Wireless Module Expert

# UC15 FILE

# AT Commands Manual

**UMTS/HSPA Module Series**

Rev. UC15_FILE_AT_Commands_Manual_V1.0

Date: 2013-11-15

**Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarters:**

**Quectel Wireless Solutions Co., Ltd.**
Office 501, Building 13, No.99, Tianzhou Road, Shanghai, China, 200233
Tel: +86 21 5108 6236
Mail: info@quectel.com

**Or our local office, for more information, please visit:**
http://www.quectel.com/support/salesupport.aspx

**For technical support, to report documentation errors, please visit:**
http://www.quectel.com/support/techsupport.aspx

## GENERAL NOTES

QUECTEL OFFERS THIS INFORMATION AS A SERVICE TO ITS CUSTOMERS. THE INFORMATION PROVIDED IS BASED UPON CUSTOMERS' REQUIREMENTS. QUECTEL MAKES EVERY EFFORT TO ENSURE THE QUALITY OF THE INFORMATION IT MAKES AVAILABLE. QUECTEL DOES NOT MAKE ANY WARRANTY AS TO THE INFORMATION CONTAINED HEREIN, AND DOES NOT ACCEPT ANY LIABILITY FOR ANY INJURY, LOSS OR DAMAGE OF ANY KIND INCURRED BY USE OF OR RELIANCE UPON THE INFORMATION. ALL INFORMATION SUPPLIED HEREIN ARE SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.

## COPYRIGHT

# About the Document

## History

| Revision | Date | Author | Description |
|---|---|---|---|
| 1.0 | 2013-11-15 | Gibson PAN | Initial |

# Contents

# 1 Introduction

Quectel module provides AT commands to operate files on different physical storage mediums. This document is a reference guide to these commands.

The supported storage mediums are as follows:

- **UFS:** User File Storage directory. It is a special directory on the flash file system.
- **RAM:** Random Access Memory. It is faster but much smaller than the UFS. And the files in the RAM will be lost when rebooting module.

The file name indicates the storage location. When the file name begins with “RAM:”, it means the file is located in RAM. If there are no prefix characters in the file name, the file is located in UFS.

## 1.1. The Process of Using File AT Commands

There are two modes to create, read and write the file in the storage:

1. The file is created and all the content of the file could be uploaded to the storage by command “AT+QFUPL”. And the content can be outputted/downloaded through the UART/USB by command “AT+QFDWL”.

2. Open the file by “AT+QFOPEN”, then the file can be written or read at any time and any location until the file is closed by “AT+QFCLOSE”.

   - When the file is opened by command “AT+QFOPEN”, you can set the file as overwrite mode or read-only mode or others by the parameter <mode>. (For more information about <mode>, see Section 2.7). After the file is opened, a <filehandle> is assigned to this file. Then you can operate this file via this <filehandle>.
   - After the file is opened, write the file by command “AT+QFWRITE” and read the file by “AT+QFREAD” from the current file position.
   - You can set the file position by “QFSEEK” and get the current position by “AT+QFPOSITION".
   - “QFFLUSH” will save the file to the physical storage immediately.
   - “QFTUCAT” will truncate the file from the current position to the end of the file.
   - Close the file by “AT+QFCLOSE”. Then the <filehandle> becomes meaningless to this file.

There are several commands to manage files in the storage.

1. “AT+QFLDS” gets the storage size.
2. “AT+QFLST” lists files information in the storage.
3. “AT+QFDEL” deletes the file(s).
4. “AT+QFMOV” moves the file from one storage to another.

