Processor SDK Linux Getting Started Guide

From Texas Instruments Wiki

Linux Software Developer's Guide → Linux Getting Started Guide

Contents

Welcome to the Linux Getting Started Guide

Thanks for your interest in learning more about the Linux Software Development Kit (SDK). The SDK as we affectionately call it is our attempt to provide a great starting point to develop an embedded system on a TI Processor running Linux. Given this goal, we wanted to provide something that is more than just a typical Board Support Package (BSP) containing a bootloader, Linux kernel, and filesystem. While these are certainly necessary elements, we feel they are just a starting point, especially for those that aren't experts in developing with Linux. So, the SDK also contains tools for developing on TI Processors (a validated cross-compiling toolchain, for example), pre-built libraries that you can use without having to rebuild them yourself, and some documentation to help explain how all of these pieces work together. We package all of this together with a working Linux Embedded System that has been built with all of the things mentioned above, and it contains a featured application called "Matrix" (derived from the fact that it is basically a simple Graphical User's Interface (GUI) of icon's arranged in a "matrix"). Matrix is a fairly simple embedded Linux system that highlights some of the key features of the TI Processor offering (LCD display, graphics, networking, etc.).

What it really serves as is a "known good" starting point. One of the big challenges with starting development on a new platform (not to mention, a new Operating System (OS) for many), is getting an environment set up where you can build and debug code on hardware. The SDK attacks this problem with providing everything you need to do development, and it is validated on standard TI hardware platforms (EVMs). It wraps all of this up into one simple installer that helps get everything you need in the right place to do development. For example, you can start off with simply re-building the Linux Embedded System that we provide to validate that everything is working on your system. This simple step gives you confidence that you can move forward from a good baseline.

As you go along your development journey and have questions, there is documentation and support available to you. Make sure to save a pointer to the Linux SDK Software Developer's Guide (SDG). If you don't find what you need, take a look at the active Sitara Forum (http://e2e.ti.com/support/arm/sitara_arm/f/791.aspx) on the E2E Community and see if the topic has been covered before. If not, post a new thread and we'll do our best to provide some guidance.

What would you like to do with the SDK?

As described above, the SDK has a lot to it. Let's break it down to two pieces to simplify things a bit:

- The example embedded Linux system starring Matrix. Essentially, a working bootloader (U-Boot), Linux kernel, and filesystem that can be put on an SD card and ran on a TI EVM, or even one of the very popular Beaglebones (either the original "white" or the newer "black").
- Everything needed to create the above embedded system from "scratch":
  - U-Boot sources and configuration files
  - Kernel sources and configuration files
  - A Linaro cross-compiling toolchain as well as other host binaries and components
  - A Yocto/OE compliant filesystem and sources for example applications in Matrix
  - A variety of scripts and Makefiles to automate certain tasks
  - Other components needed to build an embedded Linux system that don't fit neatly into one of the above buckets

With these two pieces more clearly defined, we can now get back to that all important question, "What would you like to do with the SDK?". If the answer is clearly "I want to build something and I'm ready to start developing now!", then go ahead and skip down to the "I want to Develop!" (or, Developing with the SDK section below for instructions on installing the SDK on a Linux Host System. This is a somewhat involved process focusing on the second of the two parts of the SDK listed above and may be more than some people want to start with. However, it provides access to the full spectrum of development from rebuilding the SDK from sources to fully adapting it with new device drivers and applications.

So, if you're not quite there yet, let's discuss some other options. Maybe you'd like to evaluate the SDK a bit to see if it is how you'd like to get started.

If this is not good enough and you really want to get your hands on something, check out the next section which shares how to play with the embedded Linux system featuring Matrix, the first piece of the SDK mentioned earlier. All you'll need is access to a Windows computer, a SD card, a SD card reader, some free, open-source software, and a supported Hardware platform.

Evaluating the SDK Embedded Linux System and Matrix

If you're a hands on person, reading documentation and looking at presentations gets old fast. So, if you want to see an example of what you can build with the SDK and actually hold it in your hands and play with it (or show it to someone else that needs help understanding what you want to do with it), with minimal effort, you can simply run the SDK Embedded Linux System with Matrix on a supported Hardware platform. This will allow you to poke and prod and interact. It's a powerful way to get the imagination active and engaged.

If you've recently purchased a TI EVM or Starterkit, it should have came with a SD card with the SDK on it. If that is the case, simply plug the card in, boot it up, and let your imagination run wild. However, if you're like us and the boards you are given never have all of the stuff they came with, or if you purchased a Beaglebone (http://beagleboard.org/Products/BeagleBone) or Beaglebone Black (http://beagleboard.org/Products/BeagleBone%20Black), you might not have a SD card with the SDK on it. Or, maybe, the SDK on your SD card is simply a few revisions old and you want the latest and greatest. If that is the case, check out the Creating a SD Card with Windows page. Just remember, you won't be able to build or change anything, simply evaluate the SDK Embedded Linux System with Matrix as delivered. But, even this is enough to get the imagination going and all some folks want to do.

