.fig

For each GPIO/GPO pin of the corresponding chipset type MT6xxx, it describes the mode names for each four mode. Also, it describes each pin's pull-up(PU)/pull-down(PD)/high-impedance(Z) for GPIO but not for GPO pins. For example:

[GPIO]

```
GPIO0 = Mode0(GPIO0) Mode1(DICL) Mode2(DSP_GPO3) Mode3() PD
GPIO1 = Mode0(GPIO1) Mode1(BSI_RFIN) Mode2() Mode3() PD
GPIO2 = Mode0(GPIO2) Mode1(DID) Mode2() Mode3() PU
GPIO3 = Mode0(GPIO3) Mode1(DIMS) Mode2() Mode3() PD
GPIO4 = Mode0(GPIO4) Mode1(DSP_CLK) Mode2(DSPCLK) Mode3(EDICK) Z
```

[GPO]

```
GPO0 = Mode0(GPO0) Mode1(SRCLKENA) Mode2() Mode3()
GPO1 = Mode0(GPO1) Mode1(SRCLKENAN) Mode2() Mode3()
GPO2 = Mode0(GPO2) Mode1(EPDN#) Mode2(6.5MHz) Mode3(26MHz)
```

Inside the parenthesis (), it's the mode name for the corresponding mode (mode0~mode3). If there's nothing in the parenthesis, then it means this mode for the pin is reserved.

2.2.2 In GPIO.cmp

GPIO.cmp will be loaded into the Tool for editing GPIO settings. It contains two parts: **1. GPIO variables** and **2. Header and tailer** of the generated GPIO files.

Preliminary Information

2.2.2.1 GPIO variables

This file lists the entire software GPIO variables that may be assigned a GPIO port number. So these GPIO variables should be synchronized with the latest codes and can be added or removed (not recommended because of backward compatibility).

For example:

```
[GPIO_variables]
; gpio_bt_reset_pin is for bluetooth reset
gpio_bt_reset_pin
; gpio_bt_dsc_pin is for bluetooth disconnection
gpio_bt_dsc_pin
; gpio_bt_power_pin is for bluetooth power on
gpio_bt_power_pin
; gpio_bt_dataselect_pin is for bluetooth data selection
gpio_bt_dataselect_pin
; gpio_rf_control1_pin is for RF control 1
gpio_rf_control1_pin
; gpio_rf_control2_pin is for RF control 2
gpio_rf_control2_pin
; gpio_rf_control3_pin is for RF control 3
gpio_rf_control3_pin
; gpio_bmt_chr_ctrl_pin is for controlling charging circuit
gpio_bmt_chr_ctrl_pin
; gpio_usb_enable_pin is for USB power enable
gpio_usb_enable_pin
```

Note that the semicolon (;) is used for comments.

2.2.2.2 Header and Trailer of the generated GPIO files

The tool will generate two files for GPIO as `gpio_var.c` and `gpio_drv.h` which are introduced in section 2.2.5. And in `GPIO.cmp`, the header and trailer of these two files will be described for Tool to generate with the setting data.

For example:

```
[gpio_drv.h_HEADER]
#ifndef _GPIO_DRV_H
#define _GPIO_DRV_H

[gpio_drv.h_TAILER]
#endif /* _GPIO_DRV_H */

[gpio_var.c_HEADER]
#ifdef __CUST_NEW__
#include "gpio_drv.h"

[gpio_var.c_TAILER]
#endif /* __CUST_NEW__ */
```

2.2.3 GPIO Editing Window

The GPIO editing window contains five categories: **GPIO mode settings**, **GPIO pull-up/pull down enable**, **GPIO direction settings**, **GPIO data inversion enable** and **GPIO variables assignment** as below. Note that it will list all the GPIO pins for the chipset selected in the Main Menu window (section 2.1.6). Actually the Tool loads the GPIO configuration from the corresponding MT6xxx.fig file.

Preliminary Information

D:\Work\customization\Doc\codegen\TOP_6227_BB.dws

GPIO Setting | GPO Setting | EINT Setting | ADC Setting | KEYPAD Setting

	Def.Mode	M0	M1	M2	M3	Pull	Def.Dir	In	Out	INV	VarName1	VarName2
GPIO0	0:GPIO0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PD	OUT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	gpio_led_status1_en_pin	
GPIO1	0:GPIO1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PD	OUT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	gpio_camera_power_en_pin	
GPIO2	NC	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PD		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
GPIO3	0:GPIO3	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PD	OUT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	gpio_bt_dataselect_pin	
GPIO4	0:GPIO4	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PD	OUT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	gpio_bt_dsc_pin	
GPIO5	0:GPIO5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PD	OUT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	gpio_led_mainbl_en_pin	
GPIO6	0:GPIO6	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PD	OUT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	gpio_flashlight_en_pin	
GPIO7	0:GPIO7	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PD	OUT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	gpio_vibrator_en_pin	
GPIO8	0:GPIO8	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PD	OUT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	gpio_sccb_serial_clk_pin	
GPIO9	0:GPIO9	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PD	OUT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	gpio_sccb_serial_data_pin	
GPIO10	0:GPIO10	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PD	OUT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
GPIO11	NC	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PD		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
GPIO12	0:GPIO12	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PD	OUT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	gpio_bt_power_pin	
GPIO13	0:GPIO13	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PD	OUT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	gpio_rf_control1_pin	
GPIO14	1:MCINS	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PU		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
GPIO15	1:MCWP	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PU		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
GPIO16	0:GPIO16	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PU	OUT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	gpio_tp_spi_clk_pin	
GPIO17	0:GPIO17	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PU	N	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	gpio_tp_spi_busy_pin	
GPIO18	0:GPIO18	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PU	N	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	gpio_tp_spi_dout_pin	
GPIO19	0:GPIO19	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PU	OUT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	gpio_tp_spi_cs_pin	
GPIO20	NC	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PU		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
GPIO21	1:PWM1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PD		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
GPIO22	NC	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PD		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

確定 取消

GPIO Mode settings

GPIO Pull-up/Pull down enable

GPIO Direction settings

GPIO data inversion enable

GPIO variables assignment

2.2.3.1 GPIO Mode Settings

The first column of this category is to set the **DEFAULT MODE** for each GPIO pin. It's a selection menu include 4 mode with the mode names plus NC. Note that when the mode name for one mode is blank, it means this mode is for reserved and can't be selected. And the **NC** means this GPIO pin is not used.

GPIO Setting | GPO Setting | E

	Def.Mode
GPIO0	0:GPIO0
GPIO1	0:GPIO0
GPIO2	1:DICL
GPIO3	2:DSP_GPO3
GPIO4	3:
GPIO5	NC
GPIO6	0:GPIO6
GPIO7	0:GPIO7

The **M0**, **M1**, **M2**, **M3** columns means the allowed mode (Mode 0, Mode 1, Mode2, Mode3) for the corresponding GPIO pin. Note that:

1. Only the modes that are not for reserved will show the checkboxes.
2. If **DEFAULT MODE** is Mode 0, then the check box for M0 is mandatory checked. The same as Mode 1, 2 and 3.

Preliminary Information

3. If DEFAULT MODE is NC, then there will be no checkbox.

GPIO Setting		GPO Setting		EINT Setting		ADC Setting	
	Def.Mode	M0	M1	M2	M3		
GPIO0	0:GPIO0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
GPIO1	0:GPIO1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
GPIO2	NC						
GPIO3	1:DIMS	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
GPIO4	1:DSP_CLK	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

2.2.3.2 GPIO pull-up/pull-down enable

For this category, it is just a checkbox to enable or disable pull-up(PU)/pull-down(PD)/high-impedance(Z) for the corresponding GPIO pin.

2.2.3.3 GPIO direction settings

This category contains three columns and they are only enabled when the allowed mode (M0, M1, M2 or M3) for the corresponding GPIO is checked. For example, assume Mode 0 of GPIO0 is GPIO mode and if M0 is checked, then GPIO direction settings of GPIO0 will appear for user to manipulate.

GPIO Setting | GPO Setting | EINT Setting | ADC Setting | KEYPAD Setting |

	Def.Mode	M0	M1	M2	M3	Pull	Def.Dir	In	Out	INV	VarName1
GPIO0	0:GPIO0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PD	OUT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	gpio_led_status1_en_pin
GPIO1	1:BSI_RFIN	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PD		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
GPIO2	NC	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> PD		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

1. "Def. Dir": A selectable menu to set the default direction (OUT: output, IN: input) for the corresponding GPIO pin.



2. "In": A checkbox. When it's checked, it means that input direction (GPI) is allowed for the corresponding GPIO pin. Note that if default direction for the GPIO pin is IN, then this checkbox is mandatory checked.
3. "Out": A checkbox. When it's checked, it means that output direction (GPO) is allowed for the corresponding GPIO pin. Note that if default direction for the GPIO pin is OUT, then this checkbox is mandatory checked.