# 2 Description of AT Command

## 2.1. AT+QFLDS Get the Space Information of the Storage

AT+QFLDS responds the space information of the specified storage.

| AT+QFLDS Get the Space Information of the Storage | |
|---|---|
| Test Command<br>**AT+QFLDS=?** | Response<br>**OK** |
| Write Command<br>**AT+QFLDS=<namepattern>** | Response<br>**+QFLDS: <freesize>,<totalsize>**<br><br>**OK**<br><br>**+CME ERROR: <errcode>** |
| Execution Command<br>**AT+QFLDS** | Response<br>**+QFLDS: <ufsfilesize>,<ufsfilenumber>**<br><br>**OK**<br><br>**+CME ERROR: <errcode>**<br>Return the UFS information |

### Parameter

| | | |
|---|---|---|
| **<namepattern>** | Pattern. | |
| | "UFS" | UFS files in flash. |
| | "RAM" | RAM files in the random storage. |
| **<freesize>** | The size of the free space in <namepattern>. | |
| **<totalsize>** | The total size of the storage <namepattern>. | |
| **<ufsfilesize>** | The size in bytes of all files in UFS. | |
| **<ufsfilenumber>** | The number of files in UFS. | |
| **<errcode>** | The error code from the module (see the Appendix A). | |

### Example

```
AT+QFLDS="UFS"
+QFLDS: 69019072,73453568

OK
```

## 2.2. AT+QFLST List Files

AT+QFLST lists the information of a single file or all files in the required storage medium.

| AT+QFLST List Files | |
|---|---|
| Test Command<br>**AT+QFLST=?** | Response<br>**OK** |
| Write Command<br>**AT+QFLST=<namepattern>** | Response<br>**[+QFLST: <filename>,<filesize>**<br>**+QFLST: <filename>,<filesize>**<br>**[…]]**<br><br>**OK**<br><br>**+CME ERROR: <errcode>** |
| Execution Command<br>**AT+QFLST** | Response<br>Return the information of the UFS files:<br>**[+QFLST: <filename>,<filesize>**<br>**+QFLST: <filename>,<filesize>**<br>**[…]]**<br><br>**OK**<br><br>**+CME ERROR: <errcode>** |

### Parameter

| | | |
|---|---|---|
| **<namepattern>** | The file to be listed. | |
| | "*" | All the files in UFS. |
| | "RAM:*" | All the files in RAM. |
| | "<filename>" | The specified file <filename> in UFS |
| | "RAM:<filename>" | The specified file <filename> in RAM. |
| **<filename>** | Name of the file. | |
| **<filesize>** | Size in bytes of the file. | |
| **<errcode>** | The error code from the module (see the Appendix A). | |

### Example

```
AT+QFLST="*"
+QFLST: "F_M12-1.bmp",562554
+QFLST: "F_M12-10.bmp",562554
+QFLST: "F_M12-11.bmp",562554

OK
```

## 2.3. AT+QFDEL Delete the File in the Storage

AT+QFDEL deletes a single file or all the files in the specified storage.

| AT+QFDEL Delete the File in the Storage | |
|---|---|
| Test Command<br>**AT+QFDEL=?** | Response<br>**+QFDEL: <filename>**<br><br>**OK** |
| Write Command<br>**AT+QFDEL=<filename>** | Response<br>**OK**<br><br>**+CME ERROR: <errcode>** |

### Parameter

| | | |
|---|---|---|
| **<filename>** | The file to be deleted. The max length of <filename> is 80 bytes. | |
| | "*" | Delete all the files in UFS (not delete the directory). |
| | "RAM:*" | Delete all the files in RAM. |
| | "<filename>" | Delete the specified file <filename> in UFS. |
| | "RAM:<filename>" | Delete the specified file <filename> in RAM. |
| **<errcode>** | The error code from the module (see the Appendix A). | |

## 2.4. AT+QFMOV Move the File

AT+QFMOV moves the file from one storage to another storage or copies an existing file to a new file in one storage. It only supports moving the file from RAM to UFS, not from UFS to RAM.

