



Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!



Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832

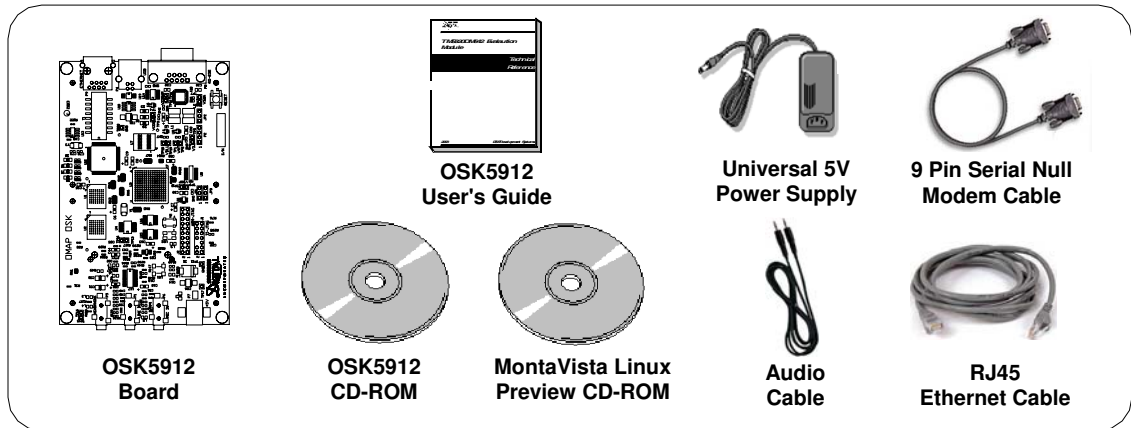
Email & Skype: info@chipsmall.com Web: www.chipsmall.com

Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China



OMAP5912 Starter Kit (OSK5912)

Users Guide



OSK5912 Package Contents

This kit contains an OSK5912 development board and associated cables, an OSK5912 CD-ROM containing a collection of board and chip specific packages and documentation and a MontaVista Linux Preview CD-ROM that contains a suite of development tools for Linux along with an OMAP5912 specific Linux kernel. A version of the kernel and a live file system are burned into the on-board Flash to provide basic standalone Linux functionality without an attached host.

To do ARM **Linux development** with the included MontaVista Linux Preview Kit, you must have:

- A Linux PC running Red Hat Linux 7.3 or 9.0
- 1 GB free disk space
- Serial port
- Local CD-ROM drive
- Ethernet adapter

To do **DSP development**, you must have:

- A Windows PC running Code Composer Studio 2.21
- A CD-ROM drive
- A Code Composer compatible JTAG emulator

or

- A Linux PC and Linux DSP Tools from Texas Instruments

Table of Contents

1 Quick Start (All Users)

- 1.1 Booting the OSK5912
- 1.2 Running the Audio Demo
- 1.3 Documentation and Support

2 Developing with Linux

- 2.1 Overview
- 2.2 Installing the MontaVista Linux Preview Kit
- 2.3 Making a local copy of the Preview Kit
- 2.4 Building the Linux Kernel
- 2.5 Booting the OSK with a Target File System
- 2.6 Building and Running a "Hello World" Application

3 Developing DSP Applications

- 3.1 Overview
- 3.2 DSP Development with Code Composer Studio and a JTAG Emulator
- 3.3 DSP Application Development - Linux DSP Tools

4 Configuring Code Composer Studio

- 4.1 Overview
- 4.2 Configuring Code Composer Studio
- 4.3 The LED Example

5 Board Description

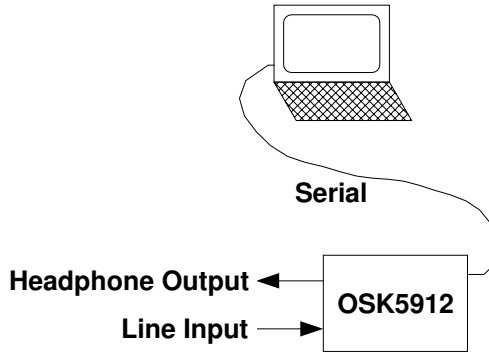
- 5.1 Board Overview

Quick Start

1.1 Booting the OSK5912

The OSK5912 ships with a Linux kernel and a JFFS2 flash file system stored into the Flash so it can boot Linux with only a terminal connected for console input/output. To boot your OSK5912 for the first time and run a quick demo:

- 1) Connect the OSK5912 to a host PC through the included 9-pin RS-232 null modem serial cable.



