DYNAMIC ENGINEERING
150 DuBois, Suite C
Santa Cruz, CA 95060
(831) 457-8891 **Fax** (831) 457-4793
http://www.dyneng.com
sales@dyneng.com
Est. 1988

Tool installation for PMC-MC-X2/X4 with P2020 series processor

6/22/2018 REV A2

Table of Contents

Introduction

On the development board a minimized version of Linux 2.6.35 is installed.

Linux minimal image is a Busy Box. Development tools are not a part of Busy Box environment. According to Linux traditions, native for the hardware platform development tools are part of full Linux image only. You cannot do development on a small, embedded platform with NXP's P2020 CPU. Cross-platform development is typical Linux way for software development for embedded devices. It is necessary to install and build cross platform toolchain on the development host.

Development platform setup

The development desktop is Intel x86-64 based computer with Linux CentOS-7.5.1804 with developer options installed. Standard development environment for that version of Linux comes with make tools version 3.82. That version has some bugs. Replace make 3.82 with make 4.1

⇔ procedure:

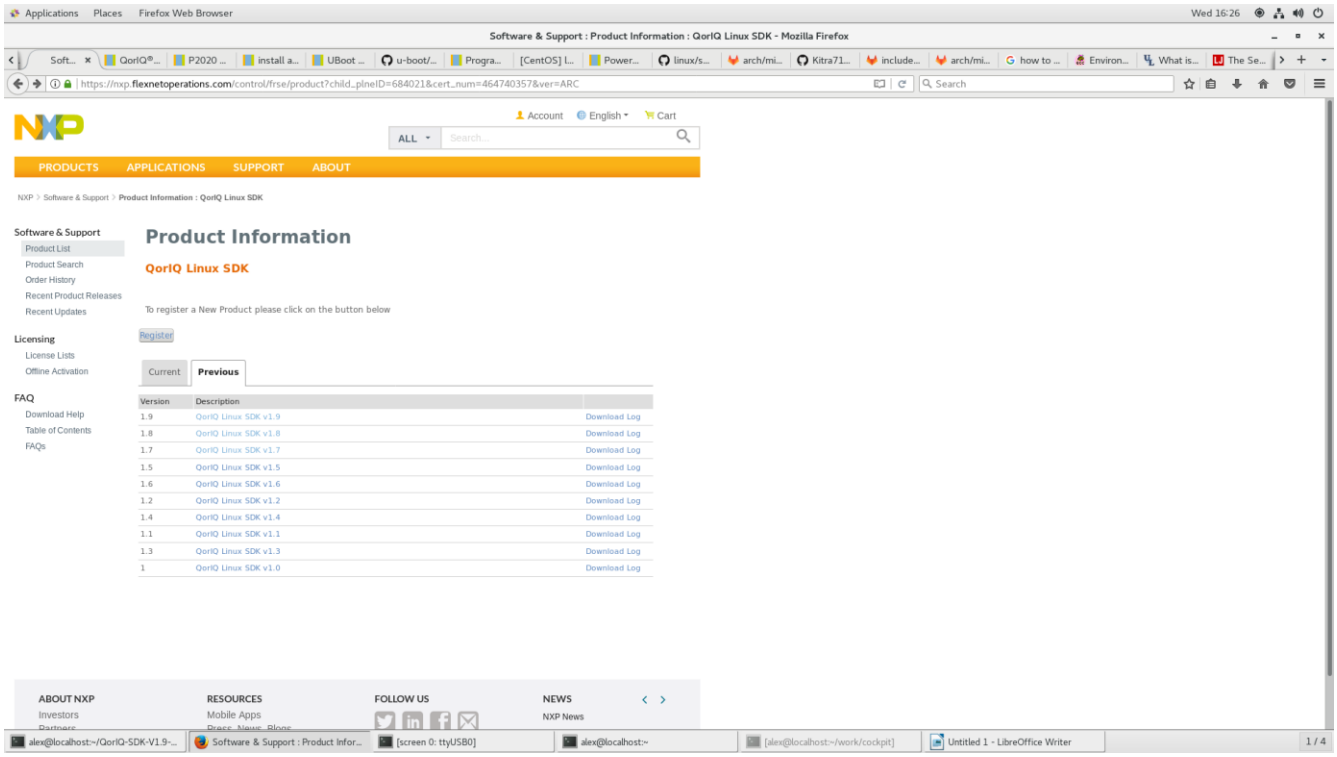
```
cd /tmp
wget http://ftp.gnu.org/gnu/make/make-4.1.tar.gz
tar xvf make-4.1.tar.gz
cd make-4.1/
./configure
make
sudo make install
rm -rf make-4.1.tar.gz make-4.1
```

After these few steps, the make files are installed in /usr/local/bin/make.

=> Skip if you already have installed make tools 4.1 or above.

