

Eport&HF51 Series Product Linux SDK User Manual

This SDK is applicable for the following products. .

	Eport Pro-EP10
	Eport Pro-EP20
	HF5111A

History:

Update Time	Author	Content	Note
2017.1.18	Ke Zhang	V1.0, first Edition	
2018.2.09	Ke Zhang	V1.1, Update to 1.30 Version	
2019.5.05	Ke Zhang	V1.2, Update to 1.34 Version	

1. Install Compilation Environment

Eport Linux SDK is managed by OpenWRT and should be built under Linux environment. The recommended environment is Fedora Core 12, which is already well tested.

During Fedora system installation, it is recommended to select 'Developing' mode, as some developing tools needed by this SDK will be installed by default. You can also add those missing tools or commands during compilation phase according to the log or error raised.



SDK and Fedora(Recommend to use our OS provided) Download Address:

http://ftp.hi-flying.com:9000/EP10_EP20_SDK_Linux_Compile/

SDK: HiFlying_linux_API_140_20200514

Index of /EP10_EP20_SDK_Linux_Compile

- [Parent Directory](#)
- [Cisco TFTP Server.rar.rar](#)
- [Fedora-12-i386-DVD.iso](#)
- [HiFlying_linux_API_135_20191113.tgz](#)
- [HiFlying_linux_API_140_20200514.tgz](#)

SDK Update:

1. hiflying, replace the directory package/hiflying
2. ep10/ep20/hf5111a, replace the binary file epm/eport in directory utility/target/linux/hiflying/{ep10, ep20, hf5111a}/base-files/bin

2. Compilation

2.1 Build Process

When Linux environment is ready, extract the released SDK package, eg.

```
>cd ~  
>tar -zxvf HiFlying_linux_UART2ETH_r7112_API.tgz  
>cd HiFlying_linux_UART2ETH_r7112_API
```

Compile the source simply by 'make' command, add 'V=99' to print out detail debug information if needed during compilation.

```
>make V=99
```

After compilation, you can find target image file under ./image directory, something like: Hiflying-xxxx-firmware-squashfs.img, xxxx means the target board, currently, we support three boards, EP10, EP20 and HF5111A.

To compile a single package, go to utility directory, and do

```
>cd utility  
>make package/xxxxxxx/compile(install, clean)
```

This SDK is OpenWRT based, so the software architecture, package management, directory structure, and build process are all compatible with OpenWRT. It is convenient to visit OpenWRT website or go to community to get more information or help.

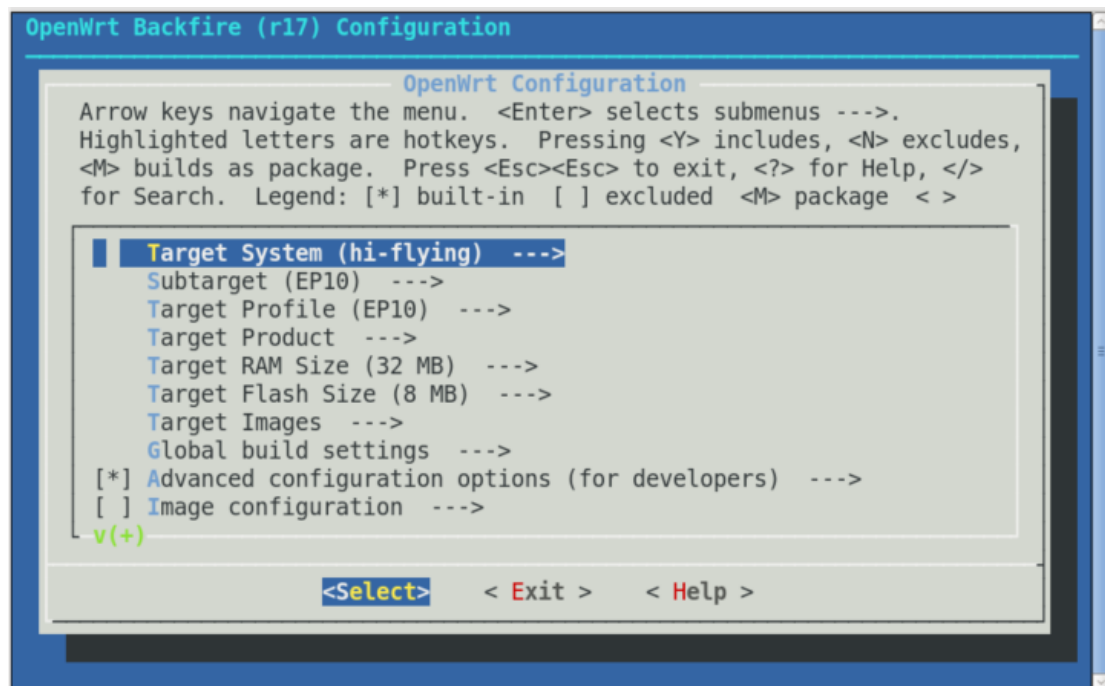
2.2 Configuration Files

There are 2 important configuration files, SDK configuration file and Linux kernel configuration file.

