ISEE IGEPv2 BOARD

IGEPv2 BOARD
SDK USER MANUAL
(Revision 1.03)

ISEE
Integration Software and Electronics Engineering

CONTENTS

0   COPYRIGHT NOTICE

VERSION CONTROL

REVISION	DATE	DESCRIPTION
1.00	19/03/2009	Initial version
1.01	17/07/2009	Revision
1.02	21/12/2009	Upgrade to igep0020b + sgx demo
1.03	29/01/2010	See changelog

1 PREFACE

1.1 VERY QUICK START GUIDE

Run IGEPv2 board

1. Get your IGEPv2 board (http://www.myigep.com)

2. Connect peripheries (monitor and keyboard) and plug power jack

3. Wait 1 minute for boot and prompt login

4. Enter default user and password:

> USERNAME: **root**
>
> PASSWORD: **letmein**

Connect to IGEPv2 board from your workstation via RS232 serial interface

5. Plug ID9 to DB9 serial RS232 cable with your serial port

6. Open serial terminal program (like minicom, hyperterminal, …)

Alternative connect to IGEPv2 board from your workstation

7. Plug Ethernet LAN cable

8. Open secure shell console on IGEPv2 default IP address:

> **192.168.254.254**

1.2 ORGANIZATION OF THE MANUAL

This manual is divided into 5 parts.

First part introduces basic concepts and useful links about IGEPv2 board environment and this manual.

Second part is a short description about IGEPv2 user hardware interface. If you would known about IGEPv2 hardware board details you should read "IGEPv2 BOARD HARDWARE MANUAL" document (public download from IGEP website).

Third part is main content of this document. It explains all details about IGEPv2 SDK (Install process, Virtual Machine format, contents, toolchain, rootfs, packages, helpers …)

Fourth part is several tricks and useful information around IGEPv2 SDK and board interconnection

Fifth part introduces about IGEPv2 support

1.3 MYIGEP ECOSYSTEM

Connect to website http://www.myigep.com

1.4 USEFUL WEB LINKS AND EMAILS

ISEE IGEP platform web site: http://www.myigep.com

ISEE IGEP wiki: http://wiki.myigep.com/trac/wiki

ISEE IGEP forum: http://wiki.myigep.com/ofurm

ISEE website: http://www.igep-platform.com

ISEE shop: http://www.igep-platform.com/shop

Other interesting links:

http://www.vmware.com/products/player/

http://www.7-zip.org/

http://www.codeblocks.org/

http://www.putty.org/

http://www.mantisbt.org/

1.5 USER REGISTRATION

Users, who want to have full access into IGEP platform web services, have to be free registered.

Goto ISEE website: http://www.igep-platform.com → Support

After that, you will receive the password access in a short time.

Users can use the same username and password to access into:

- IGEP web site

- IGEP virtual market

- Mantis Bug Tracker

- IGEP public forum

2 INTRODUCING IGEPV2 BOARD

2.1 IGEPV2 BOARD SDK FEATURES

ISEE provides customers with SDK built-in virtual machine box. Every piece of software is included, installed and configured to play and fun with our ISEE IGEPv2 BOARD. These easy deploy form speed up getting starter with ISEE IGEPv2 BOARD environment.
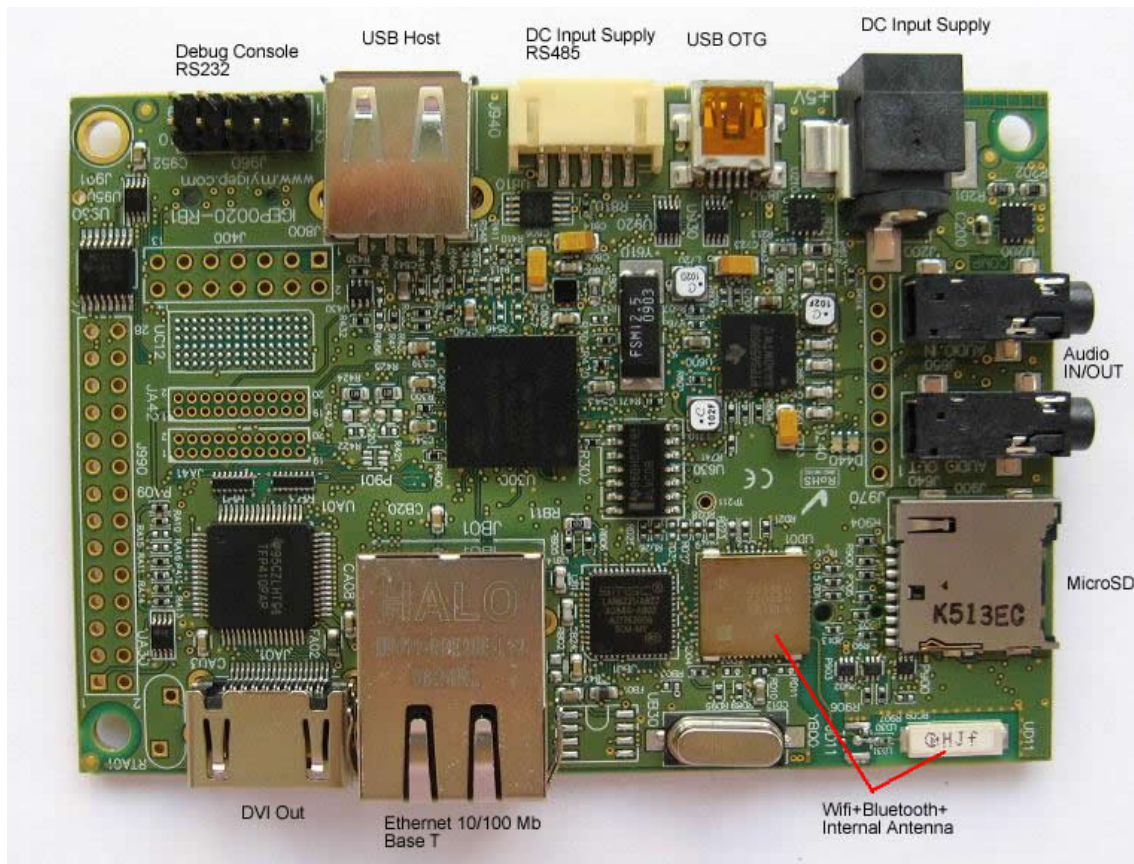
Figure 1 ISEE IGEPv2 board features

> **NOTE:**
>
> **You should read first "IGEPv2 BOARD HARDWARE MANUAL" document.**
>
> Download it from www.myigep.com

2.2 GET YOUR IGEPV2 BOARD POWER ON

1. Connect the DVI cable connector from the IGEPv2 DVI connector to the TFT DVI-D connector.

The Digital Visual Interface (DVI) is a video interface standard designed to provide very high visual quality on digital display devices such as flat panel LCD computer displays and digital projectors. It is partially compatible with the High-Definition Multimedia Interface (HDMI) standard in digital mode (DVI-D), and VGA in analog mode (DVI-A).

User will need a cable with male DVI-D connector for the TFT, and male HDMI connector from IGEPv2 Board.

Figure 2 DVI cable

Figure 4 TFT monitor example
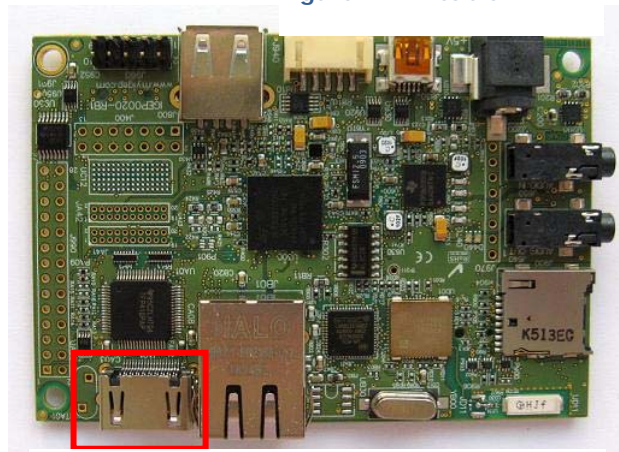
Figure 3 IGEP2 Board DVI connector

Figure 5 DVI-D Connector TFT detail example

2. Connect the stereo audio output from the IGEPv2 board to the TV Audio input.

Figure 7 Stereo audio cable
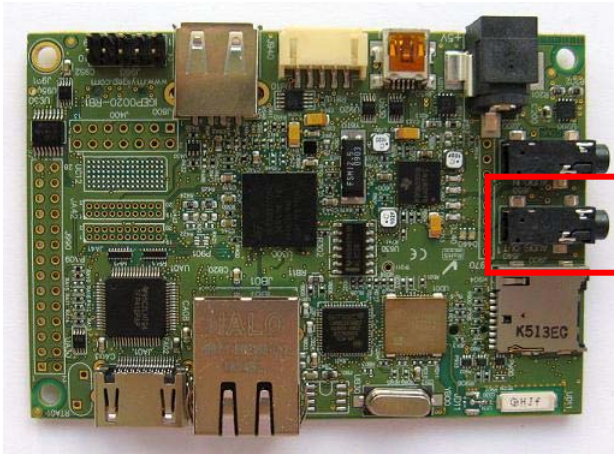
Figure 6 IGEPV2 Board stereo audio output connector

Figure 8 Stereo input Connector TFT detail example

3. Connect the Ethernet cable
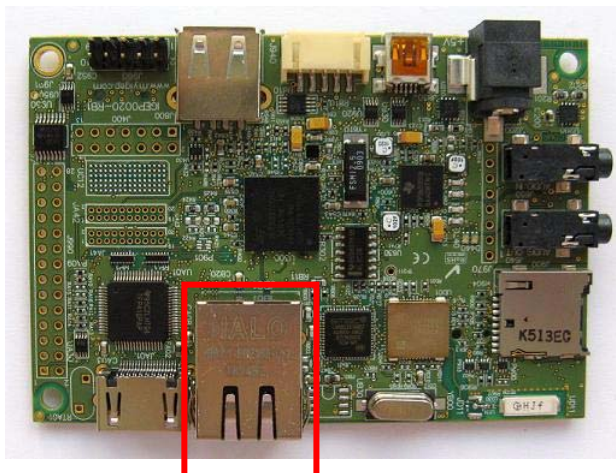
Figure 10 LAN cable

Figure 9 IGEPv2 Board Ethernet connector

4. Connect the USB keyboard and USB Mouse.

Figure 11 USB mouse

Figure 12 USB keyboard

Note: Only USB 2.0 devices work on IGEPV2 USB host connector.

a. Option1: using the IGEPV2 USB host connector and a USB hub 2.0.

Figure 13 USB hub
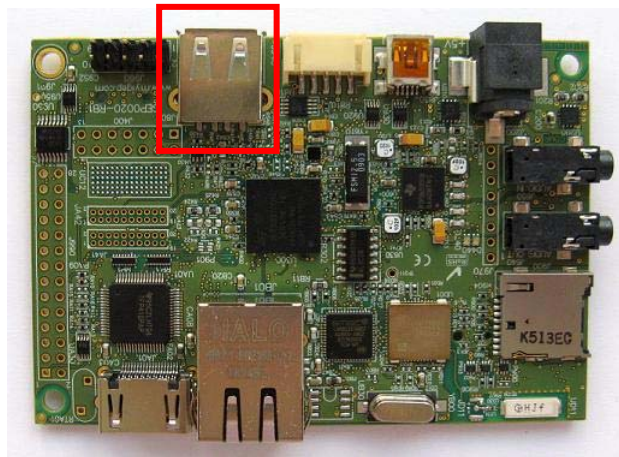
Figure 14 IGEPv2 Board USB host

Note: Do not connect a USB mouse 1.0 into the USB host connector without using a USB hub 2.0, because it will not work !!!

b. Option2: using the IGEPv2 USB mini OTG connector, a miniOTG to USB adapter and a USB hub 1.0 or 2.0.
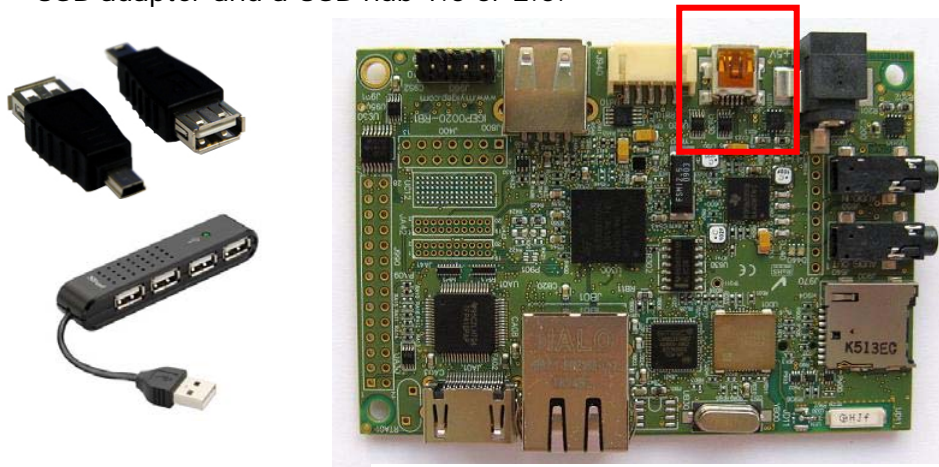
Figure 15 IGEPv2 board USB OTG

5. Power the board
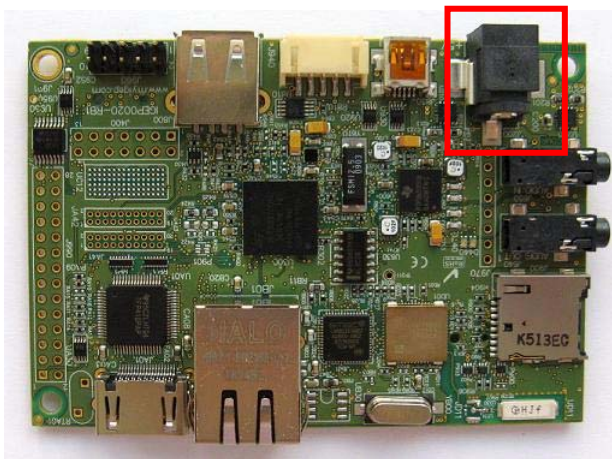
Figure 17 IGEPv2 board power jack

Figure 16 Power supply AC/DC adaptor

3   IGEP V2 SDK VIRTUAL MACHINE

3.1   WHY SDK IN A VIRTUAL MACHINE?

We use a virtual machine to easy IGEP SDK install and operating system independent, and machine independent. It is based on VMware Player (http://www.vmware.com/products/player/). It is not open source but it is freeware.

VMware Player runs virtual machines on your Windows or Linux PC. This free desktop virtualization software application makes it easy to operate any virtual machine created by VMware Workstation, VMware Fusion, VMware Server or VMware ESX, as well as Microsoft Virtual Server virtual machines or Microsoft Virtual PC virtual machines. You can also use Player to evaluate one of the many virtual appliances available from the VMware Virtual Appliance Marketplace.

- Run multiple operating systems simultaneously on a single PC

- Experience the benefits of preconfigured products without any installation or configuration hassles

- Share data between host computer and virtual machine

3.2   DOWNLOAD SDK VIRTUAL MACHINE

To download the SDK virtual machine, first you have to be registered on IGEP web site: www.myigep.com (chapter 1.5 User registration).

If so, then go to ISEE IGEP web site: http://www.myigep.com, and login with your username and password.

Press on Support button.

Press on IGEP Software button.

A new window will appear with the links of all IGEP platforms virtual machines.

Select the virtual machine you wish to download, and saved it in your PC.

Take care that all IGEP virtual machines are compressed with 7-zip compressor. We have used 7-zip compressor because of its high rate compression.

7-Zip is open source software. Most of the source code
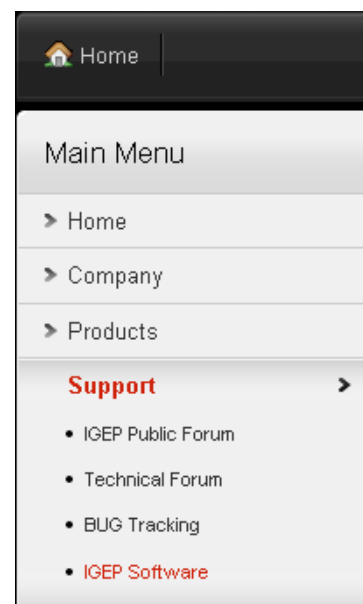
is under the GNU LGPL license. You can download a free version of 7z from: http://www.7-zip.org/

Uncompress the Vmware Image file.

