# Barcode Reader Module
# Command Protocol Programming

# release summary

| Date | Version history | Note |
|---|---|---|
| 2023.5.30 | 1.1 | Modify HIDPOS VIDPID definition<br>Modify protocol 3 barcode ID byte definition<br>Query product information Increase image resolution<br>Add reverse color barcode setting<br>Add online settings<br>Add two-dimensional barcode type Grid Matrix<br>Add some local one-dimensional code system types |
| 2023.6.01 | 1.2 | Add protocol format reply command format description |
| 2023.6.03 | 1.2.1 | Miss barcode id in protocol 6 |

# Introduction

Through the protocol programming, it is used for the host computer to realize the function configuration, information reading, control and other operations of the barcode reading module. The protocol can realize the interaction between the host computer and the barcode reading module through RS-232, USB virtual serial port and other interfaces.

This document contains the protocol format specification and the protocol instruction table.

# Protocol format

**Host commands** consists of 4 parts:

| Command type | Property ID (PID) | Functional ID (FID) | Parameter value |
|---|---|---|---|
| 1 byte | 1 byte | 1 byte | lengthen |

Description of each part

| field | length | describe |
|---|---|---|
| Command type (TYPE) | 1 byte | The host sends the command type |
| Property ID (PID) | 1 byte | A property number for a set of specific functions |
| Functional ID (FID) | 1 byte | The function number is the specific sub-function under the property.<br>Some protocol commands that contain parameter values need to set the corresponding parameter value byte data according to the two bits (7 and 6 bits) of the FID. The specific definitions are as follows:<br>7bit 6bit <PARAM>field size<br>0 0 none<br>0 1 1 byte<br>1 0 2 bytes<br>1 1 variable number of bytes |
| Parameter value (PARAM) | Variable | Some protocol commands include The parameter length is determined according to the high two bits of the FID. If it is greater than two bytes, the first two bytes of the parameter part |

| | | |
|---|---|---|
| | | determine the number of remaining bytes of the parameter. |

**Reading Module Reply Command**

There are two types of reply: reply of setup setting and reply of retrieve parameter value.

Reply of setup setting include: Setup Write Reply, Control Reply. The composition is as follows:

| Reply Command | Property ID (PID) | Functional ID (FID) | Parameter (PAR) | Result (RID) |
|---|---|---|---|---|
| 1 byte | 1 byte | 1 byte | 1 byte | 1 byte |

Description of each part

| Field | Length | Describe |
|---|---|---|
| Command type (TYPE) | 1 byte | The scanner reply command type |
| Property ID (PID) | 1 byte | PID issued by the host |
| Functional ID (FID) | 1 byte | FID issued by the host |
| Parameter (PAR) | 1 byte | The host sends the parameter value,<br>When the parameter value is 1 byte, fill in the original parameter value<br>When the parameter value is 2 or more byte parameter values, fill in 00 |
| Result(RID) | 1 byte | Parameter setting result<br>0x00 Configuration is successful<br>0x01 Illegal PID/FID |

Reply of retrieve parameter value: configuration read reply, status reply. The composition is as follows:

| command type | Property ID (PID) | Functional ID (FID) | parameter value |
|---|---|---|---|
| 1 byte | 1 byte | 1 byte | lengthen |

Description of each part

| field | length | describe |
|---|---|---|
| command type (TYPE) | 1 byte | Recognition module reply command type |
| Property ID (PID) | 1 byte | PID issued by the host |
| Functional ID (FID) | 1 byte | FID issued by the host<br>According to the upper two bits (7 and 6 bits) of FID corresponding to the parameter value byte data, the specific definition is as follows: |

| | | |
|---|---|---|
| | | 7bit 6bit <PARAM>field size<br>0 0 none<br>0 1 1 byte<br>1 0 2 bytes<br>1 1 variable number of bytes |
| Parameter (PARAM) | Variable | The parameter length is determined according to the high two bits of the FID. If it is greater than two bytes, the first two bytes of the parameter part determine the number of remaining bytes of the parameter. |

# Protocol Type

Communication programming protocols include several protocol types:

Setup Command, Control Command, Status Command, Image Command.