The default SDK configuration file is utility/.config, it is also stored in directory utility/target/linux/hiflying/xxxx/configs/default.config ('xxxx' means the target board, ie. EP10, EP20 or HF5111A). To change the configuration, just do the following:

```
>cd utility  
>make menuconfig
```

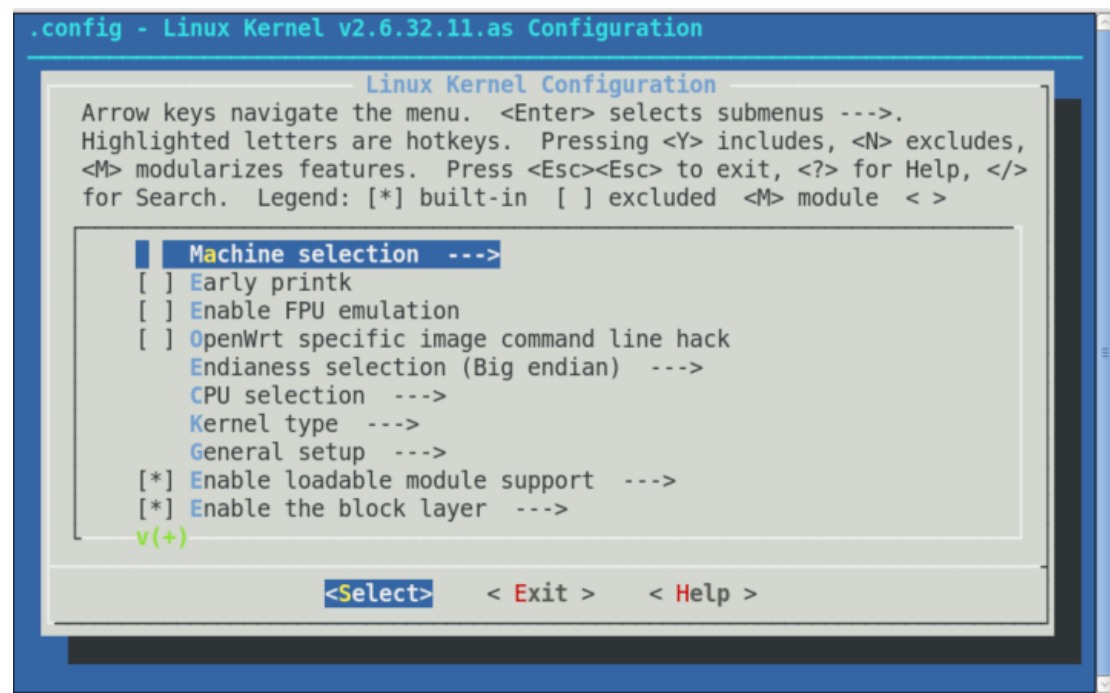
Then, custom your configuration (target selection, package add or remove...) using the menu.



The file `utility/target/linux/hiflying/xxxx/config-2.6.32-xxxxdefault` is the default kernel configuration file, where `xxxx` means board type, eg EP10, EP20 and HF5111A. It is copied to `.config` under linux kernel top lever directory during compilation, ie `kernel/linux-2.6.32.11.as/.config`. To change the configuration, do following:

```
>cd utility  
>make kernel_menuconfig
```

Then, custom your kernel configuration using the menu.



One simpler way to make target image is to run:

```
>cd utility
>make target/linux/install
```

You may find image file generated under directory utility/bin/hiflying. Note that, this should be done only when you have compiled the image at least once.