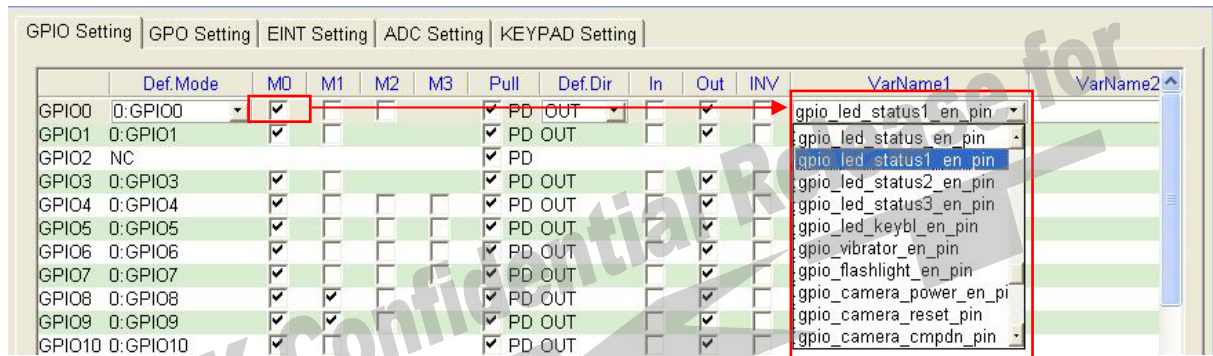
2.2.3.4 GPIO data inversion enable

It's a checkbox and is showed when the default mode for the corresponding GPIO pin is not NC. When it's check, it means to enable the data inversion for the GPIO pin. Note that it only takes effect on input signal.

2.2.3.5 GPIO variable assignment

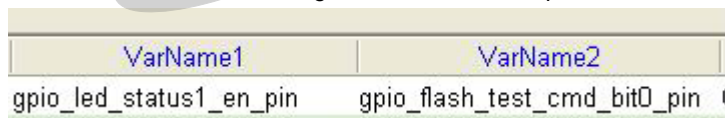
It contains two columns: VarName1 and VarName2 and is enabled only when the allowed mode (M0, M1, M2 or M3) for the corresponding GPIO pin is checked.

Preliminary Information

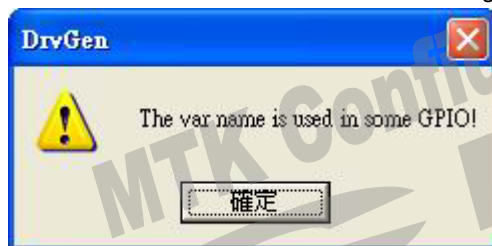


VarName1 or VarName2 is a selectable menu containing the whole software GPIO variable names defined in GPIO.cmp. User can select one of the GPIO variable names in order to assign GPIO port number to it.

In general, only VarName1 will be assigned the GPIO variable name but also can be left blank. VarName2 is for special cases when a GPIO pin is shared between two applications or modules. Therefore there will be two different GPIO variables assigned the same GPIO port number.



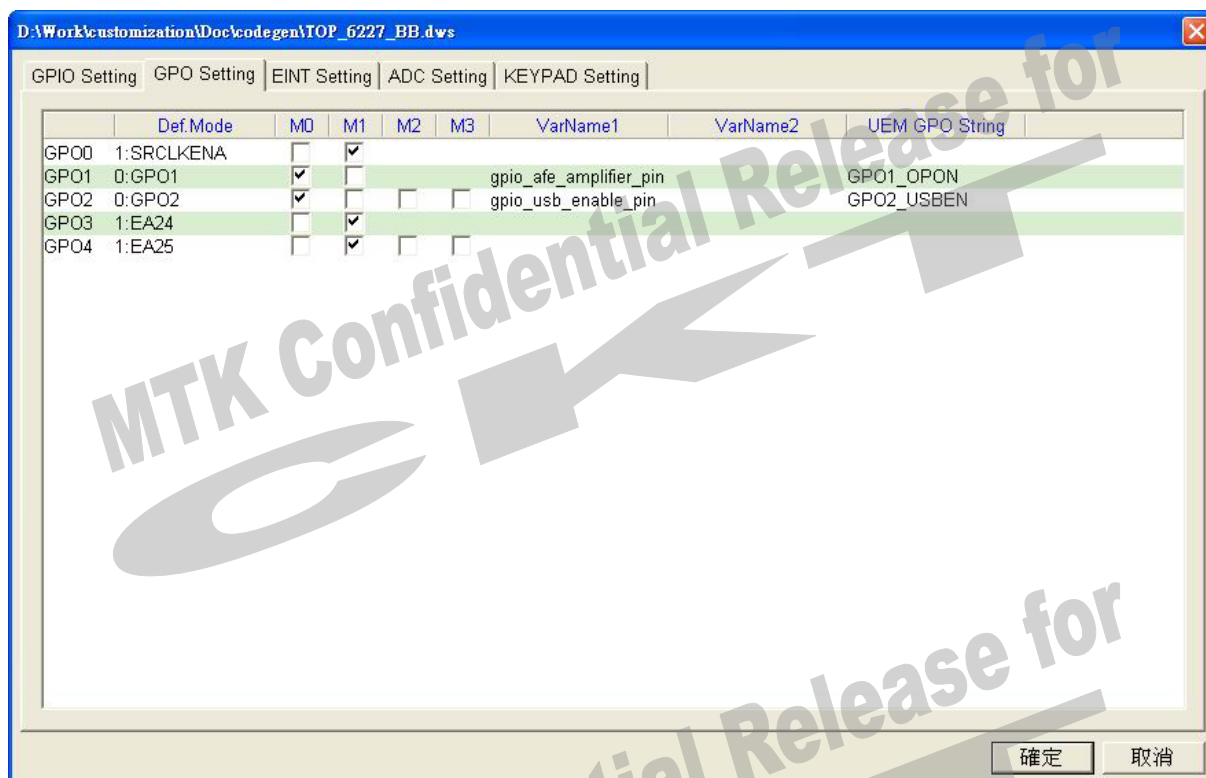
Note that no GPIO variable name can be assigned in any two different fields even in GPO Setting (GPIO variables names are shared with GPO Setting), otherwise there will be an error pop-up message as below:



2.2.4 GPO Editing Window

The GPO editing window is just like GPIO editing window except that it only contains two categories: **GPO mode settings** and **GPIO variables assignment** as below. Note that it will list all the GPO pins for the chipset selected in the Main Menu window (section 2.1.6). Actually the Tool loads the GPO configuration from the corresponding MT6xxx.fig file.

About the manipulation, please refer to section 2.2.3.1 GPIO Mode Settings and section 2.2.3.5 GPIO variable assignment.



Note that the column "UEM GPO String" is for UEM component and will be illustrated in section 2.2.6.

2.2.5 Generated Source and Header files

2.2.5.1 gpio_drv.h

- Header defined in gpio.cmp:

```
#ifndef _GPIO_DRV_H
#define _GPIO_DRV_H
```

- Defined the default mode for each GPIO port:

```
#define GPIO_PORT0_MODE    MODE_0
#define GPIO_PORT1_MODE    MODE_1
#define GPIO_PORT2_MODE    MODE_NC
#define GPIO_PORT3_MODE    MODE_0
#define GPIO_PORT4_MODE    MODE_0
.....
```

- Define the port values for those GPIOs which are permitted to set as GPIO mode:

```
#define GPIO_PORT_0    (0|0x80)
#define GPIO_PORT_1    (1|0x80)
#define GPIO_PORT_3    (3|0x80)
#define GPIO_PORT_4    (4|0x80)
#define GPIO_PORT_5    (5|0x80)
.....
```

Note these values have been "OR" with a magic value "0x80". This magic value is to be used in GPIO driver for validating operations.

- Define the permission for each mode (mode0 ~ mode3) of each GPIO port. Value "1" means permitted and value "0" means not permitted:

```
#define MODE0_GPIO0    1
#define MODE1_GPIO0    0
#define MODE2_GPIO0    0
```


Preliminary Information

```
#define MODE3_GPIO0    0
#define MODE0_GPIO1    1
#define MODE1_GPIO1    0
#define MODE2_GPIO1    0
#define MODE3_GPIO1    0
#define MODE0_GPIO2    0
#define MODE1_GPIO2    0
#define MODE2_GPIO2    0
#define MODE3_GPIO2    0
#define MODE0_GPIO3    1
.....
```

5. Define enable or disable the PULL-UP/PULL-DOWN for each GPIO port:

```
#define GPIO_PORT0_PULL    PULL_ENABLE
#define GPIO_PORT1_PULL    PULL_ENABLE
#define GPIO_PORT2_PULL    PULL_ENABLE
#define GPIO_PORT3_PULL    PULL_DISABLE
#define GPIO_PORT4_PULL    PULL_ENABLE
#define GPIO_PORT5_PULL    PULL_DISABLE
#define GPIO_PORT6_PULL    PULL_ENABLE
.....
```

6. Define the default value of GPIO direction for each port:

```
#define GPIO_PORT0_DIR    DIR_OUTPUT
#define GPIO_PORT1_DIR    DIR_OUTPUT
#define GPIO_PORT2_DIR    DIR_NULL
#define GPIO_PORT3_DIR    DIR_OUTPUT
#define GPIO_PORT4_DIR    DIR_OUTPUT
#define GPIO_PORT5_DIR    DIR_INPUT
.....
```

