iMX6 QNX

iMX6 QNX ARM: Prerequisites

Hardware

CGI Studio has been tested on Freescale i.MX6Q Sabre Board for Smart Devices

QNX Board Support Package (BSP)

QNX Neutrino for: BSP_freescale-imx6q-sabresmart_br-660_be-660 SVN Revision Number: 773534 Build Number: 313

For documentation related to this BSP please refer to:

http://community.qnx.com/sf/wiki/do/viewPage/projects.bsp/wiki/BSPAndDrivers

CMake

In order to generate makefiles for building Candera applications, the Meta build system CMake needs to be installed with at least version 3.5.2.

Running the Player Demo Application on iMX6 QNX

iMX6 System Preparation

For documentation related to how to build the BSP and prepare the bootable SDCard, please refer to the BSP User Guide:

http://community.qnx.com/sf/frs/do/viewRelease/projects.bsp/frs.freescale_i_mx6q_sabre_board_for.sdp_6_6_bsp_user_guide_for_sabre

Executing Target Player from the SD Card

The application and asset can be copied into the folder /usr/bin on the SD-Card with the target root file system, and executed like this:

```
Welcome to QNX Neutrino Initial Program Loader for Freescale i.MX6Q Sabre-Smart (ARM Cortex-A9 M
SDMMC download...
load image done.
Found image               @ 0x18000008
Jumping to startup        @ 0x10807364
ÿWelcome to QNX Neutrino 6.6.0 on the i.mx6 Smart-Device (ARM Cortex-A9 MPCore)
Starting watchdog...

Starting SD3 memory card driver...
Starting Ethernet driver
Starting screen
Running startup script...
# Mempool Map addr range[28100000-37f00000]
Mempool Map paddr range[3fd00000-4fb00000]
[Interrupt] Attached irqLine 41 with id 5.
[Interrupt] Attached irqLine 42 with id 6.
[Interrupt] Attached irqLine 43 with id 7.
Attached resmgr to /dev/galcore with id:1.

# cd /usr/bin
# CGIApplication_ARM_QNX_iMX6_Target_GCC iMX6_GettingStarted_3D_GLES20.bin

Starting standard application ...
```

Building Own Application for iMX6 QNX

Interfaces

The Feat-Standard public interface is covered by the namespaces FeatStd and various subnamespaces of namespace FeatStd. All namespaces containing either **Internal** or **Private** in the name cover private namespaces, which are not recommended to be used in any application, as it is not guaranteed that these interfaces remain stable respectively exist in any future release. Changes in any of these private namespaces can be applied without any (prior) notifications.

Namespace FeatStd

- Container
- Utilities

Namespace FeatStd::Diagnostics

- Diagnostics

Namespace FeatStd::MemoryManagement

- Memory Management

Namespace FeatStd::PerfMon

- MonitorPerformanceReporting

Namespace FeatStd::Platform

This interface defines the access to platform dependent objects. The implementation of this platform dependent code is not part of FeatStd, though there are a few reference platforms, like SimulationPlatform (implemented basically for Win32) or GenericPlatform (providing generic respectively stub implementations), which can used as templates for the customer.

This platform interface can be seen as an official porting interface of FeatStd to any customer specific platform setup, which is the reason for outlining it in the separate namespace FeatStd::Platform.

- Platform