



User Manual

SusiDevice

SusiDevice
Software APIs

ADVANTECH **EmbCore**

Edition 1.0
Jun 26 2013

Part. No.
Printed in Taiwan

Contents

| | |
|--|-----------|
| CONTENTS | 3 |
| INTRODUCTION | 6 |
| 1.1 BENEFITS | 6 |
| 1.2 ENVIRONMENT REQUIREMENTS | 6 |
| 1.2.1 Operating Systems | 6 |
| 2. DEFINITION | 7 |
| 2.1 STATUS CODES | 7 |
| 3. API | 11 |
| 3.1 DEVICE FUNCTIONS | 11 |
| 3.1.1 SusiDeviceGetValue | 11 |
| 3.1.2 SusiDeviceSetValue | 11 |
| APPENDIX A. SAB2000 | 13 |
| 1. DEFINE | 13 |
| 1.1 ITEM ID..... | 13 |
| 1.2 DEVICE INITIALIZATION | 14 |
| 1.2.1 | 15 |
| 2. FUNCTIONS | 15 |
| 2.1 DEVICE INFORMATION | 15 |
| 2.1.1 Device available | 15 |
| 2.1.2 Firmware version | 16 |
| 2.1.3 Kernel version | 16 |
| 2.1.4 EC type | 16 |
| 2.1.5 Case open..... | 16 |
| 2.1.6 Alert control..... | 16 |
| 2.1.7 Temperature | 16 |
| 2.1.8 Temperature Alert..... | 16 |
| 2.1.9 Voltage | 17 |
| 2.1.10 Fan speed..... | 17 |
| 2.1.11 G-Sensor | 17 |
| 2.1.12 LED..... | 17 |
| APPENDIX B. ADXL345(G-SENSOR) | 18 |

| | |
|---|-----------|
| 1. DEFINE | 18 |
| 1.1 ITEM ID..... | 18 |
| 1.2 DEVICE INITIALIZATION | 18 |
| 2. FUNCTIONS | 19 |
| 2.1 DEVICE AVAILABLE INFORMATION | 19 |
| 2.2 DATA OF G-SENSOR | 19 |
| 2.3 OFFSET..... | 19 |
| 2.4 MEASURE MODE..... | 20 |
| 2.5 G-RANGE OF MEASURE | 20 |
| 2.6 LOW POWER MODE | 20 |
| 2.7 SLEEP MODE | 21 |
| 2.8 OUTPUT DATA RATE IN NORMAL MODE | 21 |
| 2.9 OUTPUT DATA RATE IN SLEEP MODE | 22 |
| APPENDIX C. POE | 23 |
| 1. DEFINE | 23 |
| 1.1 ITEM ID..... | 23 |
| 1.2 DEVICE INITIALIZATION | 24 |
| 2. FUNCTIONS | 25 |
| 2.1 DEVICE AVAILABLE INFORMATION | 25 |
| 2.2 POWER SETTING | 25 |
| 2.3 DETECTION INFORMATION | 25 |
| 2.4 CLASSIFICATION INFORMATION | 25 |
| 2.5 VOLTAGE INFORMATION..... | 26 |
| 2.6 CURRENT INFORMATION | 26 |
| APPENDIX D. AMO-P008(PIC) | 27 |
| 1. DEFINE | 27 |
| 1.1 ITEM ID..... | 27 |
| 1.2 DEVICE INITIALIZATION | 30 |
| 2. FUNCTIONS | 31 |
| 2.1 DEVICE AVAILABLE INFORMATION | 31 |
| 2.2 GENERAL INFORMATION | 31 |
| 2.3 SWITCH STATUS..... | 31 |
| 2.4 F/W INFORMATION | 32 |
| 2.5 H/W CONFIG | 32 |

2.6 S/W SETTING 32

2.7 GET CURRENT SETTING 33

2.8 GET MIN SETTING..... 33

2.9 GET MAX SETTING..... 34

2.10 GET DEFAULT SETTING 34

2.11 TIMER SETTING..... 35

2.12 EEPROM DATA..... 35

2.13 SYSTEM COMMAND..... 35

Introduction

SusiDevice is auxiliary library that depend on SUSI library.

1.1 Benefits

SusiDevice use two functions *SusiDeviceGetValue* and *SusiDeviceSetValue* to control anything without learning many APIs.

1.2 Environment Requirements

1.2.1 Operating Systems

Windows XP 32-bit (All series)

Windows 7 (x86 / x64)

WES7 (x86 / x64)

Windows 8 Desktop (x86 / x64)

Others (Project based)

2. Definition

SusiDevice utilizes the API declaration in SUSI 4.0. The constants in API declaration are required for programming.

2.1 Status Codes

All functions in SusiDevice API return a status code from a common list of possible errors immediately. Each function may return any of defined status codes as following below.

```
#define SUSI_STATUS_NOT_INITIALIZED 0xFFFFFFFF
```

Description

The SUSI API library is not yet or unsuccessfully initialized. SusiLibInitialize() needs to be called prior to the first access of any other SUSI API functions.

Actions

Call SusiLibInitialize().

```
#define SUSI_STATUS_INITIALIZED 0xFFFFFFFFE
```

Description

Library has been initialized before. Since SUSI API library is only needed to initialize once, the mutiple initialization will result in this error status.

Actions

None.

```
#define SUSI_STATUS_ALLOC_ERROR 0xFFFFFFFFD
```

Description

Memory allocation error.

Actions

Free memory and try again.

```
#define SUSI_STATUS_DRIVER_TIMEOUT 0xFFFFFFFCC
```

Description

Time out in driver. This is normally caused by hardware/software semaphore timeout.

Actions

Retry.

```
#define SUSI_STATUS_INVALID_PARAMETER 0xFFFFFFF
```

Description

One or more of the function call parameters are out of the defined range.

Actions

Verify function parameters.

```
#define SUSI_STATUS_INVALID_BLOCK_ALIGNMENT 0xFFFFFEFE
```

Description

The block alignment is incorrect.

Actions

Use inputs and outputs to correctly select inputs and outputs.

```
#define SUSI_STATUS_INVALID_BLOCK_LENGTH 0xFFFFFEFD
```

Description

This means that the block length is too long.

Actions

Use alignment capabilities information to correctly align write access.

```
#define SUSI_STATUS_INVALID_DIRECTION 0xFFFFFEFC
```

Description

The current direction argument attempts to set GPIOs to a unsupported directions. I.E. Setting GPI to output.

Actions

Use inputs and outputs to correctly select input and outputs.

```
#define SUSI_STATUS_INVALID_BITMASK 0xFFFFFEFB
```

Description

The bitmask selects bits/GPIOs which are not supported for the current ID.

Actions

Use Inputs and Outputs to probe supported bits.

```
#define SUSI_STATUS_RUNNING 0xFFFFFEFA
```

Description

Watchdog timer already started.

Actions

Call SusiWDogStop(), before retrying.

```
#define SUSI_STATUS_UNSUPPORTED 0xFFFFFCFF
```

Description

The function or channel is not supported at the actual hardware environment.

Actions

None.

```
#define SUSI_STATUS_NOT_FOUND 0xFFFFFBFF
```

Description

Selected device is not found.

Actions

None.

```
#define SUSI_STATUS_TIMEOUT 0xFFFFFBFE
```

Description

Device has no response.

Actions

None.

```
#define SUSI_STATUS_BUSY_COLLISION 0xFFFFBFD
```

Description

The selected device or ID is busy or a data collision is detected.