- 2) Connect an audio source like a CD player or laptop to the line input of the OSK. Also connect a set of headphones or powered speakers to the headphone output so you can hear the output.
- 3) Start a terminal program such as HyperTerminal on Windows. The serial port should be configured to 115,200 baud, no parity, 8 data bits, 1 stop bit with no flow control.
- 4) Power up the OSK5912 by using the included AC adapter. The AC adapter should be connected to P7 which is the +5V input supply jack. U-Boot will respond with a header that looks like this:

```
U-Boot 1.1.1 (Jul 19 2004 - 09:17:40)

U-boot code: 11080000 -> 11095D6C   BSS: -> 1109A53C
DRAM Configuration:
Bank #0: 10000000 32 MB
Micron StrataFlash MT28F128J3 device initialized
Flash: 32MB
In:   serial
Out:  serial
Err:  serial

Hit any key to stop autoboot: 10
```

- 5) In a few seconds, the Linux kernel will boot like this:

```
Starting kernel ...

Uncompressing Linux ..... done, booting the kernel
```

6) When the boot sequence is complete, you will see a login prompt. Log in as **root** with no password.

MontaVista(R) Linux(R) Professional Edition 3.1, Preview Kit

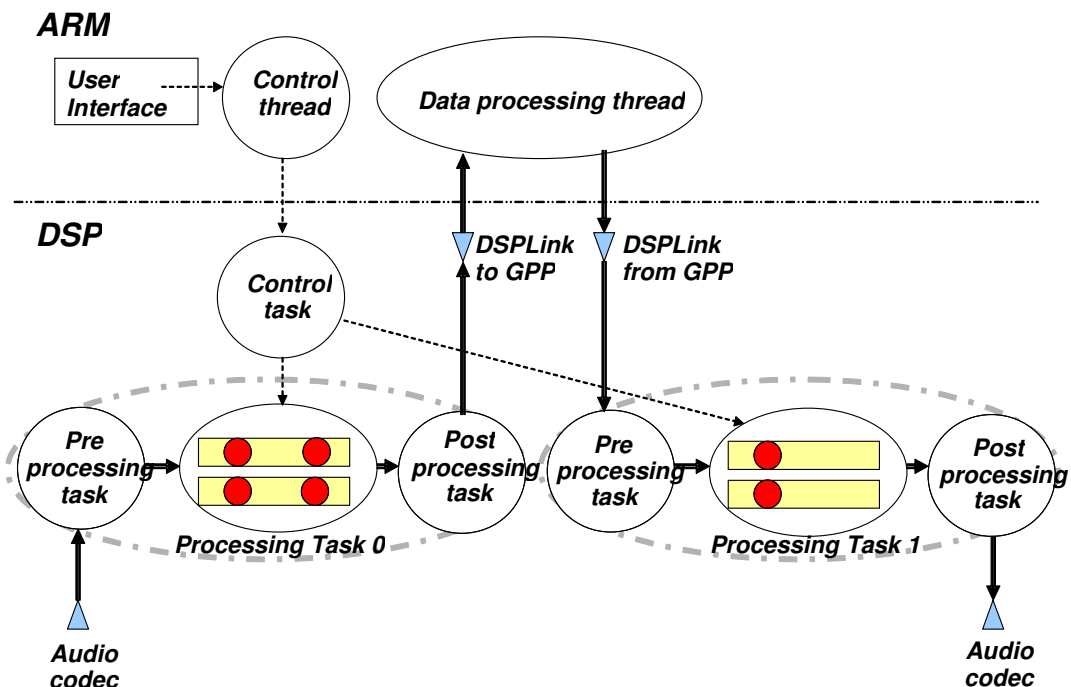
(none) login:

You are now in a full Linux shell environment and can operate as if you were running on a standalone workstation.

1.2 Running the Audio Demo

A demo that uses TI's Reference Frameworks 6 and DSP/BIOS™ Link software to filter audio through the AIC23 codec using the C55x DSP in the OMAP5912 is also stored in the file system. This demo highlights the inter-processor communication between the ARM processor running Linux and the audio application running on the DSP.

The demo processes an incoming stereo audio signal on the DSP, then sends the data to the GPP, which has the option to process the data before sending it back to the DSP for further processing. Finally, it sends the output signal to the codec. A control thread on the ARM sets DSP algorithm parameters such as volume and high-/low-pass filter coefficients. The figure below illustrates this process. For more information on this application, inter-processor communication, and DSP application development in general, please refer to section 3.1 of this document.



To run the audio demo:

- 1) Change your working directory to the demo home:

```
cd /opt
```

- 2) Now load the DSP/BIOS™ Link module by typing:

```
./dsplinkload
```

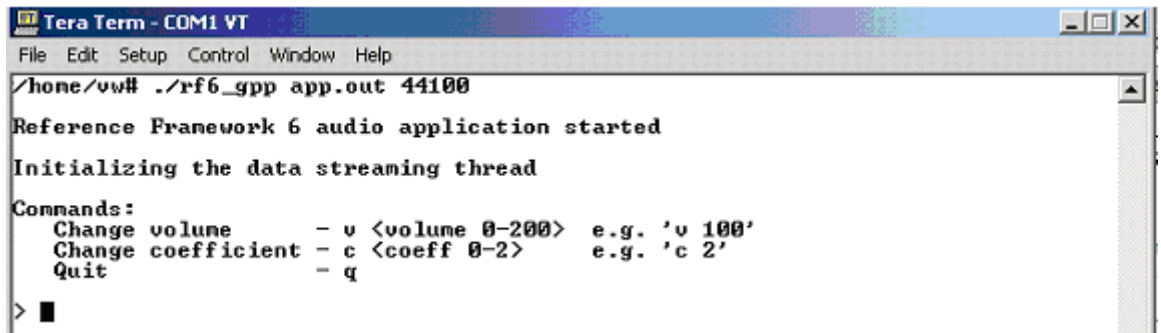
- 3) Change your working directory to the location of the demo executable:

```
cd rf6_audio
```