7. Define the permission for each direction (input, output) of each GPIO port. Value "1" means permitted and value "0" means not permitted:

```
#define DIR_IN_GPIO0      0
#define DIR_OUT_GPIO0     1
#define DIR_IN_GPIO1      0
#define DIR_OUT_GPIO1     1
#define DIR_IN_GPIO2      0
#define DIR_OUT_GPIO2     0
#define DIR_IN_GPIO3      0
#define DIR_OUT_GPIO3     1
#define DIR_IN_GPIO4      1
.....
```

8. Define the enable or disable data inversion for each GPIO port:

```
#define GPIO_PORT0_INV    INV_DISABLE
#define GPIO_PORT1_INV    INV_DISABLE
#define GPIO_PORT2_INV    INV_NULL
#define GPIO_PORT3_INV    INV_DISABLE
#define GPIO_PORT4_INV    INV_ENABLE
#define GPIO_PORT5_INV    INV_DISABLE
.....
```

9. Defined the default mode for each GPO port:

```
#define GPO_PORT0_MODE    MODE_1
#define GPO_PORT1_MODE    MODE_0
#define GPO_PORT2_MODE    MODE_0
#define GPO_PORT3_MODE    MODE_1
```

Preliminary Information

```
#define GPO_PORT4_MODE    MODE_1
```

10. Define the permission for each mode (mode0 ~ mode3) of each GPO port. Value "1" means permitted and value "0" means not permitted:

```
#define MODE0_GPO0    0
#define MODE1_GPO0    1
#define MODE2_GPO0    0
#define MODE3_GPO0    0
#define MODE0_GPO1    1
#define MODE1_GPO1    0
#define MODE2_GPO1    0
.....
```

11. Define the port values for those GPIOs which are permitted to set as GPO mode:

```
#define GPO_PORT_1    (1|0x70)
#define GPO_PORT_2    (2|0x70)
.....
```

Note these values have been "OR" with a magic value "0x70". This magic value is to be used in GPIO driver for validating operations.

12. Tailor defined in gpio.cmp:

```
#endif /* _GPIO_DRV_H */
```

2.2.5.2 gpio_var.c

1. Header defined in gpio.cmp:

```
#ifndef __CUST_NEW__
#include "gpio_drv.h"
```

2. Declare all the GPIO variables that are previously selected on the Tool and assign the port values:

```
const char gpio_led_status1_en_pin = GPIO_PORT_0;
const char gpio_camera_power_en_pin = GPIO_PORT_1;
const char gpio_bt_dataselect_pin = GPIO_PORT_3;
const char gpio_bt_dsc_pin = GPIO_PORT_4;
const char gpio_led_mainbl_en_pin = GPIO_PORT_5;
.....
```

3. Tailor defined in gpio.cmp:

```
#endif /* __CUST_NEW__ */
```

2.2.6 Rules for GPIO

1. All GPIO pin assignments and setting are done in the Tool side.
2. When a new application or module needs to use GPIOs, new GPIO variables must be added into gpio.cmp file. Thus these variables can be assigned GPIO pin numbers on Tool side.
3. If the mode or the direction for a specific GPIO won't change dynamically in runtime, user shall not call `GPIO_ModeSetup()` or `GPIO_InitIO()` in modules or tasks because the mode and the direction have been configured well in custom file gpio_setting.c.
4. For GPIO API fuctions:

4.1 `GPIO_InitIO(char direction, char port)`: The `port` MUST be an extern variable that is declared in gpio_var.c (assigned by tool). Otherwise it will be an invalid operation when debug mode is on.

4.2 `GPIO_WriteIO(char data, char port)`: The `port` MUST be an extern variable that is declared in gpio_var.c (assigned by tool). Otherwise it will be an invalid operation when debug mode is on. Also note that this function should be called when the GPIO port is configured as GPO under GPIO mode or it's invalid too. In other words, the following example may cause an error:

```
void Function(void)
{
...
}
```

Preliminary Information

```

GPIO_WriteIO(1, gpio_led_mainbl_en_pin);
GPIO_InitIO(1, gpio_led_mainbl_en_pin);
GPIO_ModeSetup(gpio_led_mainbl_en_pin, 0);
...
}

```

The function should be altered as:

```

void Function(void)
{
...
GPIO_InitIO(1, gpio_led_mainbl_en_pin);
GPIO_ModeSetup(gpio_led_mainbl_en_pin, 0);
GPIO_WriteIO(1, gpio_led_mainbl_en_pin);
...
}

```

4.3 **GPIO_ReadIO(char port)**: The **port** MUST be an extern variable that is declared in **gpio_var.c** (assigned by tool). Otherwise it will be an invalid operation when debug mode is on. Also note that this function should be called when the GPIO port is configured as GPI under GPIO mode or it's invalid too.

4.4 **GPIO_ModeSetup(kal_uint16 pin, kal_uint16 conf_dada)**: The **pin** can be a normal constant value. However when the debug mode is on, it will check if the mode is permitted to be set to or not.

4.5 There is no restriction for the usage of other GPIO API functions.

5. It is prohibited to directly access GPIO related registers.

2.3 Component: EINT (External Interrupt)

2.3.1 In MT6xxx.fig

For the corresponding chipset type MT6xxx, it describes the total external interrupt pin count and the total count of external interrupt pins that use software debounce time delay.

```

[EINT]
EINT_COUNT = 8
EINT_DEBOUNCE_TIME_COUNT = 4

```

2.3.2 In EINT.cmp

EINT.cmp will be loaded into the Tool for editing EINT settings. It contains two parts: 1. **EINT variables** and 2. **Header and tailer** of the generated EINT files.

2.3.2.1 EINT variables

This file lists the entire EINT variables that may be assigned an EINT port number. So these EINT variables should be synchronized with the latest codes and can be added or removed (not recommended because of backward compatibility).

For example:

```

[EINT_variables]
; AUX_EINT_NO is for external device interrupt
AUX_EINT_NO
; TOUCH_PANEL_EINT_NO is for touch panel interrupt
TOUCH_PANEL_EINT_NO
; CHRDET_EINT_NO is for charger detection interrupt
CHRDET_EINT_NO
; BT_EINT_NO is for Bluetooth interrupt
BT_EINT_NO
; CLAMDET_EINT_NO is for clam shell interrupt
CLAMDET_EINT_NO

```

Note that the semicolon (;) is used for comments.

2.3.2.2 Header and Tailer of the generated EINT files

The tool will generate two files for EINT as `eint_var.c` and `eint_drv.h` which are introduced in section 2.3.4. And in `EINT.cmp`, the header and tailer of these two files will be described for Tool to generate with the setting data.

For example:

```
[eint_drv.h_HEADER]
#ifndef _EINT_DRV_H
#define _EINT_DRV_H

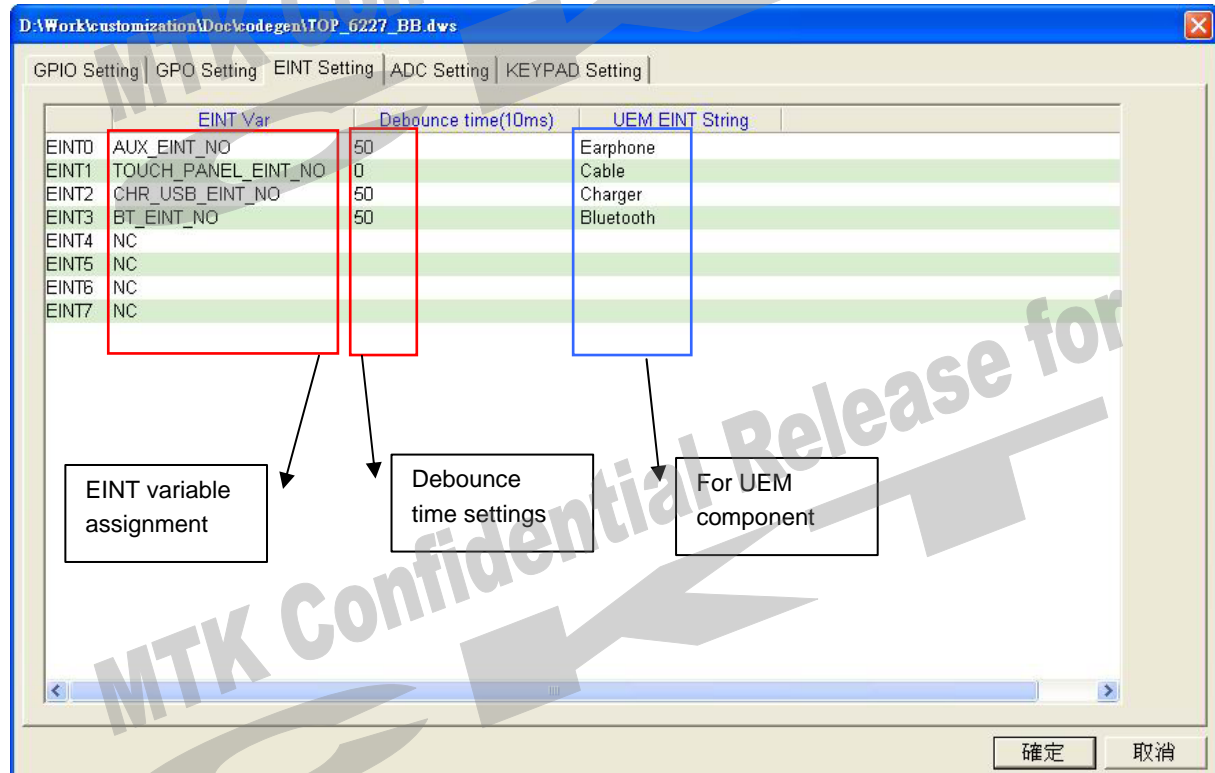
[eint_drv.h_TAILER]
#endif /* _EINT_DRV_H */

[eint_var.c_HEADER]
#ifdef __CUST_NEW__
#include "eint.h"

[eint_var.c_TAILER]
#endif /* __CUST_NEW__ */
```

