**Writing new command in U-boot.**

**Introduction.**

U-Boot providing interactive commands to end user but it may be necessary to add extra commands. There is a standard command interface.

U-boot comes with a lot of stock commands which one can run on the u-boot console similar to the Linux console commands like 'ls'. The source for every command can be found under 'common/' directory with file names starting with 'cmd\_'. However, not all commands are enabled by default.

A user may either want to enable a command in the stock or add his/her own command. Both are possible in just few simple steps.

**Enabling exciting commands.**

**Note**: You can check the list of commands already enabled from the commands list from the u-boot console by running the command '**help**' or simply **'?'**

**Enabling a command from the stock**

From the code, you can open 'common/Makefile' and under the '# command' section you can find the list of all the commands masked with config flags 'CONFIG\_\*'. To enable a command, you have to simply #define the corresponding flag under the 'include/configs/<board>.h' file and build the source. Now, you can see the command in the list of commands by running 'help'.

Example:

To enable a command 'source', in the 'common/Makefile', you can find

obj-$(CONFIG\_CMD\_SOURCE) += cmd\_source.o

Simply include the corresponding flag in 'include/configs/<board>.h' file as follows

#define CONFIG\_CMD\_SOURCE

or in another ways, simply replace $(CONFIG\_CMD\_SOURCE) with 'y' as follows.

obj-y += cmd\_source.o

will enable the command irrespective of any flags defined. However, the first method is most reliable and controllable from the board.h config file.

**Adding New commands.**

Adding and enabling a new command of your own.In this case, there are three things to be done-

1. Write an API(function) that implements your command.
2. Register your function to the command list.
3. Enable your command.

**1. Write an API that implements your command**

First thing is to write an API that implements your expected functionality. But, the API should follow a fixed prototype.

/\* Prototype \*/

int do\_funcname(cmd\_tbl\_t \*cmdtp, int flag, int argc, char \*const argv[])

where,

\*cmdtp - Command table pointer (comparable to function vector table)

flag - Unused

argc - Argument count, including command name itself

argv[] - Array of arguments (string)

**2.Register your API to the command table**

Registering your command to the command table is necessary for both, listed and command-line arguments to be properly passed to your API.

U-Boot provides the following prototype to register your API.

/\* Prototype \*/

U\_BOOT\_CMD(\_name, \_maxargs, \_rep, \_cmd, \_usage, \_help)

where,

|  |  |
| --- | --- |
| \_name | Command Name (the name you want to run the command) |
| \_maxargs | Maximum number of arguments |
| \_rep | Autorepeat allowed? (0-No/1-Yes) |
| \_cmd | Implementation function (pointer to your API) |
| \_usage | Usage - Short help message (string) |
| \_help | Long help message (string) |

**Note**: Both the above prototypes are available in 'include/command.h' and hence need to include 'command.h' to your command file header.

**3. Enable your command**

Finally, all set to have your own command except one last thing, enabling them. For that, you have to follow the same method as the first case as follow.

obj-y += <filename>.o

or have a new flag defined, say CONFIG\_CMD\_FILENAME and

obj-$(CONFIG\_CMD\_FILENAME) += <filename>.o

**Example.**

**1. Generate .c file in Uboot/common/**

# vim u-boot/common/my\_cmd.c

**2. Include headers and program**

* Include common.h and command.h
* Function name is do\_command\_name
* U\_BOOT\_CMD is defined in include/command.h
* U\_BOOT\_CMD( cmd\_name, max args, repeatable, function name, usage, help )

My my\_cmd.c file looks as below

------------------------------------------------------------------------------------------------------------------------

#include<common.h>

#include<command.h>

int do\_command(cmd\_tbl\_t \*cmdtp,int flag,int argc,char const\* argv[])

{

printf("Helow world.This is my own command\n");

printf("arg1: %s\n",argv[1]);

}

U\_BOOT\_CMD(my\_own\_cmd,5,1,do\_command,"Hellow World","arg1: start addr");

---------------------------------------------------------------------------------------------------------------------

**3.Modify Makefile**

In my case Uboot/common/Makefile, add below line

obj-y += my\_cmd.o

4. After this recompile u-boot again. And put 'help' command in u-boot prompt. Your command should added in that list.