|  |
| --- |
| hkpu |
| Energy Monitor System Protocol |
| [Type the document subtitle] |
|  |
| **hkpu** |
| **08/11/2011** |

|  |
| --- |
| [Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. Type the abstract of the document here. The abstract is typically a short summary of the contents of the document.] |

**Table of Contents**

[1 Revision History 4](#_Toc308539976)

[2 Introduction 5](#_Toc308539977)

[2.1 Purpose 5](#_Toc308539978)

[2.2 Scope 5](#_Toc308539979)

[3 Introduction to Energy Monitoring System 5](#_Toc308539980)

[3.1 Introduction 5](#_Toc308539981)

[3.2 ARM device 6](#_Toc308539982)

[3.3 Panel 6](#_Toc308539983)

[4 MZ layer 7](#_Toc308539984)

[4.1 Introduction 7](#_Toc308539985)

[4.2 Data packet format 7](#_Toc308539986)

[4.2.1 Coordinator to Middleware 7](#_Toc308539987)

[4.2.2 Middleware to Coordinator 8](#_Toc308539988)

[5 ZA layer 9](#_Toc308539989)

[5.1 Introduction 9](#_Toc308539990)

[5.2 Hardware Setup 9](#_Toc308539991)

[5.3 Connection Diagram 9](#_Toc308539992)

[5.4 Data packet format 10](#_Toc308539993)

[5.4.1 AVR to/from Zigbee 10](#_Toc308539994)

[6 Message Type 11](#_Toc308539995)

[6.1 Introduction 11](#_Toc308539996)

[6.2 Message Type List 11](#_Toc308539997)

[6.3 End Device Message - Meter 13](#_Toc308539998)

[6.3.1 MSG\_TYPE\_METER\_INT 13](#_Toc308539999)

[6.3.2 MSG\_TYPE\_METER\_LONG 13](#_Toc308540000)

[6.3.3 MSG\_TYPE\_RAW\_DATA 14](#_Toc308540001)

[6.3.4 MSG\_TYPE\_METER\_CHAR 15](#_Toc308540002)

[6.3.5 MSG\_TYPE\_METER\_INTERVAL\_INT 16](#_Toc308540003)

[6.3.6 MSG\_TYPE\_METER\_INTERVAL\_LONG 17](#_Toc308540004)

[6.3.7 MSG\_TYPE\_ INTERVAL\_RAW\_DATA 18](#_Toc308540005)

[6.3.8 MSG\_TYPE\_ INTERVAL\_RS485 20](#_Toc308540006)

[6.4 End Device Message - Panel 22](#_Toc308540007)

[6.4.1 MSG\_TYPE\_PENAL\_GENERAL 22](#_Toc308540008)

[6.4.2 MSG\_TYPE\_PENAL\_PREHEAT 23](#_Toc308540009)

[6.4.3 MSG\_TYPE\_PENAL\_REWORK 24](#_Toc308540010)

[6.4.1 MSG\_TYPE\_PENAL\_INPUTQTY 26](#_Toc308540011)

[6.4.1 MSG\_TYPE\_PENAL\_COMBINE 27](#_Toc308540012)

[6.4.2 MSG\_TYPE\_PENAL\_ SEPERATE 29](#_Toc308540013)

[6.4.3 MSG\_TYPE\_PENAL\_PREPARE 31](#_Toc308540014)

[6.4.4 MSG\_TYPE\_PENAL\_NETWORKTEST 33](#_Toc308540015)

[6.5 Zigbee remote ctrl 34](#_Toc308540016)

[6.5.1 CTRL\_TYPE\_ZIGBEE\_PING 34](#_Toc308540017)

[6.5.2 CTRL\_TYPE\_ZIGBEE\_NV\_WRITE 35](#_Toc308540018)

[6.6 AVR remote ctrl 37](#_Toc308540019)

[6.6.1 CTRL\_TYPE\_EEPROM\_WRITE\_INT 37](#_Toc308540020)

[6.6.2 CTRL\_TYPE\_EEPROM\_ WRITE\_STR 38](#_Toc308540021)

[6.6.1 CTRL\_TYPE \_PENAL\_REQUEST 39](#_Toc308540022)

[7 Definition 39](#_Toc308540023)

[7.1 Panel Configuration 39](#_Toc308540024)

[7.2 Zigbee NV memory List 40](#_Toc308540025)

[7.3 AVR EEPROM List (End device) 40](#_Toc308540026)

# Revision History

|  |  |  |
| --- | --- | --- |
| **Version** | **Description** | **Data** |
| 1.0 | Initial release | 08/11/2011 |
| 1.1 | Data packet format for M to C information corrected | 02/02/2012 |

# Introduction

## Purpose

This document is written to explain the communication protocol in PolyU energy project. The protocol is used to MZ layer(Middleware and Zigbee) and ZA layer(Zigbee and AVR).

## Scope

This document is limited to developing applications using PolyU energy monitoring system.

# Introduction to Energy Monitoring System

## Introduction

The system is used to monitor the energy using status in factory. Hardware of the system includes server, coordinator, Router and End Device. End Device includes ARM devices and Panel. ARM devices collect meter reading information. Panel collect production information form worker entering. Router is used to increasing the network cover size. Coordinator collect all information form ARM devices and Panel and pass data to server.

RS232

Server

Coordinator

Router

Router

ARM

Different Type of Energy Meter

ARM

Different Type of Energy Meter

Physical Connection

Wireless Connection

## ARM device

ARM device is used to collect reading from different type of energy meter, for example, electric meter, water meter, steam meter, yard meter, etc.

ARM device includes two microprocessor, Zigbee and AVR. AVR is used to collect meter reading form special interface. Zigbee is used to manage wireless network. Zigbee request AVR for meter reading and then send it to the coordinator.

Zigbee

AVR

Interface

Meter

## Panel

Panel is used to collect production information form worker. For example, worker began do his job. He input order number into the panel. Panel send the information to coordinator and record the start time of the production.

As same as the ARM device, it includes two microprocessor, Zigbee and AVR. AVR is used to hold human interface. Display information to work. Guide worker how to use the Panel. Receive information form worker input and then send it to Zigbee. Zigbee is used to manage wireless network. It send data to coordinator.

Zigbee

AVR

Display

Input Device

# MZ layer

## Introduction

MZ layer is between middleware(Server) and the coordinator. The protocol is design to communication between middleware and coordinator.

## Data packet format

### Coordinator to Middleware

#### Packet Structure

|  |  |  |  |
| --- | --- | --- | --- |
| Start Byte [3] | End Device ID [8] | Meter ID [1] | Len [1] |
| Type [1] | Payload [Len -1] | CheckSum [1] | Stop Byte [2] |

#### Parameter Details

|  |  |  |
| --- | --- | --- |
| **Name** | **Range** | **Representation** |
| Start Byte | ASCII ($ST) | Identify the beginning of the packet |
| End Device ID | 0~0xFFFFFFFFFFFFFFFF | 64bits Zigbee Network Address, Hex |
| Meter ID | 1~255 | Sub address in a End Device ID |
| Len | 1~255 | Sum of length of **Type** and **Payload** |
| Type | 0~255 | Message **Type**, refer to P.10 Table 1 |
| Payload | 250 bytes | Packet Content |
| CheckSum | 0~255 | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |
| Stop Byte | 0D 0A | Identify the end of the packet |

#### Example

24 53 54 00 12 4B 00 01 0B C8 77 01 05 A1 04 01 01 02 22 0D 0A

#### Description

|  |  |
| --- | --- |
| **Sample** | **Description** |
| 24 53 54 | **Start Byte**, ASCII ($ST) |
| 00 12 4B 00 01 0B C8 77 | **End Device ID**, Hex |
| 01 | **Meter ID**, Hex |
| 05 | Message **Len**, Hex, Sum of len of **Type** and **Payload** |
| A1 | Message **Type**, refer to P.10 Table 1 |
| 04 01 01 02 | **Payload**, according to different msg type, it has different meaning. |
| 22 | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |
| 0D 0A | **Stop Byte** |

### Middleware to Coordinator

#### Packet Structure

|  |  |  |
| --- | --- | --- |
| Start Byte [1] | Len [1] | End Device ID [8] |
| Type [1] | Payload [Len -2] | CheckSum [1] |