**Setup Command**

Setup command frames are used to configure the scanner. They can also be used to retrieve setup information from the scanner.

| | | |
|---|---|---|
| Setup Write | **21** | host requests modification of one or more setup parameter values<br>After the configuration is successfully written, the parameters will be directly saved. |
| Setup Write Reply | **22** | When the Setup Write is correctly received the scanner informs the host that the changes have been applied by sending a Setup Write Reply |
| Setup Read | **23** | host requests value of one or more setup parameters |
| Setup Read Reply | **24** | scanner sends setup parameter values requested by host |

**Control Command**

Control command frames are used when the host wants to control the scanner.

| | | |
|---|---|---|
| Control Command | **32** | host sends one or more control commands to be executed by scanner |
| Control Reply | **33** | Response of the scanner for control commands |

**Status Command**

Status command frames are used to send and receive information on status parameters, such as version number, serial number, etc.

| | | |
|---|---|---|
| Status Read | **43** | host requests information concerning scanner status |
| Status Reply | **44** | scanner sends status information requested by host |

**Image Data Command**

Send and receive image data

| | | |
|---|---|---|
| Image Data Read | **60** | host requests image data |
| Image Data | **61** | scanner sends image data requested by the host |

Reply

# Protocol command description

## configuration command

The commands in this part are used to configure the identification module or receive the setting information of the identification module.

### Configuration write 0x21

The configuration write command is sent by the host to send the setting command to the identification module.

The command parameter value will be applied to the corresponding function of the current reading module and saved to the storage medium.

**protocol format**

<21><PID><FID><PARAM>

### configuration code

0x21 is replaced by the string "**<AW>**", use hexadecimal characters for other parameters

After the configuration is successful, there is no return data, and the buzzer is used to prompt the configuration result. For the definition, see "Configuration code definition》

**Configuration code example**

- The command to configure the key mode is 0x21, 0x61, 0x41, 0x00, and the content of the configuration code is a string *<AW>614100*
- Configure all barcodes with suffix 0D0A, commands 0x21, 0x51 , 0xC2 , 0x00 , 0x02 , 0x0D , 0x0A, and the configuration code content is a string *<AW>51C200020D0A*

### Configure write reply 0x22

Sent by the identification module to the host as a response to a configuration write command. Used to describe configuration write execution results.

Supported configuration write commands will be executed correctly, and illegal commands will return wrong results.

**protocol format**

<22><PID><FID><PAR><RID>

The format description is as follows:

| PID/FID | Flags included in configuration writes |
|---|---|
| ABOUT | When the configuration write is a 1-byte parameter value, fill in the original parameter value<br>When the configuration write is 2 or more byte parameter values, fill in 00 |
| RID | Configuration write result<br>0x00 Configuration is successful<br>0x01 Illegal PID/FID |

**Example 1**

host 21614100
module 2261410000
Configure the reading mode as**button trigger**model

**Example 2**

host 2151C200020D0A
module 2251C20000
Configuration suffix /r/n

## Configuration read 0x23

The configuration read command is sent by the host to receive the configuration information parameter value of the identification module

**protocol format**

<23><PID><FID>

## Configuration read reply 0x24

Sent by the identification module to the host as a response to a configuration read command. Configuration parameter values for sending host requests.

If an unsupported configuration parameter is requested, no error is raised. Unsupported PIDs/FIDs are ignored.

**protocol format**

<24><PID><FID><PARAM>

The recognition module ignores unsupported set parameter requests without error.

### Example 1

host 236141
module 24614105
Read module reading mode

# control commands

The control commands are used to operate the identification module to complete corresponding operations, such as starting decoding, restoring to factory, etc.

## Control command 0x32

The control command is sent by the host to control the identification module to complete a certain operation.

**protocol format**

<32><PID><FID><PARAM>

## configuration code

0x32 is replaced by the string "**<AM>**", use hexadecimal characters for other parameters

After the configuration is successful, there is no return data, and the buzzer is used to prompt the configuration result. For the definition, see "Configuration code definition》

### Configuration code example

Restore factory default settings, the command is 0x32, 0x76, 0x01, and the content of the configuration code is a string *<AC>7601*

## Control Reply 0x33

The identification module is used to respond to the control command (0x32) sent by the host.

When the control command is executed correctly, the recognition module sends a "result command" as a response.Partial Control InstructionsNo reply, see detailscommand table。

If the control command contains illegal instructions, the illegal instructions will be included in the result command and returned to the host.

**protocol format**

<33><PID><FID><PAR><RID>

The format description is as follows:

| | |
|---|---|
| PID/FID | PID FID included in the control command |
| ABOUT | When the control command is a 1-byte parameter value, fill in the original parameter value |

| | |
|---|---|
| | When the control command has no parameter, or 2 or more byte parameter values, fill in 00 |
| RID | Control command result<br>0x00 The command is correct and successful<br>0x01 PID, FID, parameters are incorrect |

**Example 1**

host 327501

module none

Send start decoding command

**Example 2**

host 327502

module 3375020000

Send stop decoding command

## Status query command

This part of the command is used to send and receive status parameter information. Status query includes reading module software version, firmware version, serial number, etc.

