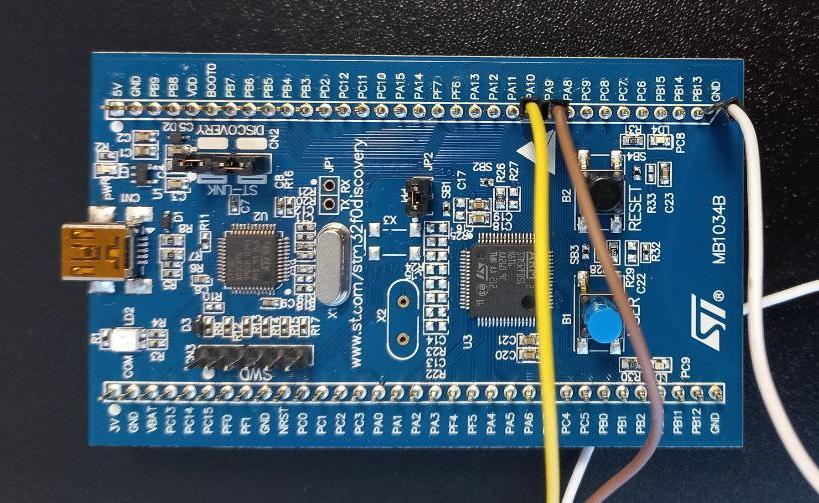


**Protocol and checksum algorithm**



This device will encrypt

and decrypt files.

Also, UART is used for

transferring commands

between PC and STM.

**Command list:**  
 All request commands are presented in six-byte numeric format:

| Command | Specific | | | | BCC |
| --- | --- | --- | --- | --- | --- |
| 1 byte | 1 byte | 1 byte | 1 byte | 1 byte | 1 byte |

In which the first byte specifies the command number, the next 4 are responsible for the type of the sent command, and the sixth byte performs the calculation of the BCC. The device uses the XOR (example below) method to calculate the BCC.

Example:

| Command | Specific | | | | BCC |
| --- | --- | --- | --- | --- | --- |
| 0x05 | 0x01 | 0x00 | 0x00 | 0x01 | 0x03 |

1. 0x05 XOR 0x01 = 0x04 - *comparing the first and second byte;*   
 2. 0x04 XOR 0x00 = 0x04 - *comparing the result of the first and second byte and the third byte*;   
 3. 0x04 XOR 0x00 = 0x04 - *comparing the result of the previous step with the third byte;*   
 4. 0x04 XOR 0x01 = 0x03 - *result of the BCC.*



1. **Is device connected**

*This command checks the connection between the PC and the STM.*

* *PC->STM sends the command*

| Command | Specific | | | | BCC |
| --- | --- | --- | --- | --- | --- |
| 0x01 | 0x00 | 0x00 | 0x00 | 0x01 | 0x00 |

* *STM->PC sends a response*

| Command | Response. | | | | | | BCC |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0x01 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x01 | 0x00 |

*If STM is connected, proceed to the next step.*

*Otherwise, it displays the message: "STM is not connected".*



1. **Encrypt**

*This command sends a file from the PC to the STM for encryption and then returns the encrypted file to the PC.*

*ー file selection*

*ー reading file by blocks*

*ー sending blocks to COM port*

*ー receiving encrypted blocks*

*ー combining blocks into output file*

* *PC->STM sends the file to be encrypted*

| Command | Specific | | | | BCC |
| --- | --- | --- | --- | --- | --- |
| 0x02 | 0x00 | 0x00 | 0x00 | 0x01 | 0x01 |

* *STM->PC reads the file encrypts and sends the encrypted file by sending the command*

| Command | Response. | | | | | | BCC |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0x02 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x01 | 0x01 |

*If the output is an encrypted file, we move on to the next step.*

*Otherwise, it displays the message: "File could not be encrypted".*



1. **Decrypt**

*This command sends an encrypted file from the PC to the STM for decryption and then returns the decrypted file to the PC. Decryption functionality is similar to encryption.*

* *PC->STM sends the file to be decrypted*

| Command | Specific | | | | BCC |
| --- | --- | --- | --- | --- | --- |
| 0x03 | 0x00 | 0x00 | 0x00 | 0x01 | 0x02 |

* *STM->PC reads the file decrypts it and sends the decrypted file by sending the command*

| Command | Response. | | | | | | BCC |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0x03 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x01 | 0x02 |

*If the output is a decrypted file, we move on to the next step.*

*Otherwise, it displays the message: "File could not be decrypted".*



1. **Ask max block size**

*This command allows you to define the max block size*

* *PC->STM sends a command requesting the maximum data size that can be processed*

| Command | Specific | | | | BCC |
| --- | --- | --- | --- | --- | --- |
| 0x04 | 0x00 | 0x00 | 0x00 | 0x01 | 0x03 |

* *STM->PC sends a command with the maximum data size*

| Command | Response. | | | | | | BCC |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0x04 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x01 | 0x03 |

*The output is the maximum data size that can be processed.*

*Otherwise, it displays a message: "Error".*

**Encryption algorithm**

***XXTEA*** *- is a block cipher designed to correct weaknesses in the original Block TEA.Is a fast and secure encryption algorithm.*

*It is different from the original XXTEA encryption algorithm. It encrypts and decrypts byte array instead of 32bit integer array, and the key is also the byte array.*

*Advantages: High speed of operation Low memory consumption Simple software implementation Relatively high reliability.*

*Description of how the algorithm works:*

*The source text is split into words of 32 bits each, and a block is formed from the resulting words. The key is also split into 4 parts, consisting of words of 32 bits each, and a key array is formed. During one round of the algorithm, one word from the block is encrypted. After the words have been encrypted, the cycle ends and a new one begins. The number of cycles depends on the number of words and is equal to 6 + 52 /n , where n is the number of words. The encryption of one word is as follows*

* *The left neighbour is shifted by two bits to the left, and the right neighbour by five bits to the right. The resulting values are added bitwise modulo 2.*
* *A bit shift operation is performed on the left neighbour by three, and a bit shift operation is performed on the right neighbour by 4. The resulting values are subjected to a bitwise addition operation modulo 2*
* *The resulting numbers add up to 2^32 modulo.*
* *The constant δ, derived from the Golden Ratio (δ = (sqrt (5 - 1) \* 2^31 = 2654435769 = 9E3779B9h) is multiplied by the cycle number (this was done to prevent simple attacks based on round symmetry).*
* *The number obtained in the previous paragraph is added bitwise modulo 2 with the right neighbour.*
* *The number obtained in step 4 is shifted bitwise to the right by 2, added bitwise modulo two with the round number, and the remainder of the division by 4 is found. Using the resulting number, select a key from the key array.*
* *The key selected in the previous round is added bitwise modulo 2 to the left neighbour.*
* *The numbers obtained in the previous and 4 paragraphs add up to 2^32 modulo.*
* *The numbers obtained in the previous and 3 paragraphs are added bit by bit modulo 2, this sum is added to the cipher word modulo 2^32.*