#### Parameter Details

|  |  |  |
| --- | --- | --- |
| **Name** | **Range** | **Representation** |
| Start Byte | FE | Identify the beginning of the packet |
| Len | 1~255 | Sum of length of **End Device ID**, **Type** and **Payload** |
| End Device ID | 0~0xFFFFFFFFFFFFFFFF | 64bits Zigbee Network Address, Hex |
| Type | 0~255 | Message **Type**, refer to P.10 Table 1 |
| Payload | 0~250 bytes | Packet Content |
| CheckSum | 0~255 | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |

#### Example

FE 11 00 12 4B 00 01 0B C7 36 C1 00 00 01 4B 00 02 01 28 AD

#### Description

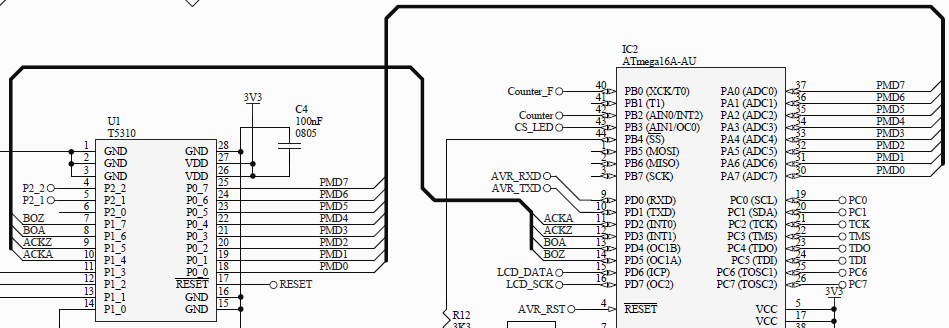
|  |  |
| --- | --- |
| **Sample** | **Description** |
| FE | **Start Byte** |
| 11 | Message **Len**, Hex, Sum of len of **End Device ID**, **Type** and **Payload** |
| 00 12 4B 00 01 0B C7 36 | **End Device ID**, Hex |
| C1 | Message **Type**, refer to P.10 Table 1 |
| 00 00 01 4B 00 02 01 28 | **Payload**, according to different msg type, it has different meaning. |
| AD | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |

# ZA layer

## Introduction

ZA layer is between Zigbee and AVR. The protocol is design to communication between Zigbee and AVR.

## Hardware Setup



## Connection Diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CC2530** | **Function** | **ATmega16L** | | **Description** |
| **Main Board** | **Panel** |
| **P0\_7** | PMD7 | PA7 | PA0 | Parallel Data Port |
| **P0\_6** | PMD6 | PA6 | PA1 |
| **P0\_5** | PMD5 | PA5 | PA2 |
| **P0\_4** | PMD4 | PA4 | PA3 |
| **P0\_3** | PMD3 | PA3 | PA4 |
| **P0\_2** | PMD2 | PA2 | PA5 |
| **P0\_1** | PMD1 | PA1 | PA6 |
| **P0\_0** | PMD0 | PA0 | PA7 |

|  |  |  |  |
| --- | --- | --- | --- |
| **CC2530** | **Function** | **ATmega16L** | **Description** |
| **P1\_7** | BOZ | PD5 | Zigbee request Bus Occupy, 0=Active |
| **P1\_6** | BOA | PD4 | AVR request Bus Occupy, 0=Active |
| **P1\_5** | ACKZ | PD3 | Zigbee Acknowledge, 0=Active |
| **P1\_4** | ACKA | PD2 | AVR Acknowledge, 0=Active |

## Data packet format

### AVR to/from Zigbee

#### Packet Structure

|  |  |  |  |
| --- | --- | --- | --- |
| Len [1] | Type[1] | Payload [Len -2] | CheckSum [1] |

#### Parameter Details

|  |  |  |
| --- | --- | --- |
| **Name** | **Range** | **Representation** |
| Len | 1~255 | Sum of Len of **Type**, **Payload** and **CheckSum** |
| Type | 0~255 | Message **Type**, refer to P.10 Table 1 |
| Payload | 0~250 bytes | Packet Content |
| CheckSum | 0~255 | AVR Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |

#### Example

06 A1 04 01 01 02 AF

#### Description

|  |  |
| --- | --- |
| **Sample** | **Description** |
| 06 | Message **Len**, Hex, Sum of Len of **Type**, **Payload** and **CheckSum** |
| A1 | Message **Type**, refer to P.10 Table 1 |
| 04 01 01 02 | **Payload**, according to different msg type, it has different meaning. |
| AF | AVR Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |

# Message Type

## Introduction

For hardware ,there are 4 main component: Server, Coordinator, Zigbee and AVR. According to the **Type** in the message packet, they can get information directly, correctly and efficiently.

## Message Type List

| **Hex** | **Type** | **Device** | **Name** | **Comment** | **Direction** |
| --- | --- | --- | --- | --- | --- |
| 0x00 | AVR internal msg |  | RECEIVE\_TYPE\_NO\_MSG | AVR sub problem return that no message is received | AVR -> AVR |
| 0x01 | End Device msg | Meter | [MSG\_TYPE\_METER\_INT](#_MSG_TYPE_METER_INT) | int format data | AVR -> Coordinator |
| 0x02 | End Device msg | Meter | [MSG\_TYPE\_METER\_CHAR](#_MSG_TYPE_METER_CHAR) | char format data | AVR -> Coordinator |
| 0x03 | End Device msg |  | MSG\_TYPE\_METER\_HEX |  | AVR -> Coordinator |
| 0x04 | End Device msg |  | MSG\_TYPE\_RFID |  | AVR -> Coordinator |
| 0x05 | End Device msg |  | MSG\_TYPE\_KEY\_PRESS |  | AVR -> Coordinator |
| 0x06 | End Device msg |  | MSG\_TYPE\_LCD\_DISPLAY |  | AVR -> Coordinator |
| 0x07 | End Device msg | Panel | [MSG\_TYPE\_PENAL\_GENERAL](#_MSG_TYPE_PENAL_GENERAL) | panel general order message | AVR -> Coordinator |
| 0x08 | End Device msg | Panel | [MSG\_TYPE\_PENAL\_PREHEAT](#_MSG_TYPE_PENAL_PREHEAT) | panel preheat message | AVR -> Coordinator |
| 0x09 | End Device msg | Panel | [MSG\_TYPE\_PENAL\_REWORK](#_MSG_TYPE_PENAL_REWORK) | panel rework message | AVR -> Coordinator |
| 0x0A | End Device msg | Panel | [MSG\_TYPE\_PENAL\_INPUTQTY](#_MSG_TYPE_PENAL_INPUTQTY) | panel qty update message | AVR -> Coordinator |
| 0x0B | End Device msg | Meter | [MSG\_TYPE\_METER\_LONG](#_MSG_TYPE_METER_LONG) | long format data | AVR -> Coordinator |
| 0x0C | End Device msg | Panel | [MSG\_TYPE\_PENAL\_COMBINE](#_MSG_TYPE_PENAL_COMBINE) | panel combine message | AVR -> Coordinator |
| 0x0D | End Device msg | Panel | [MSG\_TYPE\_PENAL\_SEPERATE](#_MSG_TYPE_PENAL__SEPERATE) | panel seperate message | AVR -> Coordinator |
| 0x0E | End Device msg | Panel | [MSG\_TYPE\_PENAL\_PREPARE](#_MSG_TYPE_PENAL_PREPARE) | panel prepare message | AVR -> Coordinator |
| 0x0F | End Device msg | Panel | Preserved for Panel |  | AVR -> Coordinator |
| 0x10 | End Device msg | Panel | Preserved for Panel |  | AVR -> Coordinator |
| 0x11 | End Device msg | Panel | Preserved for Panel |  | AVR -> Coordinator |
| 0x12 | End Device msg | Panel | Preserved for Panel |  | AVR -> Coordinator |
| 0x13 | End Device msg | Panel | Preserved for Panel |  | AVR -> Coordinator |
| 0x14 | End Device msg | Panel | [MSG\_TYPE\_PENAL\_NETWORKTEST](#_MSG_TYPE_PENAL_NETWORKTEST) | Panel heart beat signal (5mins) | AVR -> Coordinator |
| 0x15 | End Device msg | Meter | [MSG\_TYPE\_METER\_INTERVAL\_LONG](#_MSG_TYPE_METER_INTERVAL_LONG) | long format data (multi record per packet) | AVR -> Coordinator |
| 0x16 | End Device msg | Meter | [MSG\_TYPE\_METER\_INTERVAL\_INT](#_MSG_TYPE_METER_INTERVAL_INT) | int format data (multi record per packet) | AVR -> Coordinator |
| 0x17 | End Device msg | Meter | [MSG\_TYPE\_METER\_INTERVAL\_RS485](#_MSG_TYPE__INTERVAL_RS485) | RS485 format data (multi record per packet) | AVR -> Coordinator |
| ... | ... | ... | ... | ... | ... |
| 0xA0 | Zigbee remote ctrl |  | [CTRL\_TYPE\_ZIGBEE\_PING](#_CTRL_TYPE_ZIGBEE_PING) | Coordinator ping Zigbee device | Coordinator <-> Zigbee |
| 0xA1 | Zigbee remote ctrl |  | [CTRL\_TYPE\_ZIGBEE\_NV\_WRITE](#_CTRL_TYPE_ZIGBEE_NV_WRITE) | remote write value to Zigbee NV | Coordinator <-> Zigbee |
| ... | ... | ... | ... | ... | ... |
| 0xC1 | AVR remote ctrl |  | [CTRL\_TYPE\_EEPROM\_WRITE\_INT](#_CTRL_TYPE_EEPROM_WRITE_INT) | remote write value to AVR EEPROM | Coordinator <-> AVR |
| 0xC2 | AVR remote ctrl |  | [CTRL\_TYPE\_EEPROM\_WRITE\_STR](#_CTRL_TYPE_EEPROM__WRITE_STR) | remote write string to AVR EEPROM | Coordinator <-> AVR |
| 0xC3 | AVR remote ctrl | Panel | [CTRL\_TYPE\_PENAL\_REQUEST](#_CTRL_TYPE__PENAL_REQUEST) | request Panel setting form PC | Coordinator <-> AVR |
| ... | ... | ... | ... | ... | ... |
| 0xD0 | ? |  | MSG\_TYPE\_RESET |  |  |
| 0xD1 | End Device msg | Meter | [MSG\_TYPE\_RAW\_DATA](#_MSG_TYPE_RAW_DATA) | CS detail message | AVR -> Coordinator |
| 0xD3 | End Device msg | Meter | [MSG\_TYPE\_INTERVAL\_RAW\_DATA](#_MSG_TYPE__INTERVAL_RAW_DATA) | CS detail message ( Interval ) | AVR -> Coordinator |
| ... | ... | ... | ... | ... | ... |
| 0xF0 | Zigbee & AVR Com | Panel | CTRL\_TYPE\_SEND\_FAIL | Zigbee send to AVR if send message to Corrdinator fail | Zigbee -> AVR |
| 0xF1 | Zigbee & AVR Com | Panel | CTRL\_TYPE\_SEND\_SUCCESS | Zigbee send to AVR if send message to Corrdinator success | Zigbee -> AVR |
| 0xFE | --- | --- | --- | start bytes for message that coordinator send to PC | Coordinator -> PC |
| 0xFF | Zigbee & AVR Com |  | CTRL\_TYPE\_METER\_REQUEST | Zigbee sent to AVR request Data | Zigbee -> AVR |
| 0xFD | Zigbee & AVR Com |  | CTRL\_TYPE\_METER\_INTERVAL\_REQUEST | Zigbee sent to AVR request Interval Data (minute-based) | Zigbee -> AVR |