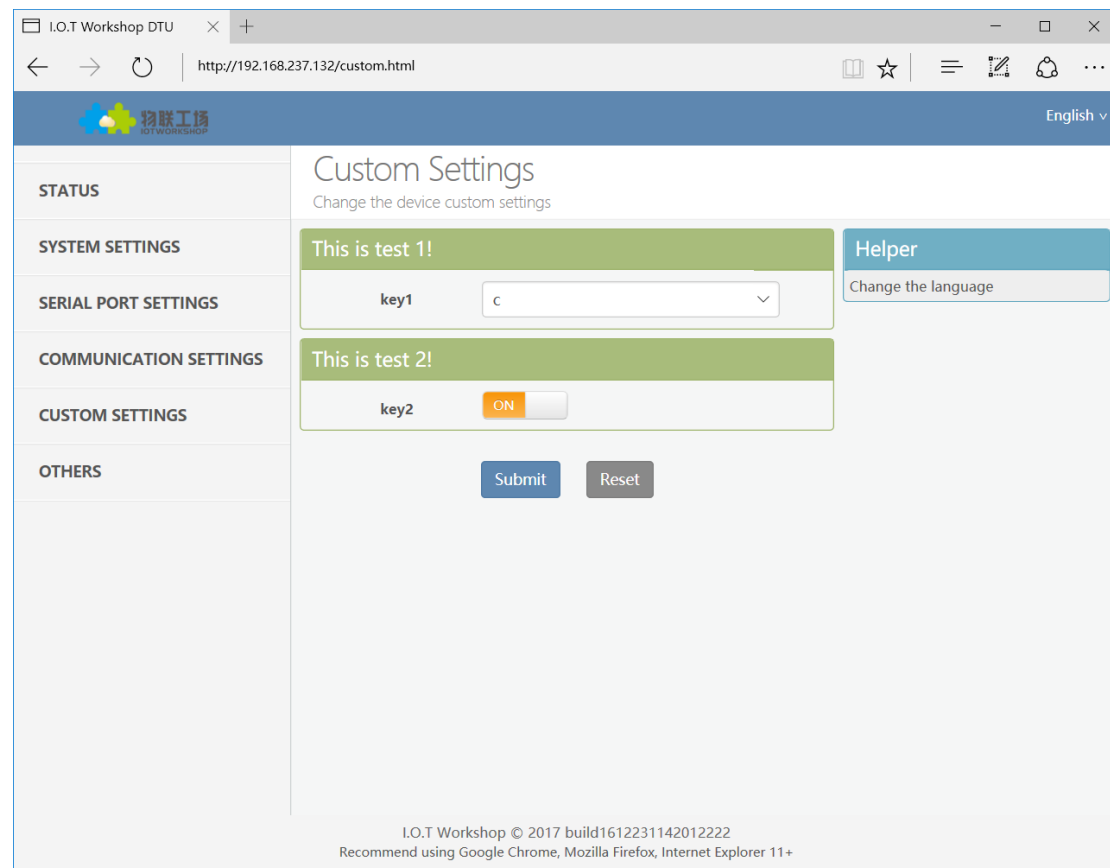
3. Work With Eport Application

In this SDK, we already integrate our Eport application and by default it starts up automatically as one of the initial service. It will perform the normal uart to ethernet, ethernet to uart functionality, cli (command line), web service, telnet, ntp... In brief, when you using the default released SDK image, it will act exactly the same as the product Hi-Flying delivered in market.

Besides all the basic features, this SDK also provides APIs to manipulate or configure these functions, to provide the SDK users to customize their own utilities based on those basic functions which are already integrated in Eport application. For example, SDK users can define their own network protocols, analysis UART data, other than

just pass through them done by Eport application.

By using the interface, SDK users can also develop custom WEB page, integrate own configuration in Eport existing WEB pages (see example below).



It is convenient for users to develop uart & ethernet applications, no need to go through everything from very beginning.

The source files to implement these are put in package 'hiflying', including some header files and two dynamic library '.so' files. For more detail, go to check the example source file 'ep_api_test.c' (under package/hiflying/src), header files (under package/hiflying/src/lib), or API documents.

