PhyFLEX-i.MX6 RDK Linux Quickstart PD14.1.0 EA2 Yocto

This Quickstart provides you with the tools and know-how to install and work with the Linux Yocto Board Support Package (BSP) for the phyFLEX-i.MX6 platform. This Quickstart shows you how to do everything from installing the appropriate tools and source, to building custom kernels, to deploying the OS, to exercising the software and hardware.

Please refer to the phyFLEX-i.MX6 Hardware Manual for specific information on board-level features such as jumper configuration, memory mapping and pin layout for the phyFLEX-i.MX6 System on Module (SOM) and baseboard. Additionally, gain access to the SOM, mapper board, and baseboard schematics for the phyFLEX-i.MX6 by registering at the following: http://phytec.com/support/registration/.

Requirements

The following system requirements are necessary to successfully complete this Quickstart. Deviations from these requirements may suffice, or may have other workarounds.

Hardware

- phyFLEX-i.MX6 System on Module (PFL-A-02)
- phyFLEX-Mapper i.MX6 (FLM-A-XL1)
- phyFLEX Baseboard (PBA-B-01)
- Serial cable (RS-232)
- Ethernet cross-over cable
- AC adapter supplying 12 VDC / min. 2 A
- LCD Display --- optional

The following are supported:- EDT ETM0700G0DH6 TTL (LCD-018-070-KAP) - default- Prime View PM070WL4 LVDS (LCD-017-070W)

- HDMI via DVI port -- optional
- TiWi WiFi module (PCM-958) --- optional

Note:

- See the latest Release Notes for up-to-date information regarding hardware revisions and display support.

Software

- A modern GNU/Linux Operating host system either natively or via a virtual machine:

- Ubuntu 12.04 LTS 64-bit recommended. Other distributions will likely work, please note that some setup information as well as OS-specific commands and paths may differ.- If using a virtual machine, VMWare Workstation, VMWare Player, and VirtualBox are all viable solutions.

- Root access to your Linux Host PC. Some commands in the Quickstart will not work if you don't have sudo access (ex. package installation, formatting SD card).
- At least 40-50GB free on target build partition.
- SD card reader operational under Linux.

- If you do not have SD card access under Linux then formatting, copying the bootloader, and mounting the root file system on an SD card will not be possible.

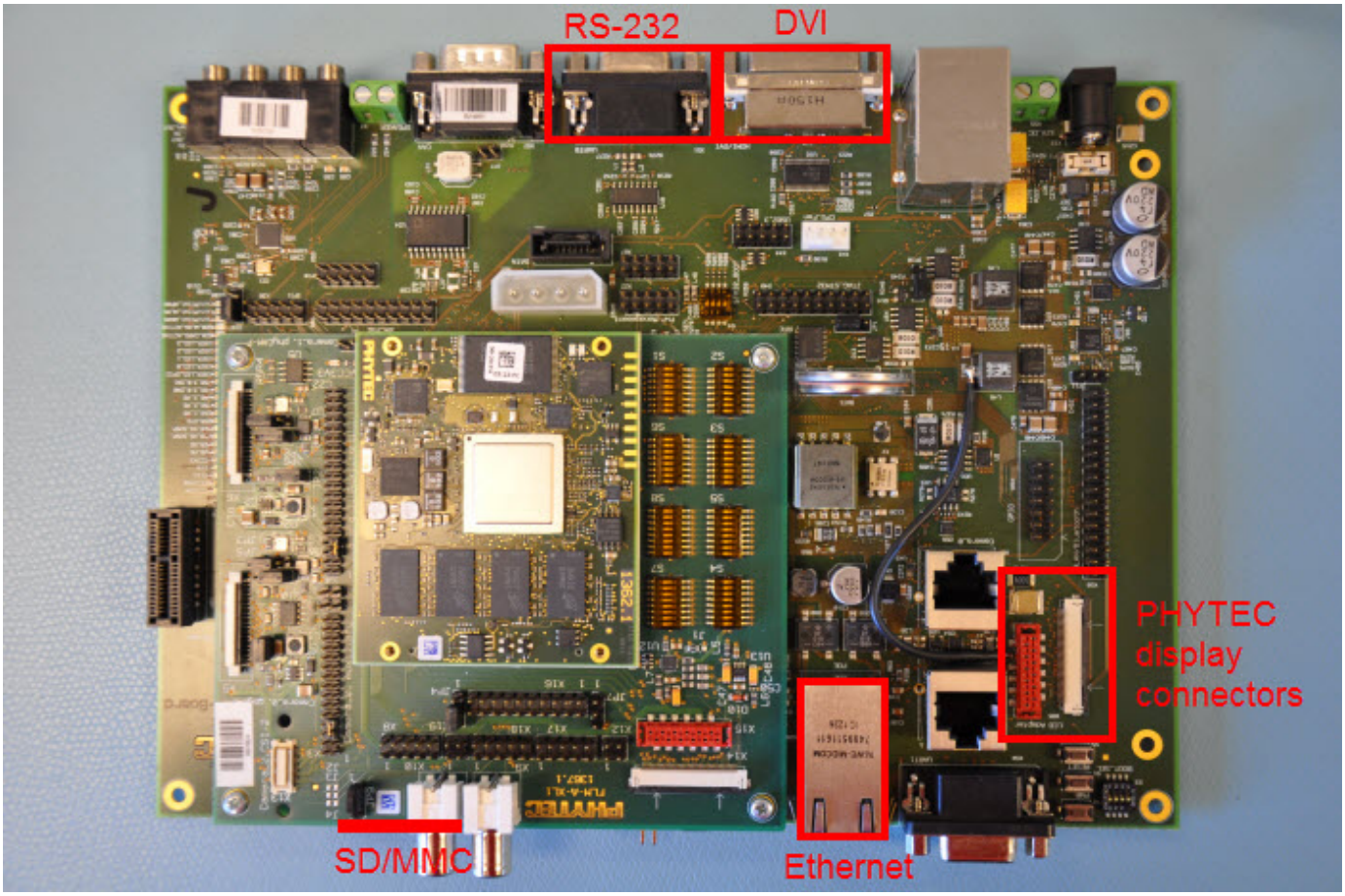
- Active Internet connection

Getting Started

This section is designed to get the board up-and-running with pre-built images.