Cross platform development tools setup

P2020 is NXP's popular communication processor with PowerPC Architecture. NXP has Linux SDK for Linux development for P2020 at https://www.nxp.com/support/developer-resources/run-time-software/linux-software-and-development-tools/linux-sdk-for-qoriq-processors:SDKLINUX
You will need to register your name and email to download the SDK.
Current SDK v.2.0 doesn't support P2020 QorIQ processor and you will need to select Previous tab



QorIQ Linux SDK v1.9 is the latest Linux development package for P2020 processor.

QorIQ Linux SDK V1.9

Files | License Keys | Notes

Download Help

Show All Files

24 Files

File Description	File Size	File Name
Cache: QorIQ SDK V1.9 CORTEXA7 CACHE	3.2 GB	QorIQ SDK V1.9 CORTEXA7 CACHE.iso
Cache: QorIQ SDK V1.9 PPC64E5500 CACHE	3.6 GB	QorIQ SDK V1.9 PPC64E5500 CACHE.iso
Cache: QorIQ SDK V1.9 PPC64E6500 CACHE	3.6 GB	QorIQ SDK V1.9 PPC64E6500 CACHE.iso
Cache: QorIQ SDK V1.9 PPCE500MC CACHE	3.3 GB	QorIQ SDK V1.9 PPCE500MC CACHE.iso
Cache: QorIQ SDK V1.9 PPCE500V2 CACHE	3.4 GB	QorIQ SDK V1.9 PPCE500V2 CACHE.iso
Cache: QorIQ SDK V1.9 PPCE5500 CACHE	3.3 GB	QorIQ SDK V1.9 PPCE5500 CACHE.iso
Cache: QorIQ SDK V1.9 PPCE6500 CACHE	3.5 GB	QorIQ SDK V1.9 PPCE6500 CACHE.iso
Document: QorIQ Linux SDK v1.9 Knowledge Center	16.9 MB	QorIQ-SDK-1.9-IC-RevA.zip
Document: QorIQ Linux SDK v1.9 Release Notes	99.3 KB	QORIQ-SDK-1-9_RN.pdf
Image: QorIQ SDK V1.9 CORTEXA7 IMAGE	767.8 MB	QorIQ SDK V1.9 CORTEXA7 IMAGE.iso
Image: QorIQ SDK V1.9 PPC64E5500 IMAGE	2.8 GB	QorIQ SDK V1.9 PPC64E5500 IMAGE.iso
Image: QorIQ SDK V1.9 PPC64E6500 IMAGE	2 GB	QorIQ SDK V1.9 PPC64E6500 IMAGE.iso
Image: QorIQ SDK V1.9 PPCE500MC IMAGE	1.2 GB	QorIQ SDK V1.9 PPCE500MC IMAGE.iso
Image: QorIQ SDK V1.9 PPCE500V2 IMAGE	1.7 GB	QorIQ SDK V1.9 PPCE500V2 IMAGE.iso
Image: QorIQ SDK V1.9 PPCE5500 IMAGE	2.5 GB	QorIQ SDK V1.9 PPCE5500 IMAGE.iso
Image: QorIQ SDK V1.9 PPCE6500 IMAGE	2 GB	QorIQ SDK V1.9 PPCE6500 IMAGE.iso
Source: QorIQ SDK V1.9 SOURCE	3.1 GB	QorIQ SDK V1.9 SOURCE.iso
Virtual Host Enviroment: QorIQ DPAA SDK v1.9 CORTEXA7 with Virtual Host Enviroment	10.2 GB	QorIQ DPAA SDK v1.9 CORTEXA7 with Virtual Host Enviroment.zip
Virtual Host Enviroment: QorIQ DPAA SDK v1.9 PPC64E5500 with Virtual Host Enviroment	11.7 GB	QorIQ DPAA SDK v1.9 PPC64E5500 with Virtual Host Enviroment.zip
Virtual Host Enviroment: QorIQ DPAA SDK v1.9 PPC64E6500 with Virtual Host Enviroment	11.1 GB	QorIQ DPAA SDK v1.9 PPC64E6500 with Virtual Host Enviroment.zip
Virtual Host Enviroment: QorIQ DPAA SDK v1.9 PPCE500MC with Virtual Host Enviroment	10.2 GB	QorIQ DPAA SDK v1.9 PPCE500MC with Virtual Host Enviroment.zip
Virtual Host Enviroment: QorIQ DPAA SDK v1.9 PPCE500V2 with Virtual Host Enviroment	10.8 GB	QorIQ DPAA SDK v1.9 PPCE500V2 with Virtual Host Enviroment.zip
Virtual Host Enviroment: QorIQ DPAA SDK v1.9 PPCE5500 with Virtual Host Environment	11.3 GB	QorIQ DPAA SDK v1.9 PPCE5500 with Virtual Host Environment.zip
Virtual Host Enviroment: QorIQ DPAA SDK v1.9 PPCE6500 with Virtual Host Enviroment	10.9 GB	QorIQ DPAA SDK v1.9 PPCE6500 with Virtual Host Enviroment.zip

You need 3 packages only:

1. QorIQ SDK V1.9 PPCE500V2 CACHE.iso
2. QorIQ SDK V1.9 PPCE500V2 IMAGE.iso
3. QorIQ SDK V1.9 SOURCE.iso

SDK Image has pre-built binaries. A good option to use them for reference. SDK Cache has pre-built binaries. T hey will save significant time for you if you install SDK Source and SDK cache into same folder.