Actions

Retry.

```
#define SUSI_STATUS_READ_ERROR 0xFFFFFAFF
```

Description

An error is detected during a read operation.

Actions

Retry.

```
#define SUSI_STATUS_WRITE_ERROR 0xFFFFFAFE
```

Description

An error is detected during a write operation.

Actions

Retry.

```
#define SUSI_STATUS_MORE_DATA 0xFFFF9FF
```

Description

The amount of available data exceeds the buffer size. Storage buffer overflow was prevented.

Read count is larger than the defined buffer length.

Actions

Either increase the buffer size or reduce the block length.

```
#define SUSI_STATUS_ERROR 0xFFFF0FF
```

Description

Generic error message. No further error details are available.

Actions

None.

```
#define SUSI_STATUS_SUCCESS 0
```

Description

The operation was successful.

Actions

None.

3. API

SusiDevice API provides the functions to control ADVANTECH platforms. The functions are based on a dynamic library. SusiDevice API can be implemented in various other programming languages.

3.1 Device Functions

In order to make SusiDevice support each device on ADVANTECH platforms. The functions standardize to two functions specially. One function is to obtain information and the other is to control the settings of device.

3.1.1 SusiDeviceGetValue

```
uint32_t SUSI_API SusiDeviceGetValue(uint32_t Id, uint32_t *pValue)
```

Description:

Get information about the firmware or AP platform in value format.

Parameters:

Id

Select target of item IDs.

pValue

A buffer pointer to the value of item's information.

3.1.2 SusiDeviceSetValue

```
uint32_t SUSI_API SusiDeviceSetValue(uint32_t Id, uint32_t Value)
```

Description:

Get information about the firmware or AP platform in value format.

Parameters:

Id

Select target of item IDs.

Value

The value of setting parameter.

Appendix A. SAB2000

1. Define

1.1 Item Id

For SAB2000 device, the item IDs are shown as below:

// Information

| | |
|-------------------------------------|------------|
| #define SAB2000_ID_DEVICE_AVAILABLE | 0x00800000 |
| #define SAB2000_ID_FW_VER | 0x00800001 |
| #define SAB2000_ID_EC_TYPE | 0x00800002 |
| #define SAB2000_ID_KERNEL_VER | 0x00800003 |
| #define SAB2000_ID_CASEOPEN | 0x00800010 |

// Alert control

| | |
|-------------------------------|------------|
| #define SAB2000_ID_CTRL_ALERT | 0x00810000 |
|-------------------------------|------------|

// Hardware monitoring - temperature

| | |
|-----------------------------------|------------|
| #define SAB2000_ID_HWM_TEMP_VTIN1 | 0x00820000 |
| #define SAB2000_ID_HWM_TEMP_VTIN2 | 0x00820001 |
| #define SAB2000_ID_HWM_TEMP_VTIN3 | 0x00820002 |
| #define SAB2000_ID_HWM_TEMP_BT1 | 0x00820003 |
| #define SAB2000_ID_HWM_TEMP_BT2 | 0x00820004 |
| #define SAB2000_ID_HWM_TEMP_BT3 | 0x00820005 |
| #define SAB2000_ID_HWM_TEMP_BT4 | 0x00820006 |

// Hardware monitoring – temperature alert

| | |
|---|------------|
| #define SAB2000_ID_HWM_TEMP_ALERT_VTIN1 | 0x00820100 |
| #define SAB2000_ID_HWM_TEMP_ALERT_VTIN2 | 0x00820101 |
| #define SAB2000_ID_HWM_TEMP_ALERT_BT1 | 0x00820103 |
| #define SAB2000_ID_HWM_TEMP_ALERT_BT2 | 0x00820104 |
| #define SAB2000_ID_HWM_TEMP_ALERT_BT3 | 0x00820105 |
| #define SAB2000_ID_HWM_TEMP_ALERT_BT4 | 0x00820106 |

// Hardware monitoring - voltage

| | |
|------------------------------------|------------|
| #define SAB2000_ID_HWM_VOLT_VCOREA | 0x00821000 |
| #define SAB2000_ID_HWM_VOLT_VCOREB | 0x00821001 |
| #define SAB2000_ID_HWM_VOLT_3V3 | 0x00821002 |
| #define SAB2000_ID_HWM_VOLT_5V | 0x00821003 |
| #define SAB2000_ID_HWM_VOLT_12V | 0x00821004 |
| #define SAB2000_ID_HWM_VOLT_12NV | 0x00821005 |
| #define SAB2000_ID_HWM_VOLT_5VSB | 0x00821006 |

```

#define SAB2000_ID_HWM_VOLT_5NV 0x00821007
#define SAB2000_ID_HWM_VOLT_VBAT 0x00821008
// Hardware monitoring - fan speed
#define SAB2000_ID_HWM_FAN_0 0x00822000
#define SAB2000_ID_HWM_FAN_1 0x00822001
#define SAB2000_ID_HWM_FAN_2 0x00822002
#define SAB2000_ID_HWM_FAN_OB1 0x00822003
#define SAB2000_ID_HWM_FAN_OB2 0x00822004
#define SAB2000_ID_HWM_FAN_OB3 0x00822005
#define SAB2000_ID_HWM_FAN_OB4 0x00822006
#define SAB2000_ID_HWM_FAN_OB5 0x00822007
#define SAB2000_ID_HWM_FAN_OB6 0x00822008
#define SAB2000_ID_HWM_FAN_OB7 0x00822009
// G sensor
#define SAB2000_ID_GSENSOR_AXIS_X 0x00830000
#define SAB2000_ID_GSENSOR_AXIS_Y 0x00830001
#define SAB2000_ID_GSENSOR_AXIS_Z 0x00830002
#define SAB2000_ID_GSENSOR_AXIS_FF_COUNT 0x00830003
#define SAB2000_ID_GSENSOR_AXIS_GVALUE 0x00830004
// LED
#define SAB2000_ID_LED_POWER 0x00831000
#define SAB2000_ID_LED_TEMP 0x00831001
#define SAB2000_ID_LED_FAN 0x00831002

```

1.2 Device Initialization

SAB2000 alarm board has 10 DIP switch to configure support functions, more detail as following tables:

| MB Fan & CPU temperature | | | | | |
|--------------------------|-----|-----|--------------|---------|----------|
| SW1 | SW2 | SW3 | Cable Status | MB FAN | CPU TEMP |
| 0 | 0 | 0 | No Connect | Disable | Disable |
| 0 | 0 | 1 | Connect | Disable | 1 |
| 0 | 1 | 0 | Connect | Disable | 2 |
| 0 | 1 | 1 | Connect | 1 | 1 |
| 1 | 0 | 0 | Connect | 2 | 1 |
| 1 | 0 | 1 | Connect | 2 | 2 |
| 1 | 1 | 0 | Connect | 3 | 1 |
| 1 | 1 | 1 | Connect | 3 | 2 |

| SW4 | SW5 | SW9 | Sys Fan Qty. |
|-----|-----|-----|--------------|
| 0 | 0 | 0 | Disable |
| 0 | 0 | 1 | 1 (FAN1) |
| 0 | 1 | 0 | 2 (FAN1~2) |
| 0 | 1 | 1 | 3 (FAN1~3) |
| 1 | 0 | 0 | 4 (FAN1~4) |
| 1 | 0 | 1 | 5 (FAN1~5) |
| 1 | 1 | 0 | 6 (FAN1~6) |
| 1 | 1 | 1 | 1 (FAN1~7) |