Connector Interfaces

Use the following as a reference for the connector interfaces on the phyFLEX-i.MX6 that will be used in this Quickstart



Booting the Pre-Built Images

The section was designed to show you how to boot the phyFLEX-i.MX6 with the pre-built demo images.

1. Insert a bootable SD card into the SD0 (X57) slot on the baseboard. Since the default boot mode for the PD14.1.0_EA2 release loads the kernel and root filesystem from SD card, a bootable SD card with *zImage*, *zImage-imx6q-phytec-pbab01.dtb*, and *fsl-image-multimedia-imx6q-pbab01.tar.bz2*extracted on the SD card is required. Instructions for creating an SD card are provided in the Creating a Bootable SD card section of the Quickstart.

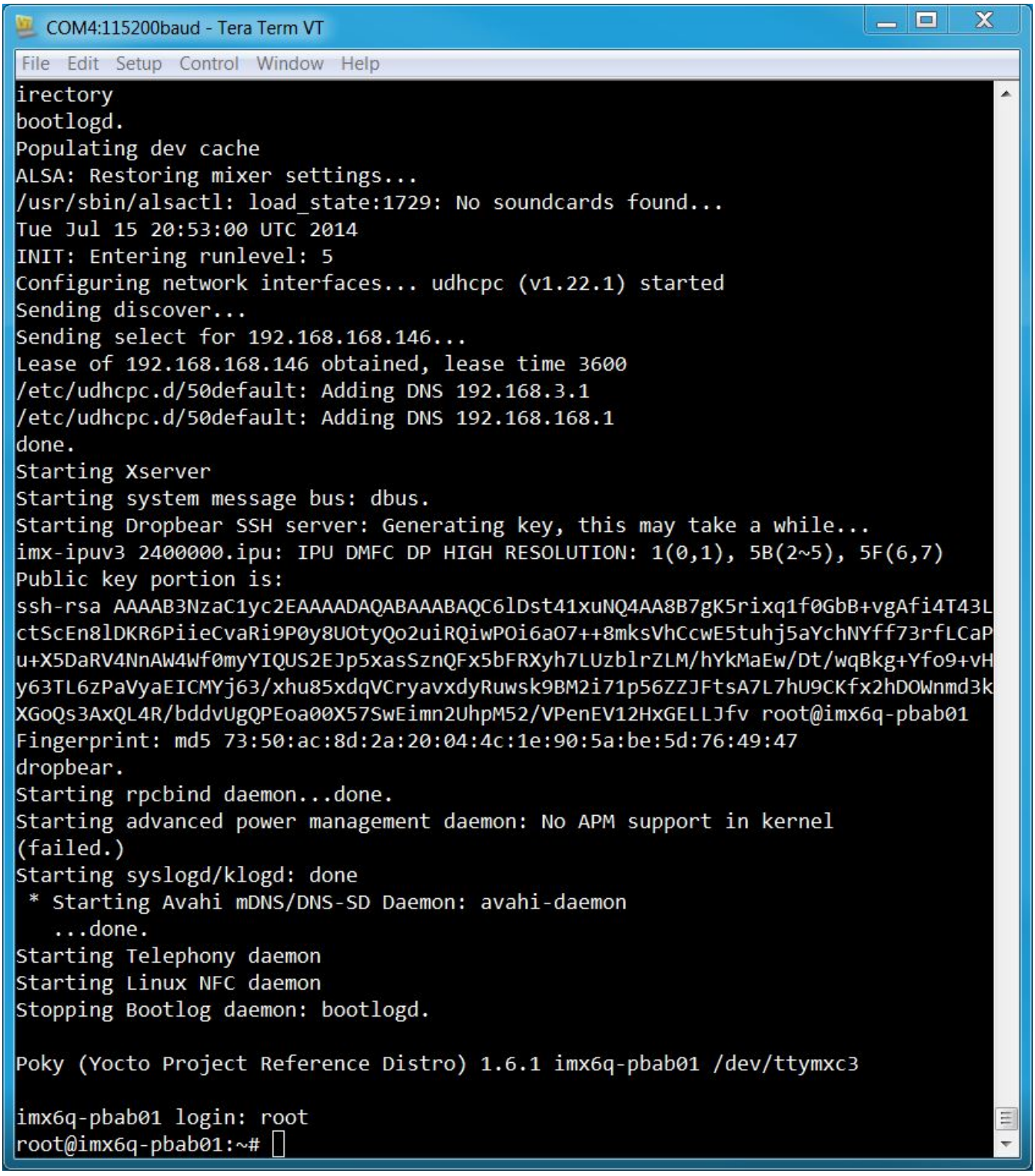
2. If you ordered a PHYTEC Display such as the LCD-018-070-KAP, plug this into the LCD power and data connectors at X65.

3. Connect the kit supplied serial cable from a free serial port on your host PC to the DB9 connector X51 on the carrier board. This is the UART0 communication channel with the i.MX6 at RS-232 levels.

4. Connect the kit supplied Ethernet cable from the Ethernet connector X28 on the carrier board to your network hub, router, or switch. If you do not have an Ethernet connection you can postpone this step, Linux will boot without the need for Ethernet connectivity but having the connection will significantly reduce your boot time.

5. Start your favorite terminal software (such as Minicom or TeraTerm) on your host PC and configure it for 115200 baud, 8 data bits, no parity, and 1 stop bit (8n1) with no handshake.