Table Message Type Table (2/11/2011)

## End Device Message - Meter

### MSG\_TYPE\_METER\_INT

#### Description

Meter data sent to Coordinator by **integer format**.

#### Packet Structure

|  |
| --- |
| Data [2] |

#### Parameter Details

|  |  |  |
| --- | --- | --- |
| **Name** | **Range** | **Representation** |
| Data | 0~0xFFFF | Count, Little Endianness , Server hold the overflow |

#### Example

24 53 54 00 12 4B 00 01 0B C8 77 01 03 01 01 02 7B 0D 0A

#### Description

|  |  |
| --- | --- |
| **Sample** | **Description** |
| 24 53 54 | **Start Byte**, ASCII ($ST) |
| 00 12 4B 00 01 0B C8 77 | **End Device ID**, Hex |
| 01 | **Meter ID**, Hex |
| 03 | Message **Len**, fixed length 3 bytes |
| 01 | Message **Type**, 0x01 = **MSG\_TYPE\_METER\_INT** |
| 01 02 | **Data**, Little Endianness, the value = 0x0201=513 (dec) |
| 7B | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |
| 0D 0A | **Stop Byte** |

### MSG\_TYPE\_METER\_LONG

#### Description

Meter data sent to Coordinator by **long format**.

#### Packet Structure

|  |
| --- |
| Data [4] |

#### Parameter Details

|  |  |  |
| --- | --- | --- |
| **Name** | **Range** | **Representation** |
| Data | 0~0xFFFFFFFF | Count, Big Endianness , Server hold the overflow |

#### Example

24 53 54 00 12 4B 00 01 0B C8 77 01 05 0B 01 02 03 04 8E 0D 0A

#### Description

|  |  |
| --- | --- |
| **Sample** | **Description** |
| 24 53 54 | **Start Byte**, ASCII ($ST) |
| 00 12 4B 00 01 0B C8 77 | **End Device ID**, Hex |
| 01 | **Meter ID**, Hex |
| 05 | Message **Len**, fixed length 5 bytes |
| 0B | Message **Type**, 0x0B = **MSG\_TYPE\_METER\_LONG** |
| 01 02 03 04 | **Data**, Big Endianness, the value = 16909060 (Dec) |
| 8E | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |
| 0D 0A | **Stop Byte** |

### MSG\_TYPE\_RAW\_DATA

#### Description

Meter data sent to Coordinator by **special format for CS module**.

#### Packet Structure

|  |  |  |  |
| --- | --- | --- | --- |
| Max [2] | Min [2] | Freq[2] | Count [2] |
| PH [2] | PL [2] | TH [2] | TL [2] |
| AVG [2] | TP\_Max [2] | TP\_Min [2] |  |

#### Parameter Details

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Range** | **Representation** | **Endianness** |
| Max | 0~0xFFFF | Maximum value of CS scanning | Little |
| Min | 0~0xFFFF | Minimum value of CS scanning | Little |
| Freq | 0~0xFFFF | CS value at sending time | Little |
| Count | 0~0xFFFF | Count, Server hold the overflow | Little |
| PH | 0~0xFFFF | Pervious count period max sensor value | Little |
| PL | 0~0xFFFF | Pervious count period min sensor value | Little |
| TH | 0~0xFFFF | Threshold high | Little |
| TL | 0~0xFFFF | Threshold low | Little |
| AVG | 0~0xFFFF | Average value of CS scanning | Little |
| TP\_Max | 0~0xFFFF | Time period max sensor value | Little |
| TP\_Min | 0~0xFFFF | Time period max sensor value | Little |

#### Example

24 53 54 00 12 4B 00 01 0B D4 C4 01 17 D1

78 00 01 00 06 00 02 00 5E 00 1D 00 51 00 28 00 00 00 79 00 00 00

A3 0D 0A

#### Description

|  |  |
| --- | --- |
| **Sample** | **Description** |
| 24 53 54 | **Start Byte**, ASCII ($ST) |
| 00 12 4B 00 01 0B D4 C4 | **End Device ID**, Hex |
| 01 | **Meter ID**, Hex |
| 17 | Message **Len**, fixed length 23 bytes |
| D1 | Message **Type**, 0xD1 = **MSG\_TYPE\_RAW\_DATA** |
| 78 00 | **Max** |
| 01 00 | **Min** |
| 06 00 | **Freq** |
| 02 00 | **Count** |
| 5E 00 | **PH** |
| 1D 00 | **PL** |
| 51 00 | **TH** |
| 28 00 | **TL** |
| 00 00 | **AVG** |
| 79 00 | **TP\_Max** |
| 00 00 | **TP\_Min** |
| 8E | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |
| 0D 0A | **Stop Byte** |

### MSG\_TYPE\_METER\_CHAR

#### Description

Meter data sent to Coordinator by **special format**. It is normally used to RS485 connection meter. Server get data according to the configuration form database. Additional documentation provided such as **RS485 data format** take a more in depth look at application development for RS485 connection meter.