1.2.1

| SW7 | SW8 | SW9 | Thermistor Qty. |
|--------|-----|-----|-----------------|
| 0 | 0 | 0 | Disable |
| 0 | 0 | 1 | 1 (TR1) |
| 0 | 1 | 0 | 2 (TR1~2) |
| 0 | 1 | 1 | 3 (TR1~3) |
| 1 | 0 | 0 | 4 (TR1~4) |
| Others | | | Reserved |

2. Functions

Bits 31-12 (0xFFFF000) of ID is separate different functions. Example: ID SAB2000_ID_FW_VER that code is 0x00800001, 0x00800000 represent SAB2000 information part. All items have same statue codes after called *SusiDeviceGetValue* and *SusiDeviceSetValue* that as following table:

Return Status Code:

| Condition | Return Value |
|----------------|-------------------------------|
| Success | SUSI_STATUS_SUCCESS |
| Find no device | SUSI_STATUS_NOT_FOUND |
| Value invalid | SUSI_STATUS_INVALID_PARAMETER |
| Else | SUSI_STATUS_ERROR |

2.1 Device Information

2.1.1 Device available

ID SAB2000_ID_DEVICE_AVAILABLE can get SAB2000 is available or not. The value of parameter is 1 or 0

while the device is found and not respectively.

2.1.2 Firmware version

ID `SAB2000_ID_FW_VER` can get SAB2000 firmware version. The value format as below:

| Bit [31-24] | Bit [23-16] | Bit [15-8] | Bit [7-0] |
|-------------|-------------|------------|-----------|
| Reserved | Characters | Major | Minor |

2.1.3 Kernel version

ID `SAB2000_ID_FW_VER` can get SAB2000 firmware version. The value format likes 2.1.2.

2.1.4 EC type

ID `SAB2000_ID_EC_TYPE` can get SAB2000 EC type. The value format as below:

| Bit [31-24] | Bit [23-16] | Bit [15-8] | Bit [7-0] |
|-------------|----------------------------------|------------|-----------|
| Reserved | Characters I = ITE N = ENE | Type (HEX) | TBD |

2.1.5 Case open

ID `SAB2000_ID_CASEOPEN` can get case open state. The value of parameter is 1 or 0 while the device is opened or closed.

2.1.6 Alert control

ID `SAB2000_ID_CTRL_ALERT` can get or set alert state. The value of parameter is 1 or 0 while the device is alarm or normal.

2.1.7 Temperature

Using temperature ID likes `SAB2000_ID_HWM_TEMP_VTIN1` can get temperature value that is in 0.1 Kelvin unit.

2.1.8 Temperature Alert

Using temperature alert ID likes `SAB2000_ID_HWM_TEMP_ALERT_VTIN1` can get or set temperature alert limit value that is in 0.1 Kelvin unit.

2.1.9 Voltage

Using voltage ID likes `SAB2000_ID_HWM_VOLT_VCOREA` can get voltages value that is in 0.001 volt unit with sign.

2.1.10 Fan speed

Using fan speed ID likes `SAB2000_ID_HWM_FAN_0` can get fan speed value that is in RPM unit.

2.1.11 G-Sensor

Using G-sensor ID likes `SAB2000_ID_GSENSOR_AXIS_X` can get or set G-sensor settings or g values.

Parameter of GValue:

| Value | Description |
|-------|------------------------|
| 0b00 | g-range is ± 2 g. |
| 0b 01 | g-range is ± 4 g. |
| 0b 10 | g-range is ± 8 g. |
| 0b 11 | g-range is ± 16 g. |

2.1.12 LED

Using LED ID likes `SAB2000_ID_LED_POWER` can get LED state.

Parameter Value:

| Value | Description |
|--------|-------------|
| 0b001 | Green |
| 0b010 | Red |
| 0b101 | Green Blink |
| 0b110 | Red Blink |
| others | N/A |

Appendix B. ADXL345(G-Sensor)

1. Define

1.1 Item Id

For ADXL345 device, the item IDs are shown as below:

| | |
|---|------------|
| <code>#define ADXL345_ID_INFO_AVAILABLE</code> | 0x00400000 |
| <code>#define ADXL345_ID_DATA_X</code> | 0x00410000 |
| <code>#define ADXL345_ID_DATA_Y</code> | 0x00410001 |
| <code>#define ADXL345_ID_DATA_Z</code> | 0x00410002 |
| <code>#define ADXL345_ID_OFFSET_X</code> | 0x00420000 |
| <code>#define ADXL345_ID_OFFSET_Y</code> | 0x00420001 |
| <code>#define ADXL345_ID_OFFSET_Z</code> | 0x00420002 |
| <code>#define ADXL345_ID_MEASURE_CTRL</code> | 0x00430000 |
| <code>#define ADXL345_ID_MEASURE_RANGE</code> | 0x00430001 |
| <code>#define ADXL345_ID_POWER_LOWPPOWER</code> | 0x00440000 |
| <code>#define ADXL345_ID_POWER_SLEEP</code> | 0x00440001 |
| <code>#define ADXL345_ID_DATARATE_NORMAL</code> | 0x00450000 |
| <code>#define ADXL345_ID_DATARATE_SEELP</code> | 0x00450001 |

1.2 Device Initialization

In initialization, the device is set in measure mode with the range from -2 g to 2 g, and not in low power mode with 100Hz of output data rate. User can modify these setting by function *SusiDeviceSetValue* with the item id:

ADXL345_ID_MEASURE_CTRL
ADXL345_ID_MEASURE_RANGE
ADXL345_ID_POWER_LOWPPOWER
ADXL345_ID_DATARATE_NORMAL.

2. Functions

The first parameter of *SusiDeviceGetValue* and *SusiDeviceSetValue* both input item id which occupies 4-byte memories. If the first parameter input is not the item ids of ADXL345, the status code will return SUSI_STATUS_UNSUPPORTED.

2.1 Device Available Information

ID `ADXL345_ID_INFO_AVAILABLE` can get ADXL345 is available or not. The value of parameter is 1 or 0 while the device is found and not respectively. The return status code always is SUSI_STATUS_SUCCESS.

2.2 Data of g-Sensor

ID `ADXL345_ID_DATA_X`, `ADXL345_ID_DATA_Y` and `ADXL345_ID_DATA_Z` those can get g value. The value of parameter is in 0.1 mg unit with sign. For examples, if the value is 0x00004E20 represents 2g, and 0xFFFFB1E0 represents -2g.

Return Status Code:

| Condition | Return Value |
|----------------|-----------------------|
| Success | SUSI_STATUS_SUCCESS |
| Find no device | SUSI_STATUS_NOT_FOUND |
| Else | SUSI_STATUS_ERROR |

2.3 Offset

ID `ADXL345_ID_OFFSET_X`, `ADXL345_ID_OFFSET_Y`, and `ADXL345_ID_OFFSET_Z` those can get or set offset for g value calibration. The value of parameter is in 0.1 mg unit with sign. For examples, if the value of is 0x00004E20 represents 2g, and 0xFFFFB1E0 represents -2g.

Return Status Code:

| Condition | Return Value |
|----------------|-----------------------|
| Success | SUSI_STATUS_SUCCESS |
| Find no device | SUSI_STATUS_NOT_FOUND |
| Else | SUSI_STATUS_ERROR |

2.4 Measure Mode

ID `ADXL345_ID_MEASURE_CTRL` can get or set measure mode. The value of parameter is 1 or 0 while the device is in measurement and standby mode respectively. The device powers up in standby mode with minimum power consumption.

