IPC Install Guide QNX

Introduction

Inter/Intra Processor Communication (IPC) is a product designed to enable communication between processors in a multi-processor environment. Features of IPC include message passing, multi-processor gates, shared memory primitives, and more.

IPC is designed for use with processors running SYS/BIOS applications. This is typically an ARM or DSP. IPC includes support for High Level Operating Systems (HLOS) like Linux, as well as the SYS/BIOS RTOS. The breadth of IPC features supported in an HLOS environment is reduced in an effort to simplify the product.

Install

IPC is released as a zip file. To install, simply extract the file.

```
buildhost$ unzip ipc_<version>.zip
```

This will extract the IPC product in a directory with its product name and version information (e.g. **c:\ipc_3_xx_xx_xx** or **/home/<user>/ipc_3_xx_xx_xx**)

NOTE

- This document assumes the IPC install path to be the user's home directory on a Linux host machine (**/home/<user>**) or the user's main drive on a Windows host machine (**C:**). The variable **IPC_INSTALL_DIR** will be used throughout the document. If IPC was installed at a different location, make appropriate changes to commands.
- Some customers find value in archiving the released sources in a configuration management system. This can help in identifying any changes made to the original sources - often useful when updating to newer releases.

Build

The IPC product often comes with prebuilt SYS/BIOS-side libraries, so rebuilding them isn't necessary. The QNX-side libraries/binaries may also be provided prebuilt by SDK programs, but the standalone IPC release does not currently pre-build them.

IPC provides GNU makefile(s) to rebuild all its libraries at the base of the product, details are below.

NOTE

GNU make version 3.81 or greater is required. The XDC tools (provided with most SDKs and CCS distributions) includes a pre-compiled version of GNU make 3.81 in $(XDC_INSTALL_DIR)/gmake.

products.mak

IPC contains a **products.mak** file at the root of the product that specifies the necessary paths and options to build IPC for the various OS support.

Edit **products.mak** and set the following variables:

- QNX
  - **QNX_INSTALL_DIR** - Path to your QNX installation directory.
  - **DESTDIR** - Path to which target binaries will be exported when running the 'make install' goal.
  - **DEVICE** - which device are you building for.
- SYS/BIOS

- **XDC_INSTALL_DIR** - Path to TI's XDCTools installation
- **BIOS_INSTALL_DIR** - Path to TI's SYS/BIOS installation
- **ti.targets.<device target and file format>** - Path to TI toolchain for the device.
- **gnu.targets.arm.<device target and file format>** - Path to GNU toolchain for the device.
  - Set only the variables to the targets your device supports to minimize build time.

NOTE

The dependencies applicable for each supported device can be found in the IPC Release Notes provided in the product.

ipc-qnx.mak

The QNX-side build is performed using QNX makefiles. To build using the components paths set in the **products.mak** file, issue the following command:

```
<buildhost> make -f ipc-qnx.mak all
```

ipc-bios.mak

The SYS/BIOS-side IPC is built with a GNU makefile. After editing **products.mak**, issue the following command:

```
<buildhost> make -f ipc-bios.mak all
```

Based on the number of targets you're building for, this may take some time.

Note for Windows users: If you are building with a Windows host machine and it has the QNX tools installed, you will instead need to run the following in a separate command prompt window (cmd.exe) to build the SYS/BIOS side outside of the QNX build environment:

```
<buildhost> set
PATH=C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem
<buildhost> <XDC_INSTALL_DIR>\gmake -f ipc-bios.mak all
```

where <XDC_INSTALL_DIR> should be replaced with the installation directory of your XDC tools, same as the path you have used in products.mak.

Run

The IPC product provides a way to install (copy) the necessary IPC executables and libraries onto the device's target file-system to simplify the execution of the applications. The details can vary across OS's, so this description has been separated into OS-specific sections.

Configuring the BSP (OMAP5432 uEVM)

Some of the provided IPC tests that use the SharedMemoryAllocator require a carveout to be created in the QNX-owned memory. To reserve this memory, you must make the following change in the file bsp-TI-OMAP5432-uEVM-src\src\hardware\startup\boards\omap5432uevm\build in the QNX BSP:

```
   startup-omap5432uevm -r 0xBA300000,0x5A00000 -vvvvv -P2 -W
```

Save the file then rebuild the QNX OS image (ifs-omap5432-uevm.bin) and replace your existing one with the new one.

