# Build Procedure for AM572X-BEL-MMRFIC Board

# Table of Contents

1. BSP Development Guide for Phytec Board ..... 1
    1.1. Requirements ..... 1
    1.2. Setup and Build the BSPLink to Setup and Build the BSP ..... 1
    1.3. Host Setup ..... 1
    1.4. Git Setup ..... 2
    1.5. Setup the BSP Directory ..... 2
    1.6. Install the Toolchain ..... 2
    1.7. Download the BSP Meta Layers ..... 3
    1.8. Configure the Build ..... 3
    1.9. Build the BSP ..... 4
    1.10. Build Images ..... 4
    1.11. Source Locations ..... 5
2. Fixing errors in default Phytec BSP Build ..... 6
3. Creating Yocto Meta layer for MMRFIC ..... 8
    3.1. Build the BSP ..... 8
4. Reference ..... 9

# Chapter 1. BSP Development Guide for Phytec Board

- This BSP Development Guide provides you with the tools and know-how to install and work with the Linux Board Support Package (BSP) for the phyCORE-AM57x Development Kit.

## 1.1. Requirements

- The following minimum system requirements are necessary to successfully complete this BSP Development Guide. Deviations from these requirements may or may not have other workarounds:
- Ubuntu 18.04 LTS, 64-bit Host Machine with root permission

---

**Note**

If using a virtual machine, VMWare Workstation, VMWare Player, and VirtualBox are all viable solutions.

---

- 500GB or greater free disk space
- 8GB or greater of RAM
- 4x processing cores or greater available to the Ubuntu Host Machine
- SD card reader operational under Linux
- Active Internet connection

## 1.2. Setup and Build the BSPLink to Setup and Build the BSP

- This section will show you how to configure your development host to build your own BSP images from source and how to start BSP image builds. Building BSP images from source is useful for those who require modifying the default Linux image and if you would like to deploy those changes in an easy and reproducible way.

## 1.3. Host Setup

- Yocto development requires certain packages to be installed. Run the following commands to install them. Note the packages here assume a Ubuntu 18.04 LTS Linux distribution.

Host: Ubuntu

```
sudo apt-get update
sudo apt-get install git build-essential \
           python python3 diffstat texinfo gawk chrpath \
           dos2unix wget unzip socat doxygen bison \
           flex lzop libssl-dev u-boot-tools curl vim \
           gcc-multilib g++-multilib dialog cpio
```

- Verify that the /bin/sh shell for your Host PC is `bash` and not `dash`. This step is required.

Host: Ubuntu

```
sudo dpkg-reconfigure dash
```

```
bash
```

**Note**

Respond "No" to the prompt asking "Install dash as (/bin/sh)?"

- Install the *repo* tool

Host: Ubuntu

```
curl https://storage.googleapis.com/git-repo-downloads/repo > /usr/bin/repo
chmod a+rx /usr/bin/repo
```

## 1.4. Git Setup

- If you have not yet configured your Git environment on your host machine, please execute the following commands to set your user name and email address.

Host: Ubuntu

```
git config --global user.email "your@email.com"
git config --global user.name "Your Name"
git config --global http.sslcainfo /etc/ssl/certs/ca-certificates.crt
```

## 1.5. Setup the BSP Directory

- Create a directory which will house your BSP development and navigate into it.

Host: Ubuntu

```
mkdir ~/BSP-Yocto-TISDK-AM57xx-PD20.1.2
cd ~/BSP-Yocto-TISDK-AM57xx-PD20.1.2
```

## 1.6. Install the Toolchain

- Run the following commands to install the toolchain Yocto will leverage to build all the source packages:
- /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

Host: Ubuntu

Download the toolchain by running the following command on host:

```
wget "https://developer.arm.com/-/media/Files/downloads/gnu-a/8.3-2019.03/binrel/gcc-arm-8.3-2019.03-x86_64-arm-linux-gnueabihf.tar.xz" -O /tmp/gcc-x86_64-arm-linux-gnueabihf.tar.xz
```

```
mkdir -p /opt/tools/

sudo chown -R <user>: /opt/tools

tar -Jxvf /tmp/gcc-x86_64-arm-linux-gnueabihf.tar.xz -C /opt/tools/
```

## 1.7. Download the BSP Meta Layers

- In order to successfully synchronize your host environment with the repositories called for in the BSP manifest, you will need to modify your git config in the following way:

Host: Ubuntu

```
echo -e '[url "https://github.com/"]\n insteadOf = "git://github.com/"' >> ~/.gitconfig
```

- Download the manifest file for the phyCORE-AM57x BSP

Host: Ubuntu

```
repo init -u https://stash.phytec.com/scm/pub/manifests-phytec.git -b am57xx \
     -m PD20.1.2.xml
repo sync
```

## 1.8. Configure the Build

- Run the Yocto build directory setup script. The TEMPLATECONF variable is used to set the source of the local configuration files (conf/bblayers.conf and conf/local.conf), which are located in the meta-phytec layer:

Host: Ubuntu

```
TEMPLATECONF=`pwd`/sources/meta-phytec/meta-phytec-ti/conf source \
                sources/oe-core/oe-init-build-env build
```

- Once you have run the above once, you won't need to ever again. All new terminal sessions in the future can source the build environment simply by running the following commands:

Host: Ubuntu

```
cd ~/BSP-Yocto-TISDK-AM57xx-PD20.1.2
source sources/oe-core/oe-init-build-env
```

- After running the above command, you should be automatically placed into a newly created build directory at '~/BSP-Yocto-TISDK-AM57xx-PD20.1.2/build/'. This directory can now also be easily referenced with the $BUILDDIR environment variable.
- Open the conf/local.conf file using your favorite editor:

Host: Ubuntu

```
vim conf/local.conf
```

- And make the following modifications. Select a specific machine to target the build with.

Host: Ubuntu

```
MACHINE ?= "am57xx-phycore-kit"
```

- Maximize the build's efficiency by modifying the BB_NUMBER_THREADS and PARALLEL_MAKE variables to suit your host development system. By default, these are set to 4 in `build/conf/local.conf`.

Host: Ubuntu

```
# Parallelism options - based on cpu count
BB_NUMBER_THREADS ?= "7"
PARALLEL_MAKE ?= "-j 7"
```

- Add the following to a new line at the end of the file to set the TOOLCHAIN_BASE variable to point to where you extracted the toolchain.

In build/conf/local.conf

Host: Ubuntu

```
TOOLCHAIN_BASE = "/opt/tools"
```

- Increase the limit on open file descriptors in your bash shell from the default of 1024 to 8192:

Host: Ubuntu

```
ulimit -n 8192
```

## 1.9. Build the BSP

- The setup is complete and you now have everything ready to start a build. This BSP has been tested with the arago-core-tisdk-bundle and it is suggested that you start with this image before building other images.
- The following will start a build from scratch. By default this build target will generate a bootloader, Linux kernel, root filesystem images, and a SDK by default.

Host: Ubuntu

```
cd $BUILDDIR
bitbake arago-core-tisdk-bundle
```

### Note

On a freshly installed Ubuntu machine if you face an error `Please use a locale setting which supports UTF-8` or something similar, you can solve this by running `apt install locales` followed by `locale-gen en_US en_US.UTF-8` and then run `sudo dpkg-reconfigure locales` on your Ubuntu Host

### Note

If any of the recipe fails to fetch or compile, we can clean the failed recipe and try a complete build again. Do this by running `bibtake -f -c cleanall <failed-recipe-name>` and then run `bitbake <failed-recipe-name>`, once the recipe builds you can build the complete build by running `bitbake arago-core-tisdk-bundle`

## 1.10. Build Images

- All images generated by bitbake are deployed to $BUILDDIR/arago-tmp-external-armtoolchain/deploy/images/<MACHINE>
- Bootloader: MLO, u-boot.img

- Kernel: zImage
- Kernel device tree file: am5728-phycore-kit-41300111i.dtb
- Root Filesystem: tisdk-rootfs-image-am57xx-phycore-kit.tar.xz
- SDK: processor-sdk-linux-bundle-am57xx-phycore-kit.tar.xz

## 1.11. Source Locations

- Kernel: $BUILDDIR/arago-tmp-external-arm-toolchain/work/<MACHINE>-linux-gnueabi/linuxphytec-ti/4.19.79+git_v4.19.79-phy3-r7a/git/
- The device tree file to modify within the linux kernel source is: am5728-phycore-kit41300111i.dts and its included dtsi files.
- U-boot: $BUILDDIR/arago-tmp-external-arm-toolchain/work/<MACHINE>-linux-gnueabi/u-bootphytec/2019.01+git_v2019.01-phy5-r0/git/

# Chapter 2. Fixing errors in default Phytec BSP Build

To Solve the error in base-files recipes, follow the below steps.