Return Status Code:

| Condition | Return Value |
|----------------|--|
| Success | <code>SUSI_STATUS_SUCCESS</code> |
| Find no device | <code>SUSI_STATUS_NOT_FOUND</code> |
| Value invalid | <code>SUSI_STATUS_INVALID_PARAMETER</code> |
| Else | <code>SUSI_STATUS_ERROR</code> |

2.5 g-Range of Measure

ID `ADXL345_ID_MEASURE_RANGE` can get or set measure range. The device supports 4 types of g-range: ± 2 g, ± 4 g, ± 8 g, and ± 16 g. SusiDevice initializes device in ± 2 g of g-range.

Parameter Value:

| Value | Description |
|-------|------------------------|
| 2 | g-range is ± 2 g. |
| 4 | g-range is ± 4 g. |
| 8 | g-range is ± 8 g. |
| 16 | g-range is ± 16 g. |

Return Status Code:

| Condition | Return Value |
|----------------|--|
| Success | <code>SUSI_STATUS_SUCCESS</code> |
| Find no device | <code>SUSI_STATUS_NOT_FOUND</code> |
| Value invalid | <code>SUSI_STATUS_INVALID_PARAMETER</code> |
| Else | <code>SUSI_STATUS_ERROR</code> |

2.6 Low Power Mode

ID `ADXL345_ID_POWER_LOWPOWER` can get or set low power mode state. The value of parameter is 1 or 0 while the device is in reduced power operation and normal operation respectively. In reduced power operation has somewhat higher noise.

Return Status Code:

| Condition | Return Value |
|----------------|-------------------------------|
| Success | SUSI_STATUS_SUCCESS |
| Find no device | SUSI_STATUS_NOT_FOUND |
| Value invalid | SUSI_STATUS_INVALID_PARAMETER |
| Else | SUSI_STATUS_ERROR |

2.7 Sleep Mode

ID ADXL345_ID_POWER_SLEEP can get or set sleep mode state. The value of parameter is 1 and 0 while the device is in sleep mode and normal mode of operation respectively.

Return Status Code:

| Condition | Return Value |
|----------------|-------------------------------|
| Success | SUSI_STATUS_SUCCESS |
| Find no device | SUSI_STATUS_NOT_FOUND |
| Value invalid | SUSI_STATUS_INVALID_PARAMETER |
| Else | SUSI_STATUS_ERROR |

2.8 Output Data Rate in Normal Mode

ID ADXL345_ID_DATARATE_NORMAL can get or set output data rate in normal mode (not in sleep mode). SusiDevice initializes the output data rate with 100 Hz. There are 16 different rates for the device and show below:

Parameter Value:

| Value | Description |
|----------|-------------------------------|
| 0 (0x00) | Output data rate is 0.098 Hz. |
| 1 (0x01) | Output data rate is 0.195 Hz. |
| 2 (0x02) | Output data rate is 0.390 Hz. |
| 3 (0x03) | Output data rate is 0.782 Hz. |
| 4 (0x04) | Output data rate is 1.563 Hz. |
| 5 (0x05) | Output data rate is 3.125 Hz. |
| 6 (0x06) | Output data rate is 6.25 Hz. |
| 7 (0x07) | Output data rate is 12.5 Hz. |
| 8 (0x08) | Output data rate is 25 Hz. |
| 9 (0x09) | Output data rate is 50 Hz. |

| | |
|-----------|------------------------------|
| 10 (0x0A) | Output data rate is 100 Hz. |
| 11 (0x0B) | Output data rate is 200 Hz. |
| 12 (0x0C) | Output data rate is 400 Hz. |
| 13 (0x0D) | Output data rate is 800 Hz. |
| 14 (0x0E) | Output data rate is 1600 Hz. |
| 15 (0x0F) | Output data rate is 3200 Hz. |

Return Status Code:

| Condition | Return Value |
|----------------|-------------------------------|
| Success | SUSI_STATUS_SUCCESS |
| Find no device | SUSI_STATUS_NOT_FOUND |
| Value invalid | SUSI_STATUS_INVALID_PARAMETER |
| Else | SUSI_STATUS_ERROR |

2.9 Output Data Rate in Sleep Mode

ID ADXL345_ID_DATARATE_SEELP can get or set output data rate in sleep mode. SusiDevice initializes the output data rate with 8Hz. The value of parameter is the rate in unit 1Hz. There are 4 different rates for the sleep device and show below:

Parameter Value:

| Value | Description |
|-------|---|
| 1 | Output data rate is 1 Hz in sleep mode. |
| 2 | Output data rate is 2 Hz in sleep mode. |
| 4 | Output data rate is 4 Hz in sleep mode. |
| 8 | Output data rate is 5 Hz in sleep mode. |

Return Status Code:

| Condition | Return Value |
|----------------|-------------------------------|
| Success | SUSI_STATUS_SUCCESS |
| Find no device | SUSI_STATUS_NOT_FOUND |
| Value invalid | SUSI_STATUS_INVALID_PARAMETER |
| Else | SUSI_STATUS_ERROR |

Appendix C. PoE

1. Define

1.1 Item ID

For PoE devices, the item IDs are shown as below:

| | |
|-------------------------------|------------|
| #define POE_ID_INFO_AVAILABLE | 0x00200000 |
| #define POE_ID_DETECT_PORT1 | 0x00220000 |
| #define POE_ID_DETECT_PORT2 | 0x00220001 |
| #define POE_ID_DETECT_PORT3 | 0x00220002 |
| #define POE_ID_DETECT_PORT4 | 0x00220003 |
| #define POE_ID_DETECT_PORT5 | 0x00220004 |
| #define POE_ID_DETECT_PORT6 | 0x00220005 |
| #define POE_ID_DETECT_PORT7 | 0x00220006 |
| #define POE_ID_DETECT_PORT8 | 0x00220007 |
| #define POE_ID_CLASS_PORT1 | 0x00230000 |
| #define POE_ID_CLASS_PORT2 | 0x00230001 |
| #define POE_ID_CLASS_PORT3 | 0x00230002 |
| #define POE_ID_CLASS_PORT4 | 0x00230003 |
| #define POE_ID_CLASS_PORT5 | 0x00230004 |
| #define POE_ID_CLASS_PORT6 | 0x00230005 |
| #define POE_ID_CLASS_PORT7 | 0x00230006 |
| #define POE_ID_CLASS_PORT8 | 0x00230007 |
| #define POE_ID_CURRENT_PORT1 | 0x00240000 |
| #define POE_ID_CURRENT_PORT2 | 0x00240001 |
| #define POE_ID_CURRENT_PORT3 | 0x00240002 |
| #define POE_ID_CURRENT_PORT4 | 0x00240003 |
| #define POE_ID_CURRENT_PORT5 | 0x00240004 |
| #define POE_ID_CURRENT_PORT6 | 0x00240005 |
| #define POE_ID_CURRENT_PORT7 | 0x00240006 |
| #define POE_ID_CURRENT_PORT8 | 0x00240007 |
| #define POE_ID_VOLTAGE_PORT1 | 0x00250000 |
| #define POE_ID_VOLTAGE_PORT2 | 0x00250001 |

