BOS User's Manual [v 1.00]

Contents

1. BOS Board Operation

a. BOS Boot Message.

```
=====================================================
Developed by yjw, Dasuhyun Elec. [www.dasuhyun.com],
BOS Site: www.boardose co.kr
BOS Core version m0.80,Flash [128]KB
Chip Serial No :43,16,47,69,36,30,47,30,6,6B,0,31
enter 'help' for command help
=====================================================
[BOS]
```

b. BOS Serial Number.

BOS Serial Number is unique ID for each BOS Board.
This Number is fixed, nd can't change.
It is used for certify official BOS-Board.
BOS-Board buyer must keep this Serial Number for after service.
You can see the serial No. in boot message.

```
 => Chip Serial No :43,16,47,69,36,30,47,30,6,6B,0,31
```

c. Help.

If you type 'help' at the [BOS] prompt, you can see
the available BOS commands.
For more information each commands, refer to '2. Command Interpreter'

```
------------------------------------
COMMAND              USAGE
---------------------------------------------------
ver              : Version print
help             : Help-command list
format           : BOS file system format
file             : BOS file command
del              : Delete file
down             : XMODEM[128] down [file]
bscgen           : Basic Script Generator
set              : Set internal value
setbit           : setbit $PORTX [PinNo]
resetbit         : resetbit $PORTX [PinNo]
---------------------------------------------------
```

d. Format File System.

BOS board flash have two parts.
One is BOS-Core, second is file system.
File sytem contain, script file and etc.
At first time, you need to format file system, like other OS[DOS, Linux, Windows..]
To format the file system, type 'format' at BOS prompt

```
[BOS]format

################################################################
BOS File system format complete. [ 64 file blocks ]
```

'#' mean 1 file block formated.
128K flash BOS board, 64K BOS core and 64K file system.
If 256K flash BOS board, 64K BOS core and 64K+ 128K file system.

e. Basic Script Generation.

BOS need basic command script for easy-run.
For example, 'cat' is basic command.
Like this, essential script can be auto-generated.
You can type 'bscgen'-Basic Script Generation.

```
[BOS]bscgen

File created.
File created.
File created.
File created.
```

Then 4 essential script file will be generated.
Type 'file' for view what files generated.

```
[BOS]file

led.sc                  422Bytes
cat.sc                  540Bytes
time.sc                  258Bytes
speedtest.sc               562Bytes

Total 4 file found. [61440]Bytes available.
```

Led.sc is user led on / off script.
Cat.sc is file contents view script.
Time.sc is time check script[after boot]
speedtest.sc is for script runnig speed test script.

These scripts file can be changed by BOS Core version.
And you can make new script for another use.

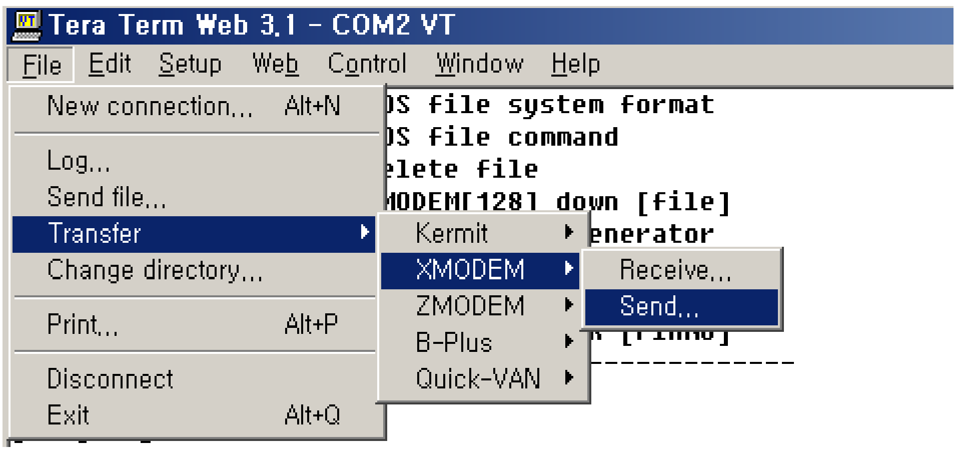
2. Command Interpreter

a. Downlaod File

BOS System doesn’t support ‘editor’ like ‘vi’ yet.
So if you want new script application, you have to write it on your PC or notebook.
And transfer it to BOS board. This called ‘download’ file.
For download script file, you need ‘X-modem’ supported hyper terminal
Download step is.
Type ‘down filename’ at the BOS-board.
Sendfile[X-Modem] form hyper-terminal.

```
[BOS]down gugudan.sc
```

… receive waiting, below example is ‘teraterm’ X-Modem send.



```
)S file system format
)S file command
elete file
MODEM[128] down [file]
enerator
```

```
Download Complete 643 bytes.

File 'gugudan.sc' created.
```