Execute your vm player, press on button Open and select the *.vmx file you will find in the uncompressed directory.

Figure 18 Virtual machine player from Vmware

If everything goes ok, you have to see a Desktop like that:
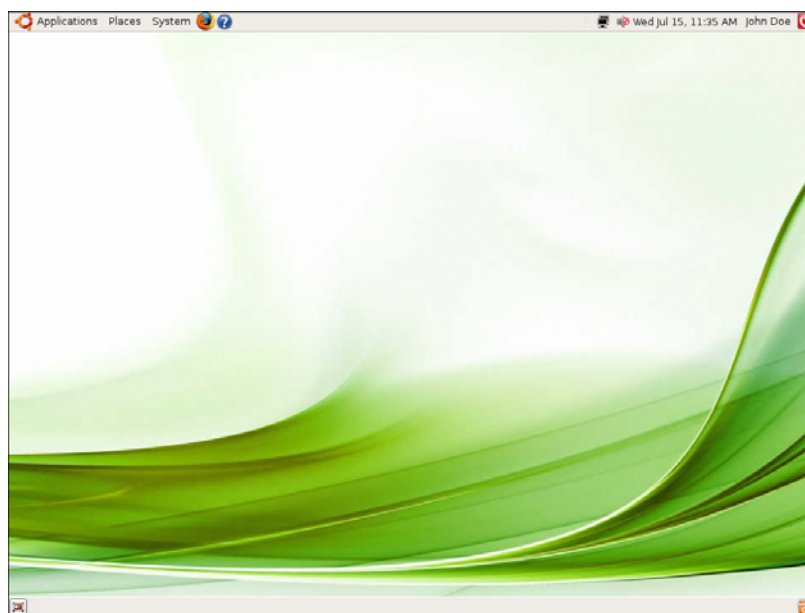
Figure 19 Ubuntu 8.04 IGEPv2
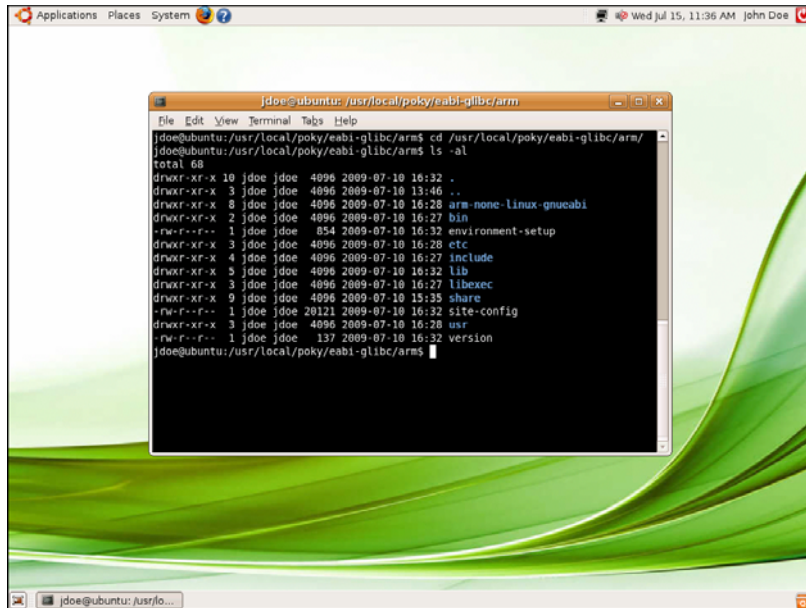
Figure 20 Ubuntu 8.04 IGEPV2 Console window

3.3 VIRTUAL IMAGE DETAILS

Ubuntu base release information:

Distribution: Ubuntu Hardy Heron

Version: 8.04

Edition: Desktop, x86

URL: http://releases.ubuntu.com/hardy/

Account information

Username: **jdoe**

Password: **letmein**

3.4 IGEPV2 SDK GENERAL SPECIFICATION

Description	Characteristics
GCC	gcc version 4.3.2 (GCC)
Libc	libc version 2.6.1
Linux	2.6.28
Rootfs (root filesystem)	Open Embeded based (opkg package manager)

The following paragraphs provide more detail on each feature and components.

GCC

GNU Compiler Collection (http://gcc.gnu.org), 4.3.2 version and "*arm-none-linux-gnueabi-*" architecture.

The GNU Compiler Collection includes front ends for C, C++, Objective-C, Fortran, Java, and Ada, as well as libraries for these languages (libstdc++, libgcj,…).

GCC development is a part of the GNU Project, aiming to improve the compiler used in the GNU system including the GNU/Linux variant. The GCC development effort uses an open development environment and supports many other platforms in order to foster a world-class optimizing compiler, to attract a larger team of developers, to ensure that GCC and the GNU system work on multiple architectures and diverse environments, and to more thoroughly test and extend the features of GCC.

Libc

GNU libc library (http://www.gnu.org/software/libc/libc.html), 2.6.1 version

Any Unix-like operating system needs a C library: the library which defines the "system calls'' and other basic facilities such as open, malloc, printf, exit…

The GNU C library is used as the C library in the GNU system and most systems with the Linux kernel.

Linux

ISEE SDK runs Linux kernel 2.6.28 and their drivers for IGEPv2 BOARD hardware (http://www.kernel.org).

Linux is a clone of the operating system Unix, written from scratch by Linus Torvalds with assistance from a loosely-knit team of hackers across the Net. It aims towards POSIX and Single UNIX Specification compliance.

It has all the features you would expect in a modern fully-fledged Unix, including true multitasking, virtual memory, shared libraries, demand loading, shared copy-on-write executables, proper memory management, and multistack networking including IPv4 and IPv6.

Although originally developed first for 32-bit x86-based PCs (386 or higher), today Linux also runs on (at least) the Alpha AXP, Sun SPARC, Motorola 68000, PowerPC, ARM, Hitachi SuperH, IBM S/390, MIPS, HP PA-RISC, Intel IA-64, AMD x86-64, AXIS CRIS, Renesas M32R, Atmel AVR32, Renesas H8/300, NEC V850, Tensilica Xtensa, and Analog Devices Blackfin architectures; for many of these architectures in both 32- and 64-bit variants.

Linux is easily portable to most general-purpose 32- or 64-bit architectures as long as they have a paged memory management unit (PMMU) and a port of the GNU C compiler (gcc) (part of The GNU Compiler Collection, GCC). Linux has also been ported to a number of architectures without a PMMU, although functionality is then obviously somewhat limited. See the µClinux project for more info.

Rootfs

It is based on Open Embedded (http://www.openembedded.org/).

Open embedded allows developers to create a complete Linux Distribution for embedded systems.

Some of the OE advantages include:

- support for many hardware architectures

- multiple releases for those architectures

- tools for speeding up the process of recreating the base after changes have been made

- easy to customize

- runs on any Linux distribution

- cross-compiles 1000's of packages including GTK+, Xwindows, Mono, Java, and about anything else you might ever need

3.5 CROSS-DEVELOPMENT AND SDK

Cross-development in general refers to the overall software development process that eventually produces a single application or a complete system running on an platform that is different from the development platform. This is accomplished using a cross-compiler toolchain and cross-compiled libraries.

Peter Seebach defines cross-compilation as follows: "Cross compilation occurs when a compiler running on one system produces executables for another system -- this is an important concept when the target system doesn't have a native set of compilation tools, or when the host system is faster or has greater resources."

Cross-development usually involves two different platforms, the host platform where actual development work takes place, and the target platform where the final application is tested and run.

The IGEPv2 SDK Virtual Machine provides a Linux based desktop computer properly configured for applications development. Development host is setup with:

- A TFTP server to get kernel image for the target

- A NFS server to use a network file system for the root filesystem for the target.

- A software development kit (SDK), a set of development tools that allows a software engineer to create applications for a certain software package, software framework or hardware platform.

- A demo root filesystem for the target

- A kernel image for the target

- A free C++ IDE : Code::blocks 8.02

If user needs to install more development software, just go to the Synaptic Package Manager.
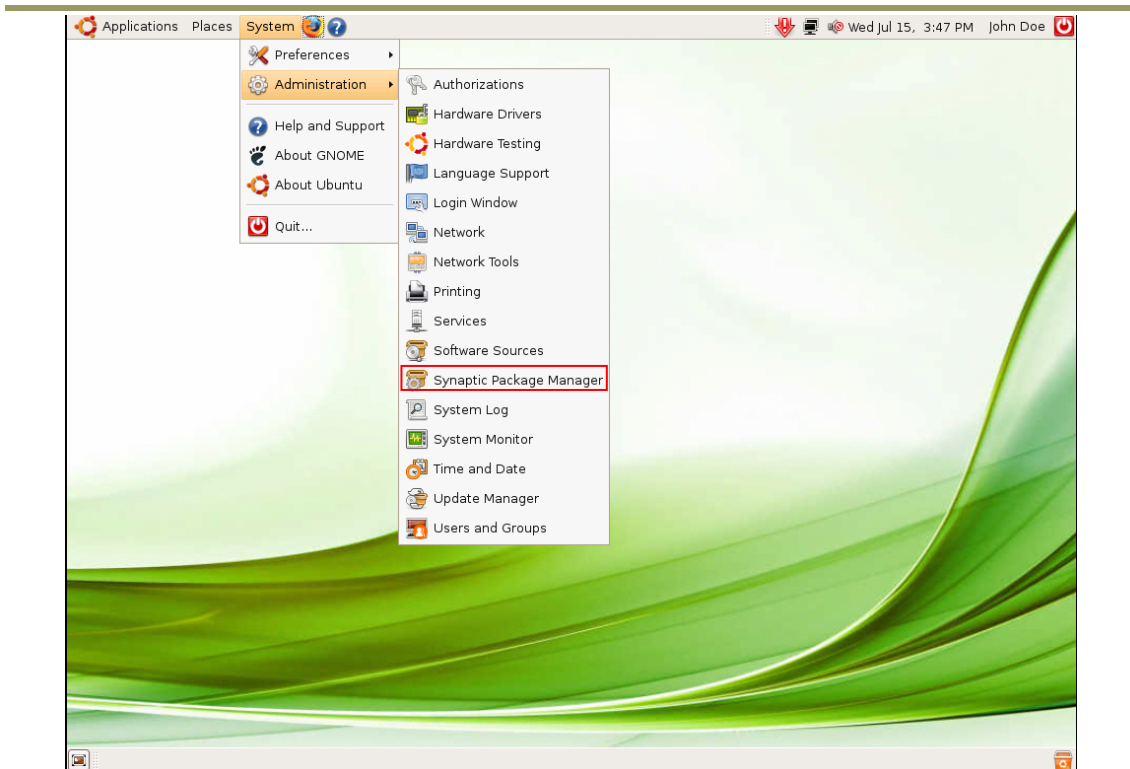
Figure 21 Synaptic Package Manager Menu

Synaptic is a graphical package management program for apt. It provides the same features as the apt-get command line utility with a GUI front-end based on Gtk+.

Features:

- Install, remove, upgrade and downgrade single and multiple packages.

- Upgrade your whole system.

- Manage package repositories (sources.list).

- Find packages by name, description and several other attributes.

- Browse all available online documentation related to a package.

- Download the latest changelog of a package.

- Lock packages to the current version.

- Force the installation of a specific package version.

- Undo/Redo of selections.

- Built-in terminal emulator for the package manager.

- And select the packages you wish to install.

…
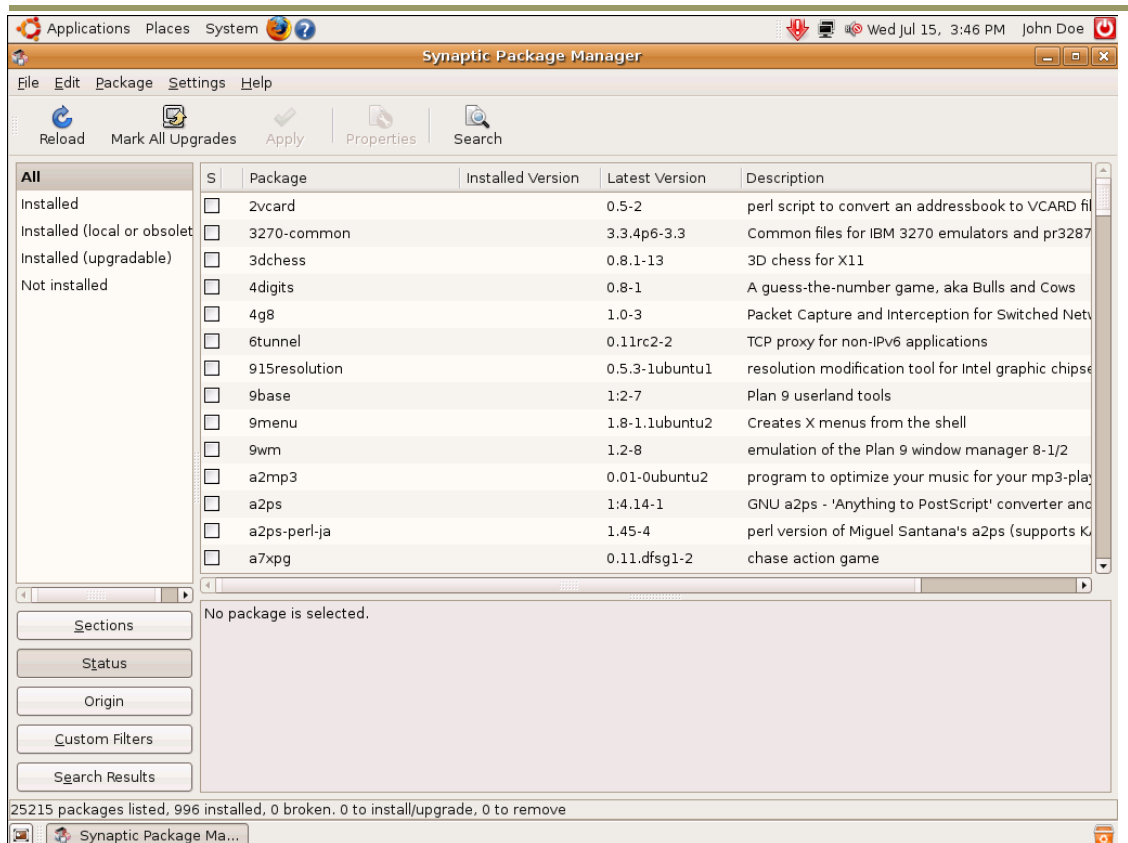
Figure 22 Synaptic Package Manager program

You can also use the apt-get command line program to install and remove software packages from your system (one at a time or many at once), as well as update your system's internal listing of what packages are available from your currently selected list or repositories.

to install a package:

```
# apt-get install NameOfPackage
```

to reinstall a package

```
# apt-get install --reinstall NameOfPackage
```

to remove a package:

```
# apt-get remove NameOfPackage
```

to remove a package and configuration file:

```
# apt-get --purge remove NameOfPackage
```

to search for a package:

```
# apt-cache search NameOfPackage
```

to update the repository (list of avalible .deb's):

```
# apt-get update
```

to upgrade your system (can be useful in maintaining an up-to-date system):

```
# apt-get upgrade
```

to see a short list of common commands:

```
# apt-get --help
```

3.6 FIRST STEPS WITH IGEPV2 SDK VIRTUAL MACHINE

Virtual Machine emphasizes "simple" and does not require you to spend few hours trying to set up a development environment.

During development, the target system can NFS-mount its root filesystem from your file server to provide a complete diskless Linux system. The target system will attempt to mount its root filesystem from the server as /srv/nfs/<distro>/<project>/<machine>, where

- <distro> is poky

- <project> is sato-demo

- <machine> is igep0020b

The target also can get its kernel image from your tftp server. Similar to the NFS root filesystem, the target system will attempt to get its kernel image from the server as /srv/tftp/<distro>/<project>/<machine>

IGEPv2 SDK Virtual Machine provides a demo root filesystem located into /srv/nfs/poky/sato-demo/igep0020b which can be mounted via NFS. Follow next steps to run the demo you can follow the next steps

First of all, connect your development board as explained in chapter "SETTING UP YOUR IGEPV2 BOARD"

Next, power up your board and stop u-boot's autoboot by pressing a key.