2.3.3 EINT Editing Window

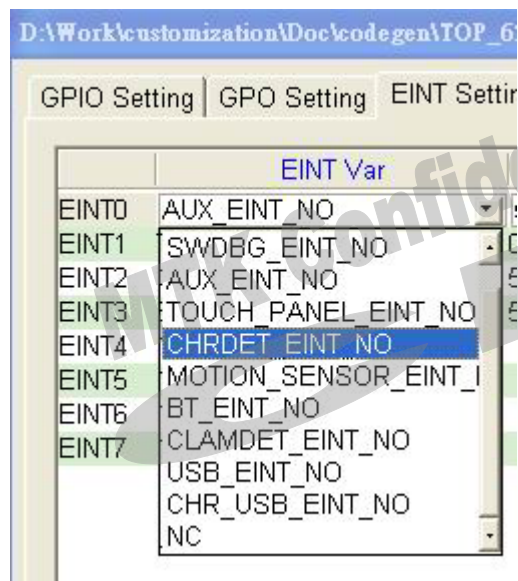
The EINT editing window contains two categories: **EINT variables assignment** and **Software ebounce Time settings** as below. Note that it will list all the EINT pins for the chipset selected in the Main Menu window (section 2.1.6). Actually the Tool loads the EINT configuration from the corresponding MT6xxx.fig file.



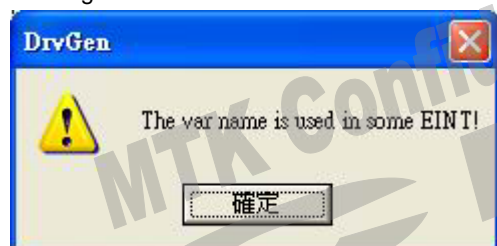
Preliminary Information

2.3.3.1 EINT variable assignment

This column is a selectable menu containing the whole software EINT variable names defined in EINT.cmp. User can select one of the EINT variable names in order to assign EINT pin number to it.

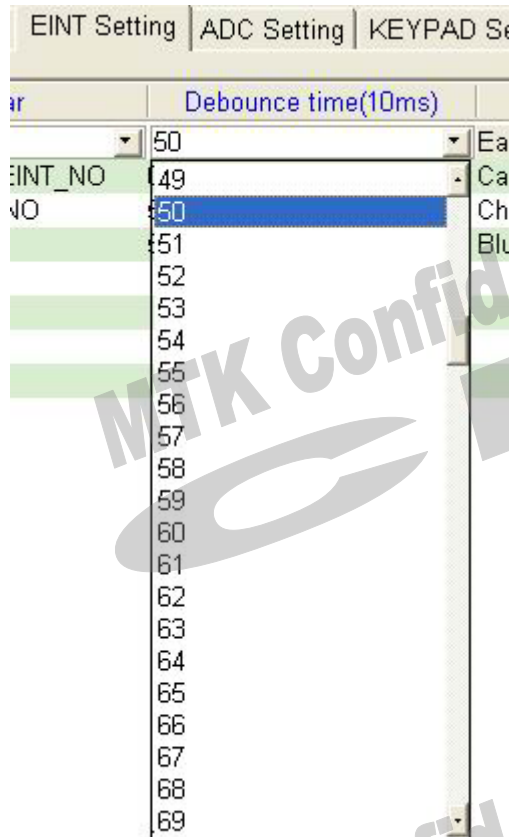


Note that no EINT variable name can be assigned in two different fields, otherwise there will be an error pop-up message as below:



2.3.3.2 Software Debounce Time Settings

This column is a selectable menu containing value 0 to 255. Note the unit is 10ms. So if user selects 10, it represents the software debounce time for this EINT pin is 100ms. Also note that not all of the EINT pin can be set for software debounce time. If the value `EINT_DEBOUNCE_TIME_COUNT` in `MT6xxx.fig` is 4, then only EINT0 to EINT3 can be set for software debounce time.



2.3.4 Generated Source and Header files

2.3.4.1 eint_drv.h

1. Header defined in EINT.cmp:

```
#ifndef __EINT_DRV_H
#define __EINT_DRV_H
```

2. Define the software debounce time for each EINT pin. Note that the unit is 10ms:

```
#define EINT0_DEBOUNCE_TIME_DELAY 50
#define EINT1_DEBOUNCE_TIME_DELAY 0
#define EINT2_DEBOUNCE_TIME_DELAY 50
#define EINT3_DEBOUNCE_TIME_DELAY 50
.....
```

3. Tailer defined in gpio.cmp:

```
#endif /* __CUST_NEW__ */
```

2.3.4.2 eint_var.c

1. Header defined in EINT.cmp:

```
#ifdef __CUST_NEW__
#include "eint.h"
```

2. Declare all the EINT variables that are previously selected on the Tool and assign the pin values:

```
const unsigned char AUX_EINT_NO = 0;
const unsigned char TOUCH_PANEL_EINT_NO = 1;
const unsigned char CHR_USB_EINT_NO = 2;
```


Preliminary Information

```
const unsigned char BT_EINT_NO = 3;
const unsigned char SWDBG_EINT_NO = EINT_CHANNEL_NOT_EXIST;
const unsigned char CHRDET_EINT_NO = EINT_CHANNEL_NOT_EXIST;
const unsigned char MOTION_SENSOR_EINT_NO = EINT_CHANNEL_NOT_EXIST;
const unsigned char CLAMDET_EINT_NO = EINT_CHANNEL_NOT_EXIST;
const unsigned char USB_EINT_NO = EINT_CHANNEL_NOT_EXIST;
.....
```

Note that for those EINT variables which haven't been selected on the Tool side, they will be assigned values by EINT_CHANNEL_NOT_EXIST.

3. Tailor defined in EINT.cmp:

```
#endif /* __CUST_NEW__ */
```

2.3.5 Rules for EINT

1. All EINT pin assignments and debounce time setting are done in the Tool side.
2. When a new application or module needs to use EINT, new EINT variables must be added into EINT.cmp file. Thus these variables can be assigned EINT pin numbers on Tool side.
3. Whoever wants to use EINT must extern the variables declared in EINT_drv.c or call the custom function `kal_uint8 custom_eint_get_channel(eint_channel_type type)` to get it. And then call EINT APIs based on these variables.

2.4 Component: ADC (Analog to Digital Converter)

2.4.1 In MT6xxx.fig

For the corresponding chipset type MT6xxx, it describes the total ADC channel count.

```
[ADC]
```

```
ADC_COUNT = 7
```

2.4.2 In ADC.cmp

ADC.cmp will be loaded into the Tool for editing ADC settings. It contains two parts: 1. EINT variables and 2. Header and tailer of the generated ADC file.

2.4.2.1 ADC variables

This file lists the entire ADC variables that may be assigned an ADC channel number. So these ADC variables should be synchronized with the latest codes and can be added or removed (not recommended because of backward compatibility).

For example:

```
[ADC_variables]
; ADC_VBAT is to measure the battery voltage.
ADC_VBAT
; ADC_VISENSE is to measure the charging current.
ADC_VISENSE
; ADC_VBATTMP is to measure the battery temperature.
ADC_VBATTMP
; ADC_VCHARGER is to measure the charger voltage.
ADC_VCHARGER
; ADC_ACCESSORYID is to measure the plug-in accessory voltage.
ADC_ACCESSORYID
```

Preliminary Information

; ADC_CHR_USB is to measure the cable voltage in order to recognize Charger or USB.
ADC_CHR_USB

Note that the semicolon (;) is used for comments.

2.4.2.2 Header and Tailer of the generated ADC file

The tool will generate one file for ADC as `adc_var.c` which is introduced in section 2.4.4. And in `ADC.cmp`, the header and tailer of this file will be described for Tool to generate with the setting data.