#### Packet Structure

|  |
| --- |
| Data [Len-2] |

#### Parameter Details

|  |  |  |
| --- | --- | --- |
| **Name** | **Range** | **Representation** |
| Data | 0~100bytes | According to the configuration in server data base to explain the value. Further information refer to additional document **RS485 data format** |

#### Example

24 53 54 00 12 4B 00 01 0B C8 77 01 0A 02 01 04 06 43 00 00 00 04 41 13 0D 0A

#### Description

|  |  |
| --- | --- |
| **Sample** | **Description** |
| 24 53 54 | **Start Byte**, ASCII ($ST) |
| 00 12 4B 00 01 0B C8 77 | **End Device ID**, Hex |
| 01 | **Meter ID**, Hex |
| 0A | Message **Len**, Hex, variable length, Sum of len of **Type** and **Payload** |
| 02 | Message **Type**, 0x02 = **MSG\_TYPE\_METER\_CHAR** |
| 01 04 06 43 00 00 00 04 41 | **Data**, according to the configuration in server data base to explain the value. Further information refer to additional document **RS485 data format** |
| 13 | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |
| 0D 0A | **Stop Byte** |

### MSG\_TYPE\_METER\_INTERVAL\_INT

#### Description

End Device send **multi records in a packet** to Coordinator by **Integer format**.

#### Packet Structure

|  |  |  |
| --- | --- | --- |
| Interval[1] | Data [2] | **Next** Data... [2] |

#### Parameter Details

|  |  |  |
| --- | --- | --- |
| **Name** | **Range** | **Representation** |
| Interval | 0~10 | Interval (Number of data stored), Hex |
| Data | 0~0xFFFF | Count, Big Endianness , Server hold the overflow |
| **Next** Data...[Optional] | 0~0xFFFF | Optional. if more than one record to be sent in a packet, it is repeatedly contained in the packet. |

#### Example

24 53 54 00 12 4B 00 01 0B D4 C4 01 0C 16

05

02 00

02 00

02 00

02 00

02 00

FE 0D 0A

#### Description

|  |  |
| --- | --- |
| **Sample** | **Description** |
| 24 53 54 | **Start Byte**, ASCII ($ST) |
| 00 12 4B 00 01 0B D4 C4 | **End Device ID**, Hex |
| 01 | **Meter ID**, Hex |
| 0C | Message **Len**, Hex, variable length, Sum of len of **Type** and **Payload** |
| 16 | Message **Type**, 0x16 = **MSG\_TYPE\_METER\_INTERVAL\_INT** |
| 05 | Interval (Number of data stored), Hex |
| 02 00 | **Data**, Little Endianness, the sample value = 2 (Dec) |
| FE | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |
| 0D 0A | **Stop Byte** |

### MSG\_TYPE\_METER\_INTERVAL\_LONG

#### Description

End Device send **multi records in a packet** to Coordinator by **long format**.

#### Packet Structure

|  |  |  |
| --- | --- | --- |
| Interval[1] | Data [4] | **Next** Data... [4] |

#### Parameter Details

|  |  |  |
| --- | --- | --- |
| **Name** | **Range** | **Representation** |
| Interval | 0~10 | Interval (Number of data stored), Hex |
| Data | 0~0xFFFFFFFF | Count, Big Endianness , Server hold the overflow |
| **Next** Data...[Optional] | 0~0xFFFFFFFF | Optional. if more than one record to be sent in a packet, it is repeatedly contained in the packet. |

#### Example

24 53 54 00 12 4B 00 01 0B D4 C4 01 16 15

05

21 00 00 00

3F 00 00 00

3F 00 00 00

40 00 00 00

41 00 00 00

1D 0D 0A

#### Description

|  |  |
| --- | --- |
| **Sample** | **Description** |
| 24 53 54 | **Start Byte**, ASCII ($ST) |
| 00 12 4B 00 01 0B D4 C4 | **End Device ID**, Hex |
| 01 | **Meter ID**, Hex |
| 16 | Message **Len**, Hex, variable length, Sum of len of **Type** and **Payload** |
| 15 | Message **Type**, 0x15 = **MSG\_TYPE\_METER\_INTERVAL\_LONG** |
| 05 | Interval (Number of data stored), Hex |
| 21 00 00 00 | **Data**, Little Endianness, the sample value = 33 (Dec) |
| D9 | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |
| 0D 0A | **Stop Byte** |

### MSG\_TYPE\_ INTERVAL\_RAW\_DATA

#### Description

End Device send **multi records in a packet** to Coordinator by **special format for CS module**.

#### Packet Structure

|  |  |  |  |
| --- | --- | --- | --- |
| Interval [1] | Max [2] | Min [2] | TH [2] |
| TL [2] | AVG [2] | TP\_Max [2] | TP\_Min [2] |
| Count [2] | **Next** Count... [2] | | |

#### Parameter Details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Range** | **Representation** | **Endian** | **Buf[n]** |
| Interval | 0~10 | **Interval** (Number of data stored), Hex | **-** | 0 |
| Max | 0~0xFFFF | Maximum value of CS scanning | Little | 1 |
| Min | 0~0xFFFF | Minimum value of CS scanning | Little | 3 |
| TH | 0~0xFFFF | Threshold high | Little | 5 |
| TL | 0~0xFFFF | Threshold low | Little | 7 |
| AVG | 0~0xFFFF | Average value of CS scanning | Little | 9 |
| TP\_Max | 0~0xFFFF | Time period max sensor value | Little | 11 |
| TP\_Min | 0~0xFFFF | Time period max sensor value | Little | 13 |
| Count | 0~0xFFFF | **Count**, Server hold the overflow | Little | 15 |
| **Next** Count...  [Optional] | 0~0xFFFF | Optional. If more than one record (**Interval** >1) to be sent in a packet, it is repeatedly contained in the packet. | Little | 15+2... |

#### Example

24 53 54 00 12 4B 00 01 0B D4 C4 01 1A D3

05 78 00 01 00 51 00 28 00 00 00 79 00 00 00

02 00

02 00

02 00

02 00

02 00

34 0D 0A

#### Description

|  |  |
| --- | --- |
| **Sample** | **Description** |
| 24 53 54 | **Start Byte**, ASCII ($ST) |
| 00 12 4B 00 01 0B D4 C4 | **End Device ID**, Hex |
| 01 | **Meter ID**, Hex |
| 1A | Message **Len**, Hex, variable length, Sum of len of **Type** and **Payload** |
| D3 | Message **Type**, 0xD3 = **MSG\_TYPE\_ INTERVAL\_RAW\_DATA** |
| 78 00 | **Max** |
| 01 00 | **Min** |
| 51 00 | **TH** |
| 28 00 | **TL** |
| 00 00 | **AVG** |
| 79 00 | **TP\_Max** |
| 00 00 | **TP\_Min** |
| 02 00 | **Count** |
| 02 00 02 00 02 00 02 00 | **Next Count**, the sample contains 4 "Next Count", Total 5 **Count** |
| 34 | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |
| 0D 0A | **Stop Byte** |

### MSG\_TYPE\_ INTERVAL\_RS485

#### Description

End Device send **multi records in a packet** to Coordinator by **special format**. It is normally used to RS485 connection meter. Server get data according to the configuration form database. Additional documentation provided such as **RS485 data format** take a more in depth look at application development for RS485 connection meter.

#### Packet Structure

|  |  |  |
| --- | --- | --- |
| Interval[1] | Data [**N**] | **Next** Data... [**N**] |

\*\* **N** = (Message **Len** - 2) / **Interval**

#### Parameter Details

|  |  |  |
| --- | --- | --- |
| **Name** | **Range** | **Representation** |
| Interval | 0~10 | Interval (Number of data stored), Hex |
| Data | N bytes | According to the configuration in server data base to explain the value. Further information refer to additional document **RS485 data format**.  Variable Length, the Data length **N** = (Message **Len** - 2) / **Interval** |
| **Next** Data...[Optional] | N bytes | Optional. if more than one record to be sent in a packet, it is repeatedly contained in the packet. |

#### Example

24 53 54 00 12 4B 00 01 0B D4 C4 01 16 17

05

02 15 01 0A

02 15 01 0A

02 14 01 0A

02 14 01 0A

02 15 01 0A

A6 0D 0A

#### Description

|  |  |
| --- | --- |
| **Sample** | **Description** |
| 24 53 54 | **Start Byte**, ASCII ($ST) |
| 00 12 4B 00 01 0B D4 C4 | **End Device ID**, Hex |
| 01 | **Meter ID**, Hex |
| 16 | Message **Len**, Hex, variable length, Sum of len of **Type** and **Payload**, the sample value = 22 (Dec) |
| 17 | Message **Type**, 0x15= **MSG\_TYPE\_METER\_INTERVAL\_RS485** |
| 05 | **Interval** (Number of data stored), Hex |
| 02 15 01 0A | According to the configuration in server data base to explain the value. Further information refer to additional document **RS485 data format**. Variable Length.  The Data length **N** = (Message **Len** - 2) / **Interval**  For this sample, **N** = (22-2)/5 = 4 (bytes) |
| A6 | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |
| 0D 0A | **Stop Byte** |

## End Device Message - Panel

### MSG\_TYPE\_PENAL\_GENERAL

#### Description

Record general production beginning and ending.