Now, it's time to change the U-Boot environment.

```
U-Boot> setenv serverip <your VM ip>

U-Boot> setenv distro poky

U-Boot> setenv machine igep0020b

U-Boot> setenv project  poky-image-sato   // poky sato

or

U-Boot> setenv project  poky-image-minimal   // poky minimal

or

U-Boot> setenv project  poky-image-demo   // poky sgx demo

U-Boot> setenv bootcmd 'run nfs-boot'
```

If you want to set as default boot option, you can also save your new environment

U-Boot> saveenv

The last step is run the boot command.

U-Boot> run bootcmd

If all is ok, something like this should appear in the monitor connected to IGEPv2 board.
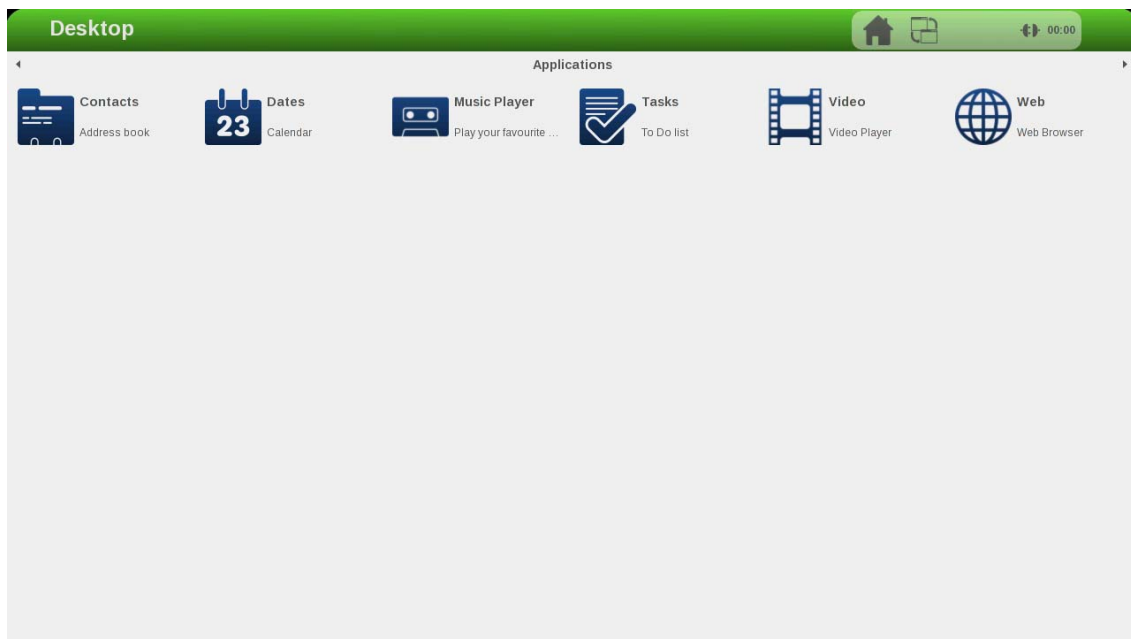
Figure 23 Poky-sato Desktop

3.7 POKY LINUX SOFTWARE DEVELOPMENT KIT

IGEPv2 SDK Virtual Machine provides a Poky Linux SDK. The Poky Linux is primarily a small linux distribution based on open source software. Poky SDK is located into the /usr/local/poky directory and contain a setup script, which can be sourced to initialise a suitable environment. After sourcing this, the compiler, a special version of pkgconfig and other useful utilities are added to the PATH. Variables to assist pkgconfig and autotools are also set.

3.7.1 ARCHITECTURE

The Poky Linux SDK will sit neatly on top of any device using the GNOME Mobile software stack, providing a well defined user experience. Poky Linux has a growing

open source community backed up by the principal developer and maintainer of Poky, OpenedHand? Ltd. The user interface environment used by Poky Linux is Sato, it is designed to work well with screens at very high DPI and restricted size. It is coded with focus on efficiency and speed so that it works smoothly on hand-held and other embedded hardware.
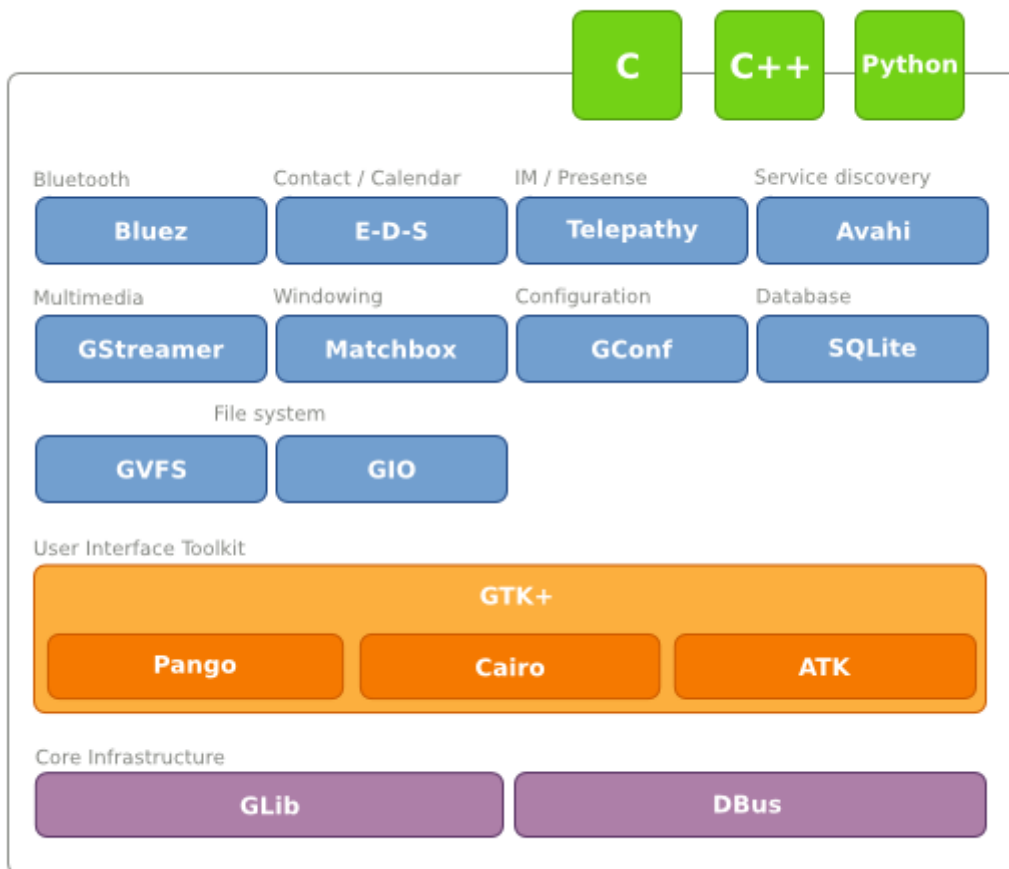
The GNOME Mobile is a diverse stack of open source, mobile application development technologies that includes the GTK toolkit for interface construction, the GConf application configuration service which leverages XML for data persistence, the extensible GnomeVFS file access abstraction layer which provides support for network transparent file manipulation, the highly flexible GStreamer multimedia framework which supports dynamic media editing as well as playback, the powerful D-Bus interprocess communication system, the BlueZ Bluetooth stack, the nascent Telepathy instant messaging and presence management framework, and the Avahi service for Zeroconf service discovery. Please refer to http://www.gnome.org/mobile for a more detailed description.

3.7.2 BUILD SOFTWARE USING THE POKY LINUX SDK

First of all you need to initialize a suitable environment, you can do this sourcing once the environment-setup script.

```
$ source /usr/local/poky/eabi-glibc/arm/environment-setup
```

Create a single .c file (hello-world.c), using your preferred editor

```
#include <stdio.h>

int main (int argc, char **argv)

{

  printf("Hello world !\n");

  return 0;

}
```

Build your program using cross-tools available into the SDK

```
$ arm-none-linux-gnuabi-gcc -o hello-world hello-world.c

$ file hello-world

hello-world: ELF 32-bit LSB executable, ARM, version 1 (SYSV), for
GNU/Linux 2.6.14, dynamically linked (uses shared libs), not stripped
```

Copy the binary result to your NFS-mounted root filesystem

```
$ sudo cp hello-world /srv/nfs/poky/poky-image-sato/igep0020a/home/root

[sudo] password for jdoe: letmein
```

And run the program in your target device (serial console)

```
# cd /home/root

# ./hello-world

Hello World !
```

Using the SDK with autotool enabled packages is straightforward, just pass the appropriate host option to configure

```
$ ./configure --target=arm-none-linux-gnueabi --host=arm-none-linux-gnueabi
```

For example, you can download the GNU Hello program and build for your device

```
$ wget http://ftp.gnu.org/gnu/hello/hello-2.4.tar.gz

$ tar xzf hello-2.4.tar.gz

$ cd hello-2.4

$ ./configure --target=arm-none-linux-gnueabi --host=arm-none-linux-gnueabi

$ make

$ file src/hello

src/hello: ELF 32-bit LSB executable, ARM, version 1 (SYSV), for GNU/Linux 2.6.14, dynamically linked (uses shared libs), not stripped
```

For makefile based projects it is usually a case of ensuring the cross tools are used e.g. CC=arm-none-linux-gnueabi-gcc and LD=arm-none-linux-gnueabi-ld.

Create a simple Makefile file for your hello-world project

```
hello-world:   hello-world.o

        $(CC) -o hello-world hello-world.o

hello-world.o: hello-world.c

        $(CC) -c hello-world.c
```

And use the make utility to automagically build your program

```
$ rm hello-world

$ make CC=arm-none-linux-gnueabi-gcc

$ file hello-world

hello-world: ELF 32-bit LSB executable, ARM, version 1 (SYSV), for GNU/Linux 2.6.14, dynamically linked (uses shared libs), not stripped
```

3.7.3 POKY DEMO

We have prepared some interesting demos inside poky-image-demo based on PowerVR SGX technology.

The PowerVR technology uses a unique approach to rendering a 3D scene, known as tile-based deferred rendering (often abbreviated as TBDR). As the polygon generating program feeds triangles to the PowerVR driver which stores them in memory in triangle strip format. Unlike other architectures, polygon rendering is not performed until all polygon information has been collated for the current frame—hence rendering is deferred.

In order to render, the display is split into rectangular sections in a grid pattern. Each section is known as a tile. With each tile is associated a list of the triangles that visibly overlap that tile. Each tile is rendered in turn to produce the final image.

Tiles are rendered using a process similar to ray-casting. Rays are cast onto the triangles associated with the tile and a pixel is rendered from the triangle closest to the camera. The PowerVR hardware typically calculates the depths associated with each polygon for one tile row in 1 cycle.

The advantage of this method is that, unlike with a more traditional z-buffered rendering pipeline, work is never done determining what a polygon looks like in an area where it is obscured by other geometry. It also allows for correct rendering of partially transparent polygons independent of the order in which they are processed by the polygon producing application. (This capability was only implemented in Series 1 and 2. It has been removed since for lack of API support and cost reasons.) More importantly, as the rendering is circumscribed to a tile at a time, the whole tile can be in fast onchip memory, which is flushed to video memory before passing on to render the next tile. Under normal circumstances, each tile is visited just once per frame.

To boot with poky-image-demo you should change U-Boot environment.

```
U-Boot> setenv project  poky-image-demo   // poky sgx demo
```

If you want to set as default boot option, you can also save your new environment

```
U-Boot> saveenv
```

The last step is run the boot command.

```
U-Boot> run bootcmd
```

If all is ok, something like this should appear in the monitor connected to IGEPv2 board.

Figure 24 Poky-demo Desktop - Applications

Just go to Applications and execute the demos:

- Coverflow Demo

- Skybox2 Demo

- Transforms Demo

SKYBOX DEMO

Skybox demo renders a scene with high-dynamic range enabling advanced effects including dynamic exposure and bloom making full usage of the variable precision supported within the POWERVR SGX IP Core.

Figure 25 Poky Desktop – Applications – Skybox2 Demo

COVERFLOW DEMO

The highly popular Cover Flow-style concept of browsing through a music collection can be implemented at 60+ FPS rates with minimal power consumption and maximal eye-candy on POWERVR enabled products this demo shows the typical mirror effects and a per-pixel specular highlight together with a variety of transition effects including a high-performance blur effect.

Figure 26 Poky Desktop – Applications – Coverflow Demo

3.8 CREATING A CUSTOM ROOT FILESYSTEM (RFS)

The simplest way to create a root filesystem is to use an already working filesystem and customize it. Prebuilt images are also available; here you have a brief description:

- poky-image-minimal - A small image, just enough to allow a device to boot

    o Get the latest stable version from:

        http://downloads.myigep.com/rootfs/*poky-image-minimal-igep0020b.cpio*

- poky-image-sato - X11 image with Sato theme and Pimlico applications. Sato is a GNOME Mobile based user interface environment.

    o Get the latest stable version from:
       http://downloads.myigep.com/rootfs/*poky-image-sato-igep0020b.cpio*

- poky-image-demo - X11 image like poky-image-sato but with PowerVR SGX demos.

    o Get the latest stable version from:

        http://downloads.myigep.com/rootfs/*poky-image-demo-igep0020b.cpio*

For example, download and install your root filesystem image (poky-image-sdk-igep0020b.cpio) as root on the NFS server,

```
$ sudo mkdir -p /srv/nfs/poky/myrootfs/igep0020b

$ cd /srv/nfs/poky/myrootfs/igep0020b

$ sudo wget <download location>/poky-image-demo-igep0020b.cpio

$ sudo cpio -idm < poky-image-demo-igep0020b.cpio
```

Copy the kernel image from sato-demo project to myrootfs project

```
$ sudo mkdir -p /srv/tftp/poky/myrootfs/igep0020b

$ sudo cp /srv/tftp/poky/poky-image-demo/igep0020b/uImage /srv/tftp/poky/myrootfs/igep0020b
```

Export the directory tree with an entry in /etc/exports on the NFS server, like

```
# My rootfs

/srv/nfs/poky/myrootfs/igep0020b      *(rw,no_root_squash,no_subtree_check,sync)
```

and restart the NFS server

```
$ sudo /etc/init.d/nfs-kernel-server restart
```

Now, you can power up your board and configure u-boot to start your new root filesystem, don't forget to change the U-Boot environment to point your new rootfs

```
U-Boot> setenv project myrootfs

U-Boot> run bootcmd
```

In normal systems you can install packages. You can do the same with your running image with opkg, a lightweight package management system. Next are some steps to install new software in your root filesystem.

The first step is editing opkg configuration file (stored in /etc/opkg/arch.conf) to add feeds locations (located to http://downloads.myigep.com/dist/poky/stable/ipk), it looks like this,

```
arch all 1

arch any 6

arch noarch 11

arch arm 16

arch armv4 21

arch armv4t 26

arch armv5te 31

arch armv6 36

arch armv7 41

arch armv7a 46

arch igep0020b 51

src oe http://downloads.myigep.com/dist/poky/stable/ipk

src oe-all http://downloads.myigep.com/dist/poky/stable/ipk/all

src oe-armv7a http://downloads.myigep.com/dist/poky/stable/ipk/armv7a

src oe-igep0020b http://downloads.myigep.com/dist/poky/stable/ipk/igep0020b
```

Next, edit the file (/etc/resolv.conf)

```
nameserver 8.8.8.8
```

Next, to update the list of available packages.

```
# opkg update
```

To view the list of available packages,

```
# opkg list
```

or view the list of installed packages.

```
# opkg list_installed
```

And you can install new packages with,

```
# opkg install <package name>
```

For more available options

```
# opkg --help
```

If you also need some software that is typically not found in our repository, either because it's too new or too specific, you can build and then copy in your root filesystem image.

For example, with an autotooled package you can do this

```
$ ./configure --target=arm-none-linux-gnueabi --host=arm-none-linux-gnueabi --prefix=/srv/nfs/poky/myrootfs/igep0020b

$ make

$ sudo make install
```

3.8.1 INSTALL ADDITIONAL LIBRARIES INTO THE SDK

Sooner or later you will want to compile an application that has dependencies which can't found inside the software development kit. Like install new packages into your rootfs you can install development packages into your SDK

```
$ source /usr/local/poky/eabi-glibc/arm/environment-setup

$ opkg-target install <package name>
```

To get a list of all available packages run

```
$ opkg-target list
```

Or

```
$ opkg-target list_installed
```

To view the list of already installed packages

If a package does not exist in our repositories, see http://www.myigep.com support.