### status read 0x43

The status read command is sent by the host to receive a certain status parameter information of the identification module.

**protocol format**

<43><PID><FID>

### configuration code

0x43 is replaced by the string "**<AS>**", use hexadecimal characters for other parameters

The configuration is successful, and the buzzer will be used to prompt the configuration result. For the definition, see "Configuration code definition", and output read status information

**Configuration code example**

The command to query the version of the reading software is 0x43, 0x02, 0xC2, and the content of the configuration code is a string *<AS>02C2*

After the scan is successful, directly output the version information content, no state reply protocol command, such as ZScan 1.2.0

### Status Reply 0x44

Sent by the identification module to the host as a response to status reading, it is used to send the status parameter value requested by the host.

Unsupported or illegal status query commands are ignored without error. If none of the queries are supported, an empty status reply is returned.

**protocol format**

<44><PID><FID><PARAM>

Unsupported or illegal status query commands are ignored without error. None of the queries are supported, an empty status reply is returned.

<44><00>

### Example 1

Get the firmware version of the reading module:

Host sends status read:

| command type | PID/FID |
|---|---|
| 43 | 02C1 |
| status read | Firmware version |

Identification module status reply:

| command type | PID/FID | parameter |
|---|---|---|
| 44 | 02C1 | 00 09 42 46 35 33 31 5F 31 2E 30 |
| state reply | Firmware version | Length BF531_1.0 |

# image command

This part of the command is used for the transmission of image data.

## image read 0x60

The image read command is issued by the host, requesting the identification module to send the currently (finally) collected image data.

**protocol format**

<60><request image header>

<Request Image Header> contains the conditional information of the requested image, such as image size, format, etc., defined as follows

| name | byte | illustrate |
|---|---|---|
| width | 2 | Image width (MSB first)<br>If the width is smaller than the maximum supported image width, the image will be scaled |
| high | 2 | Image height (MSB first)<br>If the height is smaller than the maximum supported image height, the image will be scaled |
| image type | 1 | Bit[7:4] Bit[3:0]=3, it is Jpeg format<br>Indicates the percentage of compression:<br>01 – 0A 10%-100%<br>00 50% (default)<br>Bit[3:0]0 raw grayscale data<br>3 Jpeg format Grayscale data |
| reserve | 1 | |

**Identification Module Response**

When the recognition module correctly receives the image data read command, it will send an "image reply" command as a response.

If the host request contains illegal parameter values, or the identification module returns an empty "image reply".

## Image Reply 0x61

The image reply is sent by the recognition module to the host in response to the image read

command.

**protocol format**

<61><reply image header><image data>

<Response Image Header> is defined as follows:

| name | byte | illustrate |
|---|---|---|
| width | 2 | Image width (MSB first)<br>If the width is smaller than the maximum supported image width, the image will be scaled |
| high | 2 | Image height (MSB first)<br>If the maximum image height that can be supported is small, the image will be scaled |
| compression | 1 | 0 raw data<br>3 Jpeg |
| reserve | 1 | reserved for use |
| Image data length | 4 | The total number of bytes in the image data area ((MSB first)) |

If there are illegal parameters in the image read command, the reply will return empty data.
<61><00>

**example**

The host requests the recognition module to upload an image, the size is 640x480, the original data, and the recognition module returns the currently collected image:

Host sends image data read command

| command type | request image header |
|---|---|
| 60 | 02 80 01 E0 00 00 |
| image read | Width Height Raw Data Reserved |

The recognition module sends a reply command

| command type | reply image header | image data |
| --- | --- | --- |
| 61 | 02 80 01 E0 00 00 00 1C 20 00 | 7A 76 7B 7A 7C 76 76 76 79 79 78 79 76 ..... |
| image read reply | Width Height Original Data Reserved Image Data Length | image data |

# protocol command table

## configuration command table

This list shows the definitions of PID/FID and parameter values of each configuration function of the reading module.

Complete the corresponding configuration function parameter writing or reading function, refer to the "Configuration Command" chapter.