- 4) Launch the demo with:

```
./rf6_gpp app.out 44100
```

- 5) You will hear the audio source connected to the line input on the headphone output. The audio is being passed through a FIR filter running on the DSP. A small command menu and prompt will be displayed:



```
Tera Term - COM1 VT
File Edit Setup Control Window Help
/home/vw# ./rf6_gpp app.out 44100
Reference Framework 6 audio application started
Initializing the data streaming thread
Commands:
Change volume - v <volume 0-200> e.g. 'v 100'
Change coefficient - c <coeff 0-2> e.g. 'c 2'
Quit - q
> █
```

Within this application, you can send either filter coefficient change or volume change commands. The table below shows the acceptable commands:

Command	Use	Acceptable value range
c <value>	Change FIR filter coefficients	0 (low-pass filter) 1 (high-pass filter) 2 (all-pass filter)
v <value>	Change volume	0 to 200 (% of input volume)
q	Quit application gracefully	

6) For example, to select a all-pass filter, use the following command:

```
c 2
```

You will immediately hear a different frequency response as a result of the new coefficients.

1.3 Documentation and Support

Documentation and support is limited to the contents of this package. The contents are indexed in the file:

```
readme.htm
```

at the root of the OSK5912 CD-ROM. Useful links and software updates can be found on the Internet at:

<http://omap.spectrumdigital.com>



Spectrum Digital warrants each board for a period of 90 days from the date of purchase. Hardware problems can be reported to **support@spectrumdigital.com**. OMAP related technical documentation from Texas Instruments can be found at:

<http://www.ti.com/5912osk>



Information related to Linux on OMAP processors as well as a mailing list dedicated to OMAP Linux users can be found at:

<http://linux.omap.com>

Developing with Linux

2.1 Overview

MontaVista Software is a premier vendor of embedded Linux products and services. The OSK5912 ships with a copy of the MontaVista Linux Professional Edition Preview Kit which includes all of the tools necessary to build custom kernels and applications for the OSK5912 using a Linux PC running Red Hat Linux 7.3 or 9.0.

The MontaVista Professional Edition Preview Kit included in this package is a free distribution of a Linux board support package for the OSK5912. It comes with no support from MontaVista Software, Spectrum Digital, Inc. or Texas Instruments, Inc. The full MontaVista Professional Edition BSP is available from MontaVista Software.

2.2 Installing the MontaVista Preview Kit

These instructions will install the Preview Kit in `/opt/montavista`. Before you begin, make sure your Linux host machine meets the requirements listed below:

- Linux host PC running Red Hat 7.3 or Red Hat 9.0
- This should have minimum 256MB RAM, 1Gb free hard disk space and an Ethernet connection.
- System services for TELNETD, FTPD, NFSD, (+SMBD) should be installed and started.

To make sure the services listed above are indeed running, execute the following on your Linux host machine:

```
/sbin/service nfs status
/sbin/service xinetd status
/sbin/service smb status
```

They should all be reported as running (if samba is installed). To install the Preview Kit for on your host system, complete the following steps:

- 1) **Some portions of the installation process must be done as root and some as a user. Please change to the root user with:**

```
su root
```

- 2) **Register with MontaVista to obtain the required password for installation from the following Web site:**

```
http://www.mvista.com/previewkit/index.html
```


- 3) Insert the MontaVista Preview Kit CD-ROM included in the OSK box into your host machine's CD-ROM drive. If the CD does not automount, mount the CD-ROM. Use the following command:

```
mount /mnt/cdrom
```

Note: If this command fails to mount the CD-ROM, you must specify all of the parameters to the mount command. See your system administrator if you need assistance.

- 4) Change to the directory on the CD where the installer is located. Enter the following command:

```
cd /mnt/cdrom
```

- 5) Installation of Preview Kit for Pro 3.1 is performed by the script "install_previewkit". Enter the following command:

```
./install_previewkit
```

Note: To view a list of the script's options, enter the following command:

```
./install_previewkit --help
```

- 6) The following output appears:

```
Welcome to the MontaVista® Linux® Preview Kit version 3.1.
```

- 7) When prompted to enter a password, enter the Preview Kit for Pro CD password that you obtained in step 1. You are given three attempts to enter the correct password before the installation process fails.

```
Please enter the MontaVista® Linux® Preview Kit password.  
Password:
```

- 8) When prompted to enter the installation path, enter the path where you would like the preview kit installed. The default path is /opt – to accept this path (recommended), press the enter key.

```
Where should the Preview Kit be installed? [ /opt ]
```

