**RS485通讯命令说明**

**通讯参数设置：**

波特率：115200

校验位：无校验

数据位：8位

停止位：1位

**读取UID –** **Inventory（**Read UID (64bit) and DSFID (8bit) of the TAG**）**

**Request**

Command Code : 0x62

Command Index : Request

User Data : none

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 0xAA | 0x07 | 0x07 | 0x62 | 0x00 | 0xDA | 0xE4 |

**Acknowledge**

Command Code : 0x62

Command Index : Acknowledge

User Data : none

**Response (SingleTAG-mode, CI Bit\_2 = 0)**

Command Code : 0x62

Command Index : Response

User Data : 9 byte

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5...13 | 14 | 15 |
| 0xAA | 0x10 | 0x10 | 0x62 | CI | LSB User Data MSB | CRC-16 | CRC-16 |

Byte 5: UID, LSB

...

Byte 12: UID, MSB (always **0xE0**)

Byte 13: data storage format identifier (DSFID)

发送数据：AA 07 07 **62** 00 DA E4

接收数据：AA 07 07 62 89 13 FD

AA 10 10 62 8A **78 2A 82 20 00 01 04 E0** 00 2A 63

---------------------------------------------------------------------------------------------------------------------------------

**识别载码体类型 – Get System Information（**Read UID (64bit), DSFID (8bit), AFI (8bit), memory size (16bit) and IC-reference (8bit) from TAG**）**

**Request**

Command Code : 0x70

Command Index : Request

User Data : 0...8 byte

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5...12 | 13 | 14 |
| 0xAA | 0x0F | 0x0F | 0x70 | 0x10 | LSB UID MSB | CRC-16 | CRC-16 |

Byte 5 ... 12: UID of the requested TAG (optional, if byte\_4/bit\_4 is set)

**Acknowledge**

Command Code : 0x70

Command Index : Acknowledge

User Data : none

**Response**

Command Code : 0x70

Command Index : Response

User Data : 13 byte

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5...17 | 18 | 19 |
| 0xAA | 0x14 | 0x14 | 0x70 | CI | LSB User Data MSB | CRC-16 | CRC-16 |

Byte 5: UID, LSB

...

Byte 12: UID, MSB (always **0xE0**)

Byte 13: data storage format identifier (DSFID)

Byte 14: application identifier (AFI)

Byte 15: memory size: block number (0x00...0xFF = 1...256)

Byte 16: memory size: byte/block (0x00...0x1F = 1...32)

Byte 17: IC identifier (IC-reference)

发送数据：AA 07 07 **70** 00 FB 42

接收数据：AA 07 07 70 89 32 5B

AA 14 14 70 8A **78 2A 82 20 00 01 04 E0** 00 00 **1B** **03** 01 36 88

**1B = 28 blocks (0..27)**, **03 = 4 byte / block**

---------------------------------------------------------------------------------------------------------------------------------

**设置读写头参数 – Set Parameters（**Parameterization of the transceiver**）**

**Request**

Command Code : 0x61

Command Index : Request

User Data : 2 byte

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 0xAA | 0x09 | 0x09 | 0x61 | 0x00 | Parameter 1 | Parameter  2 | CRC-16 | CRC-16 |

Byte 5: bit\_0..7 = number of blocks (0x00 = 1 block, 0xFF = 256 blocks)

Byte 6: bit\_0..4 = block size (0x00 = 1 byte/block, 0x1F = 32 byte/block)

Byte 6: bit\_7 = option flag

**Acknowledge**

without

**Response**

Command Code : 0x61

Command Index : Response

User Data : none

Remark: after PON the transceiver works with following default parameters:

- block size 4 byte, 28 blocks, option flag = 0 (equals NXP i-Code SLI/SLIX)

Fixed settings are: High Data Rate, AM with 10% modulation, Coding “1 out of 4”, ISO15693

发送数据：AA 09 09 **61** 00 **1B** **03** 53 32

接收数据：AA 07 07 61 0A E8 61

---------------------------------------------------------------------------------------------------------------------------------

**读取数据 –** **Read Multiple Blocks（**Read multiple blocks, current transceiver limit: 64 byte**）**

读取数据前，需要先执行以下命令：

1. **Get System Information**，获得载码体类型
2. **Set Parameters**，调整读写头设置来匹配正在使用的载码体

**Attention: “how many” always addresses one block more than entered, i.e. 00 = 1 block, 0A = 11 blocks, 10 = 17 blocks**

**Request**

Command Code : 0x68

Command Index : Request

User Data : 2...11 byte

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7...14 | 15 | 16 |
| 0xAA | 0x11 | 0x11 | 0x68 | 0x10 | 0x00 | 0x07 | LSB UID MSB | CRC-16 | CRC-16 |

Byte 5: **block number (start block)**

Byte 6: **number of blocks (0x00 = 1 block, 0xFF = 256 blocks)**

Byte 7...14: UID of the requested TAG (optional, if byte\_4/bit\_4 is set)

Byte 15: CustomerCommandCode (optional, if byte\_4/bit\_5 is set)

**Acknowledge**

Command Code : 0x68

Command Index : Acknowledge

User Data : none

**Response**

Command Code : 0x68

Command Index : Response

User data 1…64 byte (depending on block number and size)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5...36 | 37 | 38 |
| 0xAA | 0x27 | 0x27 | 0x68 | CI | LSB User Data MSB | CRC-16 | CRC-16 |

