Linux-c6x 2.0 Release

Introduction

This is the 2.0 GA release of Linux-c6x. This release demonstrates Linux on the c66x and c64x+ devices. Users are encouraged to update to later releases when available. Enhancements and bug fixes will be applied to 2.0.x releases.

Release highlights

- Based on Linux kernel v2.6.34
- uClibc v0.9.31-rc
- busybox v1.17.1
- Drivers such as i2c, Ethernet, UART.
- Support for peripherals such as EEPROM and NAND
- Supports SysLink IPC (not supported on c66x big endian)
- MCSDK (Multicore SDK) web control panel demo (c66x platforms)
  - includes Multicore elfloader and IPC demo through SysLink IPC and BIOS/IPC. (only c66x little endian)
- Supported devices
  - EVMC6678
  - EVMC6670
  - EVMC6474L
  - EVMC6474
  - EVMC6472
  - EVMC6457
  - DSK6455
- Ethernet boot loader support in both endians for all devices that are part of the release
- NAND boot loader support in both endians for the EVMC6678. EVMC6670, EVMC6474L, EVMC6457 and EVMC6472.

This release uses the GCC toolchain for most of the project. The correct GCC version is downloaded if needed by the ./prj config process and need not be downloaded by hand. For reference this release uses an the gcc tool chain (4.5-124) [1] from Code Sourcery.

Various other components are need to build or use all parts of this release. The table below summarizes these requirements.

Manual Dependencies

Resource	Alternate source	Required for
CCSv5.0.3.00028 [2]	(none)	JTAG debug, JTAG bootloader update, JTAG NAND update
TI CGT 7.2.2 [3]	CCSv5.0.3	building SysLink, rio-utils, bootloader
BIOS 6.32.01.38 [4]	BIOS MCSDK 2.0.5	building SysLink BIOS examples
IPC 1.23.01.26 [5]	BIOS MCSDK 2.0.5	building SysLink
XDC tools 3.22.01.21 [6]	BIOS MCSDK 2.0.5	building SysLink

This release was built on Ubuntu 10.04 32 bit and that is the recommended host configuration. Other recent 32 bit Linux distributions are expected to work but may require extra Linux knowledge. 64 bit hosts have not been verified at this time. Older Linux distributions (for example Red Hat Enterprise Linux 4) would require extra work and updated software and is not recommended. Building on non-Linux hosts is not supported at all; use a virtual Linux host if you must use Windows, Mac, etc.

This release uses the CGT based TI tool chain only to compile the BIOS example programs and the bootloader. It is not validated for producing Linux binaries although informal use has shown this to work.

New, Improved, or Fixed in the Release

Changes in 2.0 since 2.0-beta2:

- adds ./prj config to automate setup
- adds ./prj build to standardize interface to make
- enhanced top level make, reduce rebuilds
- integrates bootloader and program_evm HW updater scripts
- adds ./bootblob <template>|all facility to configure filesystem and kernel images JIT
- build JFFS images (JIT w/ templates)
- full-root fully integrated into top level build
- boot from NAND on 6670, 6474L/72/57 now works in Big Endian
- fixed i2c EEPROM read/write from Linux on 6678
- web control panel can now upadte IBL bootloader's default TFTP filename
- various fixes for DDR3 and PLL init stability

Changes in 2.0-beta2 since 2.0-beta1:

- adds hard float for C66
- adds big endian for C66 (with some limitations)
- re-adds C64x+ support
- adds C64x+ NAND boot support

Changes in 2.0-beta1 since 2.0-alpha2:

- Adds support for SysLink Interprocessor Communication on C66 devices
- Adds MCSDK (Multicore SDK) web control panel demo including Multicore IPC demo through SysLink IPC and BIOS/IPC.
- Added an elf loader application

2.0-alpha2:

- was in NAND for initial EVM6670L production

2.0-alpha1:

- was in NAND for initial EVM6678L production

Location of the Release Files

The Software can be downloaded from ti.com Software Download Site [7] or from Linux-c6x.org File Download Area [8].

Unlike previous beta releases, the IBL bootloader is fully integrated into this release. No separate download is required.

Using Prebuilt Binaries

The binaries for your selected platform can be downloaded from the locations mentioned above. Please see the User's Guide [9] for instructions on using the prebuilt binaries.

Building Software from the Source

Cloning the git repositories

Ensure you have git and your have proxy configuration correct if you are behind a non-transparent firewall.

Issue the following commands

```
mkdir ~/my-linux-c6x
cd    ~/my-linux-c6x
wget http://linux-c6x.org/bootstrap
chmod +x bootstrap
./bootstrap linux-c6x-2.0
```

If you have trouble with this setup, see the next section to start from a source tar file.