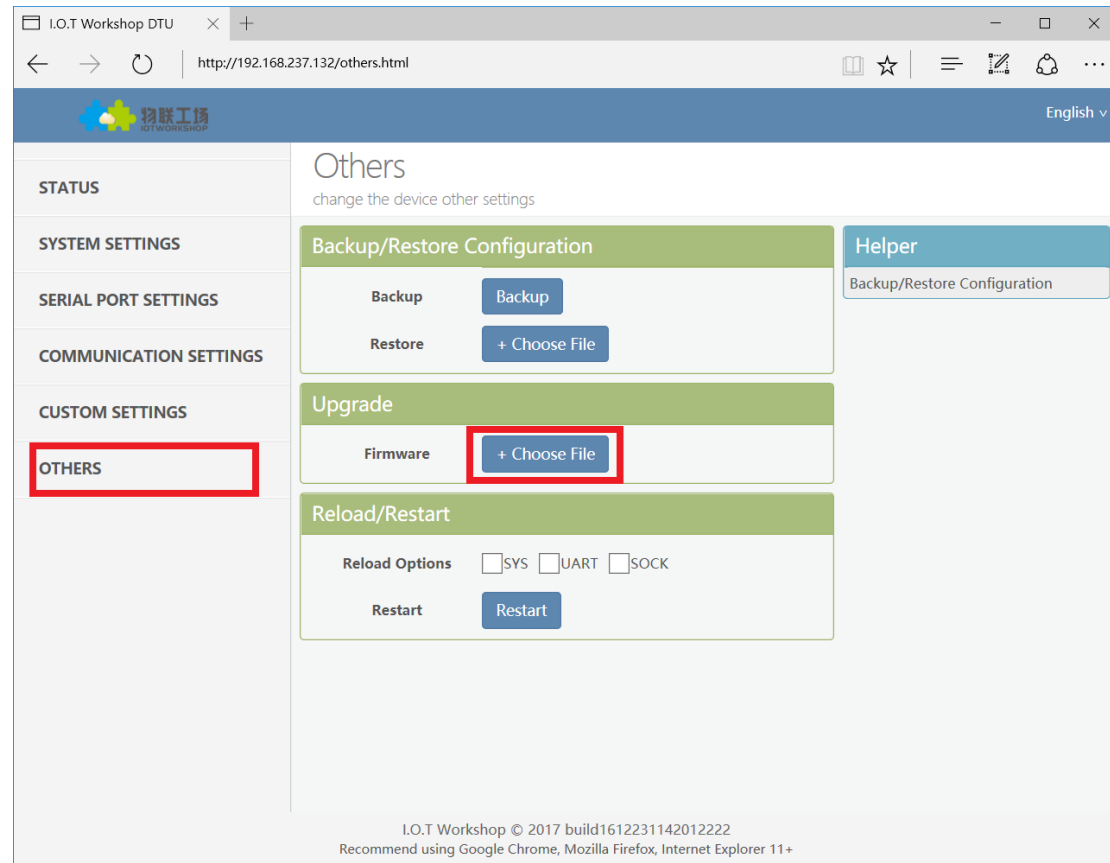
4. Image Upgrade

There are two ways to update software on the target board: web upgrade and tftp upgrade.

4.1 Web upgrade

Open the web page of the target, go to 'others' and choose the proper release file (image file). It may take about 30 seconds for the updating. After that, it will automatically go back to display the main page of the system website.

If the new firmware cause problem, then must use tftp method to upgrade.



