Hello World!

i.MX PDK Linux Application Note

Document Number: 926-77204

Revision 2010.08

Contents

Installation and Setup 3

Creating a New Application 3

Building the Application 5

Adding the Application to the Package Directory Tree 6

Running the Application 7

Files for the Application Note 8

This document shows you how to create and load a simple Hello World application into the tree directory for the packages used in the Linux distribution and then run the application. Note that the distribution contains a Hello World application; however, this procedure explains how to create a new Hello World application.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 010 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-521-6274 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2007 - 2010. All rights reserved.

Installation and Setup

Pre-requisites:

- SDK installed, as explained in the *i.MX PDK Linux User's Guide*
- NFS file system set to work with the PC host, using a serial interface between the target and the host PC (minicom for Linux or HyperTerminal for windows) at 115200

NOTE

For additional information, see the instructions for the building process and NFS procedures in the *i.MX PDK Linux User's Guide*.

Creating a New Application

In order to create the application at least three files are needed: the source file (c file), the Makefile, and the spec file. In some circumstances only the source file is needed and the information provided by the Makefile is included in another file with the extension `.spec`, which is also created when making a new application.

- **Source file** (c file): The application in C, and also the headers if the application requires them.
- **Makefile**: Each piece of code in Linux that requires a build process has a Makefile. The Makefile provides the rules, flags, includes and other elements that determine how the sources will be built.
- **.spec file**: A file with the extension `.spec:` File specification from the LTIB package that determines the instructions that the builder and installer should take when building (usually calls the make function that uses the Makefiles from each Package), installing, unpacking and even patching the package.

NOTE

To view source for the Hello World application spec file and the Makefile, see pages 8 and 9.

The first step is to obtain the sources, clean them, and then create a package. The tarball format can be `gz` or `bz2`. The example below uses `gz`. It is important to provide a version to the folder name and to the tarball as well. For the sources, see pages 8 and 9 in this document, which provides all of the source information.

```
cd hello-1.0
make clean
cd ..
tar zcvf hello-1.0.tar.gz hello-1.0
```

The next step is to copy the created package in the location where all the ltib packages are stored. By default the packages are stored in /opt/Freescale/pkgs:

```
mv hello-1.0.tar.gz /opt/freescale/pkgs/
```

The following step is to create the spec file for this package. All the packages have a spec file and they are stored in <ltib location>/dist/lfs-5.1. There is a special folder where a template of the spec file is stored (see pages 8 and 9 for the full source of the hello.spec):

```
cd <ltib location>/
mkdir dist/lfs-5.1/hello
cp dist/lfs-5.1/template/template.spec dist/lfs-5.1/hello/hello.spec
```

After the hello.spec file is created, some editing should be made in order to make it work properly. You can use any text editor to make the changes.

Field	Description
Summary	Enter a summary of what the package is/does.
Name	Enter the name of the package (usually from the tarball name) .
Version	Enter the version (usually from the tarball/directory).
Release	Begin at 1 and add a revision each time you change the spec file.
License	For example, GPL/LGPL/BSD. Find this in the package's files.
Group	If this exists on an rpm-based machine, copy from rpm –qi. If not, select something from /usr/share/doc/rpm-/GROUPS.
%Build	Just apply a Make.
%Install	Copy the executable to the usr/bin directory.

Building the Application

Now that the .spec file is created and ready, you can build and test the package.

To build the application, follow these steps:

1. Unpack the package that is stored in `/opt/Freescale/pkgs`, using the following command. This command unpacks the `hello -1.0.tar.gz` inside the `<ltib location>/rpm/Build`/:
   ```
   cd <ltib location>/
   ./ltib -m prep -p hello.spec
   ```
2. Build the package, using the following command:
   ```
   ./ltib -m scbuild -p hello.spec
   ```
3. Once the package is built without problems, install the package. (Within the hello.spec file, these are the %install section commands):
   ```
   ./ltib -m scinstall -p hello.spec
   ```
   Now the package (executable) is installed as the test package in the NFS root filesystem area (rootfs). We specified in the spec file that it needs to be installed in `rootfs/usr/bin`.
   ```
   ./ltib -m scdeploy -p hello.spec
   ```
   You will find the Hello World application file in the `rootfs/usr/bin` directory. The application is ready to be tested.

Adding the Application to the Package Directory Tree

Add the package to allow it to be visible when selecting the packages with `./ltib -c` .

To add the Hello World package, follow these steps:

1. Make sure that the package that was unpacked in <`ltib location>/rpm/BUILD/` is erased.
   ```
   cd <ltib location>/
   cd /rpm/BUILD/
   rm -r hello-1.0/
   ```
2. Go to `<ltib directory>/ config/userspace/:`
   ```
   cd <ltib location>/
   cd config/userspace/
   ```
3. Use a text editor (such as vi, emacs, or gedit) to open the file `packages.lkc`.