### Communication Interface

Among them, the definitions of PID and VID of the three interfaces of USB are detailed in "USB interface VID PID》

| Communication interface selection | | |
|---|---|---|
| describe | PID/FID | value |
| RS232 (serial port) | 4240 | 00 |
| USB emulated keyboard | | 01 |
| USB virtual serial port | | 02 |
| USB HID POS | | 03 |

### RS232 parameter

| RS232 parameter | | | |
|---|---|---|---|
| describe | | PID/FID | value |
| baud rate | 1200 | 4141 | 04 |
| | 2400 | | 05 |
| | 4800 | | 06 |
| | 9600 | | 07 |
| | 19200 | | 08 |
| | 38400 | | 09 |
| | 57600 | | 0A |
| | 115200 | | 0B |
| | 128000 | | 0C |
| data bit | 7 bits | 4142 | 00 |
| | 8 bits | | 01 |
| Check Digit | None | 4143 | 00 |
| | Even | | 01 |
| | Odd | | 02 |
| stop bit | 1 | 4144 | 00 |

| | | | |
|---|---|---|---|
| | 2 | | **01** |

## USB keyboard

| USB keyboard parameters | | | |
|---|---|---|---|
| **describe** | | **PID/FID** | **value** |
| keyboard country type | US | **4340** | **00** |
| | Spain | | **01** |
| | Germany | | **02** |
| | France | | **03** |
| | Italy | | **05** |
| | Sweden | | **06** |
| | Russia | | **0E** |
| | Turkey Q | | **0F** |
| | Turkey F | | **10** |
| | Hungary | | **11** |
| | Croatia | | **12** |
| Directly output barcode encoding type characters | ASCII | **4341** | **00** |
| | GBK<br>In Windows system, support Alt plus key combination to directly output GBK data | | **01** |
| | UTF-8<br>In Windows system, it supports Alt plus key combination to directly output UTF-8 data | | **02** |
| key delay | Set the key-press interval when the virtual keyboard is pressed continuously. The interval time is the time between the last key release and the next key press, in milliseconds<br>XX = hex value 0 to FF | **4346** | **XX** |

## Reading parameters

| Reading parameters | | | |
|---|---|---|---|
| **describe** | | **PID/FID** | **value** |
| trigger mode | button<br>Decoding starts when the trigger is pressed (trigger level is low), and decoding stops when the trigger is released. Stop decoding after successful decoding. In this mode, you can also use the communication commands "start decoding" and "stop decoding" to achieve the same decoding process. Send "start decoding" command, decoding starts, send "stop decoding" to stop. | **6141** | **00** |

| | | | |
|---|---|---|---|
| | automatic<br>After the reading engine is started, it keeps decoding. At this time, the trigger button is invalid, and the "stop decoding" communication command cannot stop the decoding state. Continue to read after successful reading | | **02** |
| | motion sensing<br>After the system is started, the reading engine will detect the captured images. When the scene changes, it will start decoding. After the reading is successfully output or no scene change is detected, it will re-enter the state of detecting scene changes.<br>In this mode, the reading engine can also respond to the low-level trigger of the trigger, or the "start decoding" communication command. At this time, it will start reading and decoding. After the reading is successfully output or no scene change is detected, it will restart Enter the detection scene change state。 | | **05** |
| Induction Mode Sensitivity | In the motion sensing mode, the sensitivity level can be set to detect scene changes. The higher the level, the lower the scene change detection threshold, and the more sensitive the detection of subtle changes. | | |
| | Class 1 (lowest sensitivity) | **6144** | **01** |
| | level 2 | | **02** |
| | Level 3 | | **03** |
| | level 4 | | **04** |
| | 5 (highest sensitivity) | | **05** |
| Inductive continuous decoding | closure | **614B** | **00** |
| | turn on | | **01** |
| Sensing continuous decoding delay | Delay time to keep the decoding state, the unit is 100ms<br>XXYY = hex value 0001 to FFFF | **618C** | **XXYY** |
| same code interval | closure | **6482** | **0000** |
| | Interval time (unit: 1ms)<br>XXYY = hex value 0001 to FFFF | | **XXYY** |
| command decoding delay | Set [decoding command/**start delayed decoding**】Command decoding time (unit: 1ms)<br>XXYY = hex value 0001 to FFFF | **618A** | **XXYY** |

## fill light

| Fill Light Settings | | | |
|---|---|---|---|
| **describe** | | **PID/FID** | **value** |
| Operating mode | Decoding is always on | **6241** | **02** |
| | Often off | | **00** |
| | Always on when powered on | | **03** |
| Working brightness | Percentage 0 to 100<br>XX = hex value 0 to 64 | **6248** | **XX** |
| Sensing detection enable | In the motion sensing mode, when there is no moving object, the system enters the idle monitoring state. At this time, there are two settings for the fill light: off, always on (low brightness) | | |
| | closure | **6244** | **00** |
| | always on | | **01** |
| Sensing detection brightness | Percentage 0 to 100<br>XX = hex value 0 to 64 | **624C** | **XX** |