6. Plug the kit supplied 12 V power adapter into the power connector X12 on the carrier board. You will instantly see power LEDs VCC12IN, VCC12, VCC5_X, VCC5, and VCC3V3 on the carrier board as well as D1 on the SOM light up solid green. You will also start to see console output on your terminal window. If everything was done correctly the board should boot completely into Linux, arriving at a *root@imx6q-pbab01* prompt. The default login account is *root* with an empty password. Note that the first time the board is booted it will takes a little while for the SSH server to generate new keys. Subsequent boots should be faster.



Troubleshooting:

Not seeing any output on the console?

- Check that you have setup the terminal software correctly per step 5.
- Flash the release images from the PHYTEC FTP by creating a bootable SD/MMC card (Section 5), booting from SD/MMC (Section 6.1), and flashing the bootloader (Section 7).

- When finished with the kit demo, from the Linux command line use the **reboot** command to restart, or the **poweroff** command to shutdown the system. It is safe to remove power from the kit when *Power down.* is printed on the console.

Development Host Setup

The following notes assume a Ubuntu 12.04 64-bit build machine.

Packages

Yocto development requires certain packages to be installed. Run the following commands to ensure you have the packages installed:

```
sudo apt-get install sed wget cvs subversion git-core coreutils \
unzip texi2html texinfo gcc-multilib libsdl1.2-dev docbook-utils gawk \
python-pysqlite2 diffstat help2man make gcc build-essential \
g++ desktop-file-utils chrpath libgl1-mesa-dev libglu1-mesa-dev \
mercurial autoconf automake groff libtool xterm curl
```

Note:

The above is the recommended package installation for phyFLEX-i.MX6 development on a Ubuntu 12.04 LTS Linux distribution. For a breakdown of the packages as well as a list of packages required for other Linux distributions, see the "Required Packages for the Host Development System" section in the Yocto Project Reference Manual: http://www.yoctoproject.org/docs/1.6/ref-manual/ref-manual.html#required-packages-for-the-host-development-system

Verify that the preferred shell for your Host PC is *bash* and not *dash*:

```
sudo dpkg-reconfigure dash
# Respond "No" to the prompt asking "Install dash as /bin/sh?"
```

Download and install the **repo** tool. This tool is used to obtain Yocto source from Git.

```
cd /opt
sudo mkdir bin

# /opt/ directory has root permission, change the permissions so your user account can access this folder. In the
following replace <user> with your specific username
sudo chown –R <user>: bin

cd bin
curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ./repo
#add directory that contains repo to your path
chmod a+x repo
```

Add the *repo* directory in your PATH, using **export** from the command line or permanently by including it in *.bashrc*:

```
PATH=/opt/bin/:$PATH
```

Server Setup

The following steps describe the setup for TFTP, NFS, and Samba servers. Server setup is not required for working with the phyFLEX-i.MX6, however they will significantly reduce time and are highly recommended during the building and development phase.

TFTP

TFTP is a "trivial" file transfer protocol used to transfer individual files across a network. Setting up a TFTP server on your Linux Host PC will allow you to exchange files with the target board. Some examples where this will be advantageous include:

- Modifying and doing development on the Linux kernel. Barebox can be configured to remotely boot the kernel so you have access to the latest build without needing to continually reflash the target board.
- Updating images from the bootloader. Transferring files over a network in Barebox is an alternative to using an SD card which eliminates some time consuming steps such as formatting an SD card.
- Individual file transfer to the root fileystem. When Linux has been fully booted you may want to copy a specific file from your Host PC to the target board (images, application executables).

Install the TFTP server on your Host PC:

```
sudo apt-get install tftpd-hpa
```

Specify a folder where the files will reside on your Host PC by replacing the folder path for *TFTP_DIRECTORY* with whatever folder you wish to use as your TFTP file storage location, or leave the folder as the default.

```
sudo gedit /etc/default/tftpd-hpa

# /etc/default/tftpd-hpa

TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/var/lib/tftpboot"
TFTP_ADDRESS="0.0.0.0:69"
TFTP_OPTIONS="--secure"
```

If you made any changes to the settings of the TFTP server, you need to restart it for them to take effect.

```
sudo restart tftpd-hpa
```

If you would like to grant every user on the system permission to place files in the TFTP directory, use the following command, replacing <*TFTP_DIRECTORY*> with your chosen location.

```
sudo chmod ugo+rwx <TFTP_DIRECTORY>
```

Files in the <*TFTP_DIRECTORY*> on your Host PC can now be accessed from another machine on the same network such as the target board by simply using the IP address of the Host PC. Take note of this IP address, in a typical wired connection this will be *inet addr* listed under *eth0*.

```
ifconfig
```

NFS

A network filesystem (NFS) server gives other systems the ability to mount a filesystem stored on the Host PC and exported over the network. Setting up an NFS server on your Linux Host PC gives you access to the target boards root filesystem which will allow you to quickly test applications and evaluate different filesystem setups for the target board. That is, the root filesystem for the phyFLEX-i.MX6 SOM will actually be located on the remote host Linux machine. This enables easy access and modifications to the root filesystem during development.

Install the NFS server on your Host PC:

```
sudo apt-get install nfs-kernel-server
```

Exported filesystems are designated in the "/etc/exports" file and allow you to choose both the directory to be exported and many settings for accessing the exports. Below is an example for exporting a folder called "nfs_export-ex" located in a user's home directory.

```
sudo gedit /etc/exports

# /etc/exports

/home/<user>/nfs_export-ex  *(rw,sync,no_root_squash,no_subtree_check)
```

The options (rw, sync, no_root_squash, no_subtree_check) for this folder are essential in setting up the NFS export correctly. For more information on additional options, refer to the man page for 'exports'.