- 9) Select "1" as the Linux Support Package to install.

```
Architecture  Linux Support Package
```

```
-----  
1. arm_v4t_le  ti-omap5912-osk
```

```
Choose the LSP/Architecture to install by number.  
Separate multiple entries by a space.  
(? to list, q to quit): 1
```

- 10) When prompted to confirm that the LSP you chose to install is correct, enter **y** to confirm. Wait for the configuration and installation to complete. Once complete, output similar to the following appears:

```
Finished Installing the MontaVista Linux Preview Kit!
```

2.3 Installing a Local Copy of the Preview Kit

The following steps will make a local working copy of the kernel and file system from the installation directory (`/opt/montavista`) in your local user directory (usually `/home/username` where *username* is your login ID):

- 1) Make a directory called *montavista* in your home directory (`/home/username`). Since we want to own this directory as user, type `'su username'` to change back to normal user mode. Change back to your home directory by executing `'cd ~'`. Then execute `'mkdir montavista'` when your current directory is your home directory. Change directory to the created MontaVista directory by executing `'cd montavista'` and create a new directory called *filesys* with the command `'mkdir filesys'`. Now copy the target file system recursively to our local directory, while preserving all file attributes. Change back to the root user by executing `'su root'` followed by the root password. Then execute `'cp -a /opt/montavista/previewkit/arm/v4t_le/target/* filesys/'`.
- 2) The reason for making a local target file system, is that we want to put our own executables on the target file system. But since we have copied the files and directories as root, we won't have permission to put our executables in the file system. Therefore we make the directory `/opt` on the target file system owned by the user by executing `'chown username:username filesys/opt'`.
- 3) Now we want to make a local embedded Linux kernel source tree. Execute `'exit'` to change back to normal user mode, then execute `'mkdir kernel'`. Now copy the kernel source tree to our local directory recursively (but not preserving file permissions, do `'man cp'` for details). We do this by executing `'cp -R /opt/montavista/previewkit/lsp/ti-omap5912_osk-previewkit-arm_v4t_le/linux-2.4.20_mvl31 kernel/'`.

↖
The mvl31 portion of this line is hard to read. In upper case, it would read MVL31 followed by a space.

- 4) A feature in the Linux file system is the concept of logical links. A logical link is created in order to access a directory or a file from another place in the file system. It is also a good way of keeping multiple versions of a file or a tree, but still only have one access point. Change directory to the *kernel* directory using `'cd kernel'` and execute `'ln -s linux-2.4.20_mvl31 linux'`. Execute `'ls -l'` and you should see the following:

```
lrwxrwxrwx  1 username  username          18 May 23 14:09 linux -> linux-
2.4.20_mvl31
drwxr-xr-x 14 username  username      4096 Jun  5 10:08 linux-2.4.20_mvl31
```

The logical link Linux points to the directory linux-2.4.20-mvl31, which means that the latter directory now can be accessed by just using the link Linux. Now we could install another kernel tree, and by just changing the Linux link, we would not have to alter any paths in our Makefiles etc.

2.4 Building the Linux Kernel

This section describes how to build the local copy of the Linux kernel. Re-building the kernel is necessary if you change any kernel code or want to compile the kernel with different features turned on/off. The build process also compiles any kernel modules that are necessary for operation.

- 1) Since we want to use the cross compiler to generate an ARM Linux image, we need to add the path to our cross compiler tools. Execute `export PATH="/opt/montavista/previewkit/arm/v4t_le/bin:/opt/montavista/previewkit/host/bin:/opt/montavista/previewkit/host/lib:$PATH"` to do this.
- 2) Change directory to the *kernel root* directory by executing `'cd linux'`. First we want to configure the kernel. We do this by executing `'make menuconfig'`. This gives you a menu of different options for the Linux kernel. Here you select which device drivers are to be compiled, which features to be turned on or off etc. The following is a list of options that were used for the factory configuration in Flash. Please examine each of these settings as they may be different from the default Preview Kit settings. You may select different options according to your needs, but the factory configuration is always a good place to start.

```
Networking Options -> IP: DHCP Support = ON
Networking Options -> IP: BOOTP Support = OFF
File Systems -> Compressed ROM file system support = OFF
Console Drivers -> Frame-buffer support --> Support for frame buffer devices = OFF
```

When you are operating the menus, use the arrow keys to switch fields, the return key to select items and the space bar to toggle option selections. When you are done, exit from the configurator and save the new kernel configuration. The kernel is now configured and can be compiled.

- 3) Since there are numerous configuration options available for the Linux kernel, the kernel needs to build internal dependencies in order to build the correct object files. Execute `'make dep'` to do this.
- 4) Upon completion we execute `'make clean'` to remove any old object files residing in the kernel tree (since this is a fresh install, there should be none).