## positioning light

| Locator Light Settings | | | |
|---|---|---|---|
| **describe** | | **PID/FID** | **value** |
| Operating mode | Decoding is always on | **6242** | **02** |
| | Often off | | **00** |
| | | | |
| Sensing detection enable | closure | **6245** | **00** |
| | always on | | **01** |

## buzzer

| Buzzer settings | | | |
|---|---|---|---|
| **describe** | | **PID/FID** | **value** |
| Prompt sound for successful reading | turn on | **6346** | **01** |
| | closure | | **00** |
| Types of reading success | Low | **6341** | **00** |
| | middle | | **01** |
| | high | | **02** |
| start beep | turn on | **6345** | **01** |
| | closure | | **00** |

## data editing

| data editing | | | |
|---|---|---|---|
| **describe** | | **PID/FID** | **value** |
| All barcode suffix content | Data length 0 to 40 bytes<br>**LD** = length (2 bytes) + data (0-40 bytes) | **51C2** | **LD** |
| All barcode prefix content | Data length 0 to 40 bytes<br>**LD** = length (2 bytes) + data (0-40 bytes) | **51C1** | **LD** |
| protocol format<br>Detailed definition | Raw data | **5143** | **00** |
| | protocol format 1 | | **01** |
| | protocol format 2 | | **02** |
| | protocol format 3 | | **03** |
| | protocol format 4 | | **04** |
| | protocol format 5 | | **05** |
| | Protocol Format 6 | | **06** |

## Barcode Global Settings

| Barcode Global Settings | | | |
|---|---|---|---|
| **describe** | | **PID/FID** | **value** |
| Inverted color barcode reading | No reverse color reading | **7144** | **00** |
| | Allow positive and negative color reading | | **02** |

## One-dimensional code settings

| One-dimensional code settings | | | |
|---|---|---|---|
| **describe** | | **PID/FID** | **value** |
| **Code 39** | | | |
| Code 39 | Do not read | **2340** | **00** |
| | Allow reading | | **01** |
| Check method | No check | **2341** | **00** |
| | Code 32 verification | | **01** |
| | Mod 43 checksum | | **02** |
| Full ASCII | prohibit | **2342** | **00** |
| | allow | | **01** |
| Output start and stop characters | prohibit | **2343** | **00** |
| | allow | | **01** |
| output verification | prohibit | **2333** | **00** |
| | allow | | **01** |
| **Code 93** | | | |
| Code 93 | Do not read | **2240** | **00** |
| | Allow reading | | **01** |
| Full ASCII | prohibit | **2241** | **00** |
| | allow | | **01** |
| **Code 128** | | | |
| Code 128 | Do not read | **2140** | **00** |
| | Allow reading | | **01** |
| **UPC-A** | | | |
| UPC-A | Do not read | **2441** | **00** |
| | Allow reading | | **01** |
| output verification | prohibit | **2451** | **00** |
| | allow | | **01** |
| **UPC-E** | | | |
| UPC-E | Do not read | **2442** | **00** |
| | Allow reading | | **01** |
| output verification | prohibit | **2452** | **00** |
| | allow | | **01** |
| **EAN 13** | | | |

| | | | |
|---|---|---|---|
| EAN 13 | Do not read | 2443 | 00 |
| | Allow reading | | 01 |
| output verification | prohibit | 2453 | 00 |
| | allow | | 01 |
| **EAN 8** | | | |
| EAN 8 | Do not read | 2444 | 00 |
| | Allow reading | | 01 |
| output verification | prohibit | 2454 | 00 |
| | allow | | 01 |
| **Codabar** | | | |
| Codabar | Do not read | 2540 | 00 |
| | Allow reading | | 01 |
| Check method | No check | 2541 | 00 |
| | check | | 01 |
| output verification | prohibit | 2542 | 00 |
| | allow | | 01 |
| Output start and stop characters | prohibit | 2543 | 00 |
| | Output start and stop characters | | 01 |
| **Industrial 2 of 5** | | | |
| Industrial 2 of 5 | Do not read | 2640 | 00 |
| | Allow reading | | 01 |
| Check method | No check | 2641 | 00 |
| | Find remainder 10 check | | 01 |
| output verification | prohibit | 2642 | 00 |
| | allow | | 01 |
| **Matrix 2 of 5** | | | |
| Matrix 2 of 5 | Do not read | 2740 | 00 |
| | Allow reading | | 01 |
| Check method | No check | 2741 | 00 |
| | check | | 01 |
| output verification | prohibit | 2742 | 00 |
| | allow | | 01 |
| **Interleaved 2 of 5** | | | |
| Interleaved 2 of 5 | Do not read | 2840 | 00 |
| | Allow reading | | 01 |
| Check method | No check | 2841 | 00 |
| | check | | 01 |
| output verification | prohibit | 2842 | 00 |
| | allow | | 01 |
| **MSI** | | | |
| MSI | Do not read | 2A40 | 00 |
| | Allow reading | | 01 |
| Check method | No check | 2A41 | 00 |
| | Find remainder 10 check | | 01 |