| | |
|---------------------------------|------------|
| #define POE_ID_VOLTAGE_PORT3 | 0x00250002 |
| #define POE_ID_VOLTAGE_PORT4 | 0x00250003 |
| #define POE_ID_VOLTAGE_PORT5 | 0x00250004 |
| #define POE_ID_VOLTAGE_PORT6 | 0x00250005 |
| #define POE_ID_VOLTAGE_PORT7 | 0x00250006 |
| #define POE_ID_VOLTAGE_PORT8 | 0x00250007 |
| | |
| #define POE_ID_CAP_PORT1 | 0x00260000 |
| #define POE_ID_CAP_PORT2 | 0x00260001 |
| #define POE_ID_CAP_PORT3 | 0x00260002 |
| #define POE_ID_CAP_PORT4 | 0x00260003 |
| #define POE_ID_CAP_PORT5 | 0x00260004 |
| #define POE_ID_CAP_PORT6 | 0x00260005 |
| #define POE_ID_CAP_PORT7 | 0x00260006 |
| #define POE_ID_CAP_PORT8 | 0x00260007 |
| | |
| #define POE_ID_PORT_POWER_PORT1 | 0x00270000 |
| #define POE_ID_PORT_POWER_PORT2 | 0x00270001 |
| #define POE_ID_PORT_POWER_PORT3 | 0x00270002 |
| #define POE_ID_PORT_POWER_PORT4 | 0x00270003 |
| #define POE_ID_PORT_POWER_PORT5 | 0x00270004 |
| #define POE_ID_PORT_POWER_PORT6 | 0x00270005 |
| #define POE_ID_PORT_POWER_PORT7 | 0x00270006 |
| #define POE_ID_PORT_POWER_PORT8 | 0x00270007 |

1.2 Device Initialization

In initialization, the device will be set to auto mode and then detect status of detection, classification, voltage and current of each ports.

2. Functions

The first parameter of *SusiDeviceGetValue* and *SusiDeviceSetValue* both input item id which occupies 4-byte memories. If the first parameter input is not the item ids of POE, then the error code will return SUSI_STATUS_UNSUPPORTED.

2.1 Device Available Information

ID `POE_ID_INFO_AVAILABLE` can get the availability of POE. The value of second parameter is 1 or 0 while the device is found or not, respectively. The return status code is always `SUSI_STATUS_SUCCESS`.
ID from `POE_ID_CAP_PORT1` to `POE_ID_CAP_PORT8` reports whether a port is available. The value of second parameter is 1 or 0 while the port is found or not, respectively.

2.2 Power Setting

ID from `POE_ID_PORT_POWER_PORT1` to `POE_ID_PORT_POWER_PORT8` control the power ON(1) or OFF(0) of each port.

2.3 Detection Information

ID from `POE_ID_DETECT_PORT1` to `POE_ID_DETECT_PORT8` can get the Detection Status of each port. The status definition of mapping of POE is shown as below:

| Value | Status |
|----------|---------------|
| 0 (0x00) | Unknown |
| 1 (0x01) | PD Error |
| 2 (0x02) | PD Error |
| 3 (0x03) | PD Error |
| 4 (0x04) | Detected Good |
| 5 (0x05) | PD Error |
| 6 (0x06) | Detect Open |
| 7 (0x07) | PD Error |

2.4 Classification Information

ID from `POE_ID_CLASS_PORT1` to `POE_ID_CLASS_PORT8` can get the Classification Status of each ports. The status definition of mapping of POE is below:

| Value | Status |
|-------|--------|
|-------|--------|

| | |
|----------|---------------|
| 0 (0x00) | Class Unknown |
| 1 (0x01) | Class 1 |
| 2 (0x02) | Class 2 |
| 3 (0x03) | Class 3 |
| 4 (0x04) | Class 4 |
| 5 (0x05) | Error |
| 6 (0x06) | Class 0 |
| 7 (0x07) | Over Current |

2.5 Voltage Information

The ID from POE_ID_VOLTAGE_PORT1 to POE_ID_VOLTAGE_PORT8 is for getting the voltage value of each port. The unit is milli volt.

2.6 Current Information

The ID from POE_ID_CURRENT_PORT1 to POE_ID_CURRENT_PORT8 is for getting the current value of each port. The unit is micro Amps. We assume the device is using 0.25 ohm.

Appendix D. AMO-P008(PIC)

1. Define

1.1 Item Id

For AMO-P008 device, the item IDs are shown as below:

| | |
|-----------------------------------|------------|
| #define PIC_ID_INFO_AVAILABLE | 0X00600008 |
| #define PIC_ID_FW_VER | 0X00600000 |
| #define PIC_ID_FW_CONFIG_MASK | 0X00600001 |
| #define PIC_ID_BOARD_ID | 0X00600002 |
| #define PIC_ID_BOARD_NAME_LEN | 0X00600003 |
| #define PIC_ID_BOARD_NAME1 | 0X00600004 |
| #define PIC_ID_BOARD_NAME2 | 0X00600005 |
| #define PIC_ID_BOARD_NAME3 | 0X00600006 |
| #define PIC_ID_BOARD_NAME4 | 0X00600007 |
| #define PIC_ID_SWITCH_STATE | 0X00610000 |
| #define PIC_ID_SWITCH1_MODE | 0X00610001 |
| #define PIC_ID_SWITCH1_CFG_SELECT | 0X00610002 |
| #define PIC_ID_SWITCH2_PWR_SELECT | 0X00610003 |
| #define PIC_ID_FW_STATE | 0X00620000 |
| #define PIC_ID_FW_SYS_STATUS | 0X00620001 |
| #define PIC_ID_FW_BAT_STATUS | 0X00620002 |
| #define PIC_ID_FW_TMR_STATUS | 0X00620003 |
| #define PIC_ID_FW_BAT_TYPE | 0X00620004 |
| #define PIC_ID_FW_BAT_VOLT | 0X00620005 |
| #define PIC_ID_FW_BAT_VOLT_STATUS | 0X00620006 |
| #define PIC_ID_FW_BAT_ADC | 0X00620007 |
| #define PIC_ID_FW_BAT_LOW_ADC | 0X00620008 |
| #define PIC_ID_FW_SYSON_LEVEL | 0X00620009 |
| #define PIC_ID_FW_IGN_LEVEL | 0X0062000A |
| #define PIC_ID_FW_V12_STATUS | 0X0062000B |
| #define PIC_ID_FW_V48_STATUS | 0X0062000C |
| #define PIC_ID_FW_CHECK_SUM | 0X0062000F |
| #define PIC_ID_HW_TAB_IGN1 | 0X00630000 |
| #define PIC_ID_HW_TAB_IGN2 | 0X00630001 |
| #define PIC_ID_HW_TAB_IGN3 | 0X00630002 |
| #define PIC_ID_HW_TAB_IGN4 | 0X00630003 |