- 5) Now we make the actual ARM Linux kernel image by executing '**make uImage**'. Execute '**ls arch/arm/boot**' and you should see a file named *uImage*. This is the kernel image.
- 6) Now we want to make the kernel modules associated with this kernel image. To do this, switch back to the Linux directory (cd /home/username/montavista/kernel/linux) and execute '**make modules**'.
- 7) We now need to install these modules in the kernel directory. Since the modules will be installed in /home/username/montavista/filesys/lib/modules (or /lib/modules from the target file systems point of view), and this directory is owned by root, we need to change to the *root user* using '**su root**' to install the modules. After doing so, execute '**make INSTALL_MOD_PATH=/home/username/montavista/filesys modules_install**'. This will install the kernel modules onto your target file system.

2.5 Booting the OSK with a Target File System

Once the kernel and kernel modules are built, the next step is to re-configure the OSK5912 to boot using the new kernel. One of the most convenient ways to do this is enable the NFS file sharing service on your Linux host PC and then have the OSK5912 boot off of the remote file system rather than the on-board Flash memory.

- 1) The target file system we have made will be mounted over NFS (network file system). This helps during development, since we don't have to make a new file system image every time we change something in the file system (create a new executable for instance). Edit the file */etc/exports* and add a line which says:

```
/home/username/montavista/filesys *(rw,no_root_squash,no_all_squash,sync)
```

- 2) Now make the NFS server aware of the change in it's configuration by executing '**exportfs -a**'. To be on the safe side we also restart the NFS server by first executing '**/sbin/service nfs stop**' followed by '**/sbin/service nfs start**'.
- 3) At this point, we either have to copy the file *uImage* we obtained in Section 2.4 from the kernel root directory to a Windows machine to access it with *HyperTerminal*, or we use samba to access the Linux host machine. If samba is installed, you will need to execute '**smbpasswd -a username**' and give it a password. Then start the samba service by executing '**/sbin/service smb start**'. From Windows Explorer access the Linux machine with [\\<Linuxhostip>\user, user=username](#) and the password you just set. See step 10 on how to get your Linux host IP.
- 4) Start HyperTerminal with the following settings: 115,200 baud, no parity, 8 data bits, 1 stop bit and no hardware flow control. Do not use minicom, it does not work properly with U-Boot.
- 5) Connect the OSK5912 to the host PC through the serial cable and power the board up. You will see the U-Boot startup messages as in Section 1.1. When you see the auto boot countdown message:

```
Hit any key to stop autoboot: 10
```

hit any key to get to the U-Boot prompt.

- 6) Initiate a kermit download from U-Boot on the OSK5912 by typing:

```
loadb
```

at the U-Boot prompt. This will load the kernel into SDRAM at address 0x10000000.

- 7) Start transferring the uImage file from the host to the OSK5912 using HyperTerminal's Transfer -> Send menu option. Make sure you are using the Kermit protocol.
- 8) Copy the new kernel into Flash at address 0x100000. **This will overwrite the factory installed kernel, so please only continue past this point if you are sure you want to proceed!**

```
erase 1:8-16
cp.b 0x10000000 0x100000 B50DC
```

The value 0xB50DC is the size of the downloaded image. It is displayed by U-Boot when the download is complete. If your kernel is a different size, change the length accordingly.

- 9) Enter the kernel parameters by issuing the following command to Uboot. Note that the `nfsroot=serverip` portion of the command line should be replaced with the IP address of your Linux host machine (i.e. `nfsroot=192.168.1.10`). To determine the IP of your Linux machine execute `'/sbin/ifconfig'` on the host and look at `inet addr` under `eth0`. After issuing this command, enter the `saveenv` command at the Uboot prompt to store these parameters to flash.

```
setenv bootargs console=ttyS0,115200n8 noinitrd rw ip=dhcp root=/dev/nfs
nfsroot=serverip:/home/username/montavista/filesys,nolock mem=30M
```

```
saveenv
```

- 10) To boot the Linux kernel from Uboot, you can use the `bootm 0x100000` command. You will see the output of the kernel as it boots. It should get an IP address from your DHCP server on your network, and mount it's root file system from the Linux host machine. If the kernel parameters come too fast for you to read, execute `'dmesg'` on the target later to see the log. When the boot is done you will see the following:

```
bootm 0x100000
```

```
[ many screens of Linux boot diagnostic output]
```

```
MontaVista(R) Linux(R) Professional Edition 3.1, Preview Kit
```

```
137.167.40.108 login:
```

- 11) To login, type `'root'` (it won't ask you for a password). You will get a prompt like you have on your Linux host machine. Note that the file system you are looking at on the target is in fact the `/home/username/montavista/filesys` directory on your Linux host PC. To verify this, type `'touch /home/username/montavista/filesys/opt/success'` on your Linux host, and execute `'ls -l /opt'` on your target. You should see a zero byte file named `success` (which is what `touch` creates).

You are now set up to run create/debug target applications from your Linux host.

2.6 Building and Running a "Hello World" Application

Now you can build applications on your host PC and run them natively on your OSK5912. There is a simple "hello world" project on the OSK5912 CD-ROM that will help you get started.

- 1) Copy the **helloWorld.tar.gz** file from the /linux/helloworld directory in the OSK5912 CD-ROM to the /home/username/montavista directory.