#### Packet Structure

|  |  |  |  |
| --- | --- | --- | --- |
| Time Out[1] | Staff Card ID[10] | Input Method[1] | Flow Direction[1] |
| Total Order Card[1] | Order Card ID[10] | Input Method[1] | QTY [2] |
| **Next** Order Card ID[10]… [Optional] | | **Next** Input Method[1]  ...[Optional] | **Next** QTY [2]...  [Optional] |

#### Parameter Details

|  |  |  |
| --- | --- | --- |
| **Name** | **Range** | **Representation** |
| Time Out | 0 | Message sent due to user complete input. |
| 1 | Message sent due to Time Out. |
| Staff Card ID | 0x30~0x39, ASCII(0~9)  fixed Length 10 bytes | Staff Card ID |
| Input Method | 0x55, ASCII(U) | Staff card is inputted by UART, eg. RFID |
| 0x4B, ASCII(K) | Staff card is inputted by Keypad |
| Flow Direction | 0x30, ASCII(0) | Output Direction, Production Beginning |
| 0x31, ASCII(1) | Input Direction, Production Ending |
| Total Order Card | 1~5 | Number of Order Card contain in the packet.  Each order card record contains **Order Card ID**, **Input Method** and **QTY** |
| Order Card ID | 0x30~0x39, ASCII(0~9)  fixed Length 10 bytes | Order Card ID |
| Input Method | 0x55, ASCII(U) | Associated order card is inputted by UART, eg. RFID, barcode |
| 0x4B, ASCII(K) | Associated order card is inputted by Keypad |
| QTY | 0~65535 | Quantity of the associated order card |
| **Next** Order Card ID, **Next** Input Method, **Next** QTY [Optional] | Refer to **Order Card ID**, **Input Method** and **QTY** | Optional. if more than one order card, it is repeatedly contained in the packet. |

#### Example

24 53 54 00 12 4B 00 01 0B D3 D6 01 4F 07

00 30 30 30 33 31 33 32 30 39 33 55 31 05

30 30 31 33 34 37 30 33 30 36 4B 00 00

30 30 31 33 34 37 30 33 30 36 55 00 00

30 30 31 33 34 37 30 33 30 36 55 00 00

30 30 31 33 34 37 30 33 30 36 55 00 00

30 30 31 33 34 37 30 33 30 36 55 00 00

2B 0D 0A

#### Description

|  |  |
| --- | --- |
| **Sample** | **Description** |
| 24 53 54 | **Start Byte**, ASCII ($ST) |
| 00 12 4B 00 01 0B D3 D6 | **End Device ID**, Hex |
| 01 | **Meter ID**, Hex |
| 4F | Message **Len**, Hex, variable length, Sum of len of **Type** and **Payload**, the sample value = 79 bytes |
| 07 | Message **Type**, 0x07 = **MSG\_TYPE\_PENAL\_GENERAL** |
| 00 | **Timeout**, 0=normal send message, 1=timeout send message |
| 30 30 30 33 31 33 32 30 39 33 | **Staff Card ID**, ASCII (0003132093) |
| 55 | **Input Method**, ASCII (U) |
| 31 | **Flow Direction**, ASCII (1) |
| 05 | **Total Order Card**, Hex, Range 0x01 to 0x05 |
| 30 30 31 33 34 37 30 33 30 36 | **Order Card ID**, ASCII (0013470306) |
| 4B | **Input Method**, ASCII (K) |
| 00 00 | **QTY**, Hex, quantity of the order |
| 2B | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |
| 0D 0A | **Stop Byte** |

### MSG\_TYPE\_PENAL\_PREHEAT

#### Description

Record machine preheat process time.

#### Packet Structure

|  |  |  |  |
| --- | --- | --- | --- |
| Time Out[1] | Staff Card[10] | Input Method[1] | Preheat Status[1] |

#### Parameter Details

|  |  |  |
| --- | --- | --- |
| **Name** | **Range** | **Representation** |
| Time Out | 0 | Message sent due to user complete input. |
| 1 | Message sent due to Time Out. |
| Staff Card ID | 0x30~0x39, ASCII(0~9)  fixed Length 10 bytes | Staff Card ID |
| Input Method | 0x55, ASCII(U) | Staff card is inputted by UART, eg. RFID |
| 0x4B, ASCII(K) | Staff card is inputted by Keypad |
| Preheat Status | 0x30, ASCII(0) | Preheat Start |

#### Example

24 53 54 00 12 4B 00 01 0B D3 D6 01 0D 08

00 30 30 30 33 31 33 32 30 39 33 55 30

6D 0D 0A

#### Description

|  |  |
| --- | --- |
| **Sample** | **Description** |
| 24 53 54 | **Start Byte**, ASCII ($ST) |
| 00 12 4B 00 01 0B D3 D6 | **End Device ID**, Hex |
| 01 | **Meter ID**, Hex |
| 0D | Message **Len**, Hex, fixed length, Sum of len of **Type** and **Payload**, the sample value = 13 bytes |
| 08 | Message **Type**, 0x08 = **MSG\_TYPE\_PENAL\_PREHEAT** |
| 00 | **Timeout**, 0=normal send message, 1=timeout send message |
| 30 30 30 33 31 33 32 30 39 33 | **Staff Card ID**, ASCII (0003132093) |
| 55 | **Input Method**, ASCII (U) |
| 30 | **Preheat Status**, ASCII (0) |
| 6D | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |
| 0D 0A | **Stop Byte** |

### MSG\_TYPE\_PENAL\_REWORK

#### Description

Record rework production information.

#### Packet Structure

|  |  |  |  |
| --- | --- | --- | --- |
| Time Out[1] | Staff Card ID[10] | Input Method[1] | Dummy Value[1] |
| Total Order Card[1] | Order Card ID[10] | Input Method[1] | QTY [2] |
| **Next** Order Card ID[10]… [Optional] | **Next** Input Method[1]  [Optional] | **Next** QTY [2]  [Optional] |

#### Parameter Details

|  |  |  |
| --- | --- | --- |
| **Name** | **Range** | **Representation** |
| Time Out | 0 | Message sent due to user complete input. |
| 1 | Message sent due to Time Out. |
| Staff Card ID | 0x30~0x39, ASCII(0~9)  fixed Length 10 bytes | Staff Card ID |
| Input Method | 0x55, ASCII(U) | Staff card is inputted by UART, eg. RFID |
| 0x4B, ASCII(K) | Staff card is inputted by Keypad |
| Dummy Value | 0x30 | Nil |
| Total Order Card | 1~5 | Number of Order Card contain in the packet.  Each order card record contains **Order Card ID**, **Input Method** and **QTY** |
| Order Card ID | 0x30~0x39, ASCII(0~9)  fixed Length 10 bytes | Order Card ID |
| Input Method | 0x55, ASCII(U) | Associated order card is inputted by UART, eg. RFID, barcode |
| 0x4B, ASCII(K) | Associated order card is inputted by Keypad |
| QTY | 0~65535 | Quantity of the associated order card |
| **Next** Order Card ID, **Next** Input Method, **Next** QTY [Optional] | Refer to **Order Card ID**, **Input Method** and **QTY** | Optional. if more than one order card, it is repeatedly contained in the packet. |

#### Example

24 53 54 00 12 4B 00 01 0B D3 D6 01 28 09

00 30 30 30 33 31 33 32 30 39 33 55 30 02

30 30 31 33 34 37 30 33 30 36 4B 00 00

30 30 31 33 34 37 30 33 30 36 55 00 00

1B 0D 0A

#### Description

|  |  |
| --- | --- |
| **Sample** | **Description** |
| 24 53 54 | **Start Byte**, ASCII ($ST) |
| 00 12 4B 00 01 0B D3 D6 | **End Device ID**, Hex |
| 01 | **Meter ID**, Hex |
| 28 | Message **Len**, Hex, variable length, Sum of len of **Type** and **Payload**, the sample value = 40 bytes |
| 09 | Message **Type**, 0x09 = **MSG\_TYPE\_PENAL\_REWORK** |
| 00 | **Timeout**, 0=normal send message, 1=timeout send message |
| 30 30 30 33 31 33 32 30 39 33 | **Staff Card ID**, ASCII (0003132093) |
| 55 | **Input Method**, ASCII (U) |
| 30 | **Dummy Value**, ASCII (0) |
| 02 | **Total Order Card**, Hex, Range 0x01 to 0x05 |
| 30 30 31 33 34 37 30 33 30 36 | **Order Card ID**, ASCII (0013470306) |
| 4B | **Input Method**, ASCII (K) |
| 00 00 | **QTY**, Hex, quantity of the order |
| 1B | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |
| 0D 0A | **Stop Byte** |

### MSG\_TYPE\_PENAL\_INPUTQTY

#### Description

Update the quantity of order.