3.8.2 DEVELOP WITH CODE::BLOCKS IDE

Code::Blocks is a free C++ IDE built to meet the most demanding needs of its users. It is designed to be very extensible and fully configurable.

An IDE with all the features user needs, having a consistent look, feel and operation across platforms.
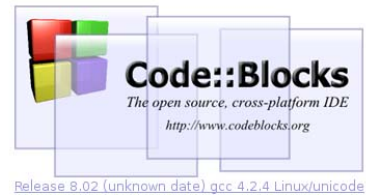
Built around a plugin framework, Code::Blocks can be extended with plugins. Any kind of functionality can be added by installing/coding a plugin. For instance, compiling and debugging functionality is already provided by plugins!
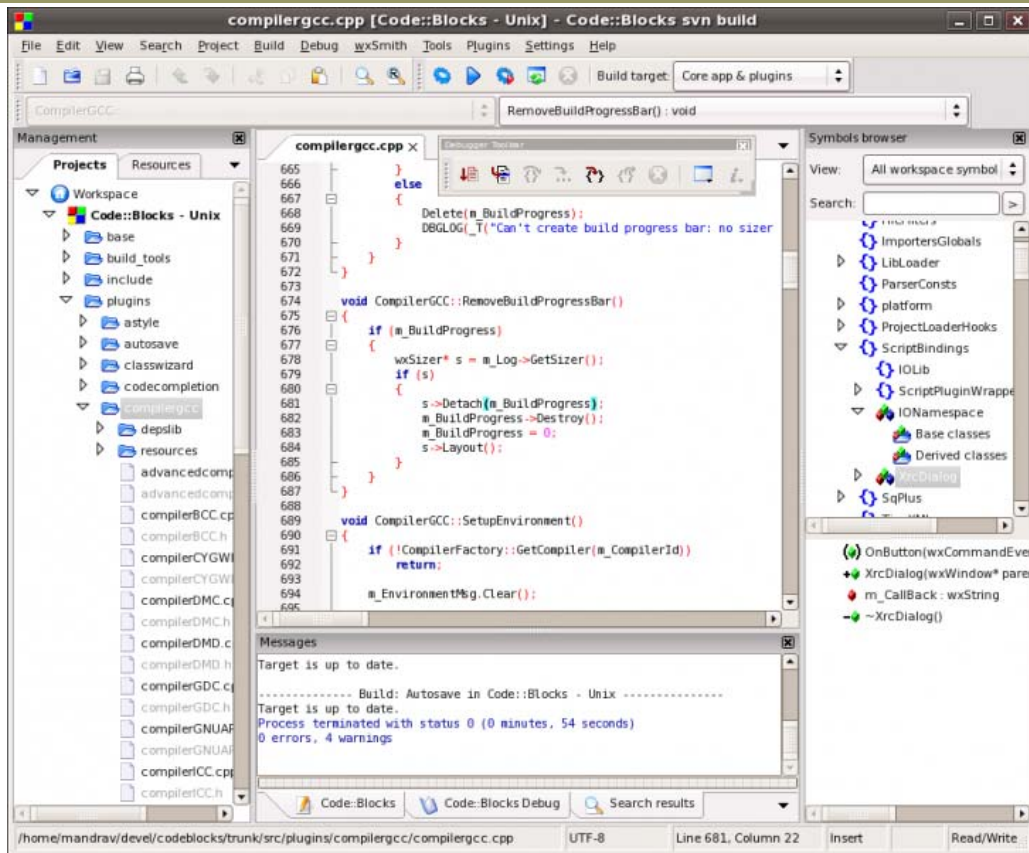
Code::Blocks configuration

Go to Compiler and debugger Setting menu: Settings--> Compiler and debugger
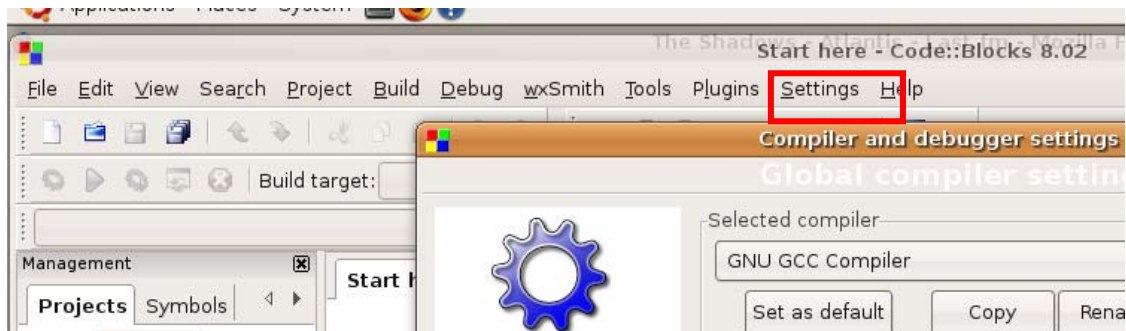
The compiler and debugger settings window will appear:
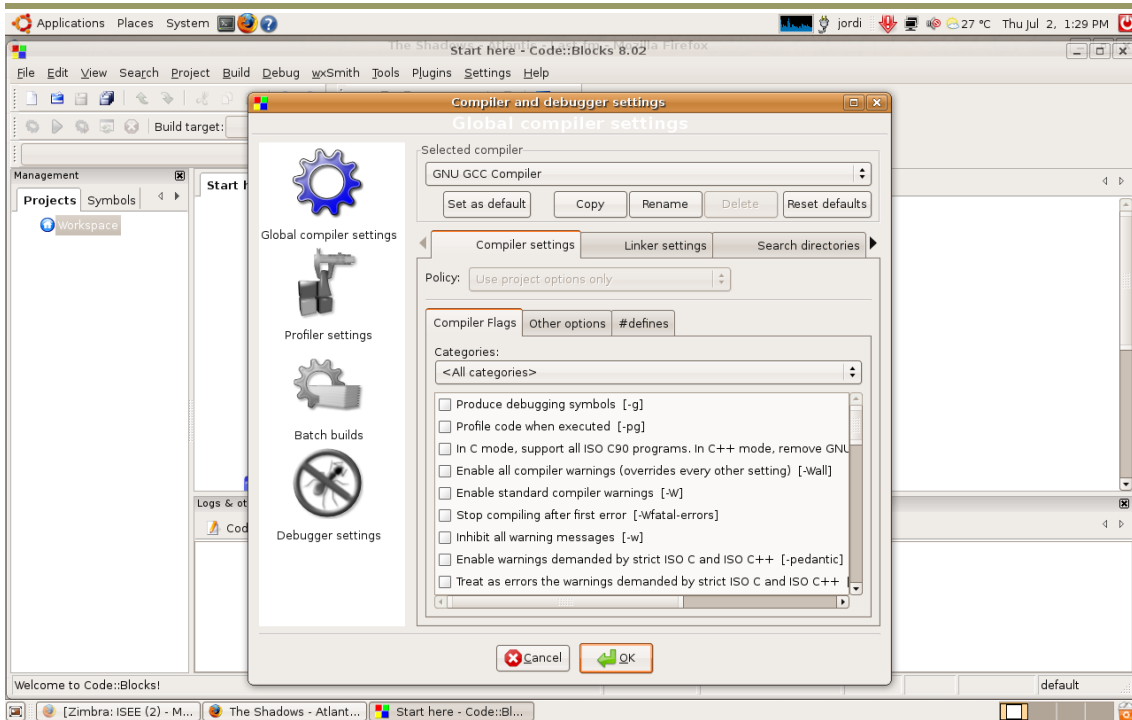
Choose GNU ARM GCC Compiler from Selected compiler window.
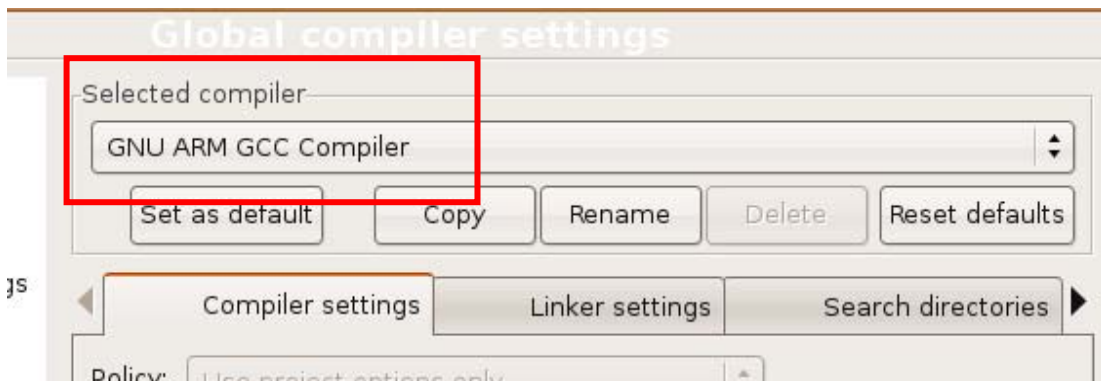
Go to Toolchain executables window.
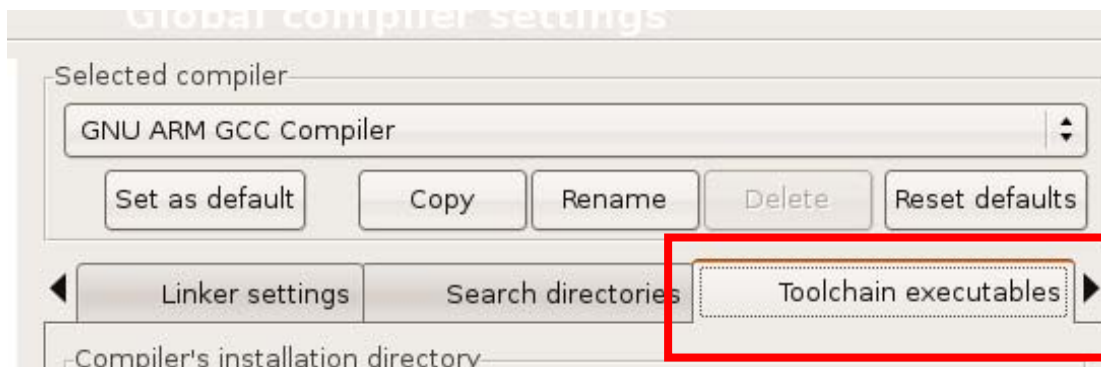
Compiler's installation directory: */usr/local/poky/eabi-glibc/arm/*

Program Files

C Compiler: *arm-none-linux-gnueabi-gcc*

C++ Compiler: *arm-none-linux-gnueabi-g++*

Linker for dynamic libs: *arm-none-linux-gnueabi-gcc*

C Linker for static libs: *arm-none-linux-gnueabi-ar*

Debugger: *arm-none-linux-gnueabi-gdb*
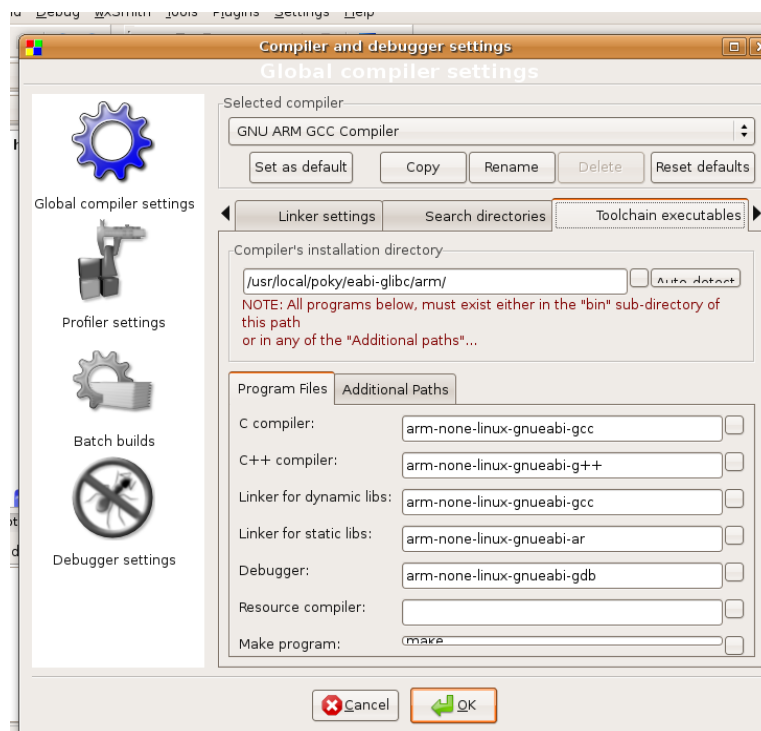
4   IGEPV2 SDK TRICKS

4.1 HOW TO CONNECT CABLE TO IGEPV2 SERIAL DEBUG PORT

Support for RS232 is provided by a 10 pin header on the IGEPV2 Board for access to an onboard RS232 transceiver.
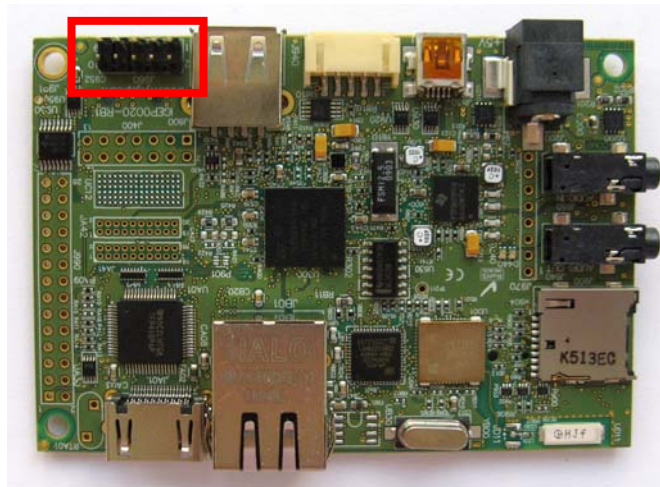
Figure 27 IDC10 header
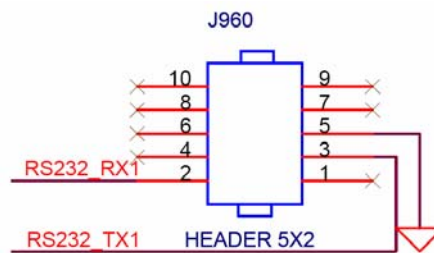
Figure 28 J960 schematic

NOTE:

Look out for pin 1, it is a polarity connector.

Wrong use could damage board

User will require an IDC10 to DB9 flat cable like the one showed below:
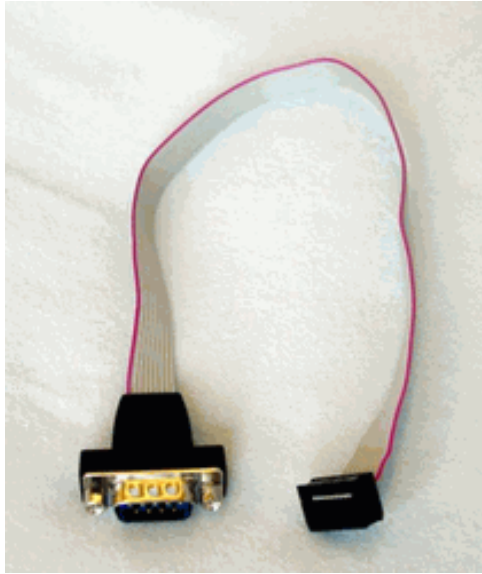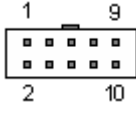
Figure 29 IDC10 to DB9 cable

Connectors are numbered:

DB9	IDC10
Pin 1	Pin 1
Pin 2	Pin 2
Pin 3	Pin 3
Pin 4	Pin 4
Pin 5	Pin 5
Pin 6	Pin 6
Pin 7	Pin 7
Pin 9	Pin 9
NC	Pin 10

4.2 HOW TO OPEN A LINUX CONSOLE TO IGEPV2 BOARD

You can user Serial debug port interface or Ethernet.