- 2) Extract the archive by typing the following command from the /home/username/montavista directory:

```
tar xvzf helloWorld.tar.gz
```

- 3) Change to the application directory.

```
cd helloWorld
```

- 4) Build the target application by using the included makefile. Type the following command:

```
make
```

You should see the following output:

```
----- Building the Hello World Application -----  
/opt/montavista/previewkit/arm/v4t_le/bin/arm_v4t_le-gcc -O2 -Wall helloApp.c -o helloWorld
```

```
The build was successful
```

- 5) Copy the resultant executable "helloWorld" to the target file system:

```
cp helloWorld /home/username/montavista/filesys/opt
```

- 6) Boot the OSK, and change to the directory where the executable is stored:

```
cd /opt
```

- 7) Execute the application by issuing the following command:

```
./helloWorld
```

- 8) You should now see "Hello World" printed in the console.

Note:

The remainder of this document refers mainly to DSP development. You do not have to perform any of these steps if you are only interested in ARM development under Linux, but it is useful for general understanding of the development environment.

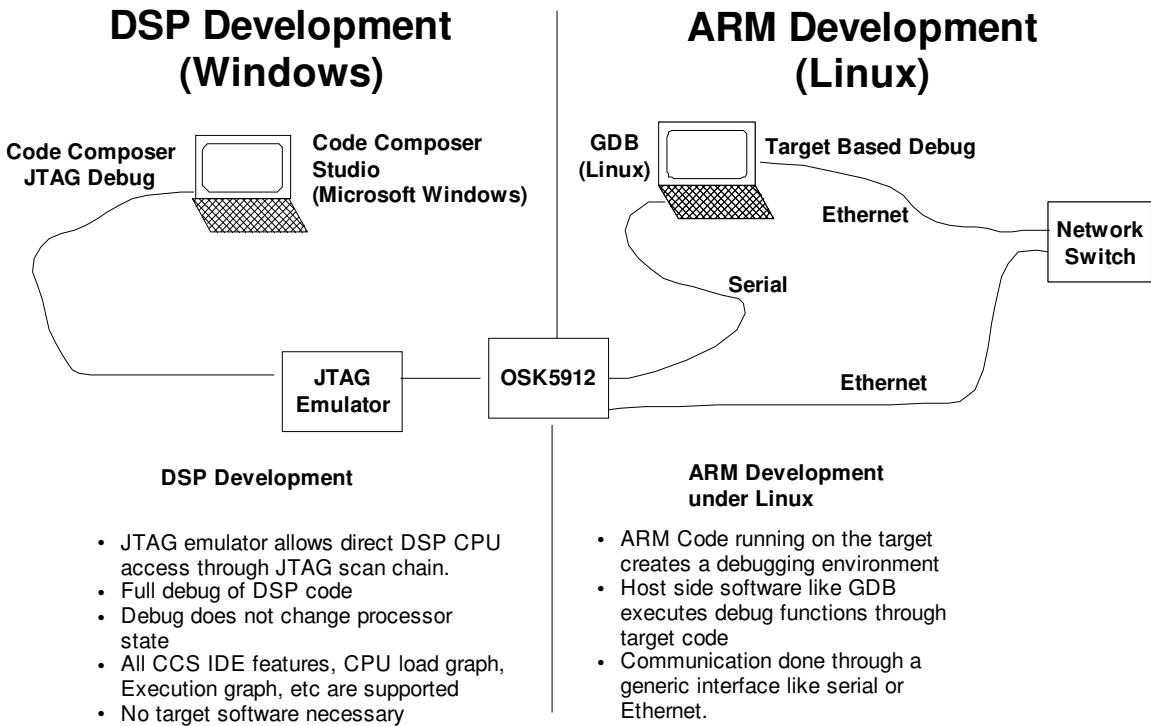
Developing DSP Applications

3.1 Overview

In addition to having an ARM926EJ-S core, the OMAP5912 processor has a C55x DSP core. The DSP core can be used by the ARM to offload signal processing intensive applications (multimedia encoding/decoding, medical imaging, etc). The DSP/BIOS™ Link and Reference Frameworks 6 (RF6) products included on the OSK5912 CDROM provide easy to use software that allows the user to leverage the DSP from ARM Linux applications.

DSP/BIOS™ Link is a communications library that allows the ARM to control and communicate with the 55x DSP through a standard API. Reference Frameworks 6 for eXpressDSP Software is provided as starterware for developing applications that use DSP/BIOS™ and the TMS320 DSP Algorithm Standard (also known as XDAIS). The audio demo referred to in section 1.1 uses both DSP/BIOS™ Link (for loading the demo onto the DSP and for streaming audio data from the DSP to ARM) and RF6 (to manage the FIR filter and volume algorithms on the DSP).

The following sections describe how to build and debug DSP application code for two types of users: Those that want to use a JTAG emulator and Code Composer Studio for OMAP (on Windows), and those that would prefer to build and debug the DSP code within Linux.