Byte 5: data, lowest address

…

Byte 36: data, highest address

发送数据：AA 09 09 **68** 00 **00** **00** 92 88（已设置4 byte / block，从第0个block读取1个block数据，即从第0个字节开始，读取4个字节）

接收数据：AA 07 07 68 09 6B 84

AA 0B 0B 68 0A **31 32 33 34** A4 D5（读取4个字节数据**31 32 33 34**）

===============================================================================

发送数据：AA 09 09 **68** 00 **00** **02** 80 AB（已设置4 byte / block，从第0个block读取3个block数据，即从第0个字节开始，读取12个字节）

接收数据：AA 07 07 68 09 6B 84

AA 13 13 68 0A **31 32 33 34 35 36 37 38 39 30 61 62** 2C 40（读取12个字节数据**31 32 33 34 35 36 37 38 39 30 61 62**）

===============================================================================

发送数据：AA 09 09 **68** 00 **01** **01** C3 80（已设置4 byte / block，从第1个block读取2个block数据，即从第4个字节开始，读取8个字节）

接收数据：AA 07 07 68 09 6B 84

AA 0F 0F 68 0A **35 36 37 38 39 30 61 62** 0C D0（读取8个字节数据**35 36 37 38 39 30 61 62**）

---------------------------------------------------------------------------------------------------------------------------------

**写入数据 – Write Multiple Blocks（**Write multiple blocks, current transceiver limit: 64 byte**）**

写入数据前，需要先执行以下命令：

**1．Get System Information**，获得载码体类型

**2．Set Parameters**，调整读写头设置来匹配正在使用的载码体

**Attention: “how many” always addresses one block more than entered, i.e. 00 = 1 block, 0A = 11 blocks, 10 = 17 blocks**

**Request**

Command Code : 0x69

Command Index : Request

User Data : 3...75 byte (depending on block number and size)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7...38 | 39...46 | 47 | 48 |
| 0xAA | 0x31 | 0x31 | 0x69 | 0x10 | 0x00 | 0x07 | LSB UserData MSB | LSB UID MSB | CRC | CRC |

Byte 5: **block number (start block)**

Byte 6: **number of blocks (0x00 = 1 block, 0xFF = 256 blocks)**

Byte 7: data, lowest address

...

Byte 38: data, highest address

Byte 39...46: UID of the requested TAG (optional, if byte\_4/bit\_4 is set)

Byte 47: CustomerCommandCode (optional, if byte\_4/bit\_5 is set)

**Acknowledge**

Command Code : 0x69

Command Index : Acknowledge

User Data : none

**Response**

Command Code : 0x69

Command Index : Response

User Data : none

发送数据：AA 15 15 **69** 00 **00** **02** **11 22 33 44 11 22 33 44 11 22 33 44** 72 1A（已设置4 byte / block，从第0个block写入3个block数据，即从第0个字节开始，写入12个字节**11 22 33 44 11 22 33 44 11 22 33 44**）

接收数据：AA 07 07 69 09 B3 9D

AA 07 07 69 09 B3 9D

AA 07 07 69 0A 28 AF

===============================================================================

发送数据：AA 0D 0D **69** 00 **01** **00** **55 66 77 88** FD 36（已设置4 byte / block，从第1个block写入1个block数据，即从第4个字节开始，写入4个字节**55 66 77 88**）

接收数据：AA 07 07 69 09 B3 9D

AA 07 07 69 0A 28 AF

* **CRC-16校验算法**

**CRC-16 Calculation, C-SourceCode**

A CRC-16 following the ISO/IEC 13239 is used:

Polynomial: x^16 +x^12 + x^5 + x^0

Preset-value: 0xFFFF

Direction: Backward

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**/\* sample code \*/**

**/\* CRC-calculation \*/**

**/\* based on the ISO/IEC 15693 \*/**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**#include <stdio.h>**

**#define POLYNOMIAL 0x8408 // x^16 + x^12 + x^5 + 1**

**#define PRESET\_VALUE 0xFFFF**

**#define NUMBER\_OF\_BYTES 4 // Example: 4 data bytes**

**void main()**

**{**

**unsigned int current\_crc\_value;**

**unsigned char array\_of\_databytes[NUMBER\_OF\_BYTES] = {0xAA, 0x06, 0x49, 0x00};**

**int number\_of\_databytes = NUMBER\_OF\_BYTES;**

**int calculate\_or\_check\_crc;**

**int i, j;**

**number\_of\_databytes = NUMBER\_OF\_BYTES;**

**current\_crc\_value = PRESET\_VALUE;**

**for (i = 0; i < number\_of\_databytes; i++)**

**{**

**current\_crc\_value = current\_crc\_value ^ ((unsigned int)array\_of\_databytes[i]);**

**for (j = 0; j < 8; j++)**

**{**

**if (current\_crc\_value & 0x0001)**

**{**

**current\_crc\_value = (current\_crc\_value >> 1) ^ POLYNOMIAL;**

**}**

**else**

**{**

**current\_crc\_value = (current\_crc\_value >> 1);**

**}**

**}**

**}**

**current\_crc\_value = ~current\_crc\_value;**

**printf ("Generated CRC is '%04X'\n", current\_crc\_value);**

**printf ("The Least Significant Byte (transmitted first) is: '%02X'\n",**

**current\_crc\_value & 0xFF);**

**printf( "The Most Significant Byte (transmitted second) is: '%02X'\n",**

**(current\_crc\_value >> 8) & 0xFF);**

**}**