Getting Started with CARMA DevKit

Revision History

Revision	Date	Author	Changes
1.0	05th October 2012	DC	First release

INDEX

1 Overview

CARMA development kit is the first ARM-based platform that supports CUDA; it is based on NVIDIA Tegra 3 SoC (4 Cortex-A9 cores, 2GB RAM) and NVIDIA Quadro1000M GPU (GF108 with 2 GB of memory, 96 CUDA cores, compute capability 2.1).

The system allows the user to run CUDA applications providing the power efficiency of this innovative architecture.

It is a development platform, not intended for production use.

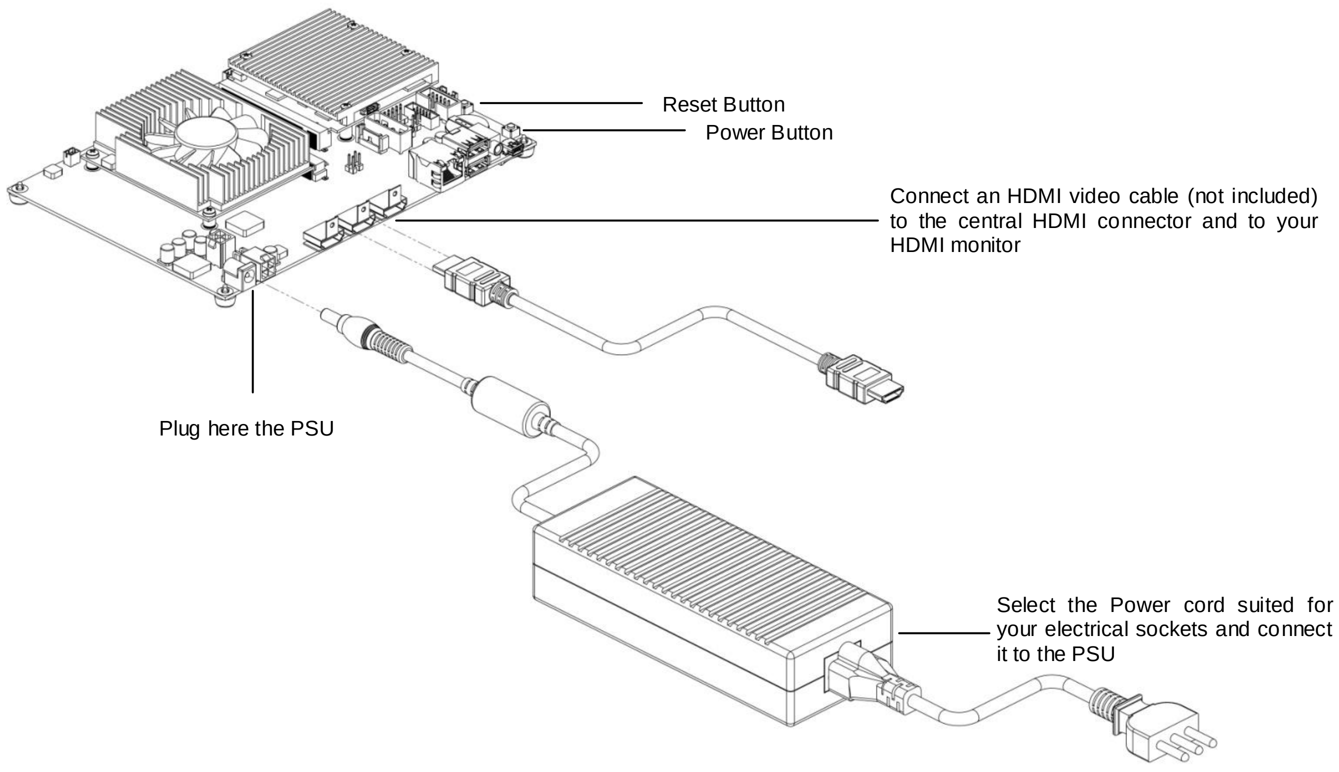
Additional information, FAQs and technical documents can be found at www.seco.com/carmakit.