3.2 DSP Development with Code Composer Studio and a JTAG Emulator

The figure above shows the development environment for those that wish to use a JTAG emulator and Code Composer Studio for OMAP. Code Composer Studio is a full Windows based IDE for the C55x DSP with optimizing compilers, performance analysis tools and DSP/BIOS™ support. A JTAG emulator allows the debug tools to view the exact processor and memory state non-intrusively through an industry standard JTAG scan chain interface designed into the chips themselves. For ARM side Linux application development, industry standard GDB and DDD interfaces can be used via the Ethernet and serial ports found on the OSK5912.

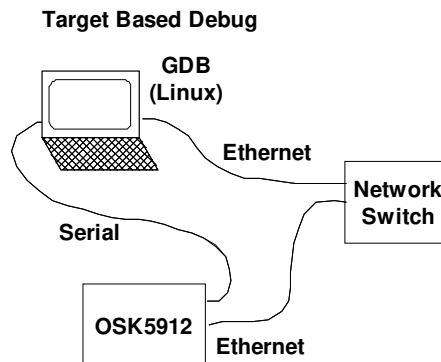
For a step by step guide on how to get started with building and debugging DSP code in this environment, please refer to the DSP Users Guide document located in the following directory on the OSK5912 CDROM:

docs\osk5912_dspusersguide.pdf

3.3 DSP Application Development – Linux DSP Tools

The second option, Linux DSP Tools, provides Linux hosted tools (C55x DSP compiler, Gnu make, Tconf) to build DSP applications and the foundational target content. These tools enable the developer to start from an existing base of robust working software on the DSP without having to use Code Composer Studio or an emulator. The Linux DSP Tools distribution bundles together five standalone products:

- C55x Codegen
- DSP/BIOS™
- DSP/BIOS™ Link
- Real Time Analysis Software Developer's kit (RTA SDK)
- RF6



- DSP executables are built under Linux
- DSP executables are loaded, run, and halted via ARM-side of DSP/BIOS Link
- To facilitate debug, LOG_printf statements in the DSP executable appear in the Linux syslog

The RTA SDK package allows for DSP application debugging via “printf” style debug. The figure above shows the development environment for those that wish to use the Linux DSP Tools.

For a step by step guide on how to get started with building and debugging DSP code in this environment, please refer to the Linux DSP Tools development guide document located here:

https://www-a.ti.com/downloads/sds_support/LinuxDspTools/index.html

Developing with Code Composer Studio

4.1 Overview

Code Composer Studio is a full Windows based IDE for the ARM926EJ-S and C55x DSP with optimizing compilers, performance analysis tools and DSP/BIOS™ support. A JTAG emulator allows the debug tools to view the exact processor and memory state non-intrusively through an industry standard JTAG scan chain interface designed into the chips themselves.

Code Composer Studio is not bundled with the OSK5912 and must be purchased separately.

4.2 Configuring Code Composer Studio

To use your OSK5912 with Code Composer Studio, you must perform the following steps:

- 1) Install Code Composer Studio for OMAP. Version 2.21 or later is recommended. The default Installation directory is c:\ti. You should use the Code Composer installation directory as the home for all installs in step 2.

- 2) Run the OSK5912 installer on the OSK5912 CD-ROM in:

`\codecomposer\SetupOSK5912.exe`

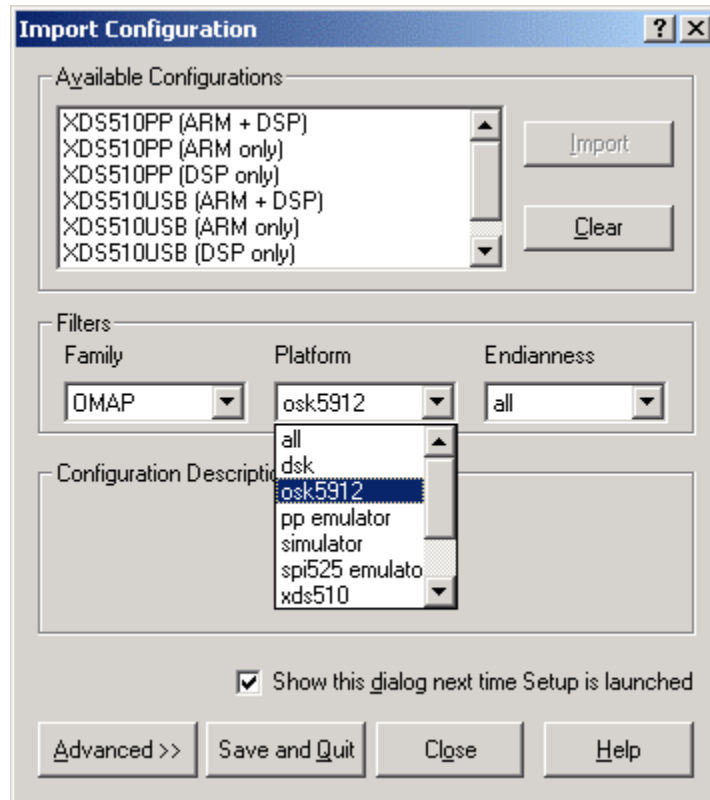
- 3) All OSK5912 specific source code and examples will be installed in:

`c:\ti\boards\osk5912`

- 4) Install JTAG emulation drivers if necessary (see documentation that comes with the emulator). For Spectrum Digital emulators, please see the "Emulators" section at:

<http://omap.spectrumdigital.com>

- 5) Launch Code Composer Setup to configure CCS with the OMAP5912 scan chain configuration. You can automatically import a configuration by using the filter to select the OSK5912 under the OMAP processor family. Most users should use the ARM + DSP configuration.