| AT+QFMOV Move the File | |
|---|---|
| Test Command<br>**AT+QFMOV=?** | Response<br>**+QFMOV: <srcfilename>,<destfilename>,(0,1),(0,1)** |

| | |
|---|---|
| | **OK** |
| Write Command<br>**AT+QFMOV=<srcfilename>,<destfilename>,<copy>,<overwrite>** | Move a single file:<br>**AT+QFMOV="RAM:filenamea","flienameb",<copy>,<overwrite>**<br>Move the file named "filenamea" in RAM to UFS, and renamed it as "filenameb".<br>Move all files:<br>**AT+QFMOV="RAM: *", "*",<copy>,<overwrite>**<br>Move all files in RAM to UFS.<br>Response<br>**OK**<br><br>**+CME ERROR: <errcode>** |

**Parameter**

| | |
|---|---|
| **<srcfilename>** | The source file. The max length is 80 bytes.<br>"<srcfilename>" The source file named <srcfilename>. |
| **<destfilename>** | The destination file. The max length is 80 bytes.<br>"<destfilename>" The destination file named <destfilename>. |
| **<copy>** | Whether or not to delete the source file after the file is moved.<br>0 Delete the source file after it is moved, i.e., cut this file.<br>1 Do not delete the source file after it is moved, i.e., copy this file. |
| **<overwrite>** | Whether or not to overwrite the file if the destination file exists.<br>0 Do not overwrite the destination file if it exists.<br>1 Overwrite the destination file if it exists. |
| **<errcode>** | The error code from the module (see the Appendix A). |

## 2.5. AT+QFUPL Upload File to the Storage

1. AT+QFUPL uploads the file to the module directly. If there is a file in the storage which has the same name with the file to be uploaded, it will report error.

2. There are three ways to exit the transparent transmission mode:
   - The data uploaded reaches the <filesize>.
   - The time without any data inputted reaches <timeout>.
   - When the data is transmitted, the DTR PIN (AT&D1 should be set first.) is pulled high or the valid "+++" is inputted.

3. To prevent the "+++" from being misinterpreted as data, it should comply to the following sequence:
   - Do not input any character within T1 time (1 second) before inputting "+++".
   - Input "+++" during 1s, and no other characters can be inputted during this time.
   - Do not input any character within T1 time (1 second) after "+++" has been inputted.
   - The current result "+QFUPL: <uploadsize>, <checksum>" is outputted, and the module exits the transparent transmission mode, return OK.

| AT+QFUPL Upload File to the Storage | |
|---|---|
| Test Command<br>**AT+QFUPL=?** | Response<br>**+QFUPL: <filename>[,(1-<freesize>)[,(1-65535)[,(0,1)]]]**<br><br>**OK** |
| Write Command<br>**AT+QFUPL=<filename>[,<filesize>[ ,<timeout>[,<ackmode>]]]** | Response<br>**CONNECT**<br>TA switches to the transparent transmission mode, and the binary data of file can be inputted. When the total size of the inputted data reaches <filesize> (unit: byte), TA will return to command mode and reply the following codes:<br>**+QFUPL: <uploadsize>,<checksum>**<br>**OK**<br><br>**+CME ERROR: <errcode>** |

**Parameter**

| | |
|---|---|
| **<freesize>** | The size of the free space in <namepattern>. Please refer to the "+QFLDS". |
| **<filename>** | The name of file to be stored. The max length is 80 bytes.<br>"<filename>" The name of file uploaded to UFS.<br>"RAM:<filename>" The name of file uploaded to RAM. |
| **<filesize>** | The file size expected to be uploaded.<br>Default is 10240. Unit: byte. |
| **<uploadsize>** | The actual size of the uploaded data. Unit: byte. |
| **<timeout>** | The delay time in seconds of waiting for data to be inputted to USB/UART. Default is 5s. |
| **<ackmode>** | Whether to use ACK mode.<br>0 Turn off the ACK mode by default.<br>1 Turn on the ACK mode. |
| **<checksum>** | The checksum of the uploaded data. |
| **<errcode>** | The error code from the module (see the Appendix A). |