Check downloaded file, then you can see ‘gugudan.sc’
File name is determined by ‘[BOS]down gugudan.sc’ command line.

```
[BOS]file

led.sc                    422Bytes
cat.sc                    540Bytes
time.sc                    258Bytes
```

```
speedtest.sc                562Bytes
gugudan.sc                  643Bytes

Total 5 file found. [60416]Bytes available.
```

For see the gugudan.sc file contents, use ‘cat’ script.

```
[BOS]cat gugudan.sc
file gugudan.sc open ok!
file size:643
// ================================
// BOS Script Sample GuGuDan
// www.boardos.co.kr
// 구구단 및 중첩 루프 , subroutine test
// ================================

Println clock
$x =1
#loop_start
	$y=1
	#loop_start
		$y = $y+ 1
		$z = $x*$y
		// $PRINTLN x "*"  y "="  x* y 안 되는 것 처리
		PRINTLN $x "*"  $y "="  $z
		#if $y >= 9 #then
			#break
		#endif
	#loop_end
	#call sub1
	$x = $x+ 1
	#if $x >=10 #then
		#break
	#endif
#loop_end
Println clock
#quit


//=================================
// sub routine.
#function sub1
	println "------------------------"
#return
//==================================
BOS Script running end.
```

b. Delete File

BOS support fopen, fclose, fread, fwrite functions.
But file delete can't be maked by this functions.
So 'del' command is supported by BOS-core.
For delete file, type 'del' command.

```
[BOS]del gugudan.sc

 gugudan.sc file deleted.

[BOS]file

led.sc                    422Bytes
cat.sc                    540Bytes
time.sc                    258Bytes
speedtest.sc                 562Bytes

Total 4 file found. [61440]Bytes available.
```

c. File View

For file list view, type 'file'
For file contents view, type 'cat [filename]'

cat command need 'cat.sc' script file.
For more information, refer to [1.e – Basic Script Generation]

c. Script File Run

Window or DOS execution file have '.exe' extension.
BOS execution file have '.sc' extension.
Sc mean 'Script'
BOS execution file is not binary but 'ASCII' based text file.
And this file parsed and run by line.[interpreter] so it called 'script'
BOS Script file have control flow expression(#if, #loopstart..)
and functions(print,strcmp..).
For more information refer to '4.BOS Script' in this manual

To run BOS script file, just type the script file name['cat'] except the extension['.sc']
Then the BOS Command Interpreter find out it from the file system, and run.

c. Script Command Run

BOS Command Interpreter support few commands, like ‘file’, ‘del’..
But user’s want more commands.
BOS Command Interpreter support internal command and script function and expressions.
For example, to control BOS GPIO, type like this at BOS prompt.

```
[BOS]setbit $portc 0
```

Then BOS board user-led will be turn on.
To see the PORT value, type like this.

```
[BOS]println $portc
1
```

This is possible at BOS prompt.

```
[BOS]$a=$portc + 1
[BOS]$portc=$a
[BOS]println $portc
2
```

BOS Command prompt is so flexible and useful.
For more script internal value and expression, refer to [4.BOS Script]

3. File System

a. BOS File System Concept

BOS file system is very-simple.
And does not support multi-block. [this can be changed at later BOS version]
File size limit is differ BOS core board.
And different BOS core board use different BOS-core firm ware.
BOS-core firm ware version is two types.
(m) type is medium type and file size limit is 1K,(exactly 1024-32 bytes)
(h) type is high-density type and file size limit is 2K(exactly 2048-32 bytes)

b. BOS File System I/O Functions

You can make file control application by BOS script.
The best example is 'cat.sc'

BOS Script support 4 basic and general file I/O functions.
This is fopen, fclose, fread, fwrite.

1) [filehandle] = fopen [filename] [mode]
2) fclose [file handel]
3) fread [file handle] [buffer] [start pos] [size]
4) fwrite [file handle] [buffer] [start pos] [size]

It's similar 'C' language file operation functions. and easy to use.

For example open 'some.sc' in read mode
$file = fopen "some.sc" "r"

if write mode, then
$file = fopen "some.sc" "w"
write mode mean new file creation.

$file is BOS internal file handle.
And file open fail, then the value is 0.
other mean file open success.