| | |
|---------------------------------------|------------|
| #define PIC_ID_HW_TAB_IGN5 | 0X00630004 |
| #define PIC_ID_HW_TAB_IGN6 | 0X00630005 |
| #define PIC_ID_HW_TAB_IGN7 | 0X00630006 |
| #define PIC_ID_HW_TAB_IGN8 | 0X00630007 |
| #define PIC_ID_HW_TAB_DELAY_OFF1 | 0X00630008 |
| #define PIC_ID_HW_TAB_DELAY_OFF2 | 0X00630009 |
| #define PIC_ID_HW_TAB_DELAY_OFF3 | 0X0063000A |
| #define PIC_ID_HW_TAB_DELAY_OFF4 | 0X0063000B |
| #define PIC_ID_HW_TAB_DELAY_OFF5 | 0X0063000C |
| #define PIC_ID_HW_TAB_DELAY_OFF6 | 0X0063000D |
| #define PIC_ID_HW_TAB_DELAY_OFF7 | 0X0063000E |
| #define PIC_ID_HW_TAB_DELAY_OFF8 | 0X0063000F |
| #define PIC_ID_SET_IGN_DELAY | 0X00680000 |
| #define PIC_ID_SET_DELAY_OFF | 0X00680001 |
| #define PIC_ID_SET_HARD_OFF | 0X00680002 |
| #define PIC_ID_SET_PWR_RETRIES | 0X00680003 |
| #define PIC_ID_SET_PWR_INTERVAL | 0X00680004 |
| #define PIC_ID_SET_BL_12V | 0X00680005 |
| #define PIC_ID_SET_BL_24V | 0X00680006 |
| #define PIC_ID_SET_BL_DELAY_OFF | 0X00680007 |
| #define PIC_ID_SET_BL_HARD_OFF | 0X00680008 |
| #define PIC_ID_SET_BAT_LOW_SWITCH | 0X00680009 |
| #define PIC_ID_SET_BAT_TYPE | 0X0068000A |
| #define PIC_ID_GET_IGN_DELAY | 0X00690000 |
| #define PIC_ID_GET_DELAY_OFF | 0X00690001 |
| #define PIC_ID_GET_HARD_OFF | 0X00690002 |
| #define PIC_ID_GET_PWR_RETRIES | 0X00690003 |
| #define PIC_ID_GET_PWR_INTERVAL | 0X00690004 |
| #define PIC_ID_GET_BL_12V | 0X00690005 |
| #define PIC_ID_GET_BL_24V | 0X00690006 |
| #define PIC_ID_GET_BL_DELAY_OFF | 0X00690007 |
| #define PIC_ID_GET_BL_HARD_OFF | 0X00690008 |
| #define PIC_ID_GET_BAT_LOW_SWITCH | 0X00690009 |
| #define PIC_ID_GET_BAT_TYPE | 0X0069000A |
| #define PIC_ID_TIMER_TMR_IGN_ON | 0X006A0000 |
| #define PIC_ID_TIMER_PWR_ON_RETRIES | 0X006A0001 |
| #define PIC_ID_TIMER_PWR_ON_INTERVAL | 0X006A0002 |
| #define PIC_ID_TIMER_PWR_OFF_RETRIES | 0X006A0003 |
| #define PIC_ID_TIMER_PWR_OFF_INTERVAL | 0X006A0004 |

| | |
|--|------------|
| #define PIC_ID_TIMER_TMR_DELAY_OFF | 0X006A0005 |
| #define PIC_ID_TIMER_TMR_HARD_OFF | 0X006A0006 |
| #define PIC_ID_TIMER_TMR_BL_DELAY_OFF | 0X006A0007 |
| #define PIC_ID_TIMER_PWR_12V48V_INTERVAL | 0X006A0008 |
| #define PIC_ID_TIMER_TMR_FW_UP_TIME | 0X006A000F |
| #define PIC_ID_MIN_IGN_DELAY | 0X006B0000 |
| #define PIC_ID_MIN_DELAY_OFF | 0X006B0001 |
| #define PIC_ID_MIN_HARD_OFF | 0X006B0002 |
| #define PIC_ID_MIN_PWR_RETRIES | 0X006B0003 |
| #define PIC_ID_MIN_PWR_INTERVAL | 0X006B0004 |
| #define PIC_ID_MIN_BL_12V | 0X006B0005 |
| #define PIC_ID_MIN_BL_24V | 0X006B0006 |
| #define PIC_ID_MIN_BL_DELAY_OFF | 0X006B0007 |
| #define PIC_ID_MIN_BL_HARD_OFF | 0X006B0008 |
| #define PIC_ID_MIN_BAT_LOW_SWITCH | 0X006B0009 |
| #define PIC_ID_MIN_BAT_TYPE | 0X006B000A |
| #define PIC_ID_MAX_IGN_DELAY | 0X006C0000 |
| #define PIC_ID_MAX_DELAY_OFF | 0X006C0001 |
| #define PIC_ID_MAX_HARD_OFF | 0X006C0002 |
| #define PIC_ID_MAX_PWR_RETRIES | 0X006C0003 |
| #define PIC_ID_MAX_PWR_INTERVAL | 0X006C0004 |
| #define PIC_ID_MAX_BL_12V | 0X006C0005 |
| #define PIC_ID_MAX_BL_24V | 0X006C0006 |
| #define PIC_ID_MAX_BL_DELAY_OFF | 0X006C0007 |
| #define PIC_ID_MAX_BL_HARD_OFF | 0X006C0008 |
| #define PIC_ID_MAX_BAT_LOW_SWITCH | 0X006C0009 |
| #define PIC_ID_MAX_BAT_TYPE | 0X006C000A |
| #define PIC_ID_DEF_IGN_DELAY | 0X006D0000 |
| #define PIC_ID_DEF_DELAY_OFF | 0X006D0001 |
| #define PIC_ID_DEF_HARD_OFF | 0X006D0002 |
| #define PIC_ID_DEF_PWR_RETRIES | 0X006D0003 |
| #define PIC_ID_DEF_PWR_INTERVAL | 0X006D0004 |
| #define PIC_ID_DEF_BL_12V | 0X006D0005 |
| #define PIC_ID_DEF_BL_24V | 0X006D0006 |
| #define PIC_ID_DEF_BL_DELAY_OFF | 0X006D0007 |
| #define PIC_ID_DEF_BL_HARD_OFF | 0X006D0008 |
| #define PIC_ID_DEF_BAT_LOW_SWITCH | 0X006D0009 |
| #define PIC_ID_DEF_BAT_TYPE | 0X006D000A |
| #define PIC_ID_EEPROM_DATA1 | 0X006E0000 |

| | |
|---|------------|
| #define PIC_ID_EEPROM_DATA2 | 0X006E0001 |
| #define PIC_ID_EEPROM_DATA3 | 0X006E0002 |
| #define PIC_ID_EEPROM_DATA4 | 0X006E0003 |
| #define PIC_ID_EEPROM_DATA5 | 0X006E0004 |
| #define PIC_ID_EEPROM_DATA6 | 0X006E0005 |
| #define PIC_ID_EEPROM_DATA7 | 0X006E0006 |
| #define PIC_ID_EEPROM_DATA8 | 0X006E0007 |
| #define PIC_ID_EEPROM_DATA9 | 0X006E0008 |
| #define PIC_ID_EEPROM_DATA10 | 0X006E0009 |
| #define PIC_ID_EEPROM_DATA11 | 0X006E000A |
| #define PIC_ID_EEPROM_DATA12 | 0X006E000B |
| #define PIC_ID_EEPROM_DATA13 | 0X006E000C |
| #define PIC_ID_EEPROM_DATA14 | 0X006E000D |
| #define PIC_ID_EEPROM_DATA15 | 0X006E000E |
| #define PIC_ID_SYSTEM_GET_PIC_CHECKSUM | 0X006F0000 |
| #define PIC_ID_SYSTEM_GET_PIC_CONFIG1 | 0X006F0001 |
| #define PIC_ID_SYSTEM_GET_PIC_CONFIG2 | 0X006F0002 |
| #define PIC_ID_SYSTEM_GET_PIC_DEVICE_ID | 0X006F0003 |
| #define PIC_ID_SYSTEM_GET_PIC_USER_ID0 | 0X006F0004 |
| #define PIC_ID_SYSTEM_GET_PIC_USER_ID1 | 0X006F0005 |
| #define PIC_ID_SYSTEM_GET_PIC_USER_ID2 | 0X006F0006 |
| #define PIC_ID_SYSTEM_GET_PIC_USER_ID3 | 0X006F0007 |
| #define PIC_ID_SYSTEM_SET_DEFAULT | 0X006F000E |
| #define PIC_ID_SYSTEM_SET_PIC_RESET | 0X006F000F |

1.2 Device Initialization

2. Functions

The first parameter of *SusiDeviceGetValue* and *SusiDeviceSetValue* both input item id which occupies 4-byte memories. If the first parameter input is not the item ids of LTC4266, then the error code will return SUSI_STATUS_UNSUPPORTED.

