ATCMDS application provides the Wi-Fi capability to any device having UART interface. Host CPU uses serial commands to interface and configure the Wi-Fi module. All AT or custom commands are parsed and converted to the corresponding WLAN/Network calls. This mode returns the responses in the same ASCII format as success or failure with additional response data.

ATCMDLIB enables customers to use all standard commands in the library (libstcmd.a). Custom commands can also be added and linked to the ATCMDLIB.

# ATCMDLIB Application

This section describes the application along with code snippets. The application uses ATCMDLIB APIs to achieve functionality.

## Include ATCMDLIB Header File

|  |
| --- |
| #include “atcmd\_lib.h” |

## Initialization

|  |
| --- |
| if(ATCMDLIB\_ SUCCESS == atcmd\_init()){  os\_printf("ATCMDLIB: Ready\n");  }  else {  os\_printf("ATCMDLIB: Failed\n");  } |

## Add Sample Command Function

|  |
| --- |
| int testcmd\_func(int argc , char \*argv[])  {  int i;  os\_printf("\r\nHello");  for(i = 0 ; i <argc;i++){  os\_printf("\r\nargv[%d]=%s",i,argv[i]);  }  return ATCMDLIB\_SUCCESS;  } |

## Add New Command

|  |
| --- |
| if(ATCMDLIB\_ERROR == atcmd\_add("testcmd",&testcmd\_func))  os\_printf("Failed to add command\n"); |

## Command Status

|  |
| --- |
| /\*use this enum to send command status\*/  typedef enum ATCMDLIB\_STATUS  {  ATCMDLIB\_SUCCESS =0,  ATCMDLIB\_ERROR = 1,  ATCMDLIB\_EINVAL = 2  }; |

# Running the Application

Program at\_custom\_cmd.elf *(freertos\_sdk\_x.y\examples\at\_custom\_cmd\bin)* using the Download Tool:

1. Launch the Download tool provided with InnoPhase Talaria TWO SDK.
2. In the GUI window:
   1. Boot Target: Select the appropriate EVK from the drop-down
   2. ELF Input: Load the at\_custom\_cmd.elf by clicking on Select ELF File.
   3. Programming: Prog RAM or Prog Flash as per requirement.

For more details on using the Download tool, refer to the document: UG\_Download\_Tool.pdf (path: *freertos\_sdk\_x.y/pc\_tools/Download\_Tool/doc*).

**Note**: x and y refer to the SDK release version. For example: freertos\_sdk\_2.4/doc.

# Expected Output

The previously mentioned code will add an example command testcmd. The expected output for this command will be console pint Hello followed by arguments present in command. The code also sends OK on UART as command status, upon successful execution of the command testcmd.

|  |
| --- |
| Y-BOOT 208ef13 2019-07-22 12:26:54 -0500 790da1-b-7  ROM yoda-h0-rom-16-0-gd5a8e586  FLASH:PWWWWWAE  Build $Id: git-b8e2cc1 $  Flash detected. flash.hw.uuid: 39483937-3207-00a8-0068-ffffffffffff  $App:git-c5c0b68  SDK Ver: FREERTOS\_SDK\_1.0  At Custom Command Demo App  addr e0:69:3a:00:2c:5e  domain:1-11@20before: magic1=0x0, val=0x0, magic2=0x0  Crash detection logic initialized  after: magic1=0x11223344, val=0x0, magic2=0x55667788  ATCMDLIB: Ready  cmd:<null>  Added cmd [argc=0:name=testcmd]  starting thread-sock  CMD:atHexdump of uartbuf before processing, len=3  61 74 00 | at.  Zero arguments  cmd:at:2  Ready  resp-len:9  CMD:testcmdHexdump of uartbuf before processing, len=8  74 65 73 74 63 6D 64 00 | testcmd.  Zero arguments  cmd:testcmd:7  No arguments  Hello  resp-len:9 |

Use any serial commands terminal to issue serial interface commands, like testcmd in this example, to Talaria TWO EVB.

Open minicom on a Ubuntu terminal using the command minicom -s with 115,200 baudrate, 8 bits, no flow control, and no parity once the at\_custom\_cmd.elf is loaded on to the Talaria TWO EVB.

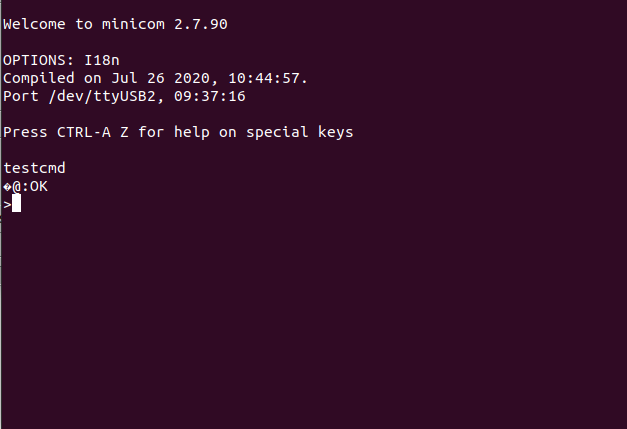


Figure : Minicom output