Start your Linux Development

OK, you're all in. Either you've known this is what you wanted to do, or you've gone through the above steps and you want to do more. It's time to develop! Here's a high level overview:

- Get a Linux host up and running if you don't already have one
- Install the SDK and run some scripts to get everything set up
- Put the SDK Embedded Linux System on a SD card to play with
- Build something to validate set up – the SDK for example
- Add something to the SDK, like a simple Hello World app

After completing these steps, you'll have a known good baseline from which you can start development.

1. **Configure a Linux Host** - If you already have a Linux host machine, go to Step 2.

   To do Linux development with the SDK, you'll need a host PC running Linux. The Linux host is generally much faster and has a lot more memory (both RAM and hard disk space) than the typical embedded system. While it is certainly possible to do all development natively, we feel the advantages of using a host provide a better way to go and what is supported out of the box with the SDK.

   There are many, many ways to get access to a Linux host. We simply can't validate all possibilities and iterations, therefore we focus on validating using Ubuntu (http://www.ubuntu.com) as the host Linux distribution, running either natively or in a Virtual Machine. We validate the Long-term Support (LTS) versions of Ubuntu at the time of a SDK release (for example, at the time of this writing, Ubuntu 12.04 and Ubuntu 14.04 are the currently supported LTS versions).

   Can you use other versions of Ubuntu or even other distributions? Theoretically, yes, as long as you can get it to work and there may be more "assembly" required. If you can use the Ubuntu version validated against the SDK, it will be the smoothest path and we will be able to help you more if you do run into trouble.

   Likewise, we would strongly recommend getting a **native 64-bit** Ubuntu LTS machine set up for development. For the cost of a little bit of hard drive space, Ubuntu can have direct access to the host's hardware. Virtual Machines (VMs) have come a long way over the years, and many people use them daily without problems. However, when you are working with a target embedded system (that may be a prototype board), whether it be a TI board or eventually your own, removing the complexity of a VM from the get go can avoid a lot of frustration (i.e. wasted time). When using a VM while connecting and disconnecting hardware components, you have to be very diligent about making sure what is connected to what. You might prefer using an hour to get more work done than debugging a perceived problem caused by the fact the virtual host grabbed a USB port when you weren't watching.

   When you're ready to proceed, Ubuntu (http://www.ubuntu.com/download/desktop/install-desktop-long-term-support) provides a great overview for how to install natively. If you choose to use a VM, here is some information to help you out:
   - **Build a Ubuntu 14.04 LTS Linux host with VMware on Win7 (preferred)**

2. **Install the SDK** - Within your Linux host machine, **Install the Linux SDK**

   > **NOTE**
   >
   > Processor SDK Installer is 64-bit, and installs only on 64-bit host machine. Support for 32-bit host is dropped as Linaro toolchain is available only for 64-bit machines

   > **NOTE**
   >
   > At least 20 GB of free space is required on the host machine for installing Processor SDK Linux

3. **Create a SD Card using the SDK Create SD Card Script**

   > **NOTE**
   >
   > You will need a >4GB SD Card and the capability to connect that card to your Linux Host machine (using a USB SD Card reader, for example).