Getting sources from the tar file

As an alternative to cloning from the git repositories, you can download the release tar file and extract it. This may be easier for you if you are behind a firewall but it will only let you build this one release.

You do not need to do this if you have already cloned the git repos.

Issue the following commands

```
cd
wget http://linux-c6x.org/files/releases/linux-c6x-2.0.0.63/linux-c6x-2.0.0.63-src.tar.gz
tar xvzf linux-c6x-2.0.0.63-src.tar.gz
mv linux-c6x-2.0.0.63 my-linux-c6x
cd my-linux-c6x
```

The rename of the directory above is of course optional. The software will work with any project level directory name and does not need to be in at the top level of the your home directory. However the above keeps the directions in this guide consistent.

Download Manual Dependencies

Download the manual dependencies based on the table above. If you want the software available to all projects you can install it into the machine (/opt/ti) or your home dir (~/ti).

Alternatively you can place the downloaded files into the project's ./download directory (ex: ~/my-linux-c6x/downloads).

At ./prj config time (in the next step), if needed resources are not found installed on the machine already, the download files will be used to installed into the projects ./opt directory.

Configuration

- For Ubuntu 10.04 Host, the ./prj config step will check or configure your host. It will instruct you to install any missing packages.

```
cd linux-c6x-project
./prj config
[Please edit setenv to specify what to build, or leave it untouched to build the example configuration]
./prj config
```

Alternatively, you may instruct ./prj config to perform the steps for you by using the command:

```
cd linux-c6x-project
AUTO_INSTALL=yes ./prj config
[Please edit setenv to specify what to build, or leave it untouched to build the example configuration]
./prj config
```

- For other Linux distros, refer to [Setting up the Linux host [10]] and then run ./prj config to verify.

Build

```
./prj build
```

Additional information

By default the build will generate kernels for the 6678 EVM. The **setenv** can be modified easily to build kernels for other EVMs. The best source of information to figure what to change in the setenv is the setenv itself. It is an excellent README in itself which also has instructions on how to build SysLink

Changes made to the setenv are picked up by the ./prj build command.

If changes are made to the external dependencies specifiers (GCC_VERSION, CCS_VERSION, CCS_DIR, etc) you will need to rerun the configure step. Issue the following commands:

```
  ./prj config
```

If you, wish to build individual components from the command line you can set the envoronment for your shell using this command:

```
source setenv
```

Note: Sourcing setenv is not needed just to run the config and build steps. If you do this your settings can interfere with the normal ./prj build logic. If in doubt use a separate shell for each type of work.

Where to find the generated files?

Once the build is successful, the generated files can be found at ~/my-c6x-linux/product. Here it is assumed building only evmc6678.

```
$ cd ~/my-c6x-linux/product
$ ls -l
total 31424
-rwxr-xr-x 1 bill bill    11018 2011-12-10 01:42 bootblob
drwxr-xr-x 4 bill bill     4096 2011-12-10 01:42 bootblob-templates
-rw-r--r-- 1 bill bill 17137926 2011-12-10 03:49 full-root-c6x-hf.cpio.gz
-rw-r--r-- 1 bill bill   146099 2011-12-10 02:25 gplv3-devtools-c6x-hf.cpio.gz
-rw-r--r-- 1 bill bill    52064 2011-12-10 03:50 i2crom_0x51_c6678_le.bin
-rwxr-xr-x 1 bill bill     7792 2011-12-10 01:42 make-filesystem
-rw-r--r-- 1 bill bill  1619305 2011-12-10 03:19 mcsdk-demo-root-c6x-hf.cpio.gz
-rw-r--r-- 1 bill bill  1537994 2011-12-10 02:25 min-root-c6x-hf.cpio.gz
-rw-r--r-- 1 bill bill   549197 2011-12-10 03:49 modules-2.6.34-evmc6678.el-linux-c6x-2.0.0.63.tar.gz
-rw-r--r-- 1 bill bill  2390479 2011-12-10 03:19 syslink-demo-evmc6678.el-hf-linux-c6x-2.0.0.63.tar.gz
-rwxr-xr-x 1 bill bill  4676407 2011-12-10 02:09 vmlinux-2.6.34-evmc6678.el-linux-c6x-2.0.0.63
-rwxr-xr-x 1 bill bill  3957248 2011-12-10 02:09 vmlinux-2.6.34-evmc6678.el-linux-c6x-2.0.0.63.bin
```

Creating bootblobs and file system images

Between the various kernels for different boards, different file system contents, different file system formats, and different kernel command lines, there are a very many possible bootblobs and/or file-system images possible. Instead of shipping each of those images pre-built, the product directory contains a "kit" for doing final assembly yourself. The bootblob script has a template mode that does this. It will assemble a filesystem image and a bootblob adjusted for that filesystem.