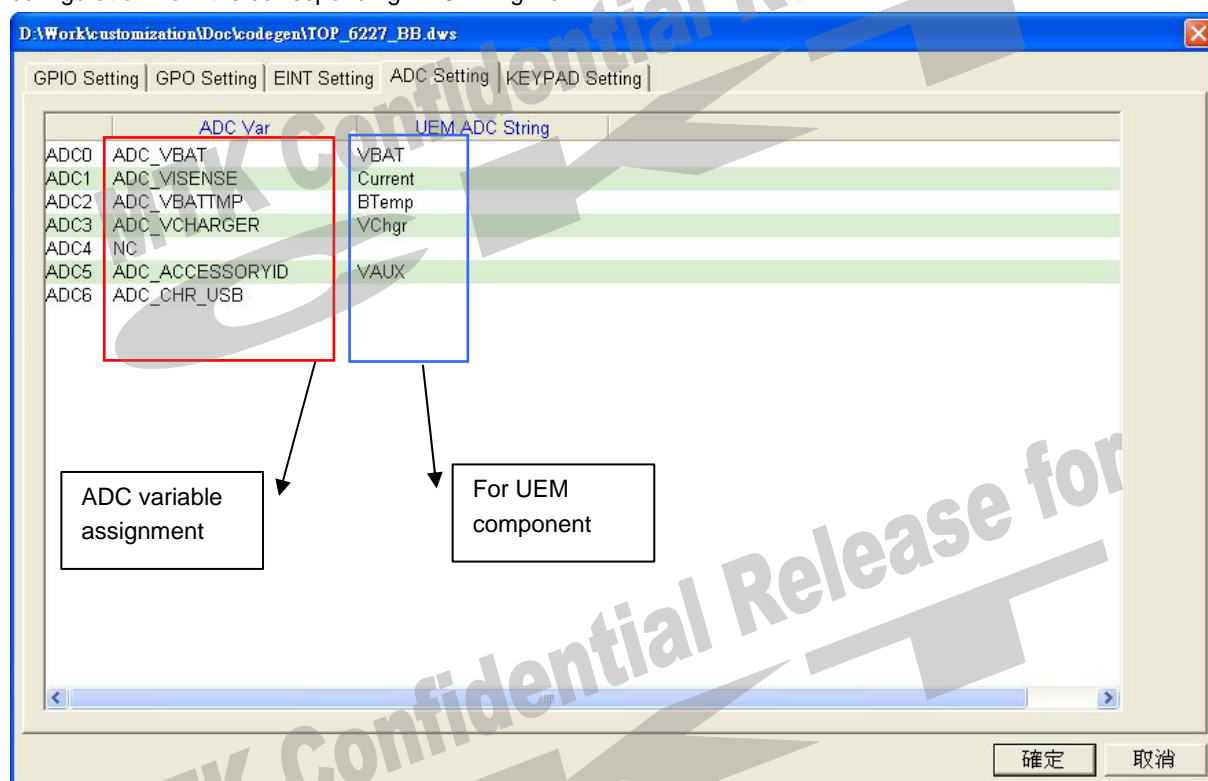
For example:

```
[adc_var.c_HEADER]
#ifdef __CUST_NEW__
#include "kal_release.h"
#include "adc.h"

[adc_var.c_TAILER]
#endif /* __CUST_NEW__ */
```

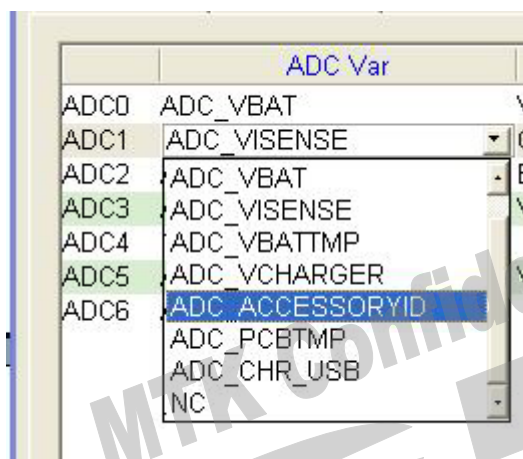
2.4.3 ADC Editing Window

The ADC editing window contains just one category: **ADC variables assignment** as below. Note that it will list all the ADC pins for the chipset selected in the Main Menu window (section 2.1.6). Actually the Tool loads the ADC configuration from the corresponding MT6xxx.fig file.

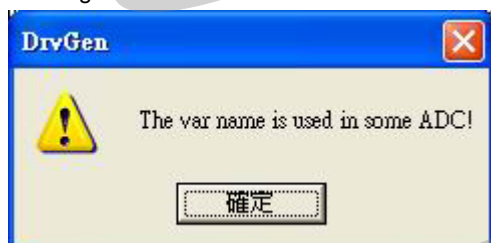


2.4.3.1 ADC variables assignment

This column is a selectable menu containing the whole software ADC variable names defined in `ADC.cmp`. User can select one of the ADC variable names in order to assign ADC channel number to it.



Note that no ADC variable name can be assigned in two different fields, otherwise there will be an error pop-up message like below:



2.4.4 Generated Source and Header files

2.4.4.1 adc_var.c

1. Header defined in ADC.cmp:

```
#ifdef __CUST_NEW__
#include "kal_release.h"
#include "adc.h"
```

2. Declare all the ADC variables that are previously selected on the Tool and assign the pin values:

```
const unsigned char ADC_VBAT = 0;
const unsigned char ADC_VISENSE = 1;
const unsigned char ADC_VBATTMP = 2;
const unsigned char ADC_VCHARGER = 3;
const unsigned char ADC_ACCESSORYID = 5;
const unsigned char ADC_CHR_USB = 6;
const unsigned char ADC_PCBTMP = ADC_ERR_CHANNEL_NO;
.....
```

Note that for those ADC variables which haven't been selected on the Tool side, they will be assigned values by ADC_ERR_CHANNEL_NO.

3. Tailor defined in ADC.cmp:

```
#endif /* __CUST_NEW__ */
```

2.4.5 Rules for ADC

1. All ADC channel assignments are done in the Tool side.

Preliminary Information

2. When a new channel needs to use ADC, new ADC variables must be added into ADC.cmp file. Thus these variables can be assigned to ADC channels on Tool side.
3. Whoever wants to measure ADC channel must extern the variables declared in ADC_drv.c or call the custom function `kal_uint8 custom_adc_get_channel (adc_channel_type type)` to get it. And then call ADC APIs based on these variables.

2.5 Component: KEYPAD

2.5.1 In MT6xxx.fig

For the corresponding chipset type MT6xxx, it describes the HW keypad matrix size.

```
[Keypad]
KEY_ROW = 6
KEY_COLUMN = 7
```

2.5.2 In KEYPAD.cmp

KEYPAD.cmp will be loaded into the Tool for editing KEYPAD settings. It contains two parts: 1. Logical key definition and 2. Header and tailer of the generated KEYPAD file.

2.5.2.1 Logical Key definition

This file lists the entire logical key definitions that may be assigned to a certain position on the keypad mapping matrix. So these KEYPAD definition should be synchronized with the latest codes.

For example:

```
[Key_definition]
DEVICE_KEY_0
DEVICE_KEY_1
DEVICE_KEY_2
DEVICE_KEY_3
DEVICE_KEY_4
DEVICE_KEY_5
DEVICE_KEY_6
DEVICE_KEY_7
DEVICE_KEY_8
DEVICE_KEY_9
DEVICE_KEY_STAR
DEVICE_KEY_HASH
DEVICE_KEY_VOL_UP
DEVICE_KEY_VOL_DOWN
DEVICE_KEY_UP
DEVICE_KEY_DOWN
DEVICE_KEY_LEFT
DEVICE_KEY_RIGHT
DEVICE_KEY_MENU
DEVICE_KEY_FUNCTION
DEVICE_KEY_SK_LEFT
DEVICE_KEY_SK_RIGHT
DEVICE_KEY_SEND
DEVICE_KEY_END
DEVICE_KEY_POWER
DEVICE_KEY_CLEAR
DEVICE_KEY_EXT_FUNC1
```

Preliminary Information

```

DEVICE_KEY_EXT_FUNC2
MAX_DEVICE_KEYS
DEVICE_KEY_NONE

```

2.5.2.2 Header and Tailer of the generated KEYPAD files

The tool will generate one file for KEYPAD as `keypad_drv.h` which is introduced in section 2.5.4. And in `KEYPAD.cmp`, the header and tailer of this file will be described for Tool to generate with the setting data. For example:

```

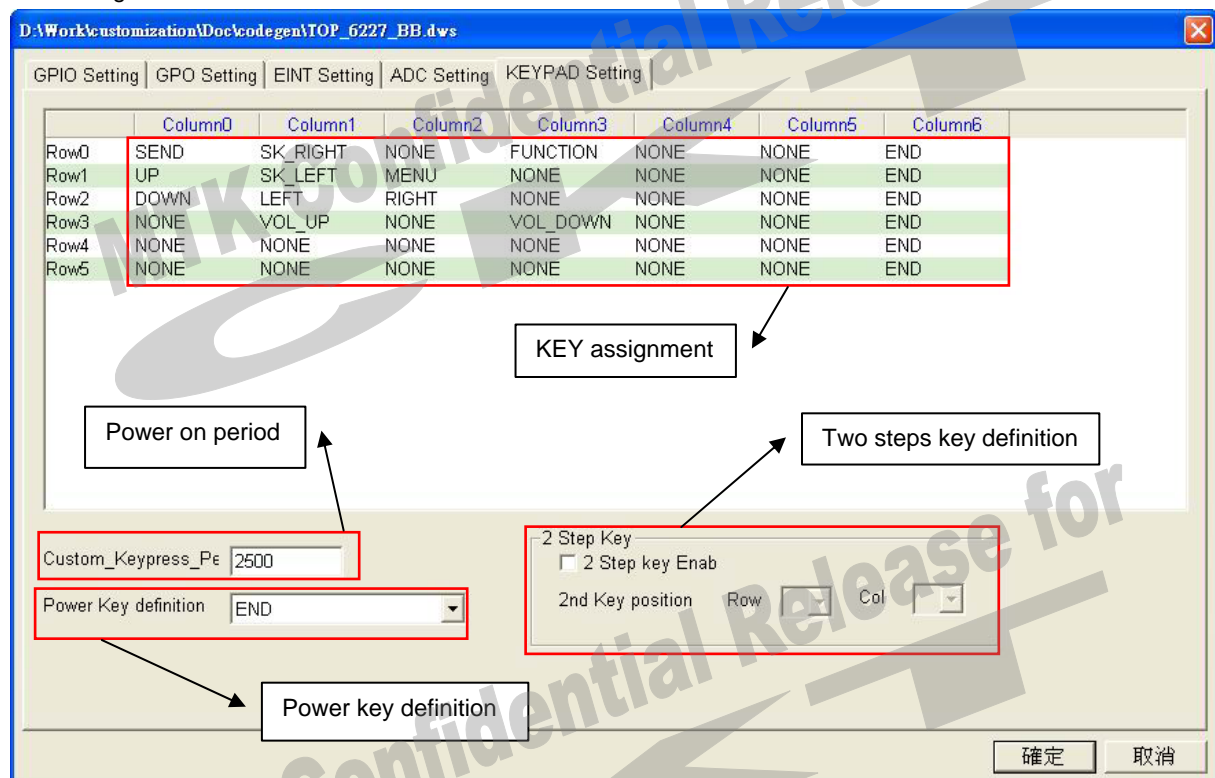
[keypad_drv.h_HEADER]
#ifndef _KEYPAD_DRV_H
#define _KEYPAD_DRV_H

[keypad_drv.h_TAILER]
#endif /* _KEYPAD_DRV_H */