- rw enables

read and write access when the directory is mounted

- sync

tells the file-system to handle local access calls before remote access

- no_root_squash

allows root access when mounting the file-system

- no_subtree_check

reduces the number of checks the server must make to ensure that an exported sub-directory is within an exported tree and also enables access to root files in conjunction with no_root_squash

After modifying this file, in order to mount the directories as an NFS, you must force the NFS server to export all of the directories listed in "/etc/exports".

```
sudo /usr/sbin/exportfs -va
```

Samba

Samba servers are an excellent way to access a Linux file-system on a Windows machine via a network connection. Using a Samba server, it is quick and easy to transfer files between systems.

To install a Samba server, use the following command:

```
sudo apt-get install samba
```

Before the Samba share can be mounted on another machine it's necessary to modify the configuration file to allow write access and access to home directories. Start by editing the "/etc/samba/smb.conf" file.

```
sudo gedit /etc/samba/smb.conf
```

Inside this file there are four specific things that need to be uncommented (remove the ';' at the beginning of the line) to enable the sharing of home folders and write access. Below is the section that must be modified:

```
#======================= Share Definitions =======================

# Un-comment the following (and tweak the other settings below to suit)
# to enable the default home directory shares. This will share each
# user's home directory as \\server\username
;[homes]
;   comment = Home Directories
;   browseable = yes

# By default, the home directories are exported read-only. Change the
# next parameter to 'no' if you want to be able to write to them.
;   read only = no
```

The outcomes after the changes are made follow:

```
#======================= Share Definitions =======================

# Un-comment the following (and tweak the other settings below to suit)
# to enable the default home directory shares. This will share each
# user's home directory as \\server\username
[homes]
   comment = Home Directories
   browseable = yes

# By default, the home directories are exported read-only. Change the
# next parameter to 'no' if you want to be able to write to them.
   read only = no
```

Note:

It might also be necessary to change the *workgroup* line to match the workgroup for your machine.

To apply the changes, the next step is to restart all Samba-related processes.

```
sudo restart smbd
sudo restart nmbd
```

Lastly, each user needs to have a password enabled to be able to use the Samba server. There are no rules for this password. The simplest method for choosing this password is to make it the same as the UNIX user's password, but it is not a requirement. After typing in the command below, you will be prompted to enter the password for the specified user.

```
sudo smbpasswd -a <user>
```

As mentioned in the configuration file, the samba share can be connected by accessing "\\<host machine ip>\\<user>" by either mounting a network share or using Windows explorer to navigate to it.

Building the BSP

Create a directory which will house your BSP development. In this example the BSP directory is */opt/PHYTEC_BSPs/*. This is not a requirement and if another location is preferred (ex. *~/PHYTEC_BSPs*) feel free to modify. We recommend using */opt* over your *HOME* directory to avoid errors attributed to ~ syntax as well as the **sudo** requirement for the root filesystem and automation package building. We also recommend creating a package download directory (*yocto_dl*) separate from the yocto tree (*yocto_fsl*), as it makes resetting the build environment easier and subsequent build times much faster.

```
sudo mkdir /opt/PHYTEC_BSPs
cd /opt/

# /opt/ directory has root permission, change the permissions so your user account can access this folder. In the
following replace <user> with your specific username
sudo chown –R <user>: PHYTEC_BSPs

cd PHYTEC_BSPs
mkdir yocto_fsl
mkdir yocto_dl
cd yocto_fsl
export YOCTO_DIR=`pwd`
```

At this point you will now be able to navigate to the Yocto directory using the *$YOCTO_DIR* environment variable.

- Install the Freescale Community BSP:

```
cd $YOCTO_DIR
repo init -u https://github.com/Freescale/fsl-community-bsp-platform -b daisy
```

- Update the *$YOCTO_DIR/.repo/manifest.xml* file by changing revisions for the fsl repositories and adding the PHYTEC git entries:

```
<project remote="yocto" revision="ba379597bf2c70e509758855733f64bef1be9326" name="meta-fsl-arm" path="sources
/meta-fsl-arm"/>
<project remote="freescale" revision="f432f685febb39dfb481cdecd77f9b2fa67cd30d" name="meta-fsl-arm-extra" path="
sources/meta-fsl-arm-extra"/>

<remote fetch="git://git.phytec.com/" name="phytec"/>
<project remote="phytec" name="meta-phytec" path="sources/meta-phytec" revision="refs/tags/phyFLEX-i.MX6-PD14.1.0
_ea2"/>
```

- Download the Yocto layers:

```
cd $YOCTO_DIR
repo sync
```

- Update the *$YOCTO_DIR/sources/base/conf/bblayers.conf* file to include *meta-phytec*:

```
BBLAYERS += " ${BSPDIR}/sources/meta-phytec "
```

- Run the Yocto build directory setup:

```
cd $YOCTO_DIR
MACHINE=imx6q-pbab01 source setup-environment build
```

- Add the new download directory to *$YOCTO_DIR/build/conf/local.conf* and well as some additional modifications:

```
DISTRO_FEATURES_remove = " wayland x11 "
DISTRO_FEATURES_append = " directfb "

# for autoinc
PRSERV_HOST = "localhost:0"

DL_DIR ?= "/opt/PHYTEC_BSPs/yocto_dl"
```

The setup is complete and you now have everything to complete a build. The following will start a build from scratch including installation of the toolchain as well as barebox, Linux, and root filesystem images.

```
cd $YOCTO_DIR/build
MACHINE=imx6q-pbab01 bitbake --continue core-image-directfb
```

Note:

If the package fetch fails, you will need to manually download the linuxptp IEEE 1588 stack to $DL_DIR (http://sourceforge.net/projects/linuxptp/files/). Once you've downloaded the archive (e.g. linuxptp-1.3.tgz), you will need to create a "done" file so fetch doesn't delete the file and try to refetch it:

```
cd yocto_dl
touch linuxptp-1.3.tgz.done
```