To create the full set of bootblobs and file-system images, do the following:

```
cd ~/my-c6x-linux/product
./bootblob all
```

Any templates that require files you don't have will be skipped.

Alternatively you can execute just a single template like so:

```
cd ~/my-c6x-linux/product
./bootblob evmc6678-initramfs-demo
```

The generated file can be used for programming the NAND flash. See the User's Guide [9] for instructions on updating NAND.

The ./prj build process invokes the bootblob script based on the BOOTBLOBS setenv setting.

For further information please refer to Using the Bootblob Utility [11]

Memory partition information

On C6670 and C6678, Linux NetCP driver uses a part of the MSMC memory. Similarly SysLink uses DDR and MSMC memory for Shared Region. Here are the memory range used...

```
User    memory   Range
======================
NetCP   MSMC     0x0C000000-0x0C003000
SysLink MSMC     0x0C008000-0x0C0F8000 (For SharedRegion 0)
Linux   DDR      0x80000000-0x8FFFFFFF (Managed by Linux on core 0, assumes mem=256M on command line)
SysLink DDR      0X9FC00000-0x9FFFFFFF (For SharedRegion 1 and User specified SharedRegion)
```

Multicore loader application

The mcoreloader program is available to load images on slave cores. The application currently supports statically linked elf images and is demonstrated through Syslink sample applications while loading BIOS IPC ELF image on slave cores. The mcoreloader is included in the mcsdk-demo file system.

Known issues

- Bootblob - evmc6474-lite IP command line is not correct by default (Workaround: export IPADDR=dhcp; ./bootblob all)
- Kernel - a single LTP test case (sgetmask01) fails on C6457 little endian only
- Kernel - Require example code demonstrating EMAC functionality across multiple cores running heterogeneous OS
- Kernel - kernel clean up pending for usage of cache.h from user space
- Kernel - No watchdog support for c66x
- SysLink - Assert log message seen when doing rmmod syslink.ko after running messageQ sample application
- SysLink - GateMP, on RTOS side sendEvent fail log message is seen (but sample application runs to completion)
- SysLink - Assert log message seen when running HeapMemMP sample application
- SysLink - c6474 and c6474L EVMs GateMP sample application fails

Support

Please send any support related questions to the TI e2e forum [12]. Please use the tag **Linux-c6x** on all questions.

References

[1] https://support.codesourcery.com/GNUToolchain/release1882
[2] http://software-dl.ti.com/dsps/forms/self_cert_export.html?prod_no=setup_CCS_5.0.3.00028.tar.gz&ref_url=http://software-dl.ti.com/dsps/dsps_public_sw/sdo_ccstudio/CCSv5/CCS_5_0_3/
[3] http://software-dl.ti.com/dsps/forms/self_cert_export.html?form_type=2&prod_no=ti_cgt_c6000_7.2.2_setup_linux_x86.bin&ref_url=http://software-dl.ti.com/dsps/dsps_registered_sw/sdo_ccstudio/codegen/C6000/7.2.2
[4] http://software-dl.ti.com/dsps/dsps_public_sw/sdo_sb/targetcontent/bios/sysbios/6_32_01_38/index_FDS.html
[5] http://software-dl.ti.com/dsps/dsps_public_sw/sdo_sb/targetcontent/ipc/1_23_01_26/index_FDS.html
[6] http://software-dl.ti.com/dsps/dsps_public_sw/sdo_sb/targetcontent/rtsc/3_22_01_21/index_FDS.html
[7] http://software-dl.ti.com/sdoemb/sdoemb_public_sw/linux_mcsdk/02_00_00_63/index_FDS.html
[8] http://linux-c6x.org/files/releases/linux-c6x-2.0.0.63/
[9] http://linux-c6x.org/files/releases/linux-c6x-2.0.0.63/linux-c6x-2.0.0.63-users-guide.pdf
[10] http://www.linux-c6x.org/wiki/index.php/Setting_up_the_Linux_Host
[11] http://linux-c6x.org/wiki/index.php/Using_the_bootblob_utility
[12] http://e2e.ti.com/support/embedded/f/354.aspx

Article Sources and Contributors

Linux-c6x 2.0 Release *Source*: http://linux-c6x.org/wiki/index.php?oldid=1636 *Contributors*: BillMills, Sandeeppaulraj

License