- Open `${BSPDIR}/sources/meta-phytec/meta-phytec-ti/recipes-core/base-files/base-files_%.bbappend` file with any text editor

```
vi sources/meta-phytec-ti/recipes-core/base-files/base-files_%.bbappend
```

- Comment the following lines in do_install_basefilesissue_append function

```
# if [[ -n ${PHYTEC_META_GIT_REL_TAG} ]]; then
# if [[ -e ${D}${sysconfdir}/issue && -e ${D}${sysconfdir}/issue.net ]];
then
# printf "PHYTEC: ${PHYTEC_META_GIT_REL_TAG}\n" >>
${D}${sysconfdir}/issue
# printf "PHYTEC: ${PHYTEC_META_GIT_REL_TAG}\n" >>
${D}${sysconfdir}/issue.net
# fi
# fi
```

To solve fetch errors fix the following:

- In file `${BSPDIR}/sources/oe-core/meta/recipes-connectivity/mobile-broadband-providerinfo/mobile-broadband-provider-info_git.bb` edit the SRC_URI line as follows:

```
SRC_URI = "git://gitlab.gnome.org/GNOME/mobile-broadband-providerinfo.git; \
           protocol=https;branch="main"; \
           file://multilibfix.patch \
"
```

- In file `${BSPDIR}/sources/oe-core/meta/recipes-rt/rt-tests/rt-tests.inc` edit the SRC_URI line as follows:

```
      SRC_URI = "git://git.kernel.org/pub/scm/utils/rt-tests/rt-tests.git; \
                 branch=main"
```

- In file `${BSPDIR}/sources/meta-virtualization/recipes-devtools/go/go-systemd_git.bb`

```
SRC_URI = "git://${PKG_NAME}.git;branch=main"
```

- In file `${BSPDIR}/sources/meta-openembedded/meta-oe/recipes-test/gtest/gtest_1.8.0.bb`

```
SRC_URI = "\
    git://github.com/google/googletest.git;protocol=https;branch=main \
    file://Add-pkg-config-support.patch \
"
```

- In file `${BSPDIR}/sources/meta-openembedded/meta-networking/recipes-support/bridge-utils/bridge-utils_1.6.bb`

```
SRC_URI = "\
    git://git.kernel.org/pub/scm/linux/kernel/git/shemminger/bridge-utils.git; \
    branch=main \
    file://kernel-headers.patch \
    file://0005-build-don-t-ignore-CFLAGS-from-environment.patch \
    file://0006-libbridge-Modifying-the-AR-to-cross-toolchain.patch \
"
```

# Chapter 3. Creating Yocto Meta layer for MMRFIC

- Open the conf/local.conf file using your favorite editor:

Host: Ubuntu

```
vim conf/local.conf
```

- Select a specific machine to target the build with.

Host: Ubuntu

```
MACHINE ?= "am572x-bel-mmrfic"
```

- To Create a Yocto meta layer, bitbake command can be used bitbake `yocto-layer create meta-mmrfic-bel-am572x`
- Edit BSP-Yocto-TISDK-AM57xx-PD20.1.2/build/conf/bblayers.conf file to add the new layer to BBLAYERS.
- Add the below line to bblayers.conf file.

```
${BSPDIR}/sources/meta-mmrfic-bel-am572x
```

- Download the recipes for U-boot, Kernel and conf from mmrfic github and copy it to the metammrfic-bel-am57x folder

```
git clone https://github.com/mmrfic/meta-mmrfic-bel-am572x.git
cd meta-mmrfic-bel-am572x
cp ./* {BSPDIR}/sources/meta-mmrfic-bel-am572x
```

## 3.1. Build the BSP

- The setup is complete and you now have everything ready to start a build. This BSP has been tested with the arago-core-tisdk-bundle and it is suggested that you start with this image before building other images.
- The following will start a build from scratch. By default this build target will generate a bootloader, Linux kernel, root filesystem images, and a SDK by default.

Host: Ubuntu

```
cd $BUILDDIR
bitbake arago-core-tisdk-bundle
```

# Chapter 4. Reference

https://develop.phytec.com/phycore-am57x/latest/software-development/bsp-development/buildthe-bsp