| | | | |
|---|---|---|---|
| | Find remainder 11 check | | 02 |
| | Remainder 10 10 Checksum | | 03 |
| | Remainder 11 10 Checksum | | 04 |
| output verification | prohibit | 2A42 | 00 |
| | allow | | 01 |
| **Code 11** | | | |
| Code 11 | Do not read | 2940 | 00 |
| | Allow reading | | 01 |
| Check method | No check | 2941 | 00 |
| | 1 checksum | | 01 |
| | 2 checksums | | 02 |
| output verification | prohibit | 2942 | 00 |
| | allow | | 01 |
| **GS1 Databar( Omnidirectional)** | | | |
| GS1 Databar (Omnidirectional) | Do not read | 2B40 | 00 |
| | Allow reading | | 01 |
| **GS1 Databar (Limited)** | | | |
| GS1 Databar (Limited) | Do not read | 2B41 | 00 |
| | Allow reading | | 01 |
| **GS1 Databar (Expanded)** | | | |
| GS1 Databar (Expanded) | Do not read | 2B42 | 00 |
| | Allow reading | | 01 |
| **China Post** | | | |
| China Post | Do not read | 2D40 | 00 |
| | Allow reading | | 01 |
| **Plessey** | | | |
| Plessey | Do not read | 2C40 | 00 |
| | Allow reading | | 01 |
| output verification | prohibit | 2C41 | 00 |
| | allow | | 01 |
| **On site** | | | |
| On site | Do not read | 2E40 | 00 |
| | Allow reading | | 01 |
| encoding type | ASCII | 2E41 | 01 |
| | default | | 00 |
| | Numeric | | 02 |

## QR code settings

| QR code settings | | | |
|---|---|---|---|
| **describe** | | **PID/FID** | **value** |
| **QR Code** | | | |
| QR Code | Do not read | **1040** | **00** |
| | Allow reading | | **01** |
| **Micro QR** | | | |
| Micro QR | Do not read | **1540** | **00** |
| | Allow reading | | **01** |
| **PDF417** | | | |
| PDF417 | Do not read | **1140** | **00** |
| | Allow reading | | **01** |
| **MicroPDF417** | | | |
| MicroPDF417 | Allow reading | **1640** | **01** |
| | Do not read | | **00** |
| **Chinese Sensible Code** | | | |
| Chinese Sensible Code | Do not read | **1240** | **00** |
| | Allow reading | | **01** |
| **Data Matrix** | | | |
| Data Matrix | Do not read | **1340** | **00** |
| | Allow reading | | **01** |
| **Aztec** | | | |
| Aztec | Do not read | **1440** | **00** |
| | Allow reading | | **01** |
| **Grid Matrix** | | | |
| Grid Matrix | Allow reading | **1740** | **01** |
| | Do not read | | **00** |

## default configuration table

| parameter name | default setting | Remark | |
|---|---|---|---|
| **Communication Interface** | | | |
| type | **USB virtual serial port** | | |
| **RS232** | | | |
| baud rate | 115200 | | |
| data bit | 8bit | | |
| Check Digit | None | | |
| stop bit | 1 | | |
| **USB keyboard** | | | |
| keyboard country type | US | | |
| Directly output barcode encoding type characters | ASCII | | |
| key delay | 0 | | |

| | | | |
|---|---|---|---|
| **Reading parameters** | | | |
| trigger mode | **motion sensing** | | |
| Induction Mode Sensitivity | Level 3 | | |
| Inductive continuous decoding | closure | | |
| Sensing continuous decoding delay | 50(5000ms) | | |
| same code interval | 0 | | |
| command decoding delay | 20000ms | | |
| **fill light** | | | |
| Operating mode | Decoding is always on | | |
| Working brightness | 100 | | |
| Sensing detection enable | closure | | |
| Sensing detection brightness | 10 | | |
| **positioning light** | | | |
| Operating mode | Decoding is always on | | |
| Sensing detection enable | closure | | |
| **buzzer** | | | |
| Prompt sound for successful reading | turn on | | |
| Types of reading success | middle | | |
| start beep | turn on | | |
| **data editing** | | | |
| All barcode suffix content | none | | |
| All barcode prefix content | none | | |
| protocol format | Raw data | | |
| **Barcode Global Settings** | | | |
| Inverted color barcode reading | prohibit | | |
| **One-dimensional code settings** | | | |
| **Code 39** | | | |
| read | allow | | |
| Check method | No check | | |
| Full ASCII | allow | | |
| Output start and stop characters | prohibit | | |
| output verification | prohibit | | |
| **Code 93** | | | |
| read | allow | | |
| Full ASCII | allow | | |
| **Code 128** | | | |
| read | allow | | |
| **UPC-A** | | | |
| read | allow | | |