2 Hardware Quick Start

The CARMA Development kit is supplied with the electronic boards already mounted and fixed on the Carrier Board, so you simply need to plug the needed cables to let the system become operative.

Connect all the peripherals needed before connecting the power adapter and/or powering on the system.

- Connect the system to an HDMI monitor using an HDMI video cable (not supplied with the system). Connect it to the system's **central** HDMI connector.
- Connect an USB Keyboard and a mouse using the two USB ports available on the front side of the CARMA Carrier board
- Plug the power jack of the Power Supply Unit (PSU, included in the kit) to the matching plug on CARMA. Connect the Power Cord suited for your electrical sockets to the PSU, and then plug it to the electrical rail.


Reset Button
Power Button
Connect an HDMI video cable (not included) to the central HDMI connector and to your HDMI monitor
Plug here the PSU
Select the Power cord suited for your electrical sockets and connect it to the PSU

CARMA Kit is factory configured with a basic Ubuntu 11.04 Linux OS and includes pre-compiled CUDA applications. When you plug the power jack of the PSU, the system will automatically power up and will present a login screen.

The system is configured to automatically login the default user '*ubuntu*' after a 15 seconds delay.

Preinstalled configuration provides lightDM as display manager, and the only preconfigured user (login name '*ubuntu*', password '*ubuntu*') uses Openbox as window manager.

Upon power up, the user can run CUDA pre-compiled examples, which are stored in */home/ubuntu/CUDA_samples* folder.

The O.S. installed is in a minimal configuration in order to leave the maximum space free for local applications and data. Most users will want to enhance the installation to provide a more complete environment.

For this reason, CARMA Kit provides Gigabit Ethernet connection, which comes preconfigured to request an IP address using DHCP.

3 Compiling CUDA applications for CARMA

The developer needs to cross-compile from a Linux x86 machine, as CUDA toolkit for ARM architecture is available for x86 platforms running Linux; using one of these platforms, the user can cross-compile the applications to be used with CARMA Kit.

3.1 Development system setup

1) Start with a clean installation of Ubuntu Desktop:

User must have a X86_64 Linux development machine, running Ubuntu 11.04, with GCC version 4.5.X. You can get an Ubuntu clean image of the OS available at http://releases.ubuntu.com/natty/

2) Installing ARM compilers

In order to perform cross compilation, a gcc ARM cross compiler must be present. This compiler will be invoked by nvcc as part of the cross compilation. Currently, only gcc 4.5.X is supported.

The following command, when executed on the development machine, can be used to obtain a gcc 4.5.X cross compiler for ARM:

```
sudo apt-get install gcc-4.5-arm-linux-gnueabi g++-4.5-arm-linux-gnueabi
```

The default cross compiler files folders will be */usr/bin/arm-linux-gnueabi-gcc* and */usr/bin/arm-linux-gnueabi-g++*.

3) Installing the CUDA toolkit

The packaged toolkit for CARMA is available at www.seco.com/carmakit in the download section.

Once you have downloaded it, execute the *.run* file on the development machine. Follow the instructions given by the installer.

4) Adding Libraries on the CARMA machine

If you need to use other libraries for ARM, you will also need to copy the libraries and corresponding header files from CARMA to the x86 PC. You can place them under */usr/local/arm_lib* and */usr/local/arm_include* or you can just put them under */usr/local/cuda/lib* and */usr/local/cuda/include*.

5) Cross Compiling

In order to let *nvcc* compiling correctly, the following command line options need to be passed to every *nvcc* invocation:

-target-cpu-arch=ARM

This is required to signal cross compilation

-ccbin=<location of g++ for ARM executable>

this is required to signal which host compiler *nvcc* should use. If during installation you used standard directories, this parameter is */usr/bin/arm-linux-gnueabi-g++-4.5*.

-m32

this is required only when the toolkit is 64-bit. It signals that the target is a 32-bit platform.