This is file copy script example. $data is internal buffer.
For more information refer to [4. BOS Script]

```
// ================================
// BOS 'file copy ' Script
// www.boardos.co.kr
// ex> copy src.txt dest.txt
// ================================
#if $argc<3 #then
        println "Usage:copy [src filename] [dest file name] "
        #Quit
#endif

$file = fopen $argv[1] "r"
#if $file ==0 #then
        println "src file " $argv[1] " open error!"
        #Quit
#endif

println "src file " $argv[1] " open ok!"
println "src file size:"  $filesize

$s = fread $file $data 0  $filesize
fclose $file
#if $s != $filesize #then
        println "src file read error! read size :" $s
        #quit
#endif

$file = fopen $argv[2] "w"
#if $file ==0 #then
        println "dest file " $argv[2] " open error!"
        #Quit
#endif

$d = fwrite $file $data 0 $s
fclose $file
#if $s != $d #then
        println "file write error! write size :" $d
        #quit
#endif

println "file copy ok!"
#quit
```

4. BOS Script

BOS keyword prefix is '#'
BOS Variable prefix is '$'
and BOS function prefix is none.

a. #define, comments, #quit

#define is similar 'C' language #define
but it just replace 1 symbol only.

For example,

```
#define LOOP_CNT 1000

$x =0
$s  = clock
println "start time sec:" $s
#loop_start
        $x = $x + 1
        #if $x >= LOOP_CNT #then   // LOO_CNT replaced by 1000
                #break
        #endif
#loop_end
#quit
```

Script comments will be started **"//"** like 'C' language.
But, BOS does not support **"/* ~ */"** comments.

And all BOS script running end by **'#quit'** keyword.

b. #if – #then – #else –#endif

```
#if [expression] #then
     [run expression is true.]
#else
     [run expression is false]
#endif
```

the **#else** can be not used, like below, but **#then,#endif** must be used.

```
#if [expression] #then
     [run expression is true.]
```

```
#endif
```

Nested **#if-#endif** is possible, like below.

```
        #if $y<1 #then
                #if $y==0 #then
                        println "y=0"
                #endif
        #else
                #if $y<2 #then
                        println "y=1"
                #else
                        println "y=2"
                #endif
        #endif
```

c. #loopstart - #loopend, #break

#loopstart -#loopend is infinite loop,
for exit the loop, use **#break** keyword.

For example,

```
$x=0
#loop_start
        $x = $x + 1
        #if $x >= 100 #then
                #break
        #endif
#loop_end
#quit
```

Nested loop is possible, like below, and **#break** keyword just exit one **#loopend.**

```
$y=0
#loop_start
        $x=0
        #loop_start
                $x = $x + 1
                print “1”
                #if $x >= 100 #then
                        #break
                #endif
        #loop_end
        $y=$y+1
        #if $y >= 10 #then
                #break
        #endif
#loop_end
#quit
```

Above example run 1000 times **print "1"** expression.

d. calculation & internal variable.

BOS script expression can use '+', '-', '/', '*' operator.
And in compare expression '<', '>', '<=', '>=' ,'==', '!=' operator can be used.

BOS expression support only binary expression only.
This means that just 2 operand is possible.

$a = $b + $c [OK]

$a = $b + $c + $d [Error!]

this limitation can be changed at later BOS version.

BOS supported internal variable is like below.

1) 26 32-bit integer **$a ~ $z.**
2) 1 character buffer **$data** [1K in medium model, 2K in high-density model]
3) 1 file handle **$file,** and **$filesize** variable determined at fopen function calls.
4) script running argument variable. **$argc, $argv[]**
5) MCU specific GPIO port variable : **$porta~porte**
6) BOS Board Port variable : **$portt, $portm, $ports, $portm**.
   * BOS tiny core board have just **$portt,** for more information refer to BOS board spec.

d. subroutine call.

BOS script support subroutine.
Subroutine start **#function** keyword, and end by **#return** keyword.
And main routine call this subroutine by **#call** keyword.

Several subroutine can be in one script file.
But subroutine must be coded at under the main routine.
This mean, **#function** keyword can't be occur before **#quit** keyword.

This is subroutine and caller example.

```
$x=1
#loopstart
        $x=$x+1

        #call sub1

        #if $x>100 #then
                #break
        #endif
#loopend
#quit

//===================================
// sub routine.
#function sub1
        println "-------------------------"
#return
//=====================================
```

d. BOS functions.

BOS script support several useful functions.
And these functions can be used at BOS command interpreter.
BOS functions are based on 'C' language. So it similar to it.
For more information about BOS script functions, refer to **"BOS reference guide."**

user's guide describe BOS functions briefly.

print : print internal variable[$x..] and string["~~"]. without CR+LF
println: print internal variable[$x..] and string["~~"]. with CR+LF (line change)
printx :print internal variable[$x..] and string["~~"]. without CR+LF, and
Hex notation.
printx :print internal variable[$x..] and string["~~"]. with CR+LF, and
Hex notation.
strcmp: string compare, if same two string it return 0, other return 1
strncmp: string compare, if same two string it return 0, other return 1 with n-size l
imit.
atoi: change asc string number to integer.
fopen, **fclose, fread, fwrite**: refer to [3.b BOS file system I/O functions]
clock: return seconds after BOS board boot.
setbit, resetbit : GPIO control functions. just change one bit to 1 or 0.

More function will be appended at later BOS-core version.