```

2.5.3 KEYPAD Editing Window

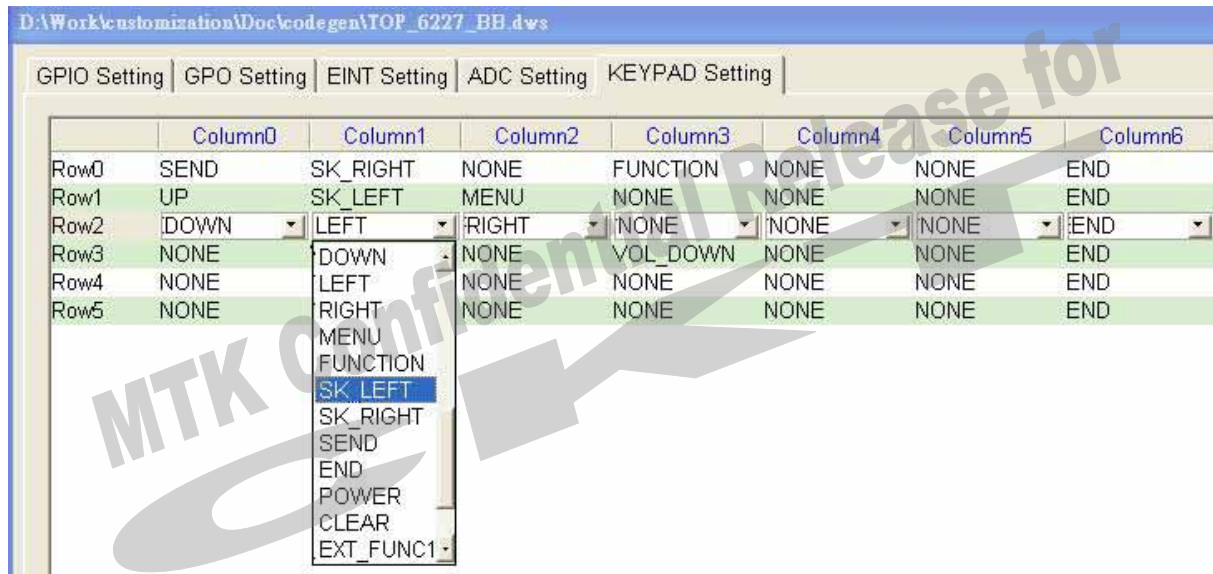
The KEYPAD editing window contains just four categories: **KEY assignment**, **Power on period**, **Power key definition** and **two steps key definition**. Note that it shows the HW keypad matrix layout for the chipset selected in the Main Menu window (section 2.1.6). Actually the Tool loads the KEYPAD configuration from the corresponding MT6xxx.fig file.



2.5.3.1 KEY assignment

These fields are selectable menu to assign a KEY defined in `KEYPAD.cmp` on each position of the KEYPAD matrix. Note that any KEY can be assigned on more than one position.

Preliminary Information



2.5.3.2 Power on period

This is a number key-in field for customized power on period (A period from power key pressed to the LCD on). The unit is ms.

Custom_Keypress_Period:

2.5.3.3 Power key definition

This field is a selectable menu to assign a KEY to be the power key.

Power Key definition:

- SK_RIGHT
- SEND
- END
- POWER
- CLEAR
- EXT_FUNC1
- EXT_FUNC2
- MAX_DEVICE_KEYS
- NONE

2.5.3.4 Two steps key definition

Two steps key is implemented by two key positions with the same key definition. One is the first key (half pressed key) and the other is the second key (full pressed key). This field is optional. It can be enabled or disabled.

2 Step Key: ☒ 2 Step key Enable

When this field is enabled, user can assign the full pressed position (2nd key position) on the keypad matrix for the two steps key.

Preliminary Information

2 Step Key

☒ 2 Step key Enab

2nd Key position Row Col

0
1
2
3
4
5

2 Step Key

☒ 2 Step key Enab

2nd Key position Row Col

0
1
2
3
4
5
6

DAWork\customization\Doc\codegen\TOP_6227_BB.dws

GPIO Setting | GPO Setting | EINT Setting | ADC Setting | KEYPAD Setting

	Column0	Column1	Column2	Column3	Column4	Column5	Column6
Row0	SEND	SK_RIGHT	FUNCTION	FUNCTION	NONE	NONE	END
Row1	UP	SK_LEFT	MENU	NONE	NONE	NONE	END
Row2	DOWN	LEFT	RIGHT	NONE	NONE	NONE	END
Row3	NONE	VOL_UP	NONE	VOL_DOWN	NONE	NONE	END
Row4	NONE	NONE	NONE	NONE	NONE	NONE	END
Row5	NONE	NONE	NONE	NONE	NONE	NONE	END

This position (with the same key definition as the 2nd key) implies it is the 1st key (half pressed key) of the two steps key.

2nd key (full pressed key) position and key definition.

Custom_Keypress_Pe

Power Key definition

2 Step Key

☒ 2 Step key Enab

2nd Key position Row Col

確定 取消

Note that the 1st key position (half pressed key position) is implied on the position which has the same key definition as the 2nd key. User should take care of having 1st key on the keypad matrix. The tool would not check it.

2.5.4 Generated Source and Header files

2.5.4.1 Keypad_drv.h

1. Header defined in KEYPAD.cmp:

```
#ifndef _KEYPAD_DRV_H
#define _KEYPAD_DRV_H
```

2. Macro of the keypad mapping:

```
#define KEYPAD_MAPPING \
DEVICE_KEY_SEND, \
DEVICE_KEY_SK_RIGHT, \
DEVICE_KEY_NONE, \
DEVICE_KEY_FUNCTION, \
DEVICE_KEY_NONE, \
DEVICE_KEY_NONE, \
DEVICE_KEY_END, \
DEVICE_KEY_UP, \
.....
```

3. Define the power on period:

```
#define KEY_PRESS_PERIOD 2500
```

4. Define the power key position:

```
#define POWERKEY_POSITION DEVICE_KEY_END
```

5. Define the 2-steps keys parameter (Optional):

```
#define SECOND_KEY_ROW 0
#define SECOND_KEY_COLUMN 3
#define TWO_STEP_KEY DEVICE_KEY_FUNCTION
```

6. Tailer defined in KEYPAD.cmp

```
#endif /* _KEYPAD_DRV_H*/
```

2.6 Component: UEM (User Equipment Management)

2.6.1 In MT6xxx.fig

None.

2.6.2 In UEM.cmp

UEM.cmp will be loaded into the Tool for editing UEM settings. It contains two parts: 1. Maximum maximum length and 2. Header and tailer of the generated UEM file.

2.6.2.1 Maximum string length

It defines the maximum length for UEM strings:

```
[UEM]
MAX_NETNAME_TEXT = 16
```

2.6.2.2 Header and Tailer of the generated UEM files

The tool will generate one file for UEM as `uem_drv.c` which is introduced in section 2.6.4. And in UEM.cmp, the header and tailer of this file will be described for Tool to generate with the setting data.