**NOTES**

1. It is strongly recommended to use DOS 8.3 file name format for <filename>.
2. <checksum> is a 16 bit checksum based on bitwise XOR.
   If the number of the characters is odd, set the last character as the high 8 bit, and the low 8 bit as 0, and then use an XOR operator to calculate the checksum.
3. "+++" sequence will cause TA to end the command and switch to command mode; however, the data previously uploaded will be preserved into the file.
4. When executing the command, the data must be entered after CONNECT appears.
5. The ACK mode is provided to avoid the loss of data when uploading large files, in case hardware flow control doesn't work. The ACK mode works as follows:
   1) Run AT+QFUPL=<filename>,<filesize>,<timeout>,1 command to enable the ACK mode.
   2) The module outputs CONNECT.
   3) MCU sends 1K bytes data, and then the module will respond with an 'A'.
   4) MCU receives this 'A' and then sends the next 1K bytes data;
   5) Repeat step 3) and 4) until the transfer is completed.

## 2.6. AT+QFDWL Download the File from the Storage

AT+QFDWL downloads the specified file from the module storage.

| AT+QFDWL Download the File from the Storage | |
|---|---|
| Test Command<br>**AT+QFDWL=?** | Response<br>**+QFDWL: <filename>**<br><br>**OK** |
| Write Command<br>**AT+QFDWL=<filename>** | Response<br>**CONNECT**<br>TA switches to data mode, and the bin data of the file will be outputted. When the file is read over, TA will return to command mode and reply the following codes:<br>**+QFDWL: <downloadsize>,<checksum>**<br><br>**OK**<br><br>**+CME ERROR: <errcode>** |

### Parameter

| | | |
|---|---|---|
| **<filename>** | The name of the file to be downloaded. The max length is 80 bytes. | |
| | "<filename>" | The downloaded file in UFS. |

| | |
|---|---|
| | "RAM: <filename>" The downloaded file in RAM. |
| **<downloadsize>** | The size of the downloaded data. |
| **<checksum>** | The checksum of the downloaded data. |
| **<errcode>** | The error code from the module (see the Appendix A). |

**NOTES**

1. "+++" sequence will cause TA to end this command and switch to command mode.
2. <checksum> is a 16 bit checksum based on bitwise XOR.

## 2.7. AT+QFOPEN Open the File

Get the file handle by executing the "AT+QFOPEN" which is used in other commands, such as "AT+QFWRITE", "AT+QFREAD", "AT+QFSEEK", "AT+QFCLOSE", "AT+QFPOSITION", "AT+QFFLUSH" and "AT+QFTUCAT".

| **AT+QFOPEN Open the File** | |
|---|---|
| Test Command<br>**AT+QFOPEN=?** | Response<br>**+QFOPEN: <filename>[,(0-2)]**<br><br>**OK** |
| Read Command<br>**AT+QFOPEN?** | Response<br>**[+QFOPEN: <filename>,<filehandle>,<mode>**<br>**+QFOPEN: <filename>,<filehandle>,<mode>**<br>**[…]]**<br><br>**OK** |
| Write Command<br>**AT+QFOPEN=<filename>[,<mode>]** | Response<br>**+QFOPEN: <filehandle>**<br><br>**OK**<br><br>**+CME ERROR: <errcode>** |

### Parameter

| | |
|---|---|
| **<filename>** | The file to be operated. The max length is 80 bytes.<br>"<filename>" The operated file in the UFS.<br>"RAM: <filename>" The operated file in the RAM. |
| **<filehandle>** | The handle of the file. The data type is 4 bytes. |
| **<mode>** | The open mode of the file. Default is 0. |

| | |
|---|---|
| | 0 If the file does not exist, it will be created; if the file exists, it will be directly opened. And both of them can be read and written.<br>1 If the file does not exist, it will be created; If the file exists, the file will be overwritten and cleared. And both of them can be read and written.<br>2 If the file exists, open it and can be read only. When the file does not exist, it will respond the error. |
| **<errcode>** | The error code from the module (see the Appendix A). |