4.2.1 SERIAL CONSOLE FROM LINUX
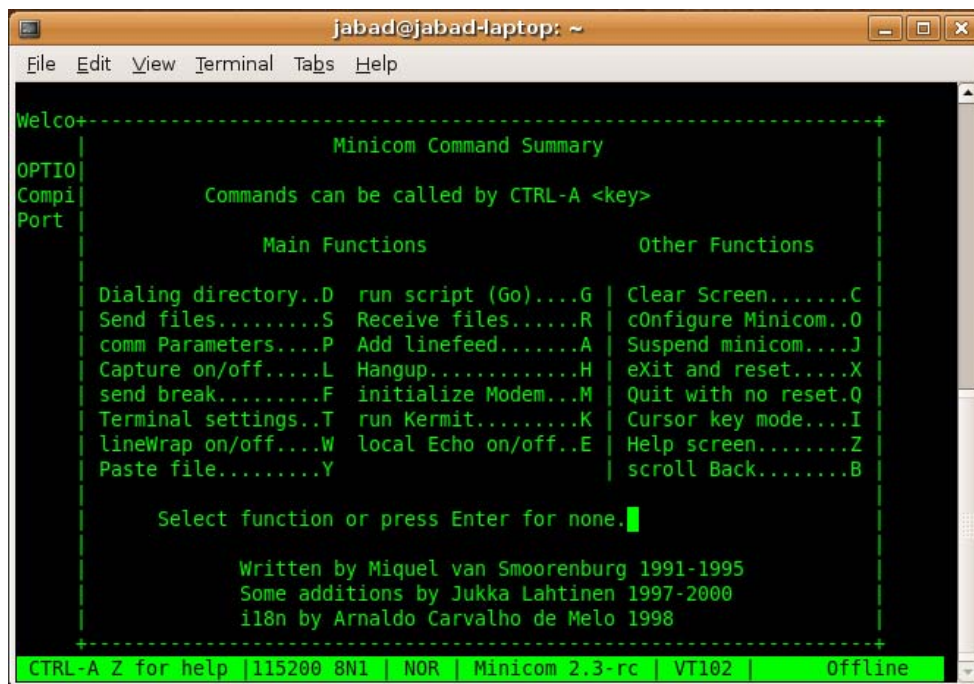
Minicom is a text-based modem control and terminal emulation program for Unix-like operating systems. Minicom includes a dialing directory, ANSI and VT100 emulation, an (external) scripting language, and other features. Minicom is a menu-driven communications program. It also has an auto zmodem download.

Minicom is installed on IGEPV2 VM SDK.

Execute minicom.

```
$ minicom
```

Go to the Minicom Command Summary: Ctrl-A Z.

Figure 30 Minicom Command Summary

Press O to Configure Minicom, and Select Serial port setup option.
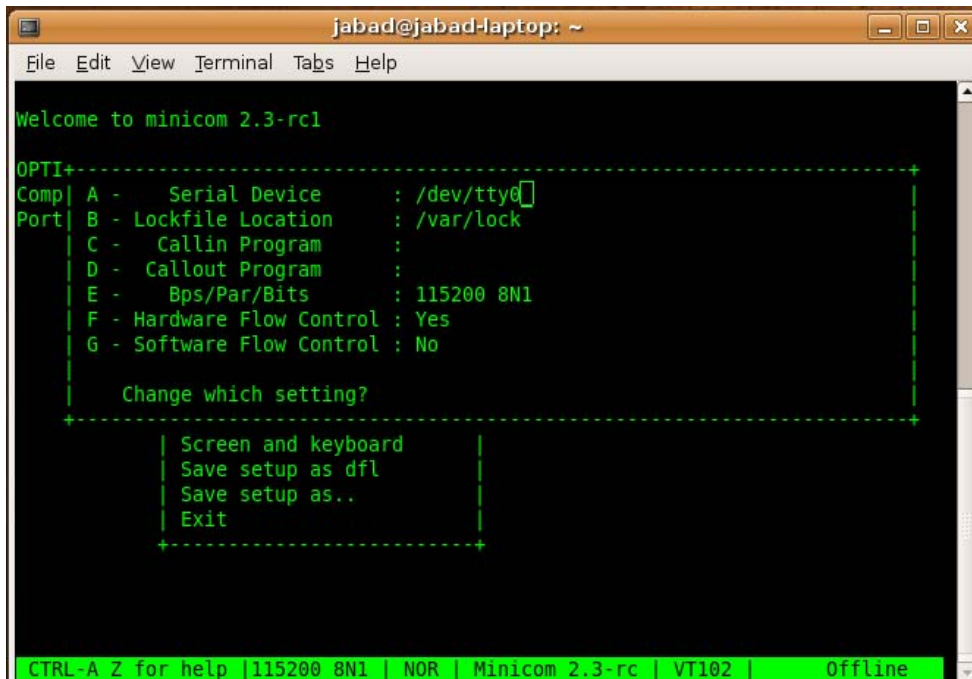
Figure 31 Minicom Configure

Chose Serial port setup option, and configure your Serial Device port, and Bps/Parity/Bits: 115200 8N1

Figure 32 Minicom Serial port Setup

4.2.2 SERIAL CONSOLE FROM WINDOWS

PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the Windows platform. PuTTY is open source software that is available with source code and is developed and supported by a group of volunteers.

You can download it from: http://www.putty.org/

You can use *putty* to open a console using the serial debug port.

Open putty. Choose Serial line. Configure Speed to 115200. Select Serial Connection type. Press on Open button.
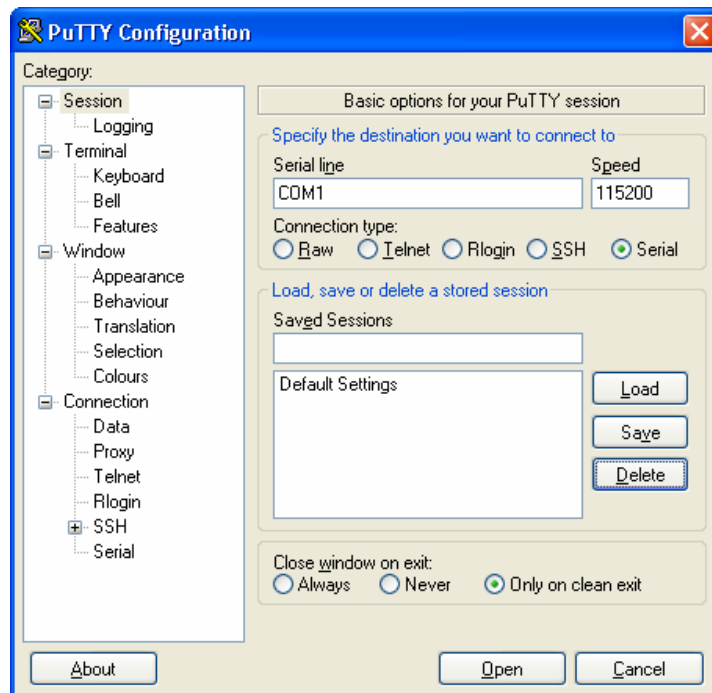
Figure 33 Putty main window

4.2.3 REMOTE TCP/IP SHELL CONSOLE FROM LINUX

Open a Linux console via Ethernet or WiFi interface link:

```
# by default IGEPV2 board comes configured with IP 192.168.254.254

$ ssh root@192.168.254.254

password: letmein
```

If you are not in the range 192.168.254.x, you can configure an alias to your Ethernet interface into your IGEPv2 SDK virtual machine:

```
$ sudo ifconfig eth0:alias0 192.168.245.x
```

If you want to change the IP address:

```
$ sudo ifconfig eth0 192.168.x.x
```

4.2.4 REMOTE TCP/IP SHELL CONSOLE FROM WINDOWS

Open putty.

Choose SSH Connection type.

Write the IP Address

Select port 22

Press on button Open.
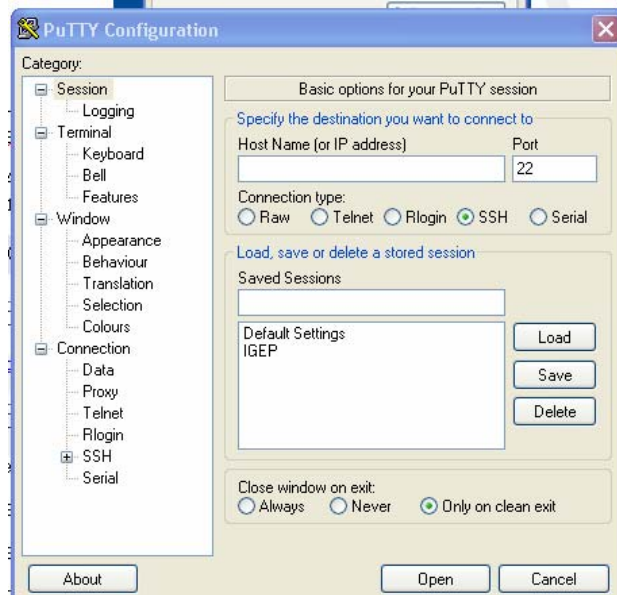
4.3 IGEPV2 BOARD BOOTLOG