4.2 tftp upgrade

1. PC direct connect to Module. And set fixed IP: 192.168.0.X

EP10、EP20 or HF5111A



PC1

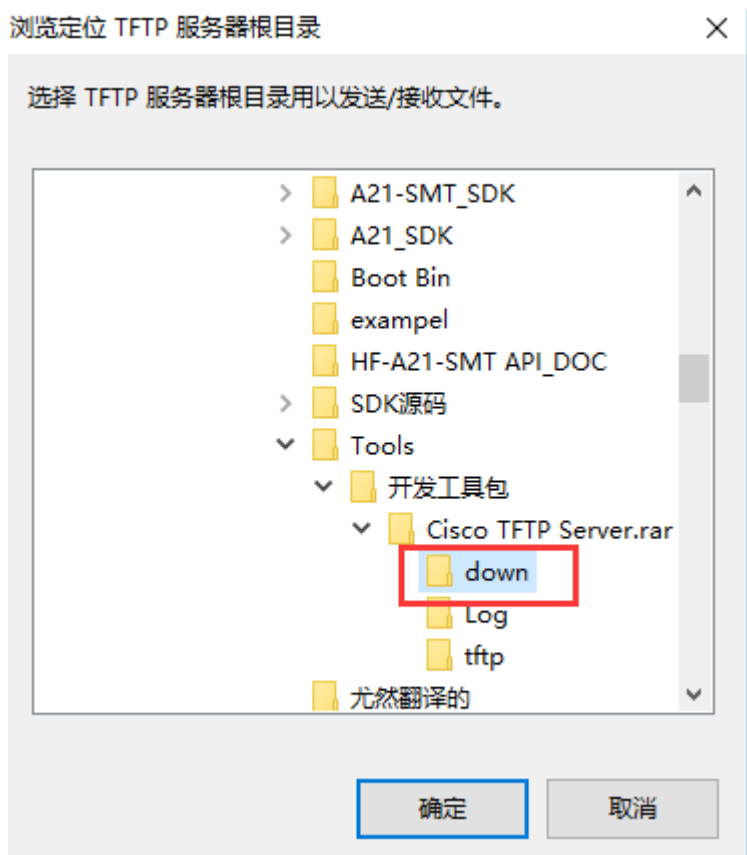
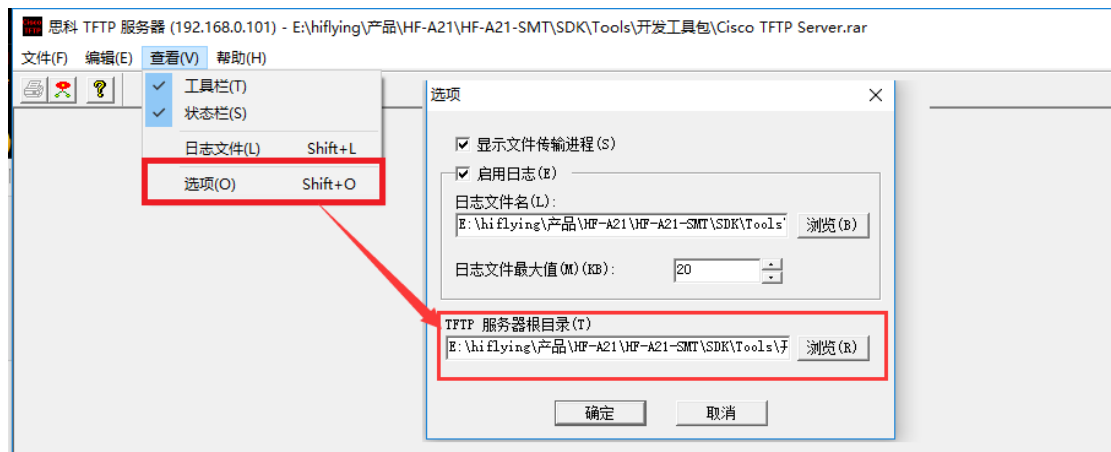
Static IP: 192.168.0.XXX



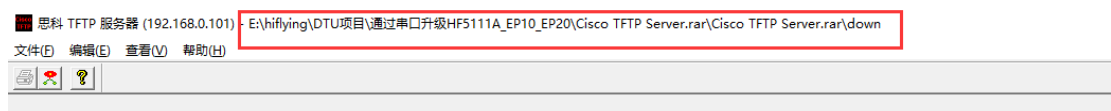
2. Put UART upgrade firmware into the TFTP "D:\Cisco TFTP Server.rar\down" directory. Such as following.



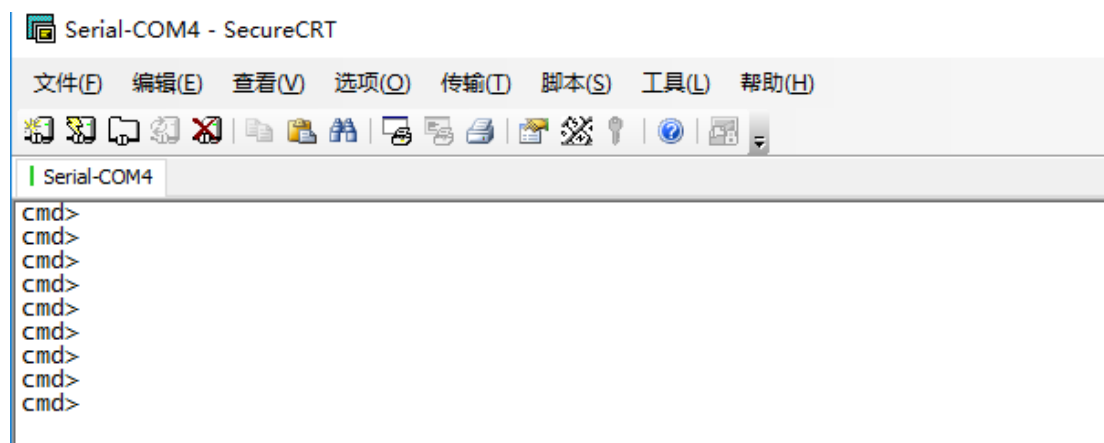
Set tftp Server directory to the above down directory.