2.1 Device Available Information

ID `PIC_ID_INFO_AVAILABLE` can get LTC4266 is available or not. The value of second parameter is 1 or 0 while the device is found and not respectively. The return status code is always SUSI_STATUS_SUCCESS.

2.2 General Information

ID from `PIC_ID_FW_VER` to `PIC_ID_BOARD_NAME4` can get general board information.

| Parameter Value | Return value | Unit | Explanation |
|-----------------------|--------------|-------|------------------------|
| PIC_ID_FW_VER | 0xAD15 | value | (AD=Advantech, 15=v21) |
| PIC_ID_FW_CONFIG_MASK | 0x07FF | value | |
| PIC_ID_BOARD_ID | 0xA008 | value | Board name. |
| PIC_ID_BOARD_NAME_LEN | 8 | value | Get board name length |
| PIC_ID_BOARD_NAME | 'M', 'A' | word | |
| | '-', 'O' | word | |
| | '0', 'P' | word | |
| | '8', '0' | word | |
| | 0xEEEE | | Reserve |

2.3 Switch status

ID from `PIC_ID_SWITCH_STATE` to `PIC_ID_SWITCH2_PWR_SWLECT` can get switch status.

| Parameter Value | Return value | Unit | Explanation |
|---------------------------|--------------|-------|---|
| PIC_ID_SWITCH_STATE | | value | SW2=bit4, SW1=bit<3:0> |
| PIC_ID_SWITCH1_MODE | 0~7 | value | SW1 switch 123 mode |
| PIC_ID_SWITCH1_CFG_SELECT | 0/1 | value | 0=OFF : SW config 1=ON : HW config |
| PIC_ID_SWITCH2_PWR_SELECT | 0/1 | value | 0=OFF : Vechicle mode 1=ON : PC mode |

2.4 F/W information

ID from PIC_ID_FW_STATE to PIC_ID_FW_CHECK_SUM can get F/W information.

| Parameter Value | Return value | Unit | Explanation |
|---------------------------|--------------|-------|-----------------------------|
| PIC_ID_FW_VER | ~ | value | f/w state |
| PIC_ID_FW_SYS_STATUS | ~ | value | f/w system state |
| PIC_ID_FW_BAT_STATUS | ~ | value | f/w battery state |
| PIC_ID_FW_TMR_STATUS | ~ | value | f/w timer state |
| PIC_ID_FW_BAT_TYPE | 0/1/2 | value | 0=ERR, 1=12V, 2=24V |
| PIC_ID_FW_BAT_VOLT | ~ | 100mV | Battery voltage |
| PIC_ID_FW_BAT_VOLT_STATUS | 0/1/2/3 | value | 0=ERR, 1=LOW, 2=OK, 3=FULL |
| PIC_ID_FW_BAT_ADC | 0~1023 | value | Battery ADC |
| PIC_ID_FW_BAT_LOW_ADC | 0~1023 | value | Battery low ADC |
| PIC_ID_FW_SYSON_LEVEL | 0/1 | value | SYS_ON(0=OFF, 1=ON) |
| PIC_ID_FW_IGN_LEVEL | 0/1 | value | IGN_ON(0=IGN_OFF, 1=IGN_ON) |
| PIC_ID_FW_V12_STATUS | 0/1 | value | V12 (0=OFF, 1=ON) |
| PIC_ID_FW_V48_STATUS | 0/1 | value | V48 (0=OFF, 1=ON) |
| PIC_ID_FW_CHECK_SUM | ~ | value | EEPROM check sum |

2.5 H/W config

ID from PIC_ID_HW_TAB_IGN1 to PIC_ID_HW_TAB_DELAY_OFF_8 can get H/W config.

| Parameter Value | Return value | Unit | Explanation |
|----------------------------|--------------|-------|--|
| PIC_ID_HW_TAB_IGN1~8 | ~ | value | IGN on delay time HW config table, mapping to SW1_123 |
| PIC_ID_HW_TAB_DELAY_OFF1~8 | ~ | value | Power off delay time HW config table, mapping to SW1_123 |

2.6 S/W setting

ID from PIC_ID_SET_IGN_DELAY to PIC_ID_SET_BAT_TYPE can get S/W setting.

| Parameter Value | Setting value | Unit | Explanation |
|----------------------|---------------|------|------------------------------|
| PIC_ID_SET_IGN_DELAY | 7~65535 | sec | IGN on delay time |
| PIC_ID_SET_DELAY_OFF | 1~65535 | sec | IGN off Power off delay time |

| | | | |
|---------------------------|---------|-------|----------------------------------|
| PIC_ID_SET_HARD_OFF | 1~65535 | sec | IGN off Hard off delay time |
| PIC_ID_SET_PWR_RETRIES | 1~255 | times | Shutdown retry times |
| PIC_ID_SET_PWR_INTERVAL | 5~65535 | sec | Shutdown interval |
| PIC_ID_SET_BL_12V | 90~119 | 100mV | 12V battery judged low threshold |
| PIC_ID_SET_BL_24V | 210~239 | 100mV | 24V battery judged low threshold |
| PIC_ID_SET_BL_DELAY_OFF | 1~65535 | sec | BAT low Power off delay time |
| PIC_ID_SET_BL_HARD_OFF | 1~65535 | sec | BAT low Hard off delay time |
| PIC_ID_SET_BAT_LOW_SWITCH | 0, 1 | | Check BAT low function switch |
| PIC_ID_SET_BAT_TYPE | 0, 1, 2 | | 0=None, 1=12V, 2=24V |

2.7 Get current setting

ID from `PIC_ID_GET_IGN_DELAY` to `PIC_ID_GET_BAT_TYPE` can get current setting.

| Parameter Value | Return value | Unit | Explanation |
|---------------------------|--------------|-------|----------------------------------|
| PIC_ID_GET_IGN_DELAY | 7~65535 | sec | IGN on delay time |
| PIC_ID_GET_DELAY_OFF | 1~65535 | sec | IGN off Power off delay time |
| PIC_ID_GET_HARD_OFF | 1~65535 | sec | IGN off Hard off delay time |
| PIC_ID_GET_PWR_RETRIES | 1~255 | times | Shutdown retry times |
| PIC_ID_GET_PWR_INTERVAL | 5~65535 | sec | Shutdown interval |
| PIC_ID_GET_BL_12V | 90~119 | 100mV | 12V battery judged low threshold |
| PIC_ID_GET_BL_24V | 210~239 | 100mV | 24V battery judged low threshold |
| PIC_ID_GET_BL_DELAY_OFF | 1~65535 | sec | BAT low Power off delay time |
| PIC_ID_GET_BL_HARD_OFF | 1~65535 | sec | BAT low Hard off delay time |
| PIC_ID_GET_BAT_LOW_SWITCH | 0, 1 | | Check BAT low function switch |
| PIC_ID_GET_BAT_TYPE | 0, 1, 2 | | 0=None, 1=12V, 2=24V |