Just for user information here you have the boot up traces…

This traces has been captured through the serial debug port.

```
exas Instruments X-Loader 1.4.2 (Sep  7 2009 - 18:18:18)
Detected Numonyx OneNAND 4G Flash
Loading u-boot.bin from onenand


U-Boot 2009.08-0-dirty (Sep 22 2009 - 13:31:59)

OMAP3530-GP ES3.1, CPU-OPP2 L3-165MHz
IGEP v2.x rev. B + LPDDR/ONENAND
DRAM:  512 MB
Muxed OneNAND(DDP) 512MB 1.8V 16-bit (0x58)
OneNAND version = 0x0031
Chip support all block unlock
Chip has 2 plane
Scanning device for bad blocks
Bad eraseblock 936 at 0x07500000
Bad eraseblock 937 at 0x07520000
Bad eraseblock 978 at 0x07a40000
Bad eraseblock 979 at 0x07a60000
Bad eraseblock 1282 at 0x0a040000
Bad eraseblock 1283 at 0x0a060000
Bad eraseblock 1872 at 0x0ea00000
Bad eraseblock 3920 at 0x1ea00000
OneNAND: 512 MB
In:    serial
Out:   serial
Err:   serial
Die ID #1fbc000400000000040365fa1801801f
Net:   smc911x-0
Warning: smc911x-0 MAC addresses don't match:
Address in SROM is       ff:ff:ff:ff:ff:ff
Address in environment is  ac:de:48:00:02:54

Hit any key to stop autoboot:  0
No MMC card found

** Unable to use mmc 0:1 for fatload **

** Unable to use mmc 0:1 for fatload **
smc911x: initializing
smc911x: detected LAN9221 controller
smc911x: phy initialized
smc911x: MAC ac:de:48:00:02:54
Using smc911x-0 device
host 192.168.254.10 is alive
smc911x: initializing
smc911x: detected LAN9221 controller
```

```
smc911x: phy initialized
smc911x: MAC ac:de:48:00:02:54
Using smc911x-0 device
TFTP from server 192.168.254.10; our IP address is 192.168.254.254
Filename 'poky/poky-image-sato/igep0020b/uImage'.
Load address: 0x80000000
Loading:
#################################################################
#################################################################
#################################################################
#################################################################
#################################################################
#################################################################
######## #############################################

done
Bytes transferred = 2332572 (23979c hex)
## Booting kernel from Legacy Image at 80000000 …
   Image Name:   Linux-2.6.28.10
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2332508 Bytes = 2.2 MB
   Load Address: 80008000
   Entry Point:  80008000
   Verifying Checksum … OK
   Loading Kernel Image … OK
OK

Starting kernel …

Uncompressing Linux.................................................................................
[    0.000000] Linux version 2.6.28.10 (jabad@jabad-laptop) (gcc version 4.3.2 (GCC) ) #2 Mon
Dec 21 12:49:37 CET 2009
[    0.000000] CPU: ARMv7 Processor [411fc083] revision 3 (ARMv7), cr=10c5387f
[    0.000000] CPU: VIPT nonaliasing data cache, VIPT nonaliasing instruction cache
[    0.000000] Machine: IGEP v2.x board
[    0.000000] Memory policy: ECC disabled, Data cache writeback
[    0.000000] OMAP3430 ES3.1
[    0.000000] SRAM: Mapped pa 0x40200000 to va 0xd7000000 size: 0x100000
[    0.000000] Reserving 4194304 bytes SDRAM for VRAM
[    0.000000] Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 130048
[    0.000000] Kernel command line: mem=512M console=ttyS2,115200n8 console=tty0
omapfb.mode=dvi:1024x768MR-16@60 root=:
[    0.000000] Clocking rate (Crystal/DPLL/ARM core): 26.0/332/500 MHz
[    0.000000] GPMC revision 5.0
[    0.000000] IRQ: Found an INTC at 0xd8200000 (revision 4.0) with 96 interrupts
[    0.000000] Total of 96 interrupts on 1 active controller
[    0.000000] OMAP34xx GPIO hardware version 2.5
[    0.000000] PID hash table entries: 2048 (order: 11, 8192 bytes)
[    0.000000] OMAP clockevent source: GPTIMER12 at 32768 Hz
[    0.000000] Console: colour dummy device 80x30
[    0.000000] console [tty0] enabled
[    0.000000] Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)
[    0.000000] Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)
[    0.000000] Memory: 512MB = 512MB total
[    0.000000] Memory: 510464KB available (4264K code, 509K data, 168K init)
[    0.000000] Calibrating delay loop… 503.12 BogoMIPS (lpj=1966080)
[    0.000000] Mount-cache hash table entries: 512
```

```
[    0.000000] CPU: Testing write buffer coherency: ok
[    0.000000] net_namespace: 532 bytes
[    0.000000] regulator: core version 0.5
[    0.000000] NET: Registered protocol family 16
[ 5090.714172] OMAP DMA hardware revision 4.0
[ 5090.715270] USB: No board-specific platform config found
[ 5090.794036] OMAP DSS rev 2.0
[ 5090.794158] OMAP DISPC rev 3.0
[ 5090.794342] OMAP DSI rev 1.0
[ 5090.796569] i2c_omap i2c_omap.1: bus 1 rev3.12 at 2600 kHz
[ 5090.800048] twl4030: PIH (irq 7) chaining IRQs 368..375
[ 5090.800109] twl4030: power (irq 373) chaining IRQs 376..383
[ 5090.800598] twl4030: gpio (irq 368) chaining IRQs 384..401
[ 5090.803314] i2c_omap i2c_omap.3: bus 3 rev3.12 at 100 kHz
[ 5090.804992] SCSI subsystem initialized
[ 5090.808441] twl4030_usb twl4030_usb: Initialized TWL4030 USB module
[ 5090.809814] usbcore: registered new interface driver usbfs
[ 5090.810333] usbcore: registered new interface driver hub
[ 5090.810699] usbcore: registered new device driver usb
[ 5090.810729] musb_hdrc: version 6.0, musb-dma, otg (peripheral+host), debug=0
[ 5090.811431] musb_hdrc: USB OTG mode controller at d80ab000 using DMA, IRQ 92
[ 5090.813476] regulator: VMMC1: 1850 <--> 3150 mV normal standby
[ 5090.813934] regulator: VUSB1V5: 1500 mV normal standby
[ 5090.814361] regulator: VUSB1V8: 1800 mV normal standby
[ 5090.814788] regulator: VUSB3V1: 3100 mV normal standby
[ 5090.865234] cfg80211: Using static regulatory domain info
[ 5090.865264] cfg80211: Regulatory domain: US
[ 5090.865295]  (start_freq - end_freq @ bandwidth), (max_antenna_gain, max_eirp)
[ 5090.865356]  (2402000 KHz - 2472000 KHz @ 40000 KHz), (600 mBi, 2700 mBm)
[ 5090.865386]  (5170000 KHz - 5190000 KHz @ 40000 KHz), (600 mBi, 2300 mBm)
[ 5090.865417]  (5190000 KHz - 5210000 KHz @ 40000 KHz), (600 mBi, 2300 mBm)
[ 5090.865447]  (5210000 KHz - 5230000 KHz @ 40000 KHz), (600 mBi, 2300 mBm)
[ 5090.865478]  (5230000 KHz - 5330000 KHz @ 40000 KHz), (600 mBi, 2300 mBm)
[ 5090.865539]  (5735000 KHz - 5835000 KHz @ 40000 KHz), (600 mBi, 3000 mBm)
[ 5090.865570] cfg80211: Calling CRDA for country: US
[ 5090.869110] NET: Registered protocol family 2
[ 5090.935638] IP route cache hash table entries: 16384 (order: 4, 65536 bytes)
[ 5090.936340] TCP established hash table entries: 65536 (order: 7, 524288 bytes)
[ 5090.938018] TCP bind hash table entries: 65536 (order: 6, 262144 bytes)
[ 5090.938995] TCP: Hash tables configured (established 65536 bind 65536)
[ 5090.939025] TCP reno registered
[ 5090.959075] NET: Registered protocol family 1
[ 5090.962310] VFS: Disk quotas dquot_6.5.1
[ 5090.962463] Dquot-cache hash table entries: 1024 (order 0, 4096 bytes)
[ 5090.963531] JFFS2 version 2.2. (NAND) (SUMMARY)  �© 2001-2006 Red Hat, Inc.

[ 5090.967987] msgmni has been set to 997
[ 5090.976104] alg: No test for stdrng (krng)
[ 5090.976257] io scheduler noop registered
[ 5090.976287] io scheduler anticipatory registered
[ 5090.976318] io scheduler deadline registered
[ 5090.976531] io scheduler cfq registered (default)
[ 5090.996582] Serial: 8250/16550 driver4 ports, IRQ sharing enabled
[ 5091.019165] serial8250.0: ttyS0 at MMIO 0x4806a000 (irq = 72) is a ST16654
[ 5091.040344] serial8250.0: ttyS1 at MMIO 0x4806c000 (irq = 73) is a ST16654
[ 5091.061767] serial8250.0: ttyS2 at MMIO 0x49020000 (irq = 74) is a ST16654
[ 5091.061859] console [ttyS2] enabled
[ 5091.532958] brd: module loaded
```

```
[ 5091.541351] loop: module loaded
[ 5091.544647] smsc911x: Driver version 2008-10-21.
[ 5091.551879] smsc911x-mdio: probed
[ 5091.555541] eth0: attached PHY driver [Generic PHY] (mii_bus:phy_addr=ffffffff:01, irq=-1)
[ 5091.564147] net eth0: MAC Address: a2:50:65:29:c5:5c
[ 5091.569488] i2c /dev entries driver
[ 5091.575103] input: triton2-pwrbutton as /devices/virtual/input/input0
[ 5091.599487] triton2 power button driver initialized
[ 5091.604431] st: Version 20080504, fixed bufsize 32768, s/g segs 256
[ 5091.611053] Driver 'st' needs updating - please use bus_type methods
[ 5091.617767] osst :I: Tape driver with OnStream support version 0.99.4
[ 5091.617797] osst :I: $Id: osst.c,v 1.73 2005/01/01 21:13:34 wriede Exp $
[ 5091.631286] Driver 'osst' needs updating - please use bus_type methods
[ 5091.638305] Driver 'sd' needs updating - please use bus_type methods
[ 5091.644958] Driver 'sr' needs updating - please use bus_type methods
[ 5091.651885] SCSI Media Changer driver v0.25
[ 5091.656463] Driver 'ch' needs updating - please use bus_type methods
[ 5091.663513] OneNAND driver initializing
[ 5091.668395] omap2-onenand omap2-onenand: initializing on CS0, phys base 0x20000000,
virtual base d0900000
[ 5091.678253] Muxed OneNAND(DDP) 512MB 1.8V 16-bit (0x58)
[ 5091.683532] OneNAND version = 0x0031
[ 5091.688446] block = 2048, wp status = 0x2
[ 5091.696746] Scanning device for bad blocks
[ 5091.745697] Bad eraseblock 936 at 0x07500000
[ 5091.752014] Bad eraseblock 978 at 0x07a40000
[ 5091.770843] Bad eraseblock 1282 at 0x0a040000
[ 5091.803466] onenand_bbt_wait: ecc error = 0x2222, controller error 0x2400
[ 5091.810302] Bad eraseblock 1872 at 0x0ea00000
[ 5091.814758] Bad eraseblock 1873 at 0x0ea20000
[ 5091.915893] onenand_bbt_wait: ecc error = 0x2222, controller error 0x2400
[ 5091.922790] Bad eraseblock 3920 at 0x1ea00000
[ 5091.927185] Bad eraseblock 3921 at 0x1ea20000
[ 5091.939849] Creating 5 MTD partitions on "omap2-onenand":
[ 5091.945343] 0x00000000-0x00080000 : "X-Loader"
[ 5091.951568] 0x00080000-0x00200000 : "U-Boot"
[ 5091.957000] 0x00200000-0x00280000 : "U-Boot Env"
[ 5091.962829] 0x00280000-0x00580000 : "Kernel"
[ 5091.968292] 0x00580000-0x20000000 : "File System"
[ 5091.978271] ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
[ 5091.985260] ehci-omap ehci-omap.0: OMAP-EHCI Host Controller
[ 5091.991882] ehci-omap ehci-omap.0: new USB bus registered, assigned bus number 1
[ 5091.999755] ehci-omap ehci-omap.0: irq 77, io mem 0x48064800
[ 5092.013580] ehci-omap ehci-omap.0: USB 2.0 started, EHCI 1.00
[ 5092.020080] usb usb1: configuration #1 chosen from 1 choice
[ 5092.026184] hub 1-0:1.0: USB hub found
[ 5092.030090] hub 1-0:1.0: 3 ports detected
[ 5092.035766] Initializing USB Mass Storage driver…
[ 5092.041076] usbcore: registered new interface driver usb-storage
[ 5092.047210] USB Mass Storage support registered.
[ 5092.052154] usbcore: registered new interface driver libusual
[ 5092.058074] g_ether gadget: using random self ethernet address
[ 5092.063995] g_ether gadget: using random host ethernet address
[ 5092.070587] usb0: MAC 42:32:fa:3e:82:37
[ 5092.074462] usb0: HOST MAC 3a:a8:bf:19:be:34
[ 5092.078887] g_ether gadget: Ethernet Gadget, version: Memorial Day 2008
[ 5092.085632] g_ether gadget: g_ether ready
```

```
[ 5092.089691] musb_hdrc musb_hdrc: MUSB HDRC host driver
[ 5092.095642] musb_hdrc musb_hdrc: new USB bus registered, assigned bus number 2
[ 5092.103576] usb usb2: configuration #1 chosen from 1 choice
[ 5092.109649] hub 2-0:1.0: USB hub found
[ 5092.113494] hub 2-0:1.0: 1 port detected
[ 5092.119262] mice: PS/2 mouse device common for all mice
[ 5092.155456] twl4030_rtc twl4030_rtc: rtc core: registered twl4030_rtc as rtc0
[ 5092.165252] OMAP Watchdog Timer Rev 0x31: initial timeout 60 sec
[ 5092.172149] mmci-omap-hs mmci-omap-hs.0: Failed to get debounce clock
[ 5092.287628] mmci-omap-hs mmci-omap-hs.1: Failed to get debounce clock
[ 5092.296081] usbcore: registered new interface driver usbhid
[ 5092.301727] usbhid: v2.6:USB HID core driver
[ 5092.306427] Advanced Linux Sound Architecture Driver Version 1.0.18rc3.
[ 5092.313598] ASoC version 0.13.2
[ 5092.317047] IGEP v2.x SoC init
[ 5092.320526] TWL4030 Audio Codec init
[ 5092.325744] asoc: twl4030 <-> omap-mcbsp-dai-(link_id) mapping ok
[ 5092.347747] ALSA device list:
[ 5092.350830]   #0: igep (twl4030)
[ 5092.354492] oprofile: using arm/armv7
[ 5092.358520] TCP cubic registered
[ 5092.361785] NET: Registered protocol family 17
[ 5092.366363] NET: Registered protocol family 15
[ 5092.371307] RPC: Registered udp transport module.
[ 5092.376129] RPC: Registered tcp transport module.
[ 5092.380981] ieee80211: 802.11 data/management/control stack, git-1.1.13
[ 5092.387664] ieee80211: Copyright (C) 2004-2005 Intel Corporation
<jketreno@linux.intel.com>
[ 5092.396179] ThumbEE CPU extension supported.
[ 5092.400543] Power Management for TI OMAP3.
[ 5092.413024] SmartReflex driver initialized
[ 5092.423889] Disabling unused clock "sr2_fck"
[ 5092.428314] Disabling unused clock "sr1_fck"
[ 5092.432647] Disabling unused clock "mcbsp_fck"
[ 5092.437164] Disabling unused clock "mcbsp_fck"
[ 5092.441650] Disabling unused clock "mcbsp_fck"
[ 5092.446166] Disabling unused clock "mcbsp_ick"
[ 5092.450653] Disabling unused clock "mcbsp_ick"
[ 5092.455169] Disabling unused clock "mcbsp_ick"
[ 5092.459686] Disabling unused clock "gpt2_ick"
[ 5092.464080] Disabling unused clock "gpt3_ick"
[ 5092.468505] Disabling unused clock "gpt4_ick"
[ 5092.472900] Disabling unused clock "gpt5_ick"
[ 5092.477325] Disabling unused clock "gpt6_ick"
[ 5092.481750] Disabling unused clock "gpt7_ick"
[ 5092.486145] Disabling unused clock "gpt8_ick"
[ 5092.490570] Disabling unused clock "gpt9_ick"
[ 5092.494995] Disabling unused clock "wdt3_ick"
[ 5092.499420] Disabling unused clock "wdt3_fck"
[ 5092.503814] Disabling unused clock "gpio2_dbck"
[ 5092.508392] Disabling unused clock "gpio3_dbck"
[ 5092.512969] Disabling unused clock "gpio4_dbck"
[ 5092.517578] Disabling unused clock "gpio5_dbck"
[ 5092.522186] Disabling unused clock "gpio6_dbck"
[ 5092.526763] Disabling unused clock "gpt9_fck"
[ 5092.531188] Disabling unused clock "gpt8_fck"
[ 5092.535583] Disabling unused clock "gpt7_fck"
```

```
[ 5092.540008] Disabling unused clock "gpt6_fck"
[ 5092.544403] Disabling unused clock "gpt5_fck"
[ 5092.548828] Disabling unused clock "gpt4_fck"
[ 5092.553253] Disabling unused clock "gpt3_fck"
[ 5092.557647] Disabling unused clock "gpt2_fck"
[ 5092.562072] Disabling unused clock "gpt1_ick"
[ 5092.566467] Disabling unused clock "wdt1_ick"
[ 5092.570892] Disabling unused clock "wdt2_ick"
[ 5092.575286] Disabling unused clock "wdt2_fck"
[ 5092.579711] Disabling unused clock "gpio1_dbck"
[ 5092.584320] Disabling unused clock "gpt1_fck"
[ 5092.588714] Disabling unused clock "cam_ick"
[ 5092.593078] Disabling unused clock "cam_mclk"
[ 5092.597503] Disabling unused clock "des1_ick"
[ 5092.601928] Disabling unused clock "sha11_ick"
[ 5092.606414] Disabling unused clock "rng_ick"
[ 5092.610748] Disabling unused clock "aes1_ick"
[ 5092.615173] Disabling unused clock "ssi_ick"
[ 5092.619476] Disabling unused clock "mailboxes_ick"
[ 5092.624328] Disabling unused clock "mcbsp_ick"
[ 5092.628814] Disabling unused clock "mcbsp_ick"
[ 5092.633331] Disabling unused clock "gpt10_ick"
[ 5092.637817] Disabling unused clock "gpt11_ick"
[ 5092.642333] Disabling unused clock "i2c_ick"
[ 5092.646667] Disabling unused clock "hdq_ick"
[ 5092.650970] Disabling unused clock "mspro_ick"
[ 5092.655487] Disabling unused clock "des2_ick"
[ 5092.659881] Disabling unused clock "sha12_ick"
[ 5092.664398] Disabling unused clock "aes2_ick"
[ 5092.668792] Disabling unused clock "icr_ick"
[ 5092.673126] Disabling unused clock "pka_ick"
[ 5092.677459] Disabling unused clock "ssi_ssr_fck"
[ 5092.682128] Disabling unused clock "hdq_fck"
[ 5092.686492] Disabling unused clock "mcbsp_fck"
[ 5092.690979] Disabling unused clock "mcbsp_fck"
[ 5092.695495] Disabling unused clock "i2c_fck"
[ 5092.699798] Disabling unused clock "mspro_fck"
[ 5092.704315] Disabling unused clock "gpt11_fck"
[ 5092.708801] Disabling unused clock "gpt10_fck"
[ 5092.713317] Disabling unused clock "dpll4_m6x2_ck"
[ 5092.718170] Disabling unused clock "dpll3_m3x2_ck"
[ 5092.723022] Disabling unused clock "sys_clkout1"
[ 5092.727752] VFP support v0.3: implementor 41 architecture 3 part 30 variant c rev 1
[ 5092.740386] fbcvt: 1024x768@60: CVT Name - .786M3-R
[ 5092.772338] Console: switching to colour frame buffer device 128x48
[ 5092.792724] clock: clksel_round_rate_div: dpll4_m4_ck target_rate 48000000
[ 5092.799835] clock: new_div = 9, new_rate = 48000000
[ 5092.807952] twl4030_rtc twl4030_rtc: setting system clock to 2000-01-01 01:23:47 UTC
(946689827)
[ 5093.334106] net eth0: SMSC911x/921x identified at 0xd08c4000, IRQ: 336
[ 5094.342285] IP-Config: Complete:
[ 5094.345458]     device=eth0, addr=192.168.254.254, mask=255.255.255.0,
gw=192.168.254.10,
[ 5094.354064]     host=192.168.254.254, domain=, nis-domain=(none),
[ 5094.360504]     bootserver=192.168.254.10, rootserver=192.168.254.10, rootpath=
[ 5094.368774] Looking up port of RPC 100003/2 on 192.168.254.10
[ 5096.381866] Looking up port of RPC 100005/1 on 192.168.254.10
```

```
[ 5096.406219] VFS: Mounted root (nfs filesystem).
[ 5096.411132] Freeing init memory: 168K
[ 5099.306121] udev: starting version 141
[ 5101.099151] Bluetooth: Core ver 2.13
[ 5101.919006] NET: Registered protocol family 31
[ 5101.923553] Bluetooth: HCI device and connection manager initialized
[ 5101.929992] Bluetooth: HCI socket layer initialized
[ 5102.846618] Bluetooth: L2CAP ver 2.11
[ 5102.850433] Bluetooth: L2CAP socket layer initialized
[ 5103.657775] Bluetooth: HIDP (Human Interface Emulation) ver 1.2
[ 5109.079742] NET: Registered protocol family 10
[ 5109.776397] Bluetooth: RFCOMM socket layer initialized
[ 5109.781707] Bluetooth: RFCOMM TTY layer initialized
[ 5109.786621] Bluetooth: RFCOMM ver 1.10

OpenedHand Linux (Poky) 3.2+snapshot-20090721 igep0020b ttyS2

igep0020b login:
```

5   SUPPORT
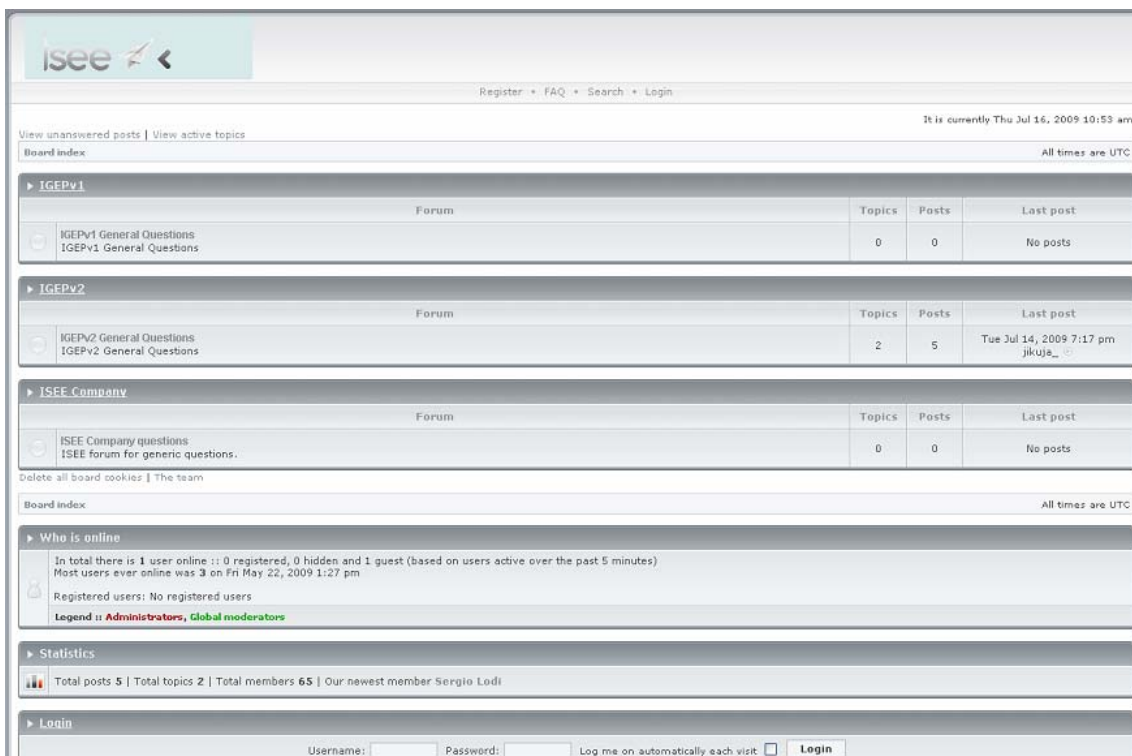
5.1   IGEP PLATFORM PUBLIC FORUM

There is a public forum where users can learn and contribute about the IGEP platform.