Then run "TFTPServer.exe" .



3. Power On or Reset module and continues click keyboard "Enter" key until enter the bootloader mode as following pic. (UART baud rate 115200, 8, No flow control, 1)

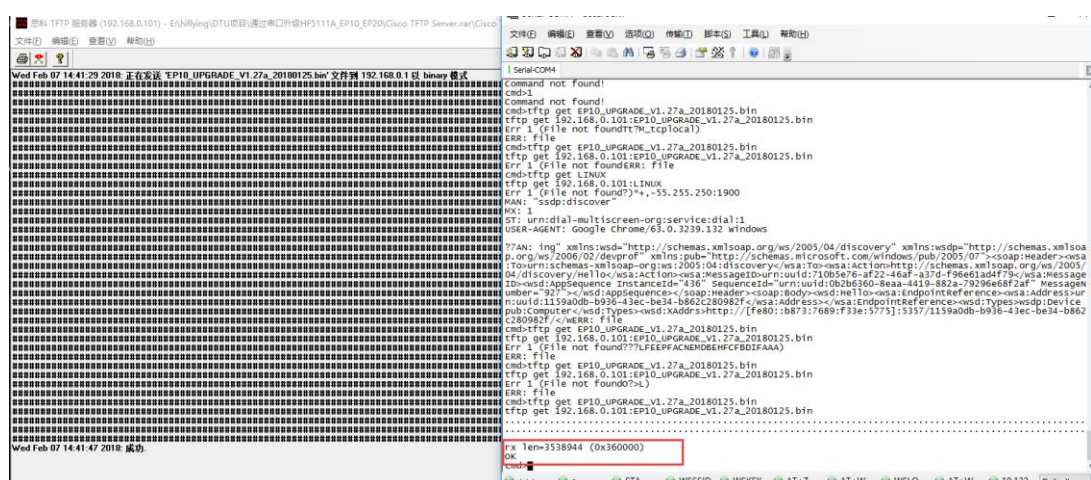


4. After the module enter bootloader mode, then set the following command :

```
cmd>set server 192.168.0.101 (PC IP address)
```

```
cmd>set server 192.168.0.101
OK
```

```
cmd>tftp get EP10 UPGRADE V1.27a 20180125.bin      (firmware name)
```



cmd>fa (Writethe download firmware into module flash)

```
cmd>fa
flash program dst=20000 len=360000 src=80500000
.....
OK
```

5. Reboot module and finish upgrade.

In original SDK, 'ep_api_test' is an test application to run manually.

You can access the device by telnet (port 2323 for Linux shell login,

user:root, password:admin) to run this application.

As this test code uses service provided by 'eport', make sure 'eport' application already there.

4.3 example

If want to add some application during boot, you need to add it in package/zrootfs

Here is an example:

1. See the attachment 'hello_world.c', it adds an application

"/usr/bin/hello_world" in boot process.

Note: as it is going to run script like following, don't add "while(1)" in source code.

```
if(f_exists("/usr/bin/hello_world"))
```

```
    exec_cmd("/usr/bin/hello_world 1>/dev/ttyS0 2>&1 & "); //start it as a
```

daemon, and print log to serial port

2. add in zrootfs/makefile

```
rm -f $(1)/etc/init.d/hello_world
```

```
$(INSTALL_BIN)
```

```
$(PKG_BUILD_DIR)/init/hello_world $(1)/etc/init.d/hello_world
```

```
ln -sf ../init.d/hello_world $(1)/etc/rc.d/S96hello_world
```

3. add in zrootfs/src/init/makefile

```
image = rcs wdk boot dbus avahi cdb telnet done sysctl hello_world
```

hello_world: \$(OBS)

@echo "Building \$@" ..."

\$(CC) -o \$@ \$@.o \$(LDFLAGS) \$(LIBS)

4. add in rcs.c

```
static rcs_process start_process[] = {
```

```
    { "S01alsa"  },
```

```
    { "S05wdk"   },
```

```
//    { "S06usb"  },
```

```
//    { "S07lcd"  },
```

```
    { "S10boot"  },
```

```
    { "S11dbus"  },
```

```
    { "S12avahi" },
```

```
//    { "S15crond" },
```

```
    { "S20cdb"   },
```

```
    { "S50telnetd" },
```

```
//    { "S94ocfg"  },
```

```
    { "S95done"  },
```

```
    { "S96hello_world" },
```

```
    { "S99sysctl" },
```