Another issue that may happen is ncurses5.9 won't be built in time for a different package in the build (says libpanelw.so is missing). Forcing the reinstall of the base ncurses package unsticks this:

```
bitbake ncurses -f -c install
```

All images generated by bitbake are deployed to *$YOCTO_DIR/build/tmp/deploy/images/imx6q-pbab01*:

- **Barebox**: barebox-imx6q-pbab01.imx
- **Kernel**: zImage
- **Kernel device tree file**: zImage-imx6q-phytec-pbab01.dtb
- **Root Filesystem**: fsl-image-multimedia-imx6q-pbab01.tar.bz2
- **Kernel modules**: Useful if building a filesystem outside of the Yocto environment modules - modules-imx6q-pbab01.tgz

Build Time Optimizations

The build time will vary depending on the package selection and Host performance. Beyond the initial build, after making modifications to the BSP, a full build is not required. Use the following as a reference to take advantage of optimized build options and reduce the build time.

To rebuild Barebox:

```
bitbake barebox -f -c compile && bitbake barebox
```

To rebuild the Linux kernel:

```
bitbake linux-fslc -f -c compile && bitbake -f linux-fslc
```

Create a Bootable SD card

The process requires an SD card reader operational under Linux to format and access the Linux partition of the card. If you do not have SD card access under Linux then copying the bootloader and mounting the root filesystem on SD/MMC card will not be possible.

If using a SD/MMC card that has already been formatted skip to the appropriate steps below -- Step 5 if the Linux kernel is changing or Step 7 if the root filesystem is the only change and continue the steps from there.

1. Determine the SD card device name

- The SD card device name is of the form /dev/sd[b|c|d|e]. Run the following without and with the SD card connected:

```
ls /dev/sd*
```

- The device that appeared on the call to ls with the SD card but not the call without is the SD card device.

2. Unmount all partitions of the SD card, using the SD card device name from Step 1:

```
umount /dev/sd[b|c|d|e]*
```

3. Make two partitions on the SD card using *fdisk* from the Linux kernel, specifying the SD card device name from Step 1:

```
sudo fdisk /dev/[b|c|d|e]

Command (m for help): o
Building a new DOS disklabel with disk identifier 0x2fe3ef94.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.
```

- Create a new primary partition (n command) with partition id C, leaving 8 MB of free space at the beginning of the card:

```
Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-7626751, default 2048): 17432
Last sector, +sectors or +size{K,M,G} (17432-7626751, default 7626751): 1267432

Command (m for help): t
Partition number (1-4): 1
Hex code (type L to list codes): C
Changed system type of partition 1 to c (W95 FAT32 (LBA))
```

- Create a new Linux partition (n command) with partition id 83

```
Command (m for help): n
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p): p
Partition number (1-4, default 2): 2
First sector (2048-7626751, default 2048): 1267433
Last sector, +sectors or +size{K,M,G} (1267433-7626751, default 7626751):
Using default value 7626751

Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): 83
```

- Write the partition table to the card (**w** command) which will destroy all data on the SD card

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
```

- Create a filesystem on the partitions from the Linux command line:

```
sudo mkfs.vfat /dev/sd[b|c|d|e]1 -F 32 -n boot
sudo mkfs.ext3 /dev/sd[b|c|d|e]2 -L rootfs
```

4. Mount all partitions

- Remove and reinsert the SD card, automount will mount */media/boot* and */media/rootfs*

Kernel

5. If modifying the Linux kernel, remove the existing zImage and zImage-imx6q-phytec-pbab01.dtb files:

```
rm -rf /media/boot/*
```

6. Load the new Linux kernel and device tree binary to the SD Card:

```
cp zImage /media/boot; sync
cp zImage-imx6q-phytec-pbab01.dtb /media/boot; sync
```

Root Filesystem

7. If modifying the root filesystem, remove the existing:

```
sudo rm -rf /media/rootfs/*
```

8. Load the new filesystem to the SD Card:

```
sudo tar -jxf fsl-image-multimedia-imx6q-pbab01.tar.bz2 -C /media/rootfs/; sync
```

9. Unmount each partition before copying the bootloader or removing the SD Card:

```
umount /media/boot /media/rootfs
```

Bootloader

10. Use the dd command to copy barebox-image to the SD Card:

```
dd if= barebox-imx6q-pbab01.imx  of=/dev/sd[b|c|d|e] bs=512 skip=2 seek=2
```

11. Use the dd command to copy the barebox default environment to the SD Card:

```
dd if=barebox_default_env of=/dev/sd[b|c|d|e] bs=512 seek=4096
```

Additionally, if you plan on using the SD/MMC card to flash images, copy the files to the */boot* partition. Since the */boot* partition is FAT32 file copying can be handled using a Windows machine and a Linux machine is not required.

Boot Configurations

The default boot mode for the PD14.1.0_EA2 Yocto BSP is the following:

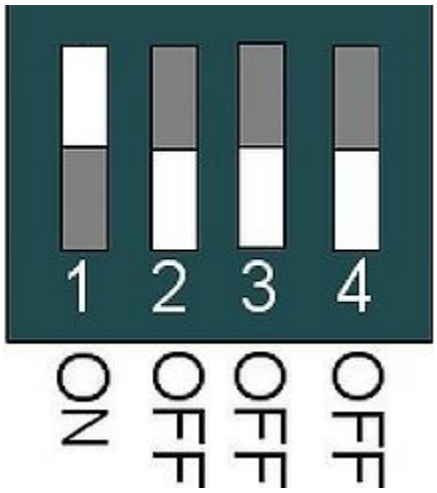
- **Barebox**: NOR flash
- **Kernel and device tree file**: SD card
- **Root Filesystem**: SD card

Selecting Boot Modes

The bootloader, one of the key software components included in the BSP, completes the required hardware initializations to download and run operating system images. The boot mode, selected from the S3 dipswitch on the Carrier Board, determines the location of the primary bootloader. Set the S3 switches correspondingly:

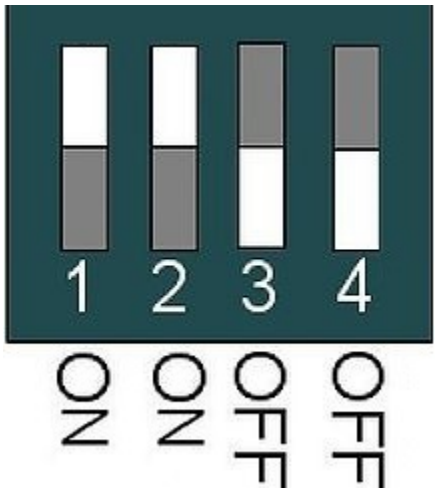
SPI NOR (default)

S3-1 ONS3-2, S3-3, S3-4 OFF



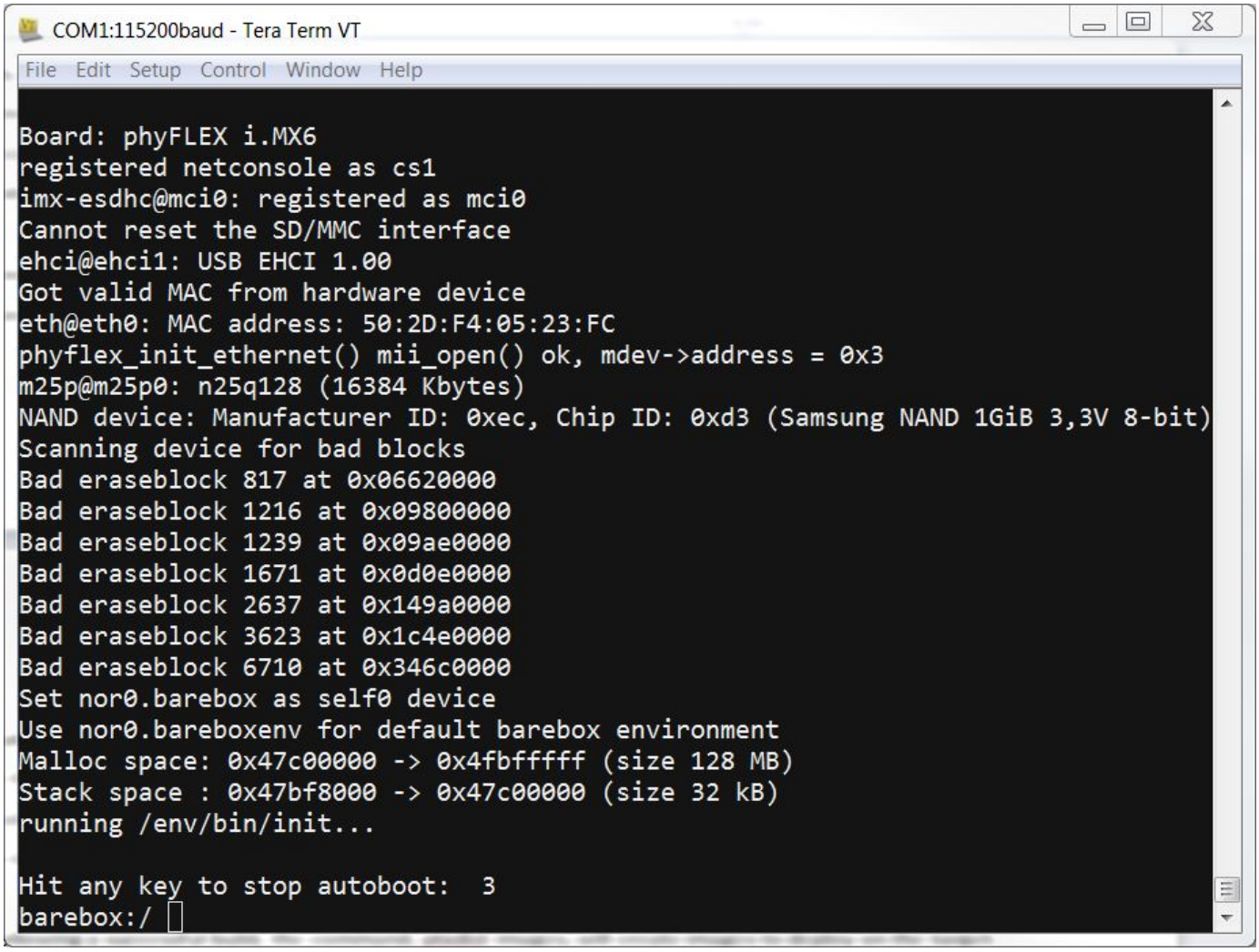
SD Card

S3-1, S3-2 ONS3-3, S3-4 OFF



Basic Settings

After application of power, approximately three seconds are allotted for the user to hit any key which will halt autoboot and enter Barebox.



Note:

help is a useful tool in Barebox to show available commands and usage.

Use the Barebox configuration files to modify environment variables, select boot settings, and setup networking parameters.

Required Settings

The following changes to the Barebox environment are required to work with the Yocto images:

- Specify the default boot option to be from SD card:

```
edit env/init/general

#Modify global.boot.default to use sd-ext3:
global.boot.default=sd-ext3
```

- Update the SD boot script to reflect the new kernel format (zImage and device tree binary):

```
edit env/boot/sd-ext3

#Modify the following lines
global.bootm.image="/mnt/mmc/zImage"
global.bootm.oftree="/mnt/mmc/zImage-imx6q-phytec-pbab01.dtb"
```

Network Settings

You can check the targets network settings by running the following:

```
devinfo eth0
```

The *ethaddr* variable is the MAC id of the target. This is a pre-programmed value which is read from the EEPROM and matches the sticker on the SOM. To modify any of the network settings, type:

```
edit /env/network/eth0
```

You should see something similar to the following, modify the variables to specify your network configuration for ETH0:

```
ipaddr=###.###.###.###
netmask=###.###.###.###
gateway=###.###.###.###
serverip=###.###.###.###
```

- **ipaddr**

A dedicated IP address for the SOM. This is crucial if TFTP will be used for updating the device's images at any point.

- **netmask**

Netmask for the network: typically 255.255.255.0. This is only necessary if the TFTP directory is located on another network.

- **gateway**

Gateway IP for the network. This is only necessary if the TFTP directory is located on another network.

- **serverip**

IP address of the host or another machine. serverip corresponds to where the TFTP directory, if it exists, is located.

Display Settings

To use the PHYTEC display connector as your primary output, add *LVDS-1* to your bootargs in Barebox:

```
edit env/bin/bootargs-display

# change
# global.linux.bootargs.dyn.display="video=mxcfb0:$display video=mxcfb1:$hdmi_res"global.linux.bootargs.dyn.
display="video=mxcfb0:$display video=mxcfb1:$hdmi_res"
# to the following:
global.linux.bootargs.dyn.display="video=LVDS-1"
```

Note:

- In the PD14.1.0_EA2 Yocto BSP release, LCD-018-070-KAP (**default**) and LCD-017-070W are currently the only panels added to the ldb.c file. The remainder will be added on the next release.

Saving Configurations

From any of the Barebox scripts, return to the Barebox prompt by pressing **CTL+D** to apply changes or **CTL+C** to cancel. To retain changes, at the Barebox prompt save the environment.

```
saveenv
```

Boot Options

The target can be booted from on-board media or from a development host via network. In our standard configuration, the PD14.1.0_EA2 BSP release loads the kernel and root filesystem from SD card. This process requires a properly formatted SD card which can be prepared using the instructions in the Creating a Bootable SD card section of the Quickstart.

The other method is to provide needed components via network. In this case the development host is connected to the phyFLEX-i.MX6 with a serial cable and via Ethernet; the embedded board boots into the bootloader, then issues a TFTP request on the network and boots the kernel from the TFTP server on the host. Then, after decompressing the kernel into RAM and starting it, the kernel mounts its root filesystem via the NFS server on the host. This method is especially useful for development purposes as it provides a quick turnaround while testing the kernel and root filesystem.

Stand-Alone NAND Boot

To use the target stand-alone, the kernel and root filesystem have to be made persistent in the on-board media of the phyFLEX-i.MX6. Due to UBIFS format issues, the root filesystem can not be flashed to NAND and the feature is currently not supported in the PD14.1.0_EA2 Yocto BSP. This support is expected to be added in the next BSP release.

Remote Boot

The network-remote boot variant is intended to be used during development because of the frequent need to rebuild the Linux kernel and root filesystem. Reflashing the newest kernel and root file system to the SOM after every new build would be very cumbersome and time consuming, by using TFTP and NFS you can accelerate the development process. All that is needed is an Ethernet connection and a network aware bootloader which can fetch the kernel from a TFTP server.

Restart the board and stop autoboot by pressing *m*. You'll get a menu:

```
Welcome to Barebox
      1: Boot: Kernel:nand;rootfs:nand
      2: Boot: network (Kernel:tftp;rootf:nfs)
      3: Boot: MMC (ext3)
      4: Settings
      5: Shell
      6: Reset
```

From this menu you can select the interface in which to boot from. Press *2* and then the enter key.

The net boot entry can also be run from the Barebox shell:

```
boot net
```

Stand-Alone SD/MMC card Boot

The SD/MMC card boot variant is an alternative stand-alone boot option. All that is needed is a properly formatted (see the Creating a Bootable SD card se ction of the Quickstart) SD/MMC card

Restart the board and stop autoboot by pressing *m*. You'll get a menu:

```
Welcome to Barebox
      1: Boot: Kernel:nand;rootfs:nand
      2: Boot: network (Kernel:tftp;rootf:nfs)
      3: Boot: MMC (ext3)
      4: Settings
      5: Shell
      6: Reset
```

From this menu you can select the interface in which to boot from. Press *3* and then the enter key.

The SD/MMC card boot entry can also be run from the Barebox shell:

```
boot sd-ext3
```

Custom Boot

You may have custom boot requirements that are not covered by the three available boot files (nand, net, sd-ext3). If this is the case you can create your own custom boot entry specifying the kernel and root filesystem location.

- In Barebox, create your own boot entry, for example named *custom*:

```
edit /env/boot/custom
```

- Use the following template to specify the location of the Linux kernel and root filesystem. Please note that the text in <> such as <*kernel_loc*>, <*rootfs_loc*>, <*kernel_loc_bootm.image*>, <*rootfs_loc_dyn.root*>, and <*nfs_root_path*> are intended to be replaced with user specified values.