- 6) Save your new board configuration and close Code Composer Setup. You can launch the Code Composer Studio IDE now. For more advanced or manual configurations please see the Emulators section at <http://omap.spectrumdigital.com>.

4.3 The LED Example

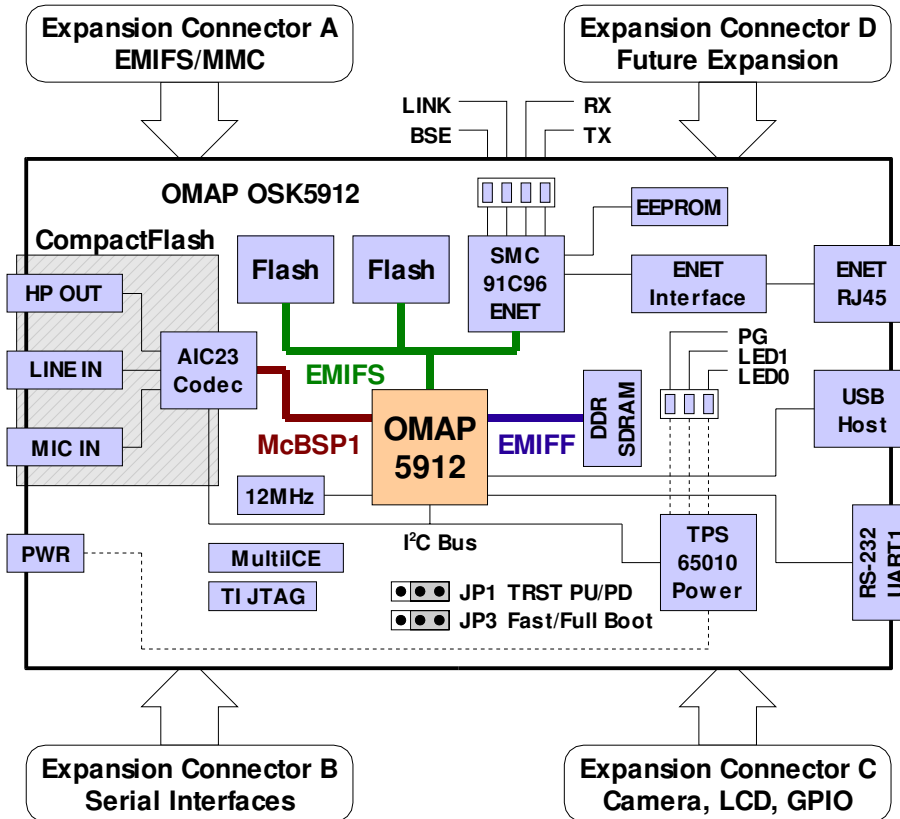
The Board Support Library provides basic functions to access many of the on and off chip peripherals on the OSK5912. It is written as a Code Composer Studio project in C, but its source code serves as a valuable example of OMAP programming and initialization. Some simple examples are also included. To run a simple LED blinking example, launch Code Composer Studio and follow the LED example instructions on the OSK5912 CD-ROM in:

docs\osk5912_bsl_examples.pdf

Board Description

5.1 Overview

The following is a block diagram of the OSK5912:



A full description of the board can be found in the OSK5912 Hardware Specification on the OSK5912 CD-ROM. The main features of the board are:

- OMAP5912 processor (192MHz ARM, 192MHz DSP)
- 32Mb Mobile DDR SDRAM
- 32Mb on-board Flash
- 10 Mbit Ethernet interface
- USB Host interface
- AIC23 stereo codec
- RS-232 serial port

The board is powered from a +5V DC input using the included AC power adapter. The OMAP5912 uses a 12MHz oscillator as a clock input with internal operating frequencies generated by on-chip PLLs. JP3 is typically the only jumper a user will modify. The Flash memory is physically connected to CS3. When JP3

is in the 2-3 position (default), the OMAP5912 boots in fast boot mode where CS3 is swapped with CS0 so the Flash starts at address 0. When the processor starts running, it will execute the code in Flash (U-Boot is stored in Flash at the factory).

If JP3 is in the 1-2 position, the OMAP5912 comes up in full boot mode with the internal ROM at address 0. The internal bootloader supports additional boot modes such as booting from the serial port. At any time, the state of the CS0-CS3 mapping can be changed by modifying the BM (bit 1) of the EMIFS_CONFIG register (address 0xFFFECC00).

For Further Information

While no specific software support is included in this package, you should visit:

<http://omap.spectrumdigital.com>

as a starting point for links, errata and other OSK5912 related resources.