| | | | |
|---|---|---|---|
| output verification | allow | | |
| **UPC-E** | | | |
| read | allow | | |
| output verification | allow | | |
| **EAN 13** | | | |
| read | allow | | |
| output verification | allow | | |
| **EAN 8** | | | |
| read | allow | | |
| output verification | allow | | |
| **Codabar** | | | |
| read | prohibit | | |
| Check method | No check | | |
| output verification | prohibit | | |
| Output start and stop characters | prohibit | | |
| **Industrial 2 of 5** | | | |
| read | prohibit | | |
| Check method | No check | | |
| output verification | prohibit | | |
| **Matrix 2 of 5** | | | |
| read | prohibit | | |
| Check method | No check | | |
| output verification | prohibit | | |
| **Interleaved 2 of 5** | | | |
| read | allow | | |
| Check method | No check | | |
| output verification | prohibit | | |
| **MSI** | | | |
| read | prohibit | | |
| Check method | No check | | |
| output verification | allow | | |
| **Code 11** | | | |
| read | prohibit | | |
| Check method | No check | | |
| output verification | allow | | |
| **GS1 Databar** | | | |
| Omnidirectional | prohibit | | |
| Limited | prohibit | | |
| Expanded | prohibit | | |
| **China Post** | | | |
| read | prohibit | | |
| **Plessey** | | | |
| read | prohibit | | |
| output verification | prohibit | | |

| | | | |
|---|---|---|---|
| **On site** | | | |
| read | prohibit | | |
| encoding type | ASCII | | |
| **QR Code** | | | |
| read | allow | | |
| **Micro QR** | | | |
| read | prohibit | | |
| **PDF417** | | | |
| read | allow | | |
| **Chinese Sensible Code** | | | |
| read | prohibit | | |
| **Data Matrix** | | | |
| read | allow | | |
| **Aztec** | | | |
| read | prohibit | | |
| **Grid Matrix** | | | |
| read | prohibit | | |

## Control command table

To complete the corresponding control command function, it is necessary to form a complete protocol command according to the "Control Command" chapter.

| Control command table | | | |
|---|---|---|---|
| **describe** | | **PID/FID** | **value** |
| decode command | start decoding<br>**no control reply command** | **7501** | **none** |
| | stop decoding | **7502** | **none** |
| | start delayed decoding<br>**no control reply command** | **7504** | **none** |
| set command | Restore Factory | **7601** | |
| Code system reading settings | close all | **7642** | **00** |
| | Only allow to read all 1D codes | | **01** |
| | Only allow to read all QR codes | | **02** |
| | Allows to read all one-dimensional codes (two-dimensional codes remain unchanged) | | **04** |
| | Turn off reading all one-dimensional codes (two-dimensional codes remain unchanged) | | **06** |
| | Allows to read all QR codes (one-dimensional unchanged) | | **05** |
| | Turn off reading all QR codes (one-dimensional remains unchanged) | | **07** |
| | Allow all barcodes | | **03** |
| online settings | turn on<br>After the online setting is turned on, no matter the scan code or the command, any function configuration will not be saved, and it will only be valid during the current operation. After restarting or re-powering, the configuration before the online setting is turned on will remain. | **7644** | **01** |
| | closure | | **00** |

## Status Query Command Table

To complete the corresponding status information query function, you need to compose a complete protocol command according to the "Status Command" chapter.

| Device Information | | | |
|---|---|---|---|
| **describe** | | **PID/FID** | |
| Version | Read software version | **02C2** | |
| | Firmware version | **02C1** | |
| product information | serial number | **02C5** | |
| | Production Date | **02C6** | |
| | Hardware model (device type) | **02C7** | |
| | hardware specification | **02C8** | |
| | hardware version | **02C4** | |
| | Image Resolution<br>Reply current device resolution<br>4 bytes: wide (2 bytes) high (2 bytes), byte order MSB<br>Such as 640x480, full reply 44 02 C3 00 04 02 80 01 E0. | **02C3** | |

# Detailed definition of command parameters

## protocol format

**Format 1:**[0x03] + data length + data

Data length: represented by two bytes, with the high-order byte first, and the value range is 0-65535

Data: data information content, such as barcode data, including data such as prefix and suffix.