Installing Tests in QNX

To assemble the IPC resource manager, shared libraries and test executables into a directory structure suitable for running on the device's file-system, issue the following command in the IPC_INSTALL_DIR directory:

```
buildhost$ make -f ipc-qnx.mak install
```

This will install the binaries into the directory specified by DESTDIR in products.mak. It this assumed that DESTDIR is a directory visible to the target filesystem. If not, you should copy its contents to such a location (e.g. onto an SD card that can be accessed by the EVM).

When building in Windows, some users might get build messages that report a version mismatch in cygwin:

```
C:/QNX650/host/win32/x86/usr/bin/make -j 1 -Cle.v7 -fMakefile install
      1 [main] ? (5984)
C:\QNX650\host\win32\x86\usr\photon\bin\find.exe: *** fa
tal error - system shared memory version mismatch detected -
0x8A88009C/0x2D1E009C.
This problem is probably due to using incompatible versions of the
cygwin DLL.
Search for cygwin1.dll using the Windows Start->Find/Search facility
and delete all but the most recent version.  The most recent version
*should*
reside in x:\cygwin\bin, where 'x' is the drive on which you have
installed the cygwin distribution.  Rebooting is also suggested if you
are unable to find another cygwin DLL.
```

Based on what we observed the binaries are still exported correctly despite the messages. If you do want to eliminate them, you should replace the file cygwin1.dll in <QNX_INSTALL_DIR>\host\win32\x86\usr\photon\bin with the newest cygwin1.dll you can find on your host machine (do a search on your PC's filesystem in Windows).

Some of the tests rely on corresponding remote core applications to be run on the slave processor(s). The remote processor's applications are loaded when launching the resource manager. See section #IPC_resource_manager for details on launching the resource manager.

The location of the remote core applications within the IPC product varies based on device.

OMAP5432 uEVM remote core applications

The OMAP5432 remote core applications can be found in <IPC_INSTALL_DIR>/packages/ti/ipc/tests/bin/ti_platform_omap54xx_* directories.

For example, copy the messageq_single.xem3 onto the devices target filesystem into the **bin** directory:

```
buildhost$ cp
packages/ti/ipc/tests/bin/ti_platform_omap54xx_ipu/messageq_single.xem3
 <DESTDIR>/armle-v7/bin
```

IPC resource manager

Much of the functionality of IPC is provided by the resource manager. It can be launched as follows:

```
target# cd <target directory corresponding to
DESTDIR>/armle-v7/bin
target# export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<target directory
corresponding to DESTDIR>/armle-v7/usr/lib
target# syslink -f <IPU remote excutable>
```

When you are done running applications that use IPC and no longer need the resource manager, it can be terminated as follows:

```
target# cd <target directory corresponding to
DESTDIR>/armle-v7/bin
target# slay syslink
```

Running Test Applications

The QNX-side of the test applications are already on the target's filesystem in <target directory corresponding to DESTDIR>/armle-v7/bin and <target directory corresponding to DESTDIR>/armle-v7/bin/tests, assuming the #Installing Tests in QNX and #IPC resource manager sections have been followed and that the resource manager has loaded the remote core(s) with the executable corresponding to the test you'd like to run.

To find out the syntax to use for running the test (say MessageQApp), run

```
target# cd <target directory corresponding to
DESTDIR>/armle-v7/bin/tests
target# use MessageQApp
```

To run a test application, execute it on the target's filesystem:

```
target# cd <target directory corresponding to
DESTDIR>/armle-v7/bin/tests
target# ./MessageQApp 10
```

Here is a list of the main tests that are available in the IPC 3.00.00.14_eng release:

- MessageQApp: Test that creates a single thread that sends messages from host to IPU using MessageQ
  - messageq_single.xem3 needs to be loaded by the resource manager
- MessageQMulti: Test that creates multiple threads which send messages from host to IPU using MessageQ
  - messageq_multi.xem3 needs to be loaded by the resource manager
- mmrpc_test: Test that exercises MMRPC
  - test_omx_ipu_omap5.xem3 needs to be loaded by the resource manager
  - Aside from the IPC resource manager, this test also needs the shmemallocator resource manager to be launched beforehand:

```
     target# cd <target directory corresponding to
DESTDIR>/armle-v7/bin
     target# shmemallocator
     target# cd tests
     target# mmrpc_test
```

OMAP5432 uEVM

The expected output on the QNX-side should be:

```
Using numLoops: 10; procId : 1
Entered MessageQApp_execute
Local MessageQId: 0x1
Remote queueId  [0x10000]
Exchanging 10 messages with remote processor CORE0...
MessageQ_get #0 Msg = 0x11c9f0
Exchanged 1 messages with remote processor CORE0
MessageQ_get #1 Msg = 0x11c9f0
Exchanged 2 messages with remote processor CORE0
MessageQ_get #2 Msg = 0x11c9f0
...
...
Exchanged 9 messages with remote processor CORE0
MessageQ_get #9 Msg = 0x11c9f0
Exchanged 10 messages with remote processor CORE0
Sample application successfully completed!
Leaving MessageQApp_execute
```

The output on the remote processor, can be obtained by running the following on the target filesystem:

```
target# cat /dev/syslink-trace1
```