#### Packet Structure

As same as MSG\_TYPE\_PENAL\_REWORK (refer to P.18 )

#### Parameter Details

As same as MSG\_TYPE\_PENAL\_REWORK (refer to P.18 )

#### Example

24 53 54 00 12 4B 00 01 0B D3 D6 01 28 0A

00 30 30 30 33 31 33 32 30 39 33 55 30 02

30 30 31 33 34 37 30 33 30 36 4B 00 00

30 30 31 33 34 37 30 33 30 36 55 00 00

1C 0D 0A

#### Description

|  |  |
| --- | --- |
| **Sample** | **Description** |
| 24 53 54 | **Start Byte**, ASCII ($ST) |
| 00 12 4B 00 01 0B D3 D6 | **End Device ID**, Hex |
| 01 | **Meter ID**, Hex |
| 28 | Message **Len**, Hex, variable length, Sum of len of **Type** and **Payload**, the sample value = 40 bytes |
| 0A | Message **Type**, 0x0A = **MSG\_TYPE\_PENAL\_INPUTQTY** |
| 00 | **Timeout**, 0=normal send message, 1=timeout send message |
| 30 30 30 33 31 33 32 30 39 33 | **Staff Card ID**, ASCII (0003132093) |
| 55 | **Input Method**, ASCII (U) |
| 30 | **Dummy Value**, ASCII (0) |
| 02 | **Total Order Card**, Hex, Range 0x01 to 0x05 |
| 30 30 31 33 34 37 30 33 30 36 | **Order Card ID**, ASCII (0013470306) |
| 4B | **Input Method**, ASCII (K) |
| 00 00 | **QTY**, Hex, quantity of the order |
| 1C | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |
| 0D 0A | **Stop Byte** |

### MSG\_TYPE\_PENAL\_COMBINE

#### Description

Record the action of combining multi cards to one card. In the production, multi orders are combined into one order to be processing.

#### Packet Structure

|  |  |  |  |
| --- | --- | --- | --- |
| Time Out[1] | Staff Card ID[10] | | Input Method[1] |
| Dummy Value[1] | | Total Order Card[1] | |
| MOC[10] | MIM[1] | | MQTY [2] |
| SOC[10] | SIM[1] | | SQTY[2] |
| **Next** SOC[10]… [Optional] | **Next** SIM[1]...[Optional] | | **Next** SQTY [2]...[Optional] |

#### Parameter Details

|  |  |  |
| --- | --- | --- |
| **Name** | **Range** | **Representation** |
| Time Out | 0 | Message sent due to user complete input. |
| 1 | Message sent due to Time Out. |
| Staff Card ID | 0x30~0x39, ASCII(0~9)  fixed Length 10 bytes | Staff Card ID |
| Input Method | 0x55, ASCII(U) | Staff card is inputted by UART, eg. RFID |
| 0x4B, ASCII(K) | Staff card is inputted by Keypad |
| Dummy Value | 0x30 | Nil |
| Total Order Card | 1~5 | Number of Order Card contain in the packet.  Each order card record contains **Order Card ID**, **Input Method** and **QTY** |
| MOC | 0x30~0x39, ASCII(0~9)  fixed Length 10 bytes | Main Order Card ID, Following order cards are combined into this order Card. |
| MIM | 0x55, ASCII(U) | Associated order card is inputted by UART, eg. RFID, barcode |
| 0x4B, ASCII(K) | Associated order card is inputted by Keypad |
| MQTY | 0~65535 | Quantity of the associated order card |
| SOC | 0x30~0x39, ASCII(0~9)  fixed Length 10 bytes | Sub Order Card ID, this Order Cards and following Order Cards are combined into the **Main Order Card**. this Order Card will not be used. |
| SIM | Refer to **MIM** | |
| SQTY | Refer to **MQTY** | |
| **Next** SOC,  **Next** SIM,  **Next** SQTY[Optional] | Refer to **SOC, SIM** and **SQTY** | Optional. if more than one Order Card, it is repeatedly contained in the packet. All Sub Order Cards are combined into the **Main Order Card**. |

#### Example

24 53 54 00 12 4B 00 01 0B D3 D6 01 4F 0C

00 30 30 30 33 31 33 32 30 39 33 55 30 05

30 30 31 33 34 37 30 33 30 35 4B 00 00

30 30 31 33 34 37 30 33 30 36 55 00 00

30 30 31 33 34 37 30 33 30 37 55 00 00

30 30 31 33 34 37 30 33 30 38 55 00 00

30 30 31 33 34 37 30 33 30 39 55 00 00

34 0D 0A

#### Description

|  |  |
| --- | --- |
| **Sample** | **Description** |
| 24 53 54 | **Start Byte**, ASCII ($ST) |
| 00 12 4B 00 01 0B D3 D6 | **End Device ID**, Hex |
| 01 | **Meter ID**, Hex |
| 4F | Message **Len**, Hex, variable length, Sum of len of **Type** and **Payload**, the sample value = 79 bytes |
| 0C | Message **Type**, 0x0C = **MSG\_TYPE\_PENAL\_ COMBINE** |
| 00 | **Timeout**, 0=normal send message, 1=timeout send message |
| 30 30 30 33 31 33 32 30 39 33 | **Staff Card ID**, ASCII (0003132093) |
| 55 | **Input Method**, ASCII (U) |
| 30 | **Dummy Value**, ASCII (0) |
| 05 | **Total Order Card**, Hex, Range 0x01 to 0x05 |
| 30 30 31 33 34 37 30 33 30 35 | **MOC**, Main Order Card ID, ASCII (0013470305), following 4 order cards are combined into this card. |
| 4B | **MIM**, associated order card input method |
| 00 00 | **MQTY**, Hex, quantity of associated order |
| 30 30 31 33 34 37 30 33 30 36 | **SOC**, Sub Order Card ID, ASCII (0013470306), following 3 order cards and this order card are combined into the Main Order Card. |
| 55 | **SIM**, associated order card input method |
| 00 00 | **SQTY**, Hex, quantity of associated order |
| 34 | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |
| 0D 0A | **Stop Byte** |

### MSG\_TYPE\_PENAL\_ SEPERATE

#### Description

Record the action of separating one card to multi cards. In the production, one order are separated into multi orders to be processing.

#### Packet Structure

|  |  |  |  |
| --- | --- | --- | --- |
| Time Out[1] | Staff Card ID[10] | | Input Method[1] |
| Dummy Value[1] | | Total Order Card[1] | |
| MOC[10] | MIM[1] | | MQTY [2] |
| SOC[10] | SIM[1] | | SQTY[2] |
| **Next** SOC[10]… [Optional] | **Next** SIM[1]...[Optional] | | **Next** SQTY [2]...[Optional] |

#### Parameter Details

|  |  |  |
| --- | --- | --- |
| **Name** | **Range** | **Representation** |
| Time Out | 0 | Message sent due to user complete input. |
| 1 | Message sent due to Time Out. |
| Staff Card ID | 0x30~0x39, ASCII(0~9)  fixed Length 10 bytes | Staff Card ID |
| Input Method | 0x55, ASCII(U) | Staff card is inputted by UART, eg. RFID |
| 0x4B, ASCII(K) | Staff card is inputted by Keypad |
| Dummy Value | 0x30 | Nil |
| Total Order Card | 1~5 | Number of Order Card contain in the packet.  Each order card record contains **Order Card ID**, **Input Method** and **QTY** |
| MOC | 0x30~0x39, ASCII(0~9)  fixed Length 10 bytes | Main Order Card ID, this order card are separated to following order cards. This order card will not be used. |
| MIM | 0x55, ASCII(U) | Associated order card is inputted by UART, eg. RFID, barcode |
| 0x4B, ASCII(K) | Associated order card is inputted by Keypad |
| MQTY | 0~65535 | Quantity of the associated order card |
| SOC | 0x30~0x39, ASCII(0~9)  fixed Length 10 bytes | Sub Order Card ID, the **Main Order Card** will be separated to this Order Cards and following Order Cards. |
| SIM | Refer to **MIM** | |
| SQTY | Refer to **MQTY** | |
| **Next** SOC,  **Next** SIM,  **Next** SQTY[Optional] | Refer to **SOC, SIM** and **SQTY** | Optional. if more than one Order Card, it is repeatedly contained in the packet. All Sub Order Cards are combined into the **Main Order Card**. |