## 2.8. AT+QFREAD Read the File

AT+QFREAD reads the data of the file related to the handle. The data starts from the current position of the file pointer which belongs to the file handle.

| **AT+QFREAD Read the File** | |
|---|---|
| Test Command<br>**AT+QFREAD=?** | Response<br>**+QFREAD: <filehandle>[,<length>]**<br><br>**OK** |
| Write Command<br>**AT+QFREAD=<filehandle>[,<length>]** | Response<br>**CONNECT <readlength>**<br>TA switches to data mode. When the total size of the data reaches <length> (unit: byte), TA will return to command mode, display the result and then reply the following codes:<br>**OK**<br><br>**+CME ERROR: <errcode>** |

**Parameter**

| | |
|---|---|
| **<filehandle>** | The handle of the file to be operated. |
| **<length>** | The length of the file to be read out and the default is 10KB. |
| **<readlength>** | The actual read length. |
| **<errcode>** | The error code from the module (see the Appendix A). |

## 2.9. AT+QFWRITE Write the File

AT+QFWRITE writes the data to the file in the module. The data starts from the current position of the file pointer which belongs to the file handle.

| AT+QFWRITE Write the File | |
|---|---|
| Test Command<br>**AT+QFWRITE=?** | Response<br>**+QFWRITE: <filehandle>[,<length>[,<timeout>]]**<br><br>**OK** |
| Write Command<br>**AT+QFWRITE=<filehandle>[,<length>[,<timeout>]]** | Response<br>**CONNECT**<br>TA switches to data mode. When the total size of the written data reaches <length> (unit: byte) or the time, TA will return to command mode and reply the following codes:<br>**+QFWRITE: <writtenlength>,<totallength>**<br><br>**OK**<br><br>**+CME ERROR: <errcode>** |

**Parameter**

| | |
|---|---|
| **<filehandle>** | The handle of the file to be operated. |
| **<length>** | The length of the file to be written, the default length is 10K. The range of this parameter is same with the <freesize> of the "AT+QFUPL". |
| **<timeout>** | The time of waiting data to be inputted to USB/UART . Default is 5s. |
| **<writtenlength>** | The actual written length. |
| **<totallength>** | The total length of the file. |
| **<errcode>** | The error code from the module (see the Appendix A). |

## 2.10. AT+QFSEEK Seek the File

Set the current position of the file pointer which belongs to the file handle. This will decide the starting position of the "AT+QFREAD", "AT+QFWRITE", "AT+QFPOSITION" and "AT+QFTUCAT".

| AT+QFSEEK Seek the File | |
|---|---|
| Test Command<br>**AT+QFSEEK=?** | Response<br>**+QFSEEK: <filehandle>,<offset>[,<position>]**<br><br>**OK** |
| Write Command<br>**AT+QFSEEK=<filehandle>,<offset>[,<position>]** | Response<br>**OK**<br><br>**+CME ERROR: <errcode>** |

### Parameter

| | |
|---|---|
| **<filehandle>** | The handle of the file to be operated. |
| **<offset>** | The number of bytes of the file pointer movement. |
| **<position>** | Pointer movement mode. Default is 0.<br>0 The beginning of the file.<br>1 The current position of the pointer.<br>2 The end of the file. |
| **<errcode>** | The error code from the module (see the Appendix A). |

**NOTES**

1. If <position> is 0, and the <offset> exceeds the file size, the command will return ERROR.
2. If <position> is 1, and the total size of the <offset> with the current position of the pointer exceeds the file size, the command will return ERROR.
3. If <position> is 2, the handle will move forward.