Users who want to use this service, have to be registered on www.myigep.com portal.

If you are registered, just go to Support>IGEP Public forum on www.myigep.com.

More documentation coming soon.

Figure 34 IGEP public forum main page

5.2 IGEP PLATFORM BUG TRACKER

For IGEPV2 bug tracking, user will have to use Mantis Bug Tracking.

Figure 35 MantisBT main page

MantisBT is a free popular web-based bugtracking system (feature list). It is written in the PHP scripting language and works with MySQL, MS SQL, and PostgreSQL databases and a webserver. MantisBT has been installed on Windows, Linux, Mac OS, OS/2, and others. Almost any web browser should be able to function as a client. It is released under the terms of the GNU General Public License (GPL).

For more information about MantisBT:
http://www.mantisbt.org/documentation.php

Users who want to use this service, have to be registered on www.myigep.com portal.

If you are registered, just go to Support>BUG Tracking on www.myigep.com.

More documentation coming soon.
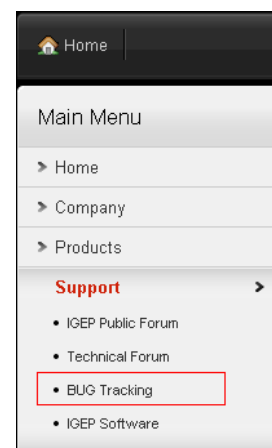
6 HOW TO..

6.1 HOW TO CROSS COMPILE X-LOADER

Overview of How-To

This How-To is meant to be a starting point for people to learn build the x-loader software for IGEP v2 devices as quickly and easily as possible.

This How-To works with the Ubuntu 8.04 IGEP v2.0 SDK Virtual Machine but most of the contents are valid also for other GNU/Linux distributions. We do not issue any guarantee that this will work on other distributions.

Requirements

Ubuntu 8.04 IGEP v2.0 SDK Virtual Machine, you can download registering as a user to http://www.myigep.com. It's free !

X-loader for IGEP v2 Rev. B

First of all setup the build environment sourcing this script

```
$ source /usr/local/poky/eabi-glibc/arm/environment-setup
```

X-loader v1.4.2-1 (stable)

This version support OneNAND DDP

Download the IGEP v2 Rev. B X-loader sources and follow next steps:

```
$ wget http://downloads.myigep.com/sources/x-loader/x-loader-1.4.2-1.tar.gz

$ tar xzf x-loader-1.4.2-1.tar.gz

$ cd x-loader-1.4.2-1

$ scripts/./autobuild.sh arm-none-linux-gnueabi-
```

The result will be :

x-load-ddp.bin.ift in autobuild/igep0020b/flash directory (for OneNAND DDP)

x-load.bin.ift in autobuild/igep0020b/sdcard directory (for SD card)

X-loader mainline tree (development)

Clone the GIT repository from git.myigep.com

```
$ git clone git://git.myigep.com/pub/scm/x-loader.git
```

```
$ cd x-loader
```

and build with

```
$ cd x-loader-1.4.2-1

$ scripts/./autobuild.sh arm-none-linux-gnueabi-
```

The result will be :

x-load-ddp.bin.ift in autobuild/igep0020b/flash directory (for OneNAND DDP)

x-load.bin.ift in autobuild/igep0020b/sdcard directory (for SD card)

6.1.1 HOW TO CROSS COMPILE U-BOOT

Overview of How-To

This How-To is meant to be a starting point for people to learn build the u-boot software for IGEP v2 devices as quickly and easily as possible.

This How-To works with the Ubuntu 8.04 IGEP v2.0 SDK Virtual Machine but most of the contents are valid also for other GNU/Linux distributions. We do not issue any guarantee that this will work on other distributions.

Requirements

Ubuntu 8.04 IGEP v2.0 SDK Virtual Machine, you can download registering as a user to http://www.myigep.com. It's free !

U-Boot for IGEP v2 Rev. B

First of all setup the build environment sourcing this script

```
$ source /usr/local/poky/eabi-glibc/arm/environment-setup
```

U-Boot v2009.11 (stable)

This version supports OneNAND DDP

Download the IGEP v2 Rev. B U-Boot sources and follow next steps:

```
$ wget http://downloads.myigep.com/sources/u-boot-arm/u-boot-arm-2009.11-
0.tar.gz

$ tar xzf u-boot-arm-2009.11-0.tar.gz
```

```
$ cd u-boot-arm-2009.11-0

$ make CROSS_COMPILE=arm-none-linux-gnueabi- omap3_igep0020b_config

$ make CROSS_COMPILE=arm-none-linux-gnueabi-
```

The result will be an u-boot.bin file in arch/arm/boot directory.

U-Boot mainline tree (development)

Clone the GIT repository from git.myigep.com

```
$ git clone git://git.myigep.com/pub/scm/u-boot-arm.git

$ cd u-boot-arm

$ git checkout origin/u-boot-2009.11.y -b u-boot-2009.11.y
```

and build with

```
$ make CROSS_COMPILE=arm-none-linux-gnueabi- omap3_igep0020b_config

$ make CROSS_COMPILE=arm-none-linux-gnueabi-
```

The result will be an u-boot.bin file in arch/arm/boot directory.

6.1.2 HOW TO CROSS COMPILE THE LINUX KERNEL

Embedded Linux is the use of a Linux operating system in embedded computer systems such as mobile phones, personal digital assistants, media players, set-top boxes, and other consumer electronics devices, networking equipment, machine control, industrial automation, navigation equipment and medical instruments.

Unlike desktop and server versions of Linux, embedded versions of Linux are designed for devices with relatively limited resources, such as cell phones and set-top boxes. Due to concerns such as cost and size, embedded devices usually have much less RAM and secondary storage than desktop computers, and are likely to use flash memory instead of a hard drive. Since embedded devices serve specific rather than general purposes, developers optimize their embedded Linux distributions to target specific hardware configurations and usage situations. These optimizations can include reducing the number of device drivers and software applications, and modifying the Linux kernel to be a real-time operating system.

Overview of How-To

This How-To is meant to be a starting point for people to learn build a kernel image for IGEP v2 devices as quickly and easily as possible.

This How-To works with the Ubuntu 8.04 IGEP v2.0 SDK Virtual Machine but most of the contents are valid also for other GNU/Linux distributions. We do not issue any guarantee that this will work on other distributions.

Requirements

Ubuntu 8.04 IGEP v2.0 SDK Virtual Machine, you can download registering as a user to http://www.myigep.com. It's free !

Supported kernels

First of all setup the build environment sourcing this script

```
$ source /usr/local/poky/eabi-glibc/arm/environment-setup
```

Kernel image for IGEP v2 Rev. B

Linux OMAP v2.6.28 (stable)

Latest stable kernel version is: 2.6.28.10-igep0020b-2

```
$ wget http://downloads.myigep.com/sources/kernel/linux-omap-[kernel version].tar.gz
```

Download the IGEP v2 Rev. B Linux kernel sources and follow next steps:

```
$ tar xzf linux-omap-[kernel version].tar.gz

$ cd [kernel version]

$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-igep0020b_defconfig

$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- uImage modules
```

The result will be an uImage file in arch/arm/boot directory. You can install the kernel modules to your target rootfs

```
$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- modules_install INSTALL_MOD_PATH=[path to your target rootfs]
```

Linux OMAP v2.6.28.10 (development)

Clone the GIT repository from git.myigep.com

```
$ git clone http://git.myigep.com/pub/scm/linux-omap-2.6.git

$ cd linux-omap-2.6
```

```
$ git checkout origin/linux-2.6.28.y-igep0020b -b linux-2.6.28.y-igep0020b
```

and build with

```
$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-
igep0020b_defconfig

$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- uImage modules
```

The result will be an uImage file in arch/arm/boot directory. You can install the kernel modules to your target rootfs

```
$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- modules_install
INSTALL_MOD_PATH=[path to your target rootfs]
```

Linux Android 2.6 (development)

Clone the GIT repository from git.myigep.com. Based on rowboat project tree.

```
$ git clone http://git.myigep.com/pub/scm/linux-omap-2.6.git

$ cd linux-omap-2.6

$ git checkout origin/linux-android-2.6 -b linux-android-2.6
```

and build with

```
$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-
igep0020_android_defconfig

$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- uImage modules
```

The result will be an uImage file in arch/arm/boot directory. You can install the kernel modules to your target rootfs

```
$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- modules_install
INSTALL_MOD_PATH=[path to your target rootfs]
```

Linux mainline tree (development)

Here is where the development work takes place and you should use this if you're after to work with the latest cutting edge developments. It is possible 'master' can suffer temporary periods of instability while new features are developed and if this is undesirable we recommend using one of the release branches. Additional patches can be found at http://patchwork.kernel.org/project/linux-omap/list

Clone GIT repository from git.kernel.org

```
$ git clone git://git.kernel.org/pub/scm/linux/kernel/git/tmlind/linux-omap-2.6.git

$ cd linux-omap-2.6
```

and build with

```
$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-
igep0020_defconfig

$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- uImage modules
```

The result will be an uImage file in arch/arm/boot directory. You can install the kernel modules to your target rootfs

```
$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- modules_install
INSTALL_MOD_PATH=[path to your target rootfs]
```

6.1.3 HOW TO UPGRADE THE FACTORY FIRMWARE

Overview of How-To

This How-To is meant to be a starting point for people to learn how to upgrade the factory firmware for IGEP v2 devices as quickly and easily as possible.

This How-To works with the Ubuntu 8.04 IGEP v2.0 SDK Virtual Machine but most of the contents are valid also for other GNU/Linux distributions. We do not issue any guarantee that this will work on other distributions.

Requirements

Ubuntu 8.04 IGEP v2.0 SDK Virtual Machine, you can download registering as a user to http://www.myigep.com. It's free !

Upgrade the factory firmware for IGEP v2 rev. B

Upgrade the firmware using an NFS-TFTP environment

First of all setup a poky-image-sato NFS-TFTP environment using latest images as explained in Ubuntu 8.04 IGEP v2.0 SDK Virtual Machine (chapter 4.1.2)

With this kernel the partitions looks like,

```
# cat /proc/mtd

dev:    size    erasesize  name

mtd0: 00080000 00040000 "X-Loader"
```

```
mtd1: 00180000 00040000 "U-Boot"

mtd2: 00080000 00040000 "U-Boot Env"

mtd3: 00300000 00040000 "Kernel"

mtd4: 1fa80000 00040000 "File System"
```

To flash files to mtd partitions just use the nandwrite program. For example, to flash the firmware factory v3.2.0-0 download from

http://downloads.myigep.com/firmwares/firmware-factory-igep0020b-3.2.0-0.tar.gz

and copy to target directory /home/root in your board, then follow next steps

```
# cd /home/root/firmware-factory-igep0020b-3.2.0-0

# for i in 0 1 2 3 4; do flash_eraseall /dev/mtd${i}; done

# nandwrite -p /dev/mtd0 x-load-ddp-1.4.2-1.igep0020b-flash.bin.ift

# nandwrite -p /dev/mtd1 u-boot-arm-2009.11-0.igep0020b.bin

# nandwrite -p /dev/mtd2 u-boot-environment.bin

# nandwrite -p /dev/mtd3 uImage-2.6.28-r3-igep0020b-20100119164554.bin

# nandwrite -p /dev/mtd4 poky-image-sato-igep0020b-20100119164554.rootfs.jffs2
```

Restart your board and enjoy your new firmware. That's all folks.

6.1.4 HOW TO SETUP WIFI

Overview of How-To

This How-To is meant to be a starting point for people to learn setup the wifi on IGEP v2 devices as quickly and easily as possible.

Requirements

IGEP v2 board (IGEP0020-RB1) with factory defaults.

Feedback and Contributing

At any point, if you see a mistake or want to contribute to this How-To, you can send me an email at support@iseebcn.com

Setup basics

- libertas: Marvell Libertas 8385/8686/8688 SDIO 802.11b/g card

- firmware: 9.70.3p24 (download from http://extranet.marvell.com/drivers/driverDisplay.do?driverId=203)

SDIO card should be showed after the image is downloaded to the board.

```
mmc1: new SDIO card at address 0001
```

The firmware binaries sd8686_helper.bin and sd8686.bin should be in /lib/firmware directory.

To make the SDIO WIFI module work load the libertas_sdio module

```
# modprobe libertas_sdio

libertas_sdio: Libertas SDIO driver

libertas_sdio: Copyright Pierre Ossman

libertas_sdio mmc1:0001:1: firmware: requesting sd8686_helper.bin

libertas_sdio mmc1:0001:1: firmware: requesting sd8686.bin

libertas: 00:13:e0:c3:0c:3c, fw 9.70.3p24, cap 0x00000303

libertas: unidentified region code; using the default (USA)

libertas: PREP_CMD: command 0x00a3 failed: 2

libertas: PREP_CMD: command 0x00a3 failed: 2

libertas: eth1: Marvell WLAN 802.11 adapter
```

Now you can connect this wifi module to an AP. First of all, you'll check if your devices is detected.

```
# iwconfig

eth1     IEEE 802.11b/g  ESSID:""

        Mode:Managed  Frequency:2.412 GHz  Access Point: Not-Associated

        Bit Rate:0 kb/s   Tx-Power=18 dBm

        Retry short limit:8   RTS thr=2347 B   Fragment thr=2346 B
```

```
      Encryption key:off

      Power Management:off

      Link Quality:0  Signal level:0  Noise level:0

      Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0

      Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

Next, you will set up the interface

```
# ifconfig eth1 up

eth1     Link encap:Ethernet  HWaddr 00:13:E0:C3:0C:3C

         UP BROADCAST MULTICAST  MTU:1500  Metric:1

         RX packets:0 errors:0 dropped:0 overruns:0 frame:0

         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0

         collisions:0 txqueuelen:1000

         RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

and you can scan for an AP

```
# iwlist eth1 scan

Cell 04 - Address: 00:18:84:81:46:E2

                ESSID:"MyPlace"

                Mode:Managed

                Frequency:2.427 GHz (Channel 4)

                Quality=100/100  Signal level=-39 dBm  Noise level=-96 dBm

                Encryption key:off

                Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 6 Mb/s; 9 Mb/s

                    11 Mb/s; 12 Mb/s; 18 Mb/s; 24 Mb/s; 36 Mb/s

                    48 Mb/s; 54 Mb/s
```

Now, is time to associate to your AP

```
# iwconfig eth1 txpower auto essid MyPlace channel 4
```

```
eth1    IEEE 802.11b/g  ESSID:"MyPlace"

        Mode:Managed  Frequency:2.427 GHz  Access Point: 00:18:84:81:46:E2

        Bit Rate:0 kb/s   Tx-Power=13 dBm

        Retry short limit:8   RTS thr=2347 B   Fragment thr=2346 B

        Encryption key:off

        Power Management:off

        Link Quality=97/100  Signal level=-43 dBm  Noise level=-94 dBm

        Rx invalid nwid:0  Rx invalid crypt:3109  Rx invalid frag:0

        Tx excessive retries:13  Invalid misc:3315   Missed beacon:0
```

and get and ip address

```
# udhcpc -i eth1

udhcpc (v1.9.1) started

Sending discover...

Sending select for 192.168.10.216...

Lease of 192.168.10.216 obtained, lease time 43200

adding dns 192.168.10.1
```

Last, you can test the network interface.

```
# ping -c 1 192.168.10.1

PING 192.168.10.1 (192.168.10.1): 56 data bytes

64 bytes from 192.168.10.1: seq=0 ttl=64 time=16.327 ms

--- 192.168.10.1 ping statistics ---

1 packets transmitted, 1 packets received, 0% packet loss

round-trip min/avg/max = 16.327/16.327/16.327 ms
```

6.1.5 HOW TO SETUP BLUETOOTH

Overview of How-To

This How-To is meant to be a starting point for people to learn setup the bluetooth on IGEP v2 devices as quickly and easily as possible.