#### Example

24 53 54 00 12 4B 00 01 0B D3 D6 01 4F 0D

00 30 30 30 33 31 33 32 30 39 33 55 30 05

30 30 31 33 34 37 30 33 30 35 4B 00 00

30 30 31 33 34 37 30 33 30 36 55 00 00

30 30 31 33 34 37 30 33 30 37 55 00 00

30 30 31 33 34 37 30 33 30 38 55 00 00

30 30 31 33 34 37 30 33 30 39 55 00 00

35 0D 0A

#### Description

|  |  |
| --- | --- |
| **Sample** | **Description** |
| 24 53 54 | **Start Byte**, ASCII ($ST) |
| 00 12 4B 00 01 0B D3 D6 | **End Device ID**, Hex |
| 01 | **Meter ID**, Hex |
| 4F | Message **Len**, Hex, variable length, Sum of len of **Type** and **Payload**, the sample value = 79 bytes |
| 0D | Message **Type**, 0x0D = **MSG\_TYPE\_PENAL\_ SEPERATE** |
| 00 | **Timeout**, 0=normal send message, 1=timeout send message |
| 30 30 30 33 31 33 32 30 39 33 | **Staff Card ID**, ASCII (0003132093) |
| 55 | **Input Method**, ASCII (U) |
| 30 | **Dummy Value**, ASCII (0) |
| 05 | **Total Order Card**, Hex, Range 0x01 to 0x05 |
| 30 30 31 33 34 37 30 33 30 35 | **MOC**, Main Order Card ID, ASCII (0013470305), this card is separated to following 4 order cards. This card will not be used |
| 4B | **MIM**, associated order card input method |
| 00 00 | **MQTY**, Hex, quantity of associated order |
| 30 30 31 33 34 37 30 33 30 36 | **SOC**, Sub Order Card ID, ASCII (0013470306), The Main Order Card is separated to following 3 cards and this card. On the other hand, this card and following 3 cards replace the main order card. |
| 55 | **SIM**, associated order card input method |
| 00 00 | **SQTY**, Hex, quantity of associated order |
| 35 | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |
| 0D 0A | **Stop Byte** |

### MSG\_TYPE\_PENAL\_PREPARE

#### Description

Record the production prepare process.

#### Packet Structure

|  |  |  |  |
| --- | --- | --- | --- |
| Time Out[1] | Staff Card ID[10] | Input Method[1] | Flow Direction[1] |
| Total Order Card[1] | Order Card ID[10] | Input Method[1] | Dummy [2] |
| **Next** Order Card ID[10]… [Optional] | | **Next** Input Method[1]  ...[Optional] | **Next** Dummy [2]...  [Optional] |

#### Parameter Details

|  |  |  |
| --- | --- | --- |
| **Name** | **Range** | **Representation** |
| Time Out | 0 | Message sent due to user complete input. |
| 1 | Message sent due to Time Out. |
| Staff Card ID | 0x30~0x39, ASCII(0~9)  fixed Length 10 bytes | Staff Card ID |
| Input Method | 0x55, ASCII(U) | Staff card is inputted by UART, eg. RFID |
| 0x4B, ASCII(K) | Staff card is inputted by Keypad |
| Flow Direction | 0x30, ASCII(0) | Output Direction, Production Beginning |
| 0x31, ASCII(1) | Input Direction, Production Ending |
| Total Order Card | 1~5 | Number of Order Card contain in the packet.  Each order card record contains **Order Card ID**, **Input Method** and **QTY** |
| Order Card ID | 0x30~0x39, ASCII(0~9)  fixed Length 10 bytes | Order Card ID |
| Input Method | 0x55, ASCII(U) | Associated order card is inputted by UART, eg. RFID, barcode |
| 0x4B, ASCII(K) | Associated order card is inputted by Keypad |
| Dummy | 0 | Nil |
| **Next** Order Card ID, **Next** Input Method, **Next** Dummy [Optional] | Refer to **Order Card ID**, **Input Method** and **QTY** | Optional. if more than one order card, it is repeatedly contained in the packet. |

#### Example

24 53 54 00 12 4B 00 01 0B D3 D6 01 28 0E

00 30 30 30 33 31 33 32 30 39 33 55 30 02

30 30 31 33 34 37 30 33 30 36 4B 00 00

30 30 31 33 34 37 30 33 30 36 55 00 00

20 0D 0A

#### Description

|  |  |
| --- | --- |
| **Sample** | **Description** |
| 24 53 54 | **Start Byte**, ASCII ($ST) |
| 00 12 4B 00 01 0B D3 D6 | **End Device ID**, Hex |
| 01 | **Meter ID**, Hex |
| 28 | Message **Len**, Hex, variable length, Sum of len of **Type** and **Payload**, the sample value = 40 bytes |
| 0E | Message **Type**, 0x0E = **MSG\_TYPE\_PENAL\_PREPARE** |
| 00 | **Timeout**, 0=normal send message, 1=timeout send message |
| 30 30 30 33 31 33 32 30 39 33 | **Staff Card ID**, ASCII (0003132093) |
| 55 | **Input Method**, ASCII (U) |
| 30 | **Dummy Value**, ASCII (0) |
| 02 | **Total Order Card**, Hex, Range 0x01 to 0x05 |
| 30 30 31 33 34 37 30 33 30 36 | **Order Card ID**, ASCII (0013470306) |
| 4B | **Input Method**, ASCII (K) |
| 00 00 | **QTY**, Hex, quantity of the order |
| 20 | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |
| 0D 0A | **Stop Byte** |

### MSG\_TYPE\_PENAL\_NETWORKTEST

#### Description

The message samples retuned by the panel debug mode 2 to sever in every 15mins. The message type = 0x14(Penal Zigbee Network (Loading) Testing) as the only message content. This message will only be sent when panel working under debug mode 2 designed to test zigbee network’s loading capacity and stability.

#### Packet Structure

Nil

#### Parameter Details

Nil

#### Example

24 53 54 00 12 4B 00 01 0B C8 C5 01 01 14 D7 0D 0A

#### Description

|  |  |
| --- | --- |
| **Sample** | **Description** |
| 24 53 54 | **Start Byte**, ASCII ($ST) |
| 00 12 4B 00 01 0B C8 C5 | **End Device ID**, Hex |
| 01 | **Meter ID**, Hex |
| 01 | Message **Len**, fixed length 1 byte, no **Payload** |
| 14 | Message **Type**, 0x14 = **MSG\_TYPE\_PENAL\_NETWORKTEST** |
| D7 | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |
| 0D 0A | **Stop Byte** |

## Zigbee remote ctrl

### CTRL\_TYPE\_ZIGBEE\_PING

#### Description

The message is used for checking Zigbee device online status. Server side send this message to router or end device. If the device is online, it return message to the server.

#### Packet Structure

Nil

#### Parameter Details

Nil

#### Request Example (Server to End device)

FE 09 00 12 4B 00 01 0B C8 C5 A0 9D

#### Description

|  |  |
| --- | --- |
| **Sample** | **Description** |
| FE | **Start Byte** |
| 09 | Message **Len**, Hex, fixed length 1 byte, no **Payload,** Sum of len of **End Device ID** and **Type** |
| 00 12 4B 00 01 0B C8 C5 | **End Device ID**, Hex |
| A0 | Message **Type**, 0xA0 = **CTRL\_TYPE\_ZIGBEE\_PING** |
| 9D | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |

#### Reply Example (End to Server)

24 53 54 00 12 4B 00 01 0B C8 C5 01 01 A0 63 0D 0A

#### Reply Description (End to Server)

|  |  |
| --- | --- |
| **Sample** | **Description** |
| 24 53 54 | **Start Byte**, ASCII ($ST) |
| 00 12 4B 00 01 0B C8 C5 | **End Device ID**, Hex |
| 01 | **Meter ID**, Hex |
| 01 | Message **Len**, fixed length 1 byte, no **Payload** |
| A0 | Message **Type**, 0xA0 = **CTRL\_TYPE\_ZIGBEE\_PING** |
| 63 | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |
| 0D 0A | **Stop Byte** |

### CTRL\_TYPE\_ZIGBEE\_NV\_WRITE

#### Description

Type for remote write Data to End Device Zigbee NV. If the message is received by End Device, it reply the server.