Preliminary Information

For example:

```
[uem_drv.c_HEADER]
#ifdef __CUST_NEW__
#include "kal_release.h"
#include "gpio_drv.h"
#include "custom_em.h"
#include "custom_equipment.h"

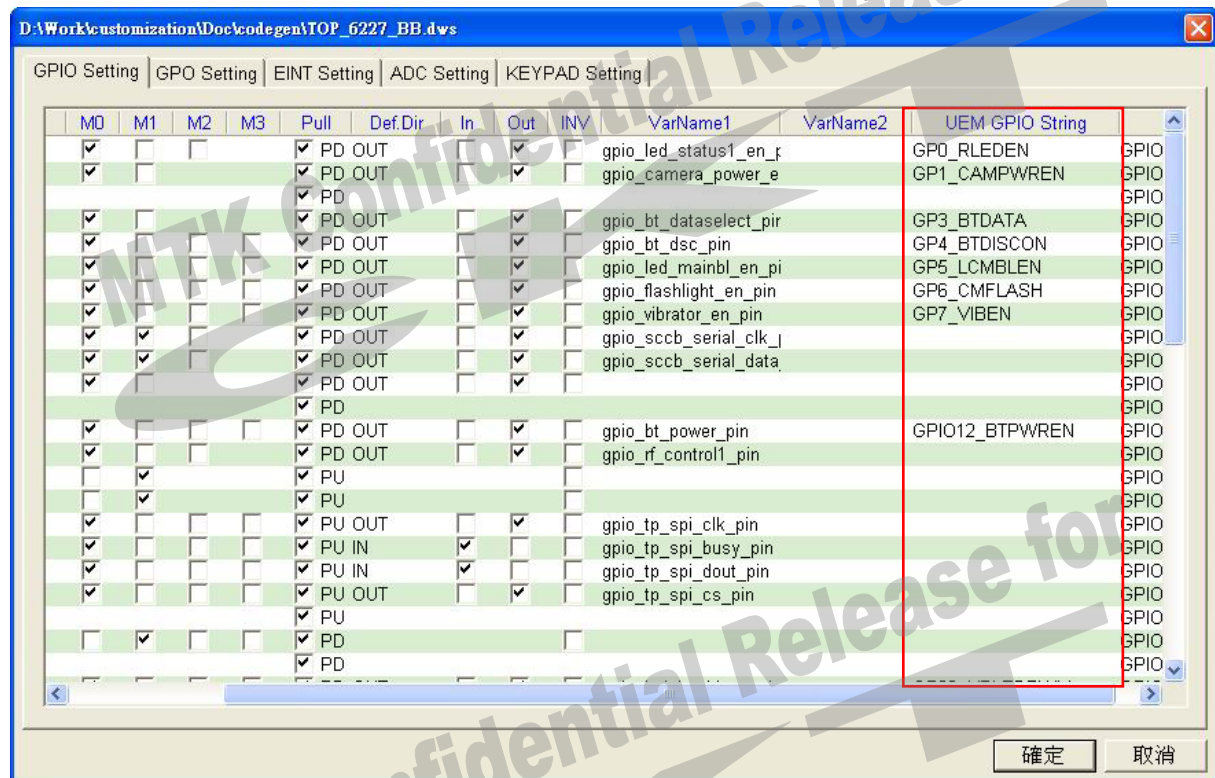
[uem_drv.c_TAILER]
#endif /* __CUST_NEW__ */
```

2.6.3 UEM Editing Window

The UEM editing windows contain four categories: **UEM GPIO String**, **UEM GPO String**, **UEM EINT String** and **UEM ADC String** as below. Note that these editing windows are attached to GPIO/GPO/EINT/ADC Editing Windows.

2.6.3.1 UEM GPIO String

The UEM GPIO String Editing field is in the last column of GPIO Setting window. User can key string names in these fields. The maximum string length is defined in UEM.comp by MAX_NETNAME_TEXT. Note that only for those GPIO ports which are permitted to set as GPIO mode, these corresponding fields can be edited.



2.6.3.2 UEM GPO String

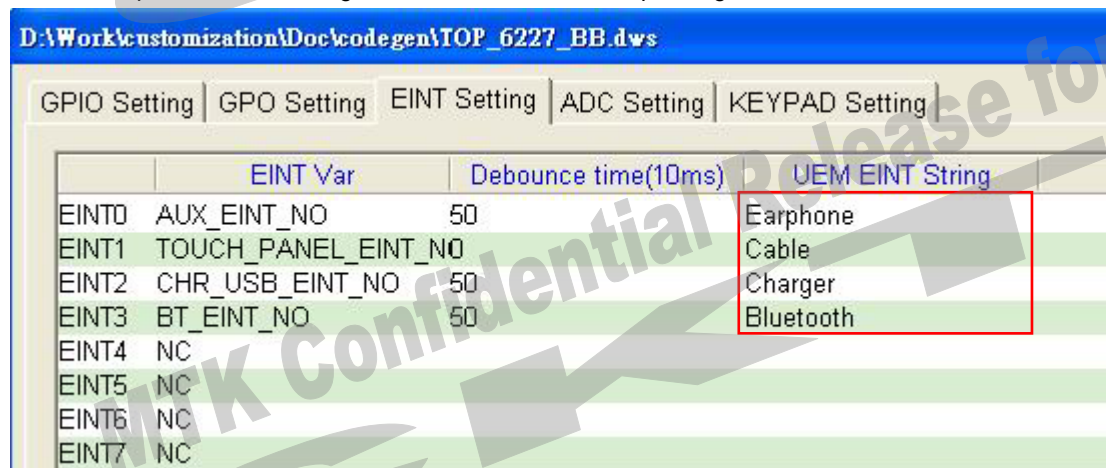
The UEM GPO String Editing field is in the last column of GPO Setting window. User can key string names in these fields. The maximum string length is defined in UEM.comp by MAX_NETNAME_TEXT. Note that only for those GPO ports which are permitted to set as GPO mode, these corresponding fields can be edited.



Also note that at least one string name must exist for UEM GPIO/GPO String. Otherwise, there will be a pop-up error message when generating codes.

2.6.3.3 UEM EINT String

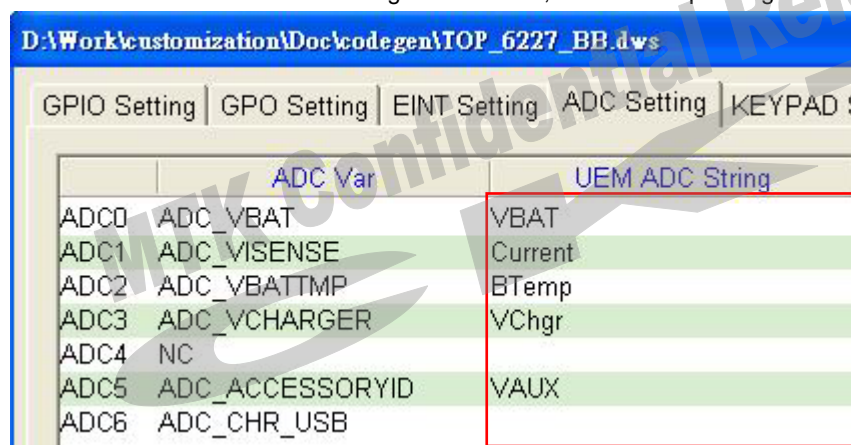
The UEM EINT String Editing field is in the last column of EINT Setting window. User can key string names in these fields. The maximum string length is defined in UEM.comp by MAX_NETNAME_TEXT. Note that only for those EINT pins which are assigned variables, these corresponding fields can be edited.



Also note that at least one string name must exist for UEM EINT String. Otherwise, there will be a pop-up error message when generating codes.

2.6.3.4 UEM ADC String

The UEM ADC String Editing field is in the last column of ADC Setting window. User can key string names in these fields. The maximum string length is defined in UEM.comp by MAX_NETNAME_TEXT. Note that only for those ADC channels which are assigned variables, these corresponding fields can be edited.



Preliminary Information

Also note that at least one string name must exist for UEM ADC String. Otherwise, there will be a pop-up error message when generating codes.

2.6.4 Generated Source and Header files

2.6.4.1 uem_drv.c

1. Header defined in UEM.cmp:

```
#ifdef __CUST_NEW__
#include "kal_release.h"
#include "gpio_drv.h"
#include "custom_em.h"
#include "custom_equipment.h"
```

2. GPIO/GPO strings and table:

```
const kal_uint8 GPIO_LABELID_MAX = 15;
kal_uint8 EmGpioIdxMenu2Tbl[15];
const unsigned char netname[][MAX_NETNAME_TEXT] = {
    "GP0_RLEDEN",
    "GP1_CAMPWREN",
    "GP3_BTDATA",
    "GP4_BTDISCON",
    "GP5_LCMBLEN",
    "GP6_CMFLASH",
    "GP7_VIBEN",
    "GPIO12_BTPWREN",
    "GP23_KPLEDPWM",
    "GP31_CHRCTL",
    "GP38_BLED_EN",
    "GP39_BTRESET",
    "GPIO52_GLED0",
    "GPO1_OPON",
    "GPO2_USBEN",
};
GPIO_MAP_ENTRY gpio_map_tbl[] = {
    {GPIO_VALID, GPIO_PORT_0, netname[0], NULL},
    {GPIO_VALID, GPIO_PORT_1, netname[1], NULL},
    {GPIO_VALID, GPIO_PORT_3, netname[2], NULL},
    {GPIO_VALID, GPIO_PORT_4, netname[3], NULL},
    {GPIO_VALID, GPIO_PORT_5, netname[4], NULL},
    {GPIO_VALID, GPIO_PORT_6, netname[5], NULL},
    {GPIO_VALID, GPIO_PORT_7, netname[6], NULL},
    {GPIO_VALID, GPIO_PORT_12, netname[7], NULL},
    {GPIO_VALID, GPIO_PORT_23, netname[8], NULL},
    {GPIO_VALID, GPIO_PORT_31, netname[9], NULL},
    {GPIO_VALID, GPIO_PORT_38, netname[10], NULL},
    {GPIO_VALID, GPIO_PORT_39, netname[11], NULL},
    {GPIO_VALID, GPIO_PORT_52, netname[12], NULL},
    {GPO_VALID, GPO_PORT_1, netname[13], NULL},
    {GPO_VALID, GPO_PORT_2, netname[14], NULL},
};
```