Requirements

IGEP v2 board (IGEP0020-RB1) with factory defaults.

Feedback and Contributing

At any point, if you see a mistake or want to contribute to this How-To, you can send me an email at support@iseebcn.com

Setup basics

The IGEP v2 has built-in a Class 2 Bluetooth 2.0 + EDR device.

Class	Maximum Permitted Power mW (dBm)	Range (approximate)
Class 1	100 mW (20 dBm)	~100 meters
Class 2	2.5 mW (4 dBm)	~10 meters
Class 3	1 mW (0 dBm)	~1 meter

For best performance you need to disable WIFI, you can do this

```
# echo 0 > /sys/class/gpio/gpio94/value
```

By default the WIFI-BT module is not configured for single antenna, so you need to configure this, set registers settings as follows:

```
Register offset        Register Name          8686 Bx

0x8000_A58CBT Switch Control     0x40865

0x8000_A5A03/4W BCA mode config       0xD24D

0x8000_A5F0BCA WLAN modes     0xA027181C

# mount -t debugfs none /sys/kernel/debug



# cd /sys/kernel/debug/lbs_wireless/eth1/registers/
```

```
# echo "0xa58c 0x40865" > wrmac

# echo "0xa5a0 0xd24d" > wrmac

# echo "0xa5f0 0xa027181c" > wrmac
```

Next, you will need to load some bluetooth modules

```
# modprobe bluetooth

# modprobe hci_uart

# modprobe l2cap

# modprobe rfcomm

# dmesg

Bluetooth: Core ver 2.13

NET: Registered protocol family 31

Bluetooth: HCI device and connection manager initialized

Bluetooth: HCI socket layer initialized

Bluetooth: L2CAP ver 2.11

Bluetooth: L2CAP socket layer initialized

Bluetooth: BNEP (Ethernet Emulation) ver 1.3

Bluetooth: BNEP filters: protocol multicast

Bluetooth: RFCOMM socket layer initialized

Bluetooth: RFCOMM TTY layer initialized

Bluetooth: RFCOMM ver 1.10
```

With your preferred editor create a bluez.psr file like this

```
// PSKEY Parameters for Bluetooth RF

// (1) 0x0031 (PSKEY_LC_ENHANCED_POWER_TABLE)

&0031 = 0900 0000 3f00 4700 ec00 0f00 0000 3f00 4a00 f000 1600 0000 3f00
4d00 f400 1e00 0000 \

3f00 5000 f800 2600 0000 3f00 5500 fc00 2e00 0000 3f00 5b00 0000 3800 0000
3f00 6900 0400
```

```
// (2) 0x01f6 (PSKEY_ANA_FTRIM)

&01f6 = 001d

// (3) 0x01fe (PSKEY_ANA_FREQ)

&01fe = 6590    // 26MHz reference clock

// (4) 0x01be (PSKEY_UART_BAUDRATE)

&01be = 0x1d8  // 115200

// (5) 0x0028 (PSKEY_LC_COMBO_DISABLE_PIO_MASK)

&0028 = 0200 0000 0000

// (6) 0x002a (PSKEY_LC_COMBO_DOT11_CHANNEL_PIO_BASE)

&002a = 0011
```

And send the file using the bccmd tool.

```
# bccmd -t bcsp -d /dev/ttyS1 psload -r bluez.psr

Loading PSKEY_LC_ENHANCED_POWER_TABLE ... done

Loading PSKEY_ANA_FTRIM ... done

Loading PSKEY_ANA_FREQ ... done

Loading PSKEY_UART_BAUDRATE...done

Loading PSKEY_LC_COMBO_DISABLE_PIO_MASK ... done

Loading PSKEY_LC_COMBO_DOT11_CHANNEL_PIO_BASE ... done
```

Finally you can attach to device an scan for other bluetooth devices.

```
# hciattach -s 115200 /dev/ttyS1 bcsp 115200 noflow

# hciconfig hci0 up

# hciconfig hci0 piscan

# hciconfig -a hci0

hci0:   Type: UART

        BD Address: 00:02:5B:00:A5:A5 ACL MTU: 310:10 SCO MTU: 64:8

        UP RUNNING PSCAN ISCAN

        RX bytes:1376 acl:0 sco:0 events:25 errors:0
```

```
        TX bytes:740 acl:0 sco:0 commands:24 errors:0

        Features: 0xff 0xff 0x8f 0xfe 0x9b 0xf9 0x00 0x80

        Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3

        Link policy: RSWITCH HOLD SNIFF PARK

        Link mode: SLAVE ACCEPT

        Name: 'igep0020b (0)'

        Class: 0x120112

        Service Classes: Networking, Object Transfer

        Device Class: Computer, Handheld

        HCI Ver: 2.0 (0x3) HCI Rev: 0xc5c LMP Ver: 2.0 (0x3) LMP Subver: 0xc5c

        Manufacturer: Cambridge Silicon Radio (10)

# hcitool -i hci0 scan

Scanning ...

     00:1E:A4:FC:EC:2E        Phone
```

6.1.6 UBUNTU 8.04 IGEP V2.0 SDK VIRTUAL MACHINE

Overview of How-To

This How-To is meant to be a starting point for people who use the Ubuntu 8.04 IGEP v2.0 SDK Virtual Machine and wants to learn how to add some enhancements to the VM.

Requirements

Ubuntu 8.04 IGEP v2.0 SDK Virtual Machine, you can download registering as a user to http://www.myigep.com. It's free !

IGEP v2 board (IGEP0020-RB1) with factory defaults.

Feedback and Contributing

At any point, if you see a mistake or want to contribute to this How-To, you can send me an email at support@iseebcn.com

Virtual Machine Enhancements

Rootfs demos using an NFS-TFTP environment for IGEP v2 Rev. B

The simplest way to create a new NFS-TFTP environment for IGEP v2 Rev. B is to use an already working filesystem and prebuilt kernel image. Prebuilt images are also available.

poky-image-minimal - A small image, just enough to allow a device to boot

Poky Purple 3.2 based image.

Download and install the root filesystem image (poky-image-minimal-igep0020b.cpio) as root on the NFS server,

```
$ sudo mkdir -p /srv/nfs/poky/poky-image-minimal/igep0020b

$ cd /srv/nfs/poky/poky-image-minimal/igep0020b

$ sudo wget http://downloads.myigep.com/rootfs/poky-image-minimal-igep0020b.cpio

$ sudo cpio -idm < poky-image-minimal-igep0020b.cpio
```

Download and copy a pre-built kernel image to poky-image-minimal project

```
$ sudo wget http://downloads.myigep.com/rootfs/uImage-2.6.28-igep0020b.bin

$ sudo mkdir -p /srv/tftp/poky/poky-image-minimal/igep0020b

$ sudo mv uImage-2.6.28-igep0020b.bin /srv/tftp/poky/poky-image-minimal/igep0020b/

$ sudo ln -s uImage-2.6.28-igep0020b.bin /srv/tftp/poky/poky-image-minimal/igep0020b/uImage
```

Export the directory tree with an entry in /etc/exports file editing with your preferred editor, like

```
/srv/nfs/poky/poky-image-minimal/igep0020b
*(rw,no_root_squash,no_subtree_check,sync)
```

and restart the NFS server

```
$ sudo /etc/init.d/nfs-kernel-server restart
```

Now, you can power up your board, stop at u-boot, set the project variable point to poky-image-minimal and boot via NFS

```
# setenv project poky-image-minimal
```

```
# run nfs-boot
```

poky-image-sato - X11 image with Sato theme and Pimlico applications.

Poky Purple 3.2 based image.

Download and install the root filesystem image (poky-image-sato-igep0020b.cpio) as root on the NFS server,

```
$ sudo mkdir -p /srv/nfs/poky/poky-image-sato/igep0020b

$ cd /srv/nfs/poky/poky-image-sato/igep0020b

$ sudo wget http://downloads.myigep.com/rootfs/poky-image-sato-igep0020b.cpio

$ sudo cpio -idm < poky-image-sato-igep0020b.cpio
```

Download and copy a pre-built kernel image to poky-image-sato project

```
$ sudo wget http://downloads.myigep.com/rootfs/uImage-2.6.28-igep0020b.bin

$ sudo mkdir -p /srv/tftp/poky/poky-image-sato/igep0020b

$ sudo mv uImage-2.6.28-igep0020b.bin /srv/tftp/poky/poky-image-
sato/igep0020b/

$ sudo ln -s uImage-2.6.28-igep0020b.bin /srv/tftp/poky/poky-image-
sato/igep0020b/uImage
```

Export the directory tree with an entry in /etc/exports file editing with your preferred editor, like

```
/srv/nfs/poky/poky-image-sato/igep0020b
*(rw,no_root_squash,no_subtree_check,sync)
```

and restart the NFS server

```
$ sudo /etc/init.d/nfs-kernel-server restart
```

Now, you can power up your board, stop at u-boot, set the project variable point to poky-image-sato and boot via NFS

```
# setenv project poky-image-sato

# run nfs-boot
```

poky-image-demo - X11 image with Sato theme with SGX demos

Poky Purple 3.2 based image.

Download and install the root filesystem image (poky-image-demo-igep0020b.cpio) as root on the NFS server,

```
$ sudo mkdir -p /srv/nfs/poky/poky-image-demo/igep0020b

$ cd /srv/nfs/poky/poky-image-demo/igep0020b

$ sudo wget http://downloads.myigep.com/rootfs/poky-image-demo-
igep0020b.cpio

$ sudo cpio -idm < poky-image-demo-igep0020b.cpio
```

Download and copy a pre-built kernel image to poky-image-demo project

```
$ sudo wget http://downloads.myigep.com/rootfs/uImage-2.6.28-igep0020b.bin

$ sudo mkdir -p /srv/tftp/poky/poky-image-demo/igep0020b

$ sudo mv uImage-2.6.28-igep0020b.bin /srv/tftp/poky/poky-image-
demo/igep0020b/

$ sudo ln -s uImage-2.6.28-igep0020b.bin /srv/tftp/poky/poky-image-
demo/igep0020b/uImage
```

Export the directory tree with an entry in /etc/exports file editing with your preferred editor, like

```
/srv/nfs/poky/poky-image-demo/igep0020b
*(rw,no_root_squash,no_subtree_check,sync)
```

and restart the NFS server

```
$ sudo /etc/init.d/nfs-kernel-server restart
```

Now, you can power up your board, stop at u-boot, set the project variable point to poky-image-minimal and boot via NFS

```
# setenv project poky-image-demo

# run nfs-boot
```

7 FAQS

7.1 HARDWARE RELATED QUESTIONS

1. Is it possible to connect a battery to maintain the real time clock in the IGEPv2 Board?

Battery backup is not implemented on IGEPv2 RA and RB.

User can use i2c external rtc or connect externally a super-cap/rechargable battery on R768 pins.

R768 is 0 Ohm resistor connected between GND and BKBAT (M14) input from TPS65950.

2. How is it connected Bluetooth chipset to OMAP?

UART2.

3. How many UARTs are available on IGEPv2 board?

OMAP3 has 3 UARTS:

UART1 --> RS485 on connector J940

UART2 --> bluetooth

UART3 --> debug console on connector J960

All of this UART are also available on other connectors:

UART1 --> on connector JA41 without DVI transceiver with CMOS 1v8 logic.

UART1 --> on connector J940 with CMOS 3V3 logic

UART2 --> on expansion connector J990 without bluetooth feature.

UART3 --> on connector JA41 without DVI transceiver with CMOS 1v8 logic.

4. Is it available s-video or video composite output on IGEPv2 board?

A dual-display interface equips the OMAP3530/25 processor. This display subsystem provides the necessary control signals to interface the memory frame buffer directly to the external displays (TV-set).

Two (one per channel) 10-bit current steering DACs are inserted between the DSS and the TV set to generate the video analog signal. One of the video DACs also includes TV detection and power-down mode. For more information, see the DSS chapter of the OMAP35x Technical Reference Manual.

IGEPv2 board hasn't s-video or video composite output connector, but there are available on the board some test points to implement a fully-functionaly s-video output.

5. Is it possible to config the wireless LAN and bluetooth so they are completely off, and not consuming any power at all?

Yes, powerdown and reset are implemented on GPIO_94 and GPIO_95.

6. Could you attach an LCD with LDVS interface to the IGEPv2?

You have to translate it to LVDS which is a differential and serialized interface. For example you can use MAX9213 serializer LVDS to translate the parallel interface to LVDS.

7. Is there a way to attach a VGA monitor directly to the IGEPv2?

You can use DVI to VGA converter connected to DVI output.

8. Is it possible to connect a CMOS sensor to the IGEPV2 board (ie are the camera signals available on a board connector)?

The camera signals are not available on IGEPv2 board.

9. Mini usb OTG must be a MINI A Male?

Review the Hardware Reference Manual: Page7: USB 2.0 LS/FS/HS OTG 1 Mini AB USB socket connector (dual slave and host role) – That means you can use MINI-A and MINI-B cable.

10. I would like too to know if the external wifi antena is easy to install? where can I aqcuire a compatible external antena?

The JD11 is a GSC connector for the external Wifi interface. It is a MURATA GSC connector, Part number MM9329-2700RA1.

You only have to find a cable that fits with this MURATA connector.

Regarding the antenna, just need a 2.4Ghz Antenna.

11. Is it possible to add analog video output to the IGEPv2 platform?

Yes it is possible. There is no connector for S-video, but signals are available on board test points.

TP400 and TP401 which are located on bottom side, belongs to luminance and chroma so:

TP401 = TVOUT_1R sch signal = PIN 3 S-video connector = Y = luminance

TP400 = TVOUT_2R sch signal = PIN 4 S-video connector= C = chroma

12. Which IGEPv2 revision is my board?

At this moment, probably it is a IGEPv2-RB. Only few premium costumers have early IGEPv2-RA prototypes boards.

If it is IGEPv2-RB revision, it could see silkscreen text "IGEP0020-RB1" near J800 (USB Host connector)

13. Is it possible to power the IGEPv2 board using USB OTG connector?

No, it is not possible.

14. What is the maximun power consumption of the IGEPv2 board ?

The power consumption of IGEPv2@600Mhz is between 690 and 720mA

The power consumption of IGEPv2@720Mhz is between 750 and 780mA.

IGEPv2 720MHz has aprox. 15% more power consumption on core and mpu_iva than 600MHZ.

7.2 SOFTWARE RELATED QUESTIONS

1. Are the sources for the xloader, uBoot and kernel supplied with the IGEPv2 board available?

All of them are available on www.myigep.com if you are a registered user.

8   CHANGELOG

IGEPv2 BOARD SDK USER MANUAL (Revision 1.00)

- Initial internal and partners version

IGEPv2 BOARD SDK USER MANUAL (Revision 1.01)

- Initial public version

IGEPv2 BOARD SDK USER MANUAL (Revision 1.02)

- General modifications to update manual to igep0020b.

- New Chapter 3.7.3 POKY-APPLICATION-SGX DEMOS

- Revision Chapter 3.8 Creating a custom root file system

- New Chapter 6. FAQS

IGEPv2 BOARD SDK USER MANUAL (Revision 1.03)

- New Chapter HOW TO…

- Corrections on chapter 3.8