**Format 2:**[0x03] + data length + number of barcodes + data length of barcode 1 + data of barcode 1 +... + CRC

Data length: the length of all data after this byte position. It is represented by two bytes, with the high-order byte first.

Number of barcodes: the number of barcodes output this time. 1 byte

Barcode 1 data length: the data length of the first barcode. 2 bytes, high byte first.

Barcode 1 data: the data information content of the first barcode

CRC: CRC check data of all previous data. 2 bytes. The algorithm uses CRC-16 BUYPASS, and the algorithm is shown in the appendix.

**Format 3:**[0x03] + data length + number of barcodes + ID number of barcode 1 + data length of barcode 1 + data of barcode 1 + ... + CRC

Data length: the length of all data after this byte position. 2 bytes, high byte first.

Number of barcodes: the number of barcodes output this time. 1 byte

Barcode 1 ID number: The first barcode type ID number. 2 bytes

Barcode 1 data length: the data length of the first barcode. 2 bytes, high byte first.

Barcode 1 data: the data information content of the first barcode

CRC: CRC check data of all previous data. 2 bytes. The algorithm uses CRC-16 BUYPASS, and the algorithm is shown in the appendix.

**Format 4:** Barcode ID number + barcode data

Barcode ID number: Barcode type ID number. 1 byte

Barcode data: data information content

**Format 5:** [AIM ID] + barcode data

AIM ID: AIM ID string

Barcode data: data information content

**Format 6:** [0x03] + data length + Barcode ID + data + CRC

Data length: the length of all data after this byte position. Two bytes, high byte first, value range 0-65535

Data: data information content, such as barcode data, including data such as prefix and suffix.

Barcode ID number: Barcode type ID number. 1 byte

CRC: CRC check data of all previous data. 2 bytes. The algorithm uses CRC-16 BUYPASS, and the algorithm is shown in the appendix.

# USB interface VID PID

### 1. USB HID keyboard

VID 0x0525 PID 0xA401

### 2. USB VCOM virtual serial port

VID 0x0525 PID 0xA4A7

### 3. USB HID POS

VID 0x0525 PID 0xA402

HIDPOS message definition

Reading device output to host data packet definition

| Message ID 0x02 | Effective data length | data | reserve | data continue |
|---|---|---|---|---|
| 1 bytes | 1 byte | 56 bytes | 5 bytes | 1 byte |

When the actual data exceeds 56 bytes, there will be more than one message data, data continue = 1, indicating that there is a continuous next message, data continue = 0, indicating that there is no data behind, and the message ends

The host sends data to the reading device data packet definition

| Message ID 0x04 | Effective data length | data | reserve | data continues |
|---|---|---|---|---|
| 1 bytes | 1 byte | 60 bytes | 1 byte | 1 byte |

When the actual data is more than 60 bytes, there will be more than one message data, data continue=1, indicating that there is a continuous next message, data continue=0, indicating that there is no data behind, and the message ends

# appendix

## Appendix A Barcode Type ID Serial Number Comparison Table

| barcode type | serial number |
|---|---|
| Code 39 | 1 |
| Codabar | 2 |
| Code 128 | 3 |
| Interleaved 2 of 5 | 6 |
| Code 93 | 7 |
| UPCA | 8 |
| UPCE | 9 |
| EAN8 | 10 |
| EAN13 | 11 |
| MSI | 14 |
| Databar Limited | 35 |
| Databar Omni | 36 |
| Databar Expanded | 37 |
| Code 11 | 38 |
| Standard 2 of 5 | 39 |
| Matrix 2 of 5 | 40 |
| Plessey | 42 |
| China Post | 43 |
| On site | 44 |
| Data Matrix | 81 |
| PDF417 | 82 |
| MicroPDF417 | 83 |
| QR Code | 87 |
| Micro QR | 88 |
| Aztec | 89 |
| Chinese Sensible Code | 96 |
| GM Code | 97 |

# Appendix B CRC Algorithm in Protocol Format

```
//Generator polynomial 0x18005 CRC-16 BUYPASS poly=0x8005 init=0x0000
unsigned short crc16_ccitt(const unsigned char *ptr, int len){
    unsigned char i;
    unsigned int ca_crc=0;
    while(len-- != 0){
        for(i=0x80; i != 0; i >>= 1){
            if((ca_crc & 0x8000) != 0){
                ca_crc <<= 1;
                ca_crc ^= 0x18005;
            }else{
                ca_crc <<= 1;
            }
            if((*ptr&i) != 0) ca_crc^=0x18005;
        }
        ptr++;
    }
    return ca_crc;
}
```