## 2.11. AT+QFPOSITION Get the Offset of the File Pointer

AT+QFPOSITION gets the current position of the file pointer which is relevant to the file handle.

| **AT+QFPOSITION Get the Offset of the File Pointer** | |
|---|---|
| Test Command<br>**AT+QFPOSITION=?** | Response<br>**+QFPOSITION: <filehandle>**<br><br>**OK** |
| Write Command<br>**AT+QFPOSITION=<filehandle>** | Response<br>**+QFPOSITION: <offset>**<br><br>**OK**<br><br>**+CME ERROR: <errcode>** |

### Parameter

| | |
|---|---|
| **<filehandle>** | The handle of the operated file. |
| **<offset>** | The offset from the beginning of the file. |
| **<errcode>** | The error code from the module (see the Appendix A). |

## 2.12. AT+QFFLUSH Force to Write the Data Remaining in the Buffer

AT+QFFLUSH will trigger the action to copy the data from the random access memory buffer to the physical flash.

| AT+QFFLUSH Force to Write the Data Remaining in the Buffer | |
|---|---|
| Test Command<br>**AT+QFFLUSH=?** | Response<br>**+QFFLUSH: <filehandle>**<br><br>**OK** |
| Write Command<br>**AT+QFFLUSH=<filehandle>** | Response<br>**OK**<br><br>**+CME ERROR: <errcode>** |

### Parameter

| | |
|---|---|
| **<filehandle>** | The handle of the operated file. |
| **<errcode>** | The error code from the module (see Appendix A). |

## 2.13. AT+QFTUCAT Truncate the File from the File Pointer

AT+QFTUCAT will truncate all the data behind the position indicated by the file pointer.

| AT+QFTUCAT Truncate the File from the File Pointer | |
|---|---|
| Test Command<br>**AT+QFTUCAT=?** | Response<br>**+QFTUCAT: <filehandle>**<br><br>**OK** |
| Write Command<br>**AT+QFTUCAT=<filehandle>** | Response<br>**OK**<br><br>**+CME ERROR: <errcode>** |

### Parameter

| | |
|---|---|
| **<filehandle>** | The handle of the operated file. |
| **<errcode>** | The error code from the module (see the Appendix A). |

## 2.14. AT+QFCLOSE Close the File

AT+QFCLOSE closes the file and ends the operation to the file. The file handle is released and should not be used again, unless open the file again by “AT+QFOPEN”.

| AT+QFCLOSE Close the File | |
|---|---|
| Test Command<br>**AT+QFCLOSE=?** | Response<br>**+QFCLOSE: <filehandle>**<br><br>**OK** |
| Write Command<br>**AT+QFCLOSE=<filehandle>** | Response<br>**OK**<br><br>**+CME ERROR: <errcode>** |

### Parameter

| | |
|---|---|
| **<filehandle>** | The handle of the operated file. |
| **<errcode>** | The error code from the module (see Appendix A). |

# 3 Example

## 3.1. Upload and Download Files

### 3.1.1. Upload the File

#### 3.1.1.1. Non ACK Mode

```
AT+QFUPL="test1.txt",10                    //Upload the text file "test1.txt" to UFS.
CONNECT

<Input file bin data>

+QFUPL: 10,613e                            //Get the bytes of the uploaded data and the checksum

OK

AT+QFUPL="RAM:test2.txt",4222              //Upload the text file "test2.txt" to RAM.
CONNECT

<Input Data>

+QFUPL: 4222,13E4                          //Get the bytes of the uploaded data and the checksum.

OK
```

#### 3.1.1.2. ACK Mode

The ACK mode can make the data transmission more reliable. When transmitting the large file without hardware flow control, the ACK mode can be used to prevent the data from being lost. About the ACK mode, please refer to the details of "AT+QFUPL".

```
AT+QFUPL="test.txt",3000,10,1              //Upload the text file "test.txt" to UFS.
CONNECT

<Input file bin data of 1024 bytes>

A                                          //After receiving 1024 bytes data, the module will respond
```

with an “A”, then the next 1024 bytes data can be inputted.

```
<Input file bin data of 1024 bytes>

A

<Input the rest file bin data>

+QFUPL: 3000,B34A

OK
```