#### Packet Structure

|  |  |  |
| --- | --- | --- |
| NVID[2] | NV\_LEN[1] | NV\_data[NV\_LEN] |

#### Parameter Details

|  |  |  |
| --- | --- | --- |
| **Name** | **Range** | **Representation** |
| NVID | 0x0401~0x0FFF | User-defined item address of NV  Refer to P.33 Table 3 (Zigbee NV memory) |
| NV\_LEN | 1~20 | **NV\_data** length in bytes |
| NV\_data | 1~20 bytes | Data to write |

#### Request Example (Server to End Device)

FE 0D 00 12 4B 00 01 0B C8 C5 A1

04 01 01 05 AD

#### Description (Server to End Device)

|  |  |
| --- | --- |
| **Sample** | **Description** |
| FE | **Start Byte** |
| 0D | Message **Len**, Hex, variable length**,** Sum of len of **End Device ID**, **Type**, and **Payload** |
| 00 12 4B 00 01 0B C8 C5 | **End Device ID**, Hex |
| A1 | Message **Type**, 0xA1 = **CTRL\_TYPE\_ZIGBEE\_NV\_WRITE** |
| 04 01 | User-defined item address of NV  Refer to P.33 Table 3 (Zigbee NV memory) |
| 01 | **NV\_data** length in bytes |
| 05 | Data to write |
| AD | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |

#### Reply Example (End Device to Server)

24 53 54 00 12 4B 00 01 0B C8 C5 01 05 A1

04 01 01 05 73 0D 0A

#### Reply Description (End Device to Server)

|  |  |
| --- | --- |
| **Sample** | **Description** |
| 24 53 54 | **Start Byte**, ASCII ($ST) |
| 00 12 4B 00 01 0B C8 C5 | **End Device ID**, Hex |
| 01 | **Meter ID**, Hex |
| 05 | Message **Len**, Hex, variable length, Sum of len of **Type** and **Payload** |
| A1 | Message **Type**, 0xA1 = **CTRL\_TYPE\_ZIGBEE\_NV\_WRITE** |
| 04 01 | User-defined item address of NV  Refer to P.33 Table 3 (Zigbee NV memory) |
| 01 | **NV\_data** length in bytes |
| 05 | Data to write |
| 73 | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |
| 0D 0A | **Stop Byte** |

## AVR remote ctrl

### CTRL\_TYPE\_EEPROM\_WRITE\_INT

#### Description

Type for remote write integer value(2bytes) to End Device AVR EEPROM. If the message is received by End Device, it reply the server.

#### Packet Structure

|  |  |
| --- | --- |
| Addr[2] | Data[2] |
| **Next** Addr[2]...[Optional] | **Next** Data[2]...[Optional] |

#### Parameter Details

|  |  |  |
| --- | --- | --- |
| **Name** | **Range** | **Representation** |
| Addr | 0x0000~0x01FD | User-defined item address of EEPROM. Increase by 2. Refer to P.35 Table 4 ( ) |
| Data | 0~65535 | **NV\_data** length in bytes, Big Endianness |
| **Next** Addr,  **Next** Data  **[Optional]** | Refer to **Addr** and **Data** | Optional. If more than one integer data(2bytes) to be saved, repeatedly contained in the packet. |

#### Request Example (Server to End Device)

FE 11 00 12 4B 00 01 0B C8 C5 C1

00 02 01 28

00 00 01 4B

3D

#### Description (Server to End Device)

|  |  |
| --- | --- |
| **Sample** | **Description** |
| FE | **Start Byte** |
| 11 | Message **Len**, Hex, variable length**,** Sum of len of **End Device ID**, **Type**, and **Payload** |
| 00 12 4B 00 01 0B C8 C5 | **End Device ID**, Hex |
| C1 | Message **Type**, 0xC1 = **CTRL\_ TYPE\_EEPROM\_WRITE\_INT** |
| 00 02 | **Addr,** user-defined item address of EEPROM. Increase by 2.  Big Endianness, the sample value = 0x0002  Refer to P.35 Table 4 ( ) |
| 01 28 | **Data**, Data to write, Big Endianness,  the sample value = 0x0128 = 296 (Dec) |
| 3D | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |

\*\* Can multi-write value to one message. But do not more than 3 value, it is not stable. No QTY in the message, QTY = ( Len-9 ) / 4.

#### Reply Example (End Device to Server)

24 53 54 00 12 4B 00 01 0B C8 C5 01 09 C1

00 02 01 28

00 00 01 4B

B7 0D 0A

#### Reply Description (End Device to Server)

|  |  |
| --- | --- |
| **Sample** | **Description** |
| 24 53 54 | **Start Byte**, ASCII ($ST) |
| 00 12 4B 00 01 0B C8 C5 | **End Device ID**, Hex |
| 01 | **Meter ID**, Hex |
| 09 | Message **Len**, Hex, variable length, Sum of len of **Type** and **Payload** |
| C1 | Message **Type**, 0xC1 = **CTRL\_ TYPE\_EEPROM\_WRITE** |
| 00 02 | **Addr,** user-defined item address of EEPROM. Increase by 2.  Big Endianness, the sample value = 0x0002  Refer to P.35 Table 4 ( ) |
| 01 28 | **Data**, Data to write, Big Endianness,  the sample value = 0x0128 = 296 (Dec) |
| B7 | Coordinator Packet **CheckSum**, Hex, Adding all bytes from **Start Byte** to **the last byte before Checksum**. |
| 0D 0A | **Stop Byte** |

\*\*After sending **CTRL\_TYPE\_EEPOM\_WRITE** to End device, it will reply message to Coordinator. Server need to check the result. If there is no any reply or the result is incorrect, server report the result to the user. There are no QTY in the message, QTY = ( Len-1 ) / 4.

### CTRL\_TYPE\_EEPROM\_ WRITE\_STR

#### Description

to be written...

### CTRL\_TYPE \_PENAL\_REQUEST

#### Description

to be written...

# Definition

## Panel Configuration

|  |  |
| --- | --- |
| **MODE** | **Operation Mode** |
| 0 (default) | General Mode |
| 1 | InputQty Mode |
|  | |
| **DIR** | **Machine Direction** |
| 0 | Preset Output |
| 1 | Preset Input |
| 2 (default) | User define |
|  | |
| **QTYs** | **Order Card QTY input requirement** |
| 0 (default) | Not Required |
| 1 | Required |
| 2 | Only Slave Card Required |
| 3 | Only Master Card Required |
|  | |
| **超時(TOUT)** | **Time Out Count Down** |
| 0 | OFF |
| 2 (default) | 2 minus count down |
| range | 0~9 (min) |
|  | |
| **KMOD** | **Keypad Operation Mode** |
| 0 (default) | Scard In by Keypad Disable |
| 1 | Scard In by Keypad Enable |
|  | |
| **PREHEAT** | **Preheat Operation Selection** |
| 0 | Disable |
| 1 (default) | Enable |
|  | |
| **REWORK** | **Rework Operation Selection** |
| 0 | Disable |
| 1 (default) | Enable |
|  | |
| **COMBINE** | **Combine Card Operation Selection** |
| 0 | Disable |
| 1 (default) | Enable |
|  | |
| **SEPARATE** | **Separate Card Operation Selection** |
| 0 | Disable |
| 1 (default) | Enable |

Table Configuration list for Panel

## Zigbee NV memory List

|  |  |  |
| --- | --- | --- |
| **NV address** | **Name** | **Description** |
| 0x0401 | NVID\_SEND\_PERIODIC | Time for request AVR data periodically |

Table Zigbee NV memory

## AVR EEPROM List (End device)

|  |  |  |
| --- | --- | --- |
| **Addr** | **Name** | **Description** |
| 0x0000 | CS\_Threshold\_High | Used by CS, Value of threshold High |
| 0x0002 | CS\_Threshold\_Low | Used by CS, Value of threshold Low |
| 0x0004 | CS\_MAX | Used by CS, Maximum value after CS scanning |
| 0x0006 | CS\_MIN | Used by CS, Minimum value after CS scanning |
| 0x0008 | CS\_AVG | Used by CS, Average value after CS scanning |

Table AVR EEPROM List (End Device)