The expected output on the remote processor should be:

```
[0][      0.000] 16 Resource entries at 0x3000
[0][      0.000] messageq_single.c:main: MultiProc id = 1
[0][      0.000] [t=0x006c565d] ti.ipc.transports.TransportVirtioSetup:
 TransportVirtio
Setup_attach: remoteProcId: 0
[0][      0.000] registering rpmsg-proto:rpmsg-proto service on 61 with
 HOST
[0][      0.000] [t=0x0072625b] ti.ipc.rpmsg.MessageQCopy:
MessageQCopy_create: endPt c
reated: 61
[0][      0.000] [t=0x0073e8d9] ti.ipc.rpmsg.MessageQCopy:
callback_availBufReady: virt
Queue_toHost kicked
[0][      0.000] [t=0x00753771] ti.ipc.rpmsg.MessageQCopy:
callback_availBufReady: virt
Queue_fromHost kicked
[0][      0.000] [t=0x0076cb49] ti.ipc.rpmsg.MessageQCopy:
MessageQCopy_swiFxn:
[0][      0.000]        Received msg: from: 0x5a, to: 0x35, dataLen: 72
[0][      0.000] [t=0x007872e9] ti.ipc.rpmsg.MessageQCopy:
MessageQCopy_send: no object
for endpoint: 53
[0][      0.000] tsk1Fxn: created MessageQ: SLAVE_CORE0; QueueID:
```

```
0x10000
[0][      0.000] Awaiting sync message from host...
[0][     51.992] [t=0x0c475268] ti.ipc.rpmsg.MessageQCopy:
callback_availBufReady: virt
Queue_fromHost kicked
[0][     51.992] [t=0x0c48eb28] ti.ipc.rpmsg.MessageQCopy:
MessageQCopy_swiFxn:
[0][     51.993]        Received msg: from: 0x400, to: 0x3d, dataLen:
176
[0][     51.993] [t=0x0c4ad220] ti.ipc.rpmsg.MessageQCopy:
MessageQCopy_send: calling c
allback with data len: 176, from: 1024
[0][     51.993]
[0][     52.995] [t=0x0c873ded] ti.ipc.rpmsg.MessageQCopy:
callback_availBufReady: virt
Queue_fromHost kicked
[0][     52.996] [t=0x0c88b029] ti.ipc.rpmsg.MessageQCopy:
MessageQCopy_swiFxn:
[0][     52.996]        Received msg: from: 0x406, to: 0x3d, dataLen:
40
[0][     52.996] [t=0x0c8a8a87] ti.ipc.rpmsg.MessageQCopy:
MessageQCopy_send: calling c
allback with data len: 40, from: 1030
[0][     52.996]
[0][     52.996] Received msg from (procId:remoteQueueId): 0x0:0x1
[0][     52.996]        payload: 8 bytes; loops: 10 with printing.
[0][     52.997] [t=0x0c8eab7e] ti.ipc.rpmsg.MessageQCopy:
callback_availBufReady: virt
Queue_fromHost kicked
[0][     52.997] [t=0x0c9031bc] ti.ipc.rpmsg.MessageQCopy:
MessageQCopy_swiFxn:
[0][     52.997]        Received msg: from: 0x406, to: 0x3d, dataLen:
40
[0][     52.997] [t=0x0c9208fa] ti.ipc.rpmsg.MessageQCopy:
MessageQCopy_send: calling c
allback with data len: 40, from: 1030
[0][     52.997]
[0][     52.997] Got msg #0 (40 bytes) from procId 0
[0][     52.997] Sending msg Id #0 to procId 0
[0][     52.998] [t=0x0c959f33] ti.ipc.rpmsg.MessageQCopy:
callback_availBufReady: virt
Queue_fromHost kicked
[0][     52.998] [t=0x0c971df7] ti.ipc.rpmsg.MessageQCopy:
MessageQCopy_swiFxn:
[0][     52.998]        Received msg: from: 0x406, to: 0x3d, dataLen:
40
[0][     52.998] [t=0x0c98f3e7] ti.ipc.rpmsg.MessageQCopy:
```

```
MessageQCopy_send: calling c
allback with data len: 40, from: 1030
[0][     52.998]
[0][     52.999] Got msg #1 (40 bytes) from procId 0
[0][     52.999] Sending msg Id #1 to procId 0
[0][     52.999] [t=0x0c9c7a00] ti.ipc.rpmsg.MessageQCopy:
callback_availBufReady: virt
Queue_fromHost kicked
[0][     52.999] [t=0x0c9df7fc] ti.ipc.rpmsg.MessageQCopy:
MessageQCopy_swiFxn:
[0][     52.999]        Received msg: from: 0x406, to: 0x3d, dataLen:
40
[0][     52.999] [t=0x0c9fce5a] ti.ipc.rpmsg.MessageQCopy:
MessageQCopy_send: calling c
allback with data len: 40, from: 1030
[0][     52.999]
[0][     53.000] Got msg #2 (40 bytes) from procId 0
[0][     53.000] Sending msg Id #2 to procId 0
[0][     53.000] [t=0x0ca36e79] ti.ipc.rpmsg.MessageQCopy:
callback_availBufReady: virt
Queue_fromHost kicked
[0][     53.000] [t=0x0ca4ea95] ti.ipc.rpmsg.MessageQCopy:
MessageQCopy_swiFxn:
[0][     53.000]        Received msg: from: 0x406, to: 0x3d, dataLen:
40
[0][     53.001] [t=0x0ca6c975] ti.ipc.rpmsg.MessageQCopy:
MessageQCopy_send: calling c
allback with data len: 40, from: 1030
[0][     53.001]
[0][     53.001] Got msg #3 (40 bytes) from procId 0
[0][     53.001] Sending msg Id #3 to procId 0
...
...
[0][     53.007] Got msg #8 (40 bytes) from procId 0
[0][     53.007] Sending msg Id #8 to procId 0
[0][     53.007] [t=0x0cccd3d7] ti.ipc.rpmsg.MessageQCopy:
callback_availBufReady: virt
Queue_fromHost kicked
[0][     53.007] [t=0x0cce50ed] ti.ipc.rpmsg.MessageQCopy:
MessageQCopy_swiFxn:
[0][     53.007]        Received msg: from: 0x406, to: 0x3d, dataLen:
40
[0][     53.007] [t=0x0cd027bd] ti.ipc.rpmsg.MessageQCopy:
MessageQCopy_send: calling c
allback with data len: 40, from: 1030
[0][     53.007]
[0][     53.008] Got msg #9 (40 bytes) from procId 0
```

```
[0][     53.008] Sending msg Id #9 to procId 0
[0][     53.008] Awaiting sync message from host...
```

Article Sources and Contributors

IPC Install Guide QNX *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?oldid=159717 *Contributors*: ChrisRing, VincentWan