### 3.1.2. Download the File

```
AT+QFDWL="test.txt"                    //Download the text file “test.txt” from UFS.
CONNECT

<Output Data>

+QFDWL: 10,613e                        //Get the bytes of the downloaded data and the checksum.

OK
```

## 3.2. Write and Read the File

### 3.2.1. Write and Read UFS File

```
AT+QFOPEN=”test”,0                     //Open the file to get the file handle.
+QFOPEN: 0

OK

AT+QFWRITE=0,10                        //Write 10 bytes to the file.
CONNECT

<Write Data>

+QFWRITE: 10,10                        //The actual bytes written and the size of the file are returned.

OK

AT+QFSEEK=0,0,0                        //Set the file pointer to the beginning of the file.
OK

AT+QFREAD=0,10                         //Read 10 bytes from the file.
CONNECT 10
```

```
<Read Data>

OK

AT+QFCLOSE=0                    //Close the file.
OK
```

### 3.2.2. Write and Read RAM File

```
AT+QFLDS="RAM"                  //Query the space information of RAM.
+QFLDS:2616320, 2621440

OK

AT+QFOPEN="RAM:1",0             //Open the file in the RAM.
+QFOPEN: 3000

OK

AT+QFWRITE=3000,10              //Write 10 bytes to the file.
CONNECT

<Write Data>

+QFWRITE: 10,10                 //The actual written bytes and the size of the file are returned.

OK

AT+QFSEEK=3000,0,0              //Set the file pointer to the beginning of the file.
OK

AT+QFREAD=3000,10               //Read the data.
CONNECT

<Read Data>

OK
AT+QFCLOSE=3000                 //Close the file.
OK
```

## 3.3. Move the File

### 3.3.1. Move a Single File

```
AT+QFLST="RAM:*"                    //List the files in the RAM.
+QFLST: "RAM:123.TXT",37
+QFLST: "RAM:test",12

OK

AT+QFMOV="RAM:test", "test",0,1     //Move the file in the RAM to the UFS.
OK

AT+QFLST="RAM:*"                    //The source file is deleted.
+QFLST: "RAM:123.TXT",37

OK

AT+QFLST="*"                        //There is a file named "test" in the UFS.
+QFLST: "test",12

OK
```

### 3.3.2. Move all Files of One Storage

```
AT+QFLST="RAM:*"                    //List the files in the RAM.
+QFLST: "RAM:1.TXT",10
+QFLST: "RAM:2.TXT",15

OK
AT+QFLST                            //List the files in the UFS.
OK
AT+QFMOVE="RAM:*","*",0,0           //Move all files in RAM to UFS, the first "0"means deleting all
                                      source files after they are moved. The second "0" means
                                      do not overwrite the destination files if exist.
OK
AT+QFLST="RAM:*"                    //Source files are deleted.
OK
AT+QFLST                            //All files in RAM are moved to UFS.
+QFLST: "1.TXT",10
+QFLST: "2.TXT",15

OK
```

# 4 Appendix A Summary of <errcode>

The result of the final error code is “+CME ERROR: <err>”. <err> indicates an error relating to the ME or Network. The operation is similar to error of result code. It will be returned when some definition error happens. The <err> codes listed here are just related to the file of the module.

| Codes of <err> | Meaning |
|---|---|
| 400 | invalid input value |
| 401 | larger than the size of the file |
| 402 | read zero bytes |
| 403 | drive is full |
| 404 | fail to move the file |
| 405 | file not found |
| 406 | invalid file name |
| 407 | file already exists |
| 409 | failed to write the file |
| 410 | failed to open the file |
| 411 | failed to read the file |
| 413 | reach the max number of file allowed to be opened |
| 414 | the file read-only |
| 416 | invalid file descriptor |
| 417 | failed to list the file |
| 418 | failed to delete the file |
| 419 | failed to get disk info |

| | |
|---|---|
| 420 | no space |
| 421 | time out |
| 423 | file too large |
| 425 | invalid parameter |
| 426 | file already operated |