Preliminary Information

3. EINT strings and table

```
const kal_uint8 EINT_LABELID_MAX = 4;
const unsigned char eintname[][MAX_NETNAME_TEXT] = {
    "Earphone",
    "Cable",
    "Charger",
    "Bluetooth",
};
GPIO_MAP_ENTRY eint_map_tbl[] = {
    {GPIO_VALID, 0, eintname[0], NULL},
    {GPIO_VALID, 1, eintname[1], NULL},
    {GPIO_VALID, 2, eintname[2], NULL},
    {GPIO_VALID, 3, eintname[3], NULL},
};
```

4. ADC strings and table

```
const kal_uint8 ADC_LABELID_MAX = 5;
const unsigned char adcname[][MAX_NETNAME_TEXT] = {
    "VBAT",
    "Current",
    "BTemp",
    "VChgr",
    "VAUX",
};
GPIO_MAP_ENTRY adc_map_tbl[] = {
    {GPIO_VALID, 0, adcname[0], NULL},
    {GPIO_VALID, 1, adcname[1], NULL},
    {GPIO_VALID, 2, adcname[2], NULL},
    {GPIO_VALID, 3, adcname[3], NULL},
    {GPIO_VALID, 5, adcname[4], NULL},
};
```

5. Tailer defined in UEM.cmp

```
#endif /* __CUST_NEW__ */
```

3 Customization Procedure for a New Project

3.1 Step 1: Turn on Feature

In the Project_Name.mak file, turn on the customization tool feature as below:

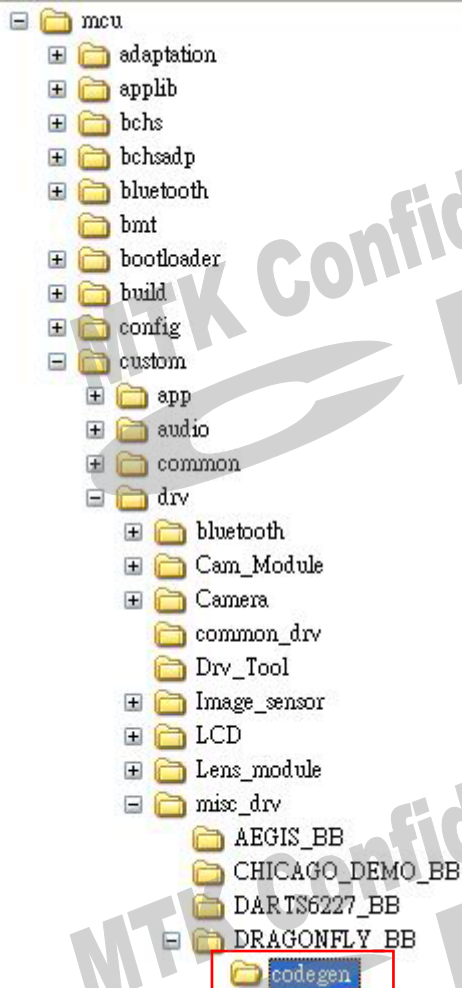
```
DRV_CUSTOM_TOOL_SUPPORT = TRUE # TRUE or FALSE
```

Note that when this feature is turned on, a compile option `__CUST_NEW__` will be defined.

3.2 Step 2: Create a data workspace file

3.2.1.1 Create a codegen folder

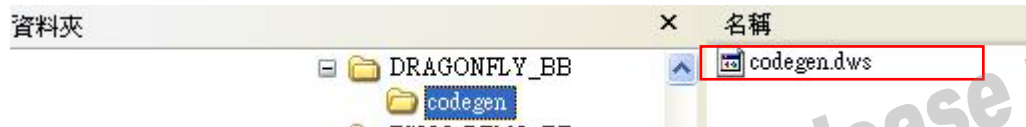
Create a folder `\mcu\custom\drv\misc_drv\BOARD_VER\codegen` where `BOARD_VER` is the Baseband main board description in the Project_Name.mak. For example, if `BOARD_VER` = `DRAGONFLY_BB` in `DRAGONFLY_DEMO_GPRS.mak`, then create a `codegen` folder under `\mcu\custom\drv\misc_drv\DRAGONFLY_BB\`.



3.2.1.2 Create a data workspace file

New a data workspace by tool as described in section 2.2.1 in the folder as mentioned above:

`\mcu\custom\drv\misc_drv\BOARD_VER\codegen`. And the file name of this data workspace must be `codegen.dws`.



3.3 Step 3: Edit settings on Tool

Based on the created data workspace file, edit the GPIO/GPO/EINT/ADC/KEYPAD/UEM settings as described in section 2. When it's done, save the workspace file.

3.4 Step 4: Settings beyond the Tool

3.4.1 Debug Mode

In `gpio_drv.c`, one global variable `kal_bool gpio_debug_enable` is declared for GPIO debug mode enable/disable. When users want to turn on GPIO debug mode, assign value `KAL_TRUE` to it otherwise just assign `KAL_FALSE` to it.

3.4.2 GPIO output initial value

Users may like to set the value of one particular GPIO port which is configured as GPO in the beginning. Then it is suggested to call `GPIO_WriteIO()` in the bottom of function `GPIO_init()` of file `gpio_drv.c`. For example as below, users want to set USB enable pin to low in the beginning:

```
void GPIO_init(void)
{
    GPIO_setting_init();

    /* Set the GPIO initial value below. */
    GPIO_WriteIO(0, gpio_usb_enable_pin);
}
```

3.4.3 Settings for unused GPIO pins

In `gpio_def.h`, it defines the mode for those unused GPIO pins (NC) as mode 0 (GPIO mode):

```
#define MODE_NC 0
```

And normally it will be GPIO mode. However not mode 0 of each GPIO pins is GPIO mode. Therefore if one GPIO pin is unused but its mode 0 is not GPIO mode and user still wants to set it to GPIO mode, then in tool side this pin should select as "GPIO" instead of "NC".

Moreover the direction of the unused GPION pins is defined as below:

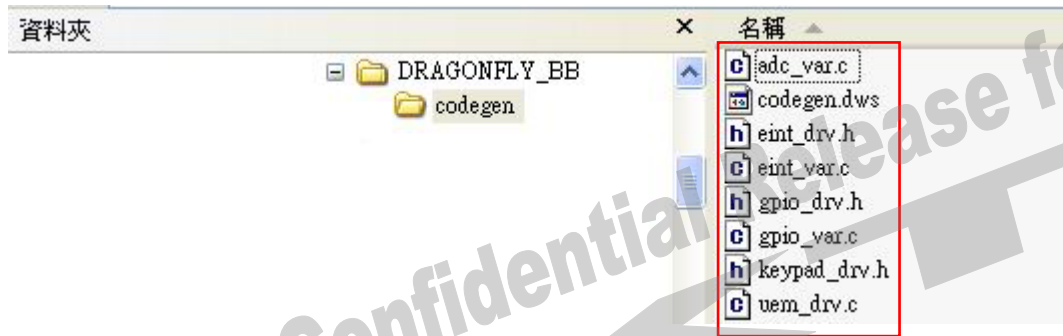
```
#define DIR_NULL 0 /* 0: Input, 1: Output */
```

User can choose to set it as GPO or GPI by the value 0 or 1.

3.5 Step 5: Build the load

Build the load on the command script window with "new" parameter. Note that users don't have to click GenCode button to generate source and header files. These files will be automatically generated in the process of building load. For example:

```
make DRAGONFLY_DEMO gprs new
```



These generated source and header files belongs to custom.lib. Therefore if there's any modification on the tool, user can re-build the custom library by "update" or "remake" parameter. For example:

```
make DRAGONFLY_DEMO gprs update custom
```

Or

```
make DRAGONFLY_DEMO gprs remake custom
```

The "update" parameter is recommended because the source and header files will be automatically generated. If "remake" parameter is used, users have to click "GenCode" button on the tool in order to generate the files in advance.

3.6 Step 6: Misc. and Debug

3.6.1 Case 1

If there is linking error as below:

Error: L6218E: Undefined symbol gpio_vibrator_en_pin (referred from uem_gpio.obj)

In this example, it means that the GPIO variable `gpio_vibrator_en_pin` is referred in `uem_gpio.c` but it is not declared. So user should check which GPIO port ought to be assigned this GPIO variable on the Tool. And rebuild the load.

3.6.2 Case 2

The load has been downloaded into the target and target powers on. Assert happens somewhere in `gpio.c` with debug mode turned on.

If assert happens in the function `GPIO_ModeSetup()`, it means that some module tries to set one disallowed mode for a GPIO port showed on the tool. User should check if the access of GPIO mode is valid or not. If it's valid, the mode for that GPIO port should be set as permitted.

If assert happens in the function `GPIO_WriteIO()`, it may be cause by

1. Invalid GPIO port number that is without `GPIO_MAGIC_NUM`.
2. This GPIO port is not in the GPIO mode or in the GPIO mode but not GPO.

[References]

MTK Confidential Release for

MTK Confidential Release for

MTK Confidential Release for