```
#!/bin/sh

if [ "$1" = menu ]; then
        boot-menu-add-entry "$0" "Kernel:<kernel_loc>;rootfs:<rootfs_loc>"
        exit
fi

global.bootm.image="<kernel_loc_bootm.image>"
#global.bootm.oftree="/env/oftree"
global.bootm.oftree="none"
<div style="color:red"><nfs_root_path></div>
bootargs-ip
bootargs-cam
bootargs-display
global.linux.bootargs.dyn.root="<rootfs_loc_dyn.root>"
```

- **<kernel_loc>**

Specifies the location of the Linux kernel to be printed in the Barebox menu. Replace this with one of the following:nand - To boot the Linux kernel from NANDtftp - To boot the Linux kernel via TFTPext3 - To boot the Linux kernel from the SD/MMC card

- **<rootfs_loc>**

Specifies the location of the root filesystem to be printed in the Barebox menu. Replace this with one of the following:nand - To mount the root filesystem from NAND Flashtftp - To mount the root filesystem via NFSext3 - To mount the root filesystem from the SD/MCC card

- **<kernel_loc_bootm.image>**

Specifies the location of the Linux kernel image/dev/nand0.kernel.bb - To boot the Linux kernel from NAND${path}/linuximage - To boot the Linux kernel via TFTP/mnt/mmc/linuximage - To boot the Linux kernel from SD/MMC card

- **<rootfs_loc_dyn.root>**

Specifies the location of the root filesystemroot=ubi0:root ubi.mtd=5 rootfstype=ubifs - To mount the root filesystem from NANDroot=/dev/nfs nfsroot=$nfsroot,v3,tcp - To mount the root filesystem via NFSroot=/dev/mmcblk0p2 rootfstype=ext3 rootwait - To mount the root filesystem from SD/MMC card

- **<nfs_root_path>**

Only required if mounting the root filesystem from NFS. Replace with the following:nfsroot="/home/${global.user}/nfsroot/${global.hostname}"

Once complete with file modifications exit the editor using **CTRL+D**. Save the environment:

```
saveenv
```

Restart the board and stop autoboot by pressing *m*. You'll see the original menu but with an additional boot entry. The following example shows an added boot entry that boots the kernel from TFTP and mounts the root filesystem from NAND Flash.

```
Welcome to Barebox
      1: Boot: Kernel:nand;rootfs:nand
      2: Boot: Kernel:tftp;rootfs:nand
      3: Boot: network (Kernel:tftp;rootf:nfs)
      4: Boot: MMC (ext3)
      5: Settings
      6: Shell
      7: Reset
```

To run your custom boot entry, press the corresponding number and then the enter key. In the example above, to boot the kernel from TFTP and mount the root filesystem from NAND Flash press *3* and then the enter key.

The custom boot entry can also be run from the Barebox shell:

```
boot custom
```

Default Boot

The phyFLEX-i.MX6 is configured by default to boot the kernel and mount the root filesystem from flash. This default setting can be changed by modifying the *global.boot.default* environment variable.

```
edit /env/config
```

Find the boot entries section, uncomment the *global.boot.default* variable and set to the desired boot source. This variable can be set to any of the entries in */env/boot*. For example to set the default boot configuration to net, *global.boot.default* should be set in the following way:

```
global.boot.default=net
```

Exit the editor by **CTRL+D** and save the environment (saveenv). On a reset or power cycle the new default boot source will take affect. Similarly it will be used in the Barebox shell when executing the following:

```
boot
```

Flashing Images

The phyFLEX-i.MX6 Rapid Development Kit is delivered with a pre-flashed bootloader. The following instructions for flashing images from TFTP or SD card will be useful if you want to:

- Flash images because NOR is empty
- Upgrade to a new release
- Use custom built images

The images to be flashed will need to be copied to the exported TFTP directory or the */boot* partition of a properly formatted SD card as described in the Cr eating a Bootable SD card section of the Quickstart.

After making all required connections, power on the board and enter Barebox:

- If flashing from TFTP, connect an Ethernet cable from the Ethernet connector *ETH0/POE* on the Carrier Board to your network hub, router, or switch.
- If flashing from SD card, insert a correctly formatted SD card into the SD/MMC card slot connector X57 on the Carrier Board.
- Connect a serial RS-232 cable from a free port on your Host PC to the connector X51 on the Carrier Board.
- Start your favorite terminal software (Windows: PuTTY or TeraTerm; Linux: Minicom) on your Host PC and configure it for 115200 baud, 8 data bits, no parity, and 1 stop bit (8n1).
- Power on your board by connecting the 12 V wall adapter to X12 on the Carrier Board.
- After application of power, within approximately three seconds, hit any key to halt autoboot and enter Barebox. If an SD card was inserted, the */boot* partition will automount to */mnt/mmc*. If an Ethernet cable is connected and the network has been configured, TFTP will automount to */mnt /tftp*.

If flashing from TFTP, additional setup to configure the Barebox environment variables to meet your network environment and development host settings is required. The current network settings can be checked by executing the following:

```
devinfo
```

If you need to change you network configuration, type:

```
edit /env/network/eth0
```

Edit the settings as described in the Network Configuration section of the Quickstart. Save the environment and reboot the board, this will automount your tftp server at boot to */mnt/tftp*.

Barebox

If you would like to upgrade, have custom Barebox requirements, or are interested in seeing the version you built in action, follow the steps below:

It should be copied to your TFTP exported directory or the */boot* partition of the SD card depending on your chosen flashing procedure.

Note:

Be sure that the *barebox-imx6q-pbab01.imx* file has the correct RAM size for your version of the phyFLEX-i.MX6 SOM. This is configured as part of the build settings.

- Copy the new Barebox from your tftp-server or SD card into the module's RAM:

Method: TFTP

```
cp /mnt/tftp/barebox-imx6q-pbab01.imx .
```

Method: SD/MMC

```
cp /mnt/mmc/barebox-imx6q-pbab01.imx .
```

- Store the Barebox into the SPI NOR Flash:

```
erase /dev/m25p0.barebox
cp barebox-imx6q-pbab01.imx /dev/m25p0.barebox
```

Note:

If something goes wrong and you don’t have a bootloader anymore on your module you need to boot from an SD card into Barebox (set DIP-switch S3 to ON-ON-OFF-OFF) and then do the flashing. See the Creating a Bootable SD card section of the Quickstart section of this Quickstart for a description of how to create a bootable SD card.