Mount SDK cache first and run ./install from the top folder. Don't do anything else with SDK cache, please! You may unmount it after installation.

Mount SDK source and run ./install. Choose same place when you have install SDK cache.

Now you can follow QorIQ SDK 1.9 Documentation to prepare Host development environment, setup Poky to build images.

Host development environment needs Python 2.7 or above. If you don't have Python
⇔ Procedure:
$ wget https://www.python.org/ftp/python/2.7.6/Python-2.7.6.tar.xz
[NOTE: Python 2.7.3 and python 2.7.5 can be used as well.]
$ tar -xf Python-2.7.6.tar.xz
$ cd Python-2.7.6
$ ./configure --prefix=/opt/python-2.7.6
$ make
$ sudo make install
Run export command [below] to ensure python 2.7.x is used for Yocto build.
$ export PATH=/opt/python-2.7.6/bin:$PATH

For a list of the Linux distributions tested by the Yocto Project community see SANITY_TESTED_DISTROS
in poky/meta-yocto/conf/distro/poky.conf.
The following is the detailed package list on the **CentOS** hosts:
$ sudo yum install gawk make wget tar bzip2 gzip python unzip perl patch \
diffutils diffstat git cpp gcc gcc-c++ glibc-devel texinfo chrpath socat SDL-devel xterm
For the **Fedora** hosts:
$ sudo yum install sudo yum install gawk make wget tar bzip2 gzip python unzip perl
patch \
diffutils diffstat git cpp gcc gcc-c++ glibc-devel texinfo chrpath \

ccache perl-Data-Dumper perl-Text-ParseWords perl-Thread-Queue socat \
findutils which SDL-devel xterm
For **Ubuntu** and **Debian** hosts:
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib \
build-essential chrpath socat libsdl1.2-dev xterm
Extra packages are needed for **Ubuntu-64b**:
$ sudo apt-get install lib32z1 lib32ncurses5 lib32bz2-1.0 ia32-libs lib32ncurses5-dev
For **OpenSUSE** host:
$ sudo zypper install python gcc gcc-c++ git chrpath make wget python-xml \
diffstat makeinfo python-curses patch socat libSDL-devel xterm

To setup Poky execute a help command
$ . ./fsl-setup-poky -h
This command output provides a list of architectures to support by QorIQ SDK V1.9
Poky installation is
$ . ./fsl-setup-env -m p2020rdb
p2020rdb is a machine to support p2020 CPU.

Build cross-development tools and images

$ cd <sdk-install-dir>/build_p2020rdb
$ bitbake <image-target>
Where <image-target> is one of the following:
• fsl-image-minimal : contains basic packages to boot up a board
• fsl-image-core : contains common open source packages and FSL specific packages.
• fsl-image-full : contains all packages in the full package list.
• fsl-image-mfgtool : contains all the user space apps needed to deploy the fsl-image-mfgtool image to
a USB stick, hard drive, or other large physical media.
• fsl-image-virt : contains toolkit to interact with the virtualization capabilities of Linux
• core-image-x11 : Freescale image with a very basic X11 image with a terminal
• fsl-toolchain : the cross compiler binary package

Select fsl-toolchain to build standalone toolchain only.

Install toolchain on your development Host

$ cd build_p2020rdb/tmp/deploy/sdk
$ ./fsl-networking-eglibc-<host_arch>-<core>-toolchain-<release>.sh

The default installation path for standalone toolchain is /opt/fsl-qoriq/1.9/. The install folder can be specified during the installation procedure.

Verify your installation

$ echo $PATH
you should see /opt/fsl-qoriq/1.9/ in the output list
/opt/fsl-qoriq/1.9/sysroots/x86_64-fslsdk-linux/usr/bin:/opt/fsl-qoriq/1.9/sysroots/x86_64-fslsdk-linux/usr/bin/powerpc-fsl-linux-gnuspe:/home/alex/QorIQ-SDK-V1.9-20151210-yocto/sources/poky/scripts:/home/alex/QorIQ-SDK-V1.9-20151210-yocto/sources/poky/bitbake/bin:/home/alex/CodeSourcery/Sourcery_G++_Lite/bin:/home/alex/CodeSourcery/Sourcery_G++_Lite/bin:/usr/lib64/qt-3.3/bin:/home/alex/perl5/bin:/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:/home/alex/.local/bin:/home/alex/bin

Verify development toolchain

$ echo $CC
output should be like that
powerpc-fsl-linux-gnuspe-gcc -m32 -mcpu=8548 -mabi=spe -mspe -mfloat-gprs=double –sysroot=/opt/fsl-qoriq/1.9/sysroots/ppce500v2-fsl-linux-gnuspe
That is a proof the development tools are installed and you are ready for Hello, World!

Application development! When you compiled your first Hello,World! Application, copy your executable on the target for execution.