WIZISP Communication Protocol

Version 1.0



<http://www.wiznet.io>

1. Memory Map in Boot Mode

0x00000000

|  |  |
| --- | --- |
| 0x20000000  0x10000000  0x00001000  0x10020000  0x1FFF1000  0x1FFF1800  0x20004000 |  |
| **SRAM**  **(16KB)** |
|  |
| **IAP API Remap**  **SW Reset**  **(2KB)** |
|  |
| **CODE FLASH**  **(128KB)** |
|
|  |
| **BOOT**  **(4KB)** |

2. Command SET

|  |  |  |
| --- | --- | --- |
| **COMMAND** | **Parameters** | **Description** |
| **ERAS** | <Param1>  <Param1> <Param2> | Erase Flash Sector |
| **DOWN** | <Param1> <Param2> | Download Data into SRAM |
| **PROG** | <Param1> <Param2>  <Param1>…<Param4> | Program SRAM data into Data Sector or Code Area |
| **DUMP** | <Param1> <Param2> | Dump Data from Flash Area |
| **XPRG** | <Param1> <Param2> | Program Data into Flash or SRAM area using XMODEM |
| **LOCK** | <Param1>  <Param1>…<Param3> | Read Lock Information or  Program Lock Information |
| **REST** | <No Param> | Reset by software |

Every command should terminate with <CR>. W7500(P) ISP Code waits <CR> to manipulate a command.

2.1 ERAS

|  |  |  |
| --- | --- | --- |
| **Command Format** | **Example** | **Description** |
| ERAS **DAT0** | ***ERAS DAT0*** | Erase Data Sector 0 |
| ERAS **DAT1** | ***ERAS DAT1*** | Erase Data Sector 1 |
| ERAS **SECT <Addr>** | ***ERAS SECT 10001000*** | Erase Code Sector the starting address of which is 0x00001000 |
| ERAS **BLCK <Addr>** | ***ERAS BLCK 10002000*** | Erase Code Block the starting address of which is 0x00002000 |
| ERAS **CHIP** | ***ERAS CHIP*** | Erase all Code Blocks |
| ERAS **MASS** | ***ERAS MASS*** | Erase all Code Blocks and Data Sectors |

2.2 DOWN

|  |  |  |
| --- | --- | --- |
| **Command Format** | **Example** | **Description** |
| DOWN **<Addr[[1]](#footnote-1)>** **<Size>[[2]](#footnote-2) <Data[[3]](#footnote-3)>** | ***DOWN 20000000 00000100 abcd….*** | Download Data into SRAM from <Addr> for <Size> bytes.  <Addr> should be belonged to SRAM area and <Size> is Hexadecimal format. |

2.3 PROG

|  |  |  |
| --- | --- | --- |
| **Command Format** | **Example** | **Description** |
| PROG **DAT0** | ***PROG DAT0 20000000*** | Program Data in SRAM into Data Sector 0 |
| PROG **DAT1** | ***PROG DAT1 20000000*** | Program Data in SRAM into Data Sector 1 |
| PROG **CODE <Addr1> <Addr2> <Size>** | ***PROG CODE 10000000 200000000 00000100*** | Program Data in SRAM from <Addr2> address by <Size> bytes into Flash starting <Addr1>.  Data should be in SRAM via ‘DOWN’ command before issuing ‘PROG’ command. |

2.4 DUMP

|  |  |  |
| --- | --- | --- |
| **Command Format** | **Example** | **Description** |
| DUMP **<Addr>** **<Size>** | ***DUMP 20000000 00000100[[4]](#footnote-4)*** | Dump Data in Flash/SRAM by <Size> bytes from <Addr> address. |

2.5 XPRG

|  |  |  |
| --- | --- | --- |
| **Command Format** | **Example** | **Description** |
| XPRG **<Addr>** **<Size>** | ***XPRG 10000000 00000100*** | Program Data into Flash or SRAM by <Size> bytes from <Addr> address and Data should be transferred via XModem protocol. |

2.6 LOCK

|  |  |  |
| --- | --- | --- |
| **Command Format** | **Example** | **Description** |
| LOCK **READ** | ***LOCK READ[[5]](#footnote-5)*** | Read current Lock Information |
| LOCK PROG <FLOCKR0[[6]](#footnote-6)> <FLOCKR1[[7]](#footnote-7)> | ***LOCK PROG 80000000 00000000*** | Program Lock Information |

2.6.1 Flash Lock Register 1 (FLOCKR0)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **31** | **30** | **29 ~ 4** | **3** | **2** | **1** | **0** |
| CRL | CBWLA | Reserved | DRL1 | DRL0 | DWL1 | DWL0 |

**CRL** : Code Read Lock, all code blocks are locked for reading with this bit set.

**CBWLA** : All Code Block Write Lock, all code blocks are locked for writing with this bit set.

**DRL1/0** : Data Sector 1/0 Read Lock

**DWL1/0** : Data Sector 1/0 Write Lock

2.6.2 Flash Lock Register 2 (FLOCKR1)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **31** | **30** | **29 ~ 4** | **3** | **2** | **1** | **0** |
| CWL31 | CWL30 | … | CWL3 | CWL2 | CWL1 | CWL0 |

**CWL[31:0]** : Write Lock for the corresponding Code Block

2.7 REST

|  |  |  |
| --- | --- | --- |
| **Command Format** | **Example** | **Description** |
| REST[[8]](#footnote-8) | ***REST*** | Reset by software |

3. Return Value

|  |  |
| --- | --- |
| **Return Value (ASCII)** | **Description** |
| 0 | Succeeded |
| 1 | Invalid Size |
| 2 | Invalid Address |
| 3 | Invalid Command |
| 4 | No Privilege to Flash |
| 5 | Invalid Parameter |
| 6 | Read Lock Protected |
| 7 | Write Lock Protected |
| 8 | Reset |

Every Response is followed by <CR><LF>

4. Working Flow

Make W7500P run in boot mode

applying LOW to BOOT pin in HW

Send ‘U’ until ‘U’ echoes back

Send ISP command according to command format

Receive Return Value to check if command succeeded

1. <Addr> Parameter should be eight Hex format ASCII characters with ‘0’s preceding if needed. [↑](#footnote-ref-1)
2. <Size> Parameter should be eight Hex format ASCII characters with ‘0’s preceding if needed. [↑](#footnote-ref-2)
3. <Data> should be binary code. For example, 0x11 0x12 0x13 …. [↑](#footnote-ref-3)
4. The format of Reply is **<Addr>:<Data>**. ***For example, 10000000:B2C06940*** [↑](#footnote-ref-4)
5. The format of Reply is two eight-length hexadecimal format Lock information strings and Return value.

   For example, ***C0000000 FFFFFFFF<CR><LF>0*** [↑](#footnote-ref-5)
6. Flash Lock Register 0 [↑](#footnote-ref-6)
7. Flash Lock Register 1 [↑](#footnote-ref-7)
8. No Parameter [↑](#footnote-ref-8)