2.8 Get min setting

ID from `PIC_ID_GET_MIN_IGN_DELAY` to `PIC_ID_GET_MIN_BAT_TYPE` can get min setting.

| Parameter Value | Return value | Unit | Explanation |
|----------------------------|--------------|-------|------------------------------|
| PIC_ID_GET_MIN_IGN_DELAY | 7~65535 | sec | IGN on delay time |
| PIC_ID_GET_MIN_DELAY_OFF | 1~65535 | sec | IGN off Power off delay time |
| PIC_ID_GET_MIN_HARD_OFF | 1~65535 | sec | IGN off Hard off delay time |
| PIC_ID_GET_MIN_PWR_RETRIES | 1~255 | times | Shutdown retry times |

| | | | |
|-------------------------------|---------|-------|----------------------------------|
| PIC_ID_GET_MIN_PWR_INTERVAL | 5~65535 | sec | Shutdown interval |
| PIC_ID_GET_MIN_BL_12V | 90~119 | 100mV | 12V battery judged low threshold |
| PIC_ID_GET_MIN_BL_24V | 210~239 | 100mV | 24V battery judged low threshold |
| PIC_ID_GET_MIN_BL_DELAY_OFF | 1~65535 | sec | BAT low Power off delay time |
| PIC_ID_GET_MIN_BL_HARD_OFF | 1~65535 | sec | BAT low Hard off delay time |
| PIC_ID_GET_MIN_BAT_LOW_SWITCH | 0, 1 | | Check BAT low function switch |
| PIC_ID_GET_MIN_BAT_TYPE | 0, 1, 2 | | 0=None, 1=12V, 2=24V |

2.9 Get max setting

ID from PIC_ID_GET_MAX_IGN_DELAY to PIC_ID_GET_MAX_BAT_TYPE can get max setting.

| Parameter Value | Return value | Unit | Explanation |
|-------------------------------|--------------|-------|----------------------------------|
| PIC_ID_GET_MAX_IGN_DELAY | 7~65535 | sec | IGN on delay time |
| PIC_ID_GET_MAX_DELAY_OFF | 1~65535 | sec | IGN off Power off delay time |
| PIC_ID_GET_MAX_HARD_OFF | 1~65535 | sec | IGN off Hard off delay time |
| PIC_ID_GET_MAX_PWR_RETRIES | 1~255 | times | Shutdown retry times |
| PIC_ID_GET_MAX_PWR_INTERVAL | 5~65535 | sec | Shutdown interval |
| PIC_ID_GET_MAX_BL_12V | 90~119 | 100mV | 12V battery judged low threshold |
| PIC_ID_GET_MAX_BL_24V | 210~239 | 100mV | 24V battery judged low threshold |
| PIC_ID_GET_MAX_BL_DELAY_OFF | 1~65535 | sec | BAT low Power off delay time |
| PIC_ID_GET_MAX_BL_HARD_OFF | 1~65535 | sec | BAT low Hard off delay time |
| PIC_ID_GET_MAX_BAT_LOW_SWITCH | 0, 1 | | Check BAT low function switch |
| PIC_ID_GET_MAX_BAT_TYPE | 0, 1, 2 | | 0=None, 1=12V, 2=24V |

2.10 Get default setting

ID from PIC_ID_GET_DEF_IGN_DELAY to PIC_ID_GET_DEF_BAT_TYPE can get default setting.

| Parameter Value | Return value | Unit | Explanation |
|--------------------------|--------------|------|------------------------------|
| PIC_ID_GET_DEF_IGN_DELAY | 7~65535 | sec | IGN on delay time |
| PIC_ID_GET_DEF_DELAY_OFF | 1~65535 | sec | IGN off Power off delay time |
| PIC_ID_GET_DEF_HARD_OFF | 1~65535 | sec | IGN off Hard off delay time |

| | | | |
|-------------------------------|---------|-------|----------------------------------|
| PIC_ID_GET_DEF_PWR_RETRIES | 1~255 | times | Shutdown retry times |
| PIC_ID_GET_DEF_PWR_INTERVAL | 5~65535 | sec | Shutdown interval |
| PIC_ID_GET_DEF_BL_12V | 90~119 | 100mV | 12V battery judged low threshold |
| PIC_ID_GET_DEF_BL_24V | 210~239 | 100mV | 24V battery judged low threshold |
| PIC_ID_GET_DEF_BL_DELAY_OFF | 1~65535 | sec | BAT low Power off delay time |
| PIC_ID_GET_DEF_BL_HARD_OFF | 1~65535 | sec | BAT low Hard off delay time |
| PIC_ID_GET_DEF_BAT_LOW_SWITCH | 0, 1 | | Check BAT low function switch |
| PIC_ID_GET_DEF_BAT_TYPE | 0, 1, 2 | | 0=None, 1=12V, 2=24V |

2.11 Timer setting

ID from PIC_ID_GET_DEF_IGN_DELAY to PIC_ID_GET_DEF_BAT_TYPE can set timer setting.

| Parameter Value | Setting value | Unit | Explanation |
|--------------------------------|---------------|-------|-------------|
| PIC_ID_GET_TMR_IGN_ON | 0~ | sec | |
| PIC_ID_GET_PWR_ON_RETRIES | 0~3 | times | |
| PIC_ID_GET_PWR_ON_INTERVAL | 0~5 | sec | |
| PIC_ID_GET_PWR_OFF_RETRIES | 0~255 | times | |
| PIC_ID_GET_PWR_OFF_INTERVAL | 0~65535 | sec | |
| PIC_ID_GET_TMR_DELAY_OFF | 0~65535 | sec | |
| PIC_ID_GET_TMR_HARD_OFF | 0~65535 | sec | |
| PIC_ID_GET_TMR_BL_DELAY_OFF | 0~65535 | sec | |
| PIC_ID_GET_PWR_12V48V_INTERVAL | 0~3 | sec | |
| PIC_ID_GET_TMR_FW_UP_TIME | 0~ | sec | |

2.12 EEPROM data

ID from PIC_ID_EEPROM_DATA1 to PIC_ID_EEPROM_DATA15 can get EEPROM data.

| Parameter Value | Return value | Unit | Explanation |
|----------------------------|--------------|------|-------------------------------|
| PIC_ID_GET_EEPROM_DATA1~15 | ~ | word | Default setting value (E0~EA) |

2.13 System Command

ID from PIC_ID_GET_PIC_CHECKSUM to PIC_ID_SET_PIC_RESET are system command.

| Parameter Value | Setting value | Unit | Explanation |
|------------------------------|---------------|------|---------------------------|
| PIC_ID_GET_PIC_CHECKSUM | | | PIC f/w check sum |
| PIC_ID_GET_PIC_CONFIG1 | | | PIC config1 |
| PIC_ID_GET_PIC_CONFIG2 | | | PIC config2 |
| PIC_ID_GET_PIC_DEVICE_ID | 0x27C5 | | Microchip PIC MCU |
| PIC_ID_GET_PIC_USER_ID (0~3) | | | |
| PIC_ID_SET_DEFAULT | | | Reset default setting |
| PIC_ID_SET_PIC_RESET | | | PIC MCU reset after 2 sec |