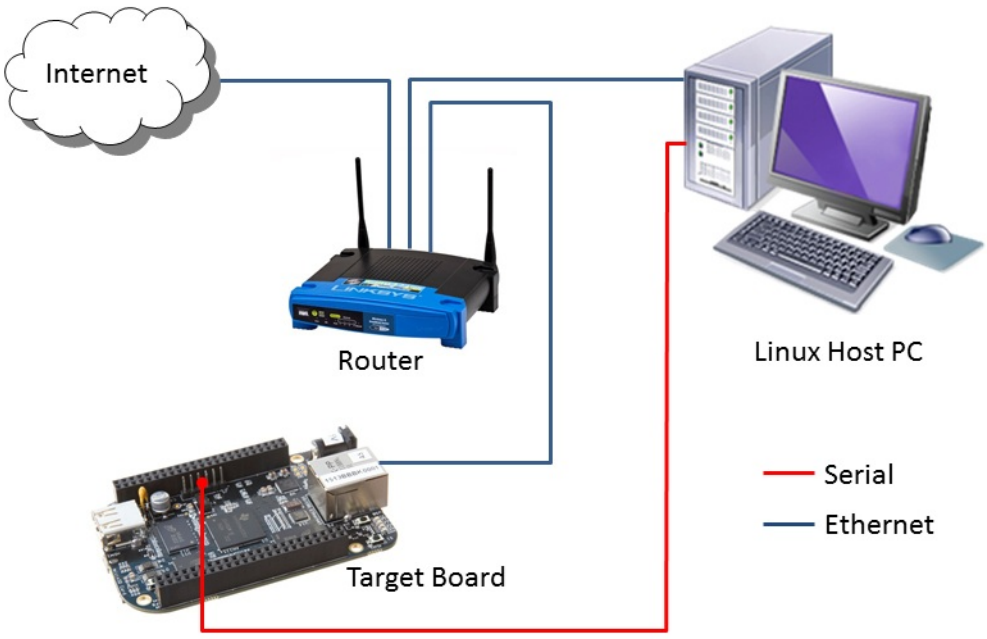
   > **NOTE**

> If using a virtual machine as your Linux host, you may need to import the SD Card reader into your virtual machine (disconnect it from the host and connect it to the VM so that the Linux VM can see it). Please see **here** for more information.

4. **Configure your development environment**

   There are many ways to connect the host development platform and the target board. These connections will vary depending on how you like to develop and what you are trying to do. Here is an example of a common set up with a serial connection for console and ethernet for networking (TFTP, NFS, etc.):

   > **NOTE**
   >
   > The recommended setup is to use TFTP for booting the kernel and NFS for hosting the target root filesystem. Since the SDK provides full cross-compile development environment for the x86 host, this configuration will simplify the transfer of files to and from the target platform.



5. **Use the SD Card to boot the target board**properly connected for your development environment.

6. **Run the Setup Script** - Once the SDK has been installed, **Run the Setup.sh Script** on your host to guide you through the remaining development environment configuration.

   > **NOTE**
   >
   > If using a virtual machine as your Linux host, you will likely need to import the target board into the virtual machine as a mass storage device using the instructions **here**

7. **Rebuild sources** using a top-level makefile in the SDK root directory. For example:

   - make all rebuilds all components in the SDK
   - make linux configures and builds the kernel
   - make u-boot-spl builds u-boot and u-boot-spl

What Would You Like to do Next?

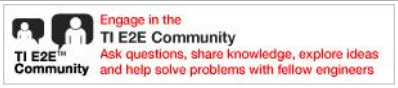
Now that you have a solid baseline set up, you can choose what you'd like to do next based on what you need to do. Here are some of the many possibilities:

Link	Summary
Processor SDK Linux Software Developer's Guide	The SDK's Homepage, a must have link for SDK users.
Processor SDK Linux Training: Hands on with the Linux SDK	The next step in learning about the Processor SDK Linux. This lab walks through how to use the SDK and Code Composer Studio with examples applications.
Processor SDK Linux How-To Guides	The SDK How-To pages. The Hands On with the SDK has some great information for developing your first Linux application.

Processor SDK Linux Kernel	More information on the Linux Kernel provided with the SDK (how to build it, for example).
Processor SDK Linux U-Boot	Everything you want to know about U-Boot, the bootloader provided with the SDK.
Processor SDK Linux Filesystem	Details about the various Filesystems delivered with the SDK, and their contents.
Processor SDK Linux Tools	Documentation for all of the various tools included with the SDK.

Archived Versions

- ***Processor SDK Linux 01.00.00.00*** (http://processors.wiki.ti.com/index.php/?title=Processor_SDK_Linux_Getting_Started_Guide&oldid=205450)
- *Sitara Linux SDK 8.00.00.00 - Getting Started Guide* (http://processors.wiki.ti.com/index.php/?title=Sitara_Linux_SDK_Getting_Started_Guide&oldid=197491)
- Sitara Linux SDK 7.0X.00.00 - Getting Started Guide (http://processors.wiki.ti.com/index.php?title=Sitara_Linux_SDK_Getting_Started_Guide&oldid=190210)



For technical support please post your questions athttp://e2e.ti.com. Please post only comments about the article ***Processor SDK Linux Getting Started Guide*** *here.*

Links

Amplifiers & Linear (http://www.ti.com/lsds/ti/analog/amplifier_and_linear.page)
Audio (http://www.ti.com/lsds/ti/analog/audio/audio_overview.page)
Broadband RF/IF & Digital Radio (http://www.ti.com/lsds/ti/analog/rfif.page)
Clocks & Timers (http://www.ti.com/lsds/ti/analog/clocksandtimers/clocks_and_timers.page)
Data Converters (http://www.ti.com/lsds/ti/analog/dataconverters/data_converter.page)
DLP & MEMS (http://www.ti.com/lsds/ti/analog/mems/mems.page)
High-Reliability (http://www.ti.com/lsds/ti/analog/high_reliability.page
Interface (http://www.ti.com/lsds/ti/analog/interface/interface.page)
Logic (http://www.ti.com/lsds/ti/logic/home_overview.page)
Power Management (http://www.ti.com/lsds/ti/analog/powermanagement/power_portal.p

Categories: AM335x | Sitara Linux SDK