   This file is ordered alphabetically, so look for the “H” section. In front of the section PKG_HELLOWORD add the following:
   ```
   config PKG_HELLO
   bool "hello"
   ```
4. Save the changes.
5. Go to `pkg_map` file that is located in the same directory (`config/userspace`).
6. Open `pkg_map` with a text editor.
7. Add the following entry anywhere (it does not need to be in alphabetical order).
   ```
   PKG_HELLO     =   hello
   ```

   Now we are ready to add the application from the directory tree.

Running the Application

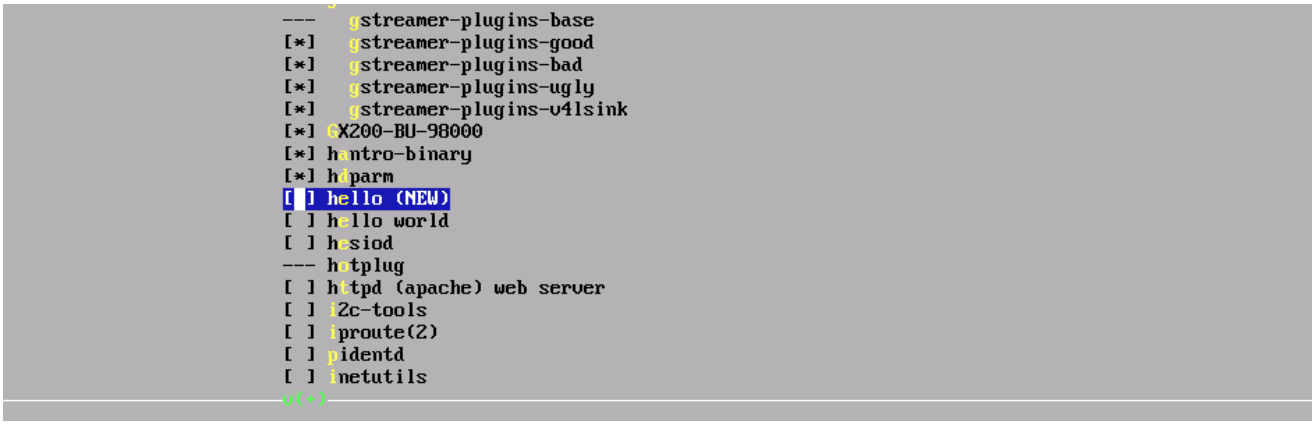
To run the application, follow these steps:

1. In the location where the ltib is installed on the host PC `<ltib directory>`. Type the following command:

   ```
   cd <ltib location>/
   ./ltib -c
   ```

   The configuration prompt is displayed (Figure 1).

Figure 1 Package List



2. In the Package List, find ‘hello’, select it, and exit, saving the changes.

   The `hello` package will build automatically and save the binary in `rootfs/usr/bin`/.

3. Run the file in the target using the following commands (nfs must be working and there must be a serial terminal at 115200).

   ```
   mx#cd /usr/bin
   mx#./hello
   ```

   The application runs, and messages are displayed in the serial console.

   All of the applications will follow this procedure. The differences are in the complexity of the Makefile, sources, or spec file.

Files for the Application Note

hello.c

```
#include <stdio.h>

int main()
{
    int i;

    printf("hello world\n");

    for ( i = 0; i < 10 ; i++ ) {
        printf("loop count = %d\n", i);
    }

    printf("hello this is the end\n");

    return 0;
}
```

hello.spec

```
%define pfx /opt/freescale/rootfs/%{_target_cpu}

Summary         : hello application for appnote
Name            : hello
Version         : 1.0
Release         : 1
License         : xxxx
Vendor          : Freescale
Packager        : User
Group           : MAD
URL             : http://xxxx
Source          : %{name}-%{version}.tar.gz
BuildRoot       : %{_tmppath}/%{name}
Prefix          : %{pfx}

%Description
%{summary}

%Prep
%setup

%Build
make

%Install
%Files
%defattr(-,root,root)
%{pfx}/*
rm -rf $RPM_BUILD_ROOT

mkdir -p $RPM_BUILD_ROOT/%{pfx}/usr/bin
cp hello $RPM_BUILD_ROOT/%{pfx}/usr/bin/
```

```
%Clean
rm -rf $RPM_BUILD_ROOT

%Files
%defattr(-,root,root)
%{pfx}/*
```

Makefile

```
EXEC = hello
OBJS = hello.o

all: $(EXEC)

$(EXEC): $(OBJS)
	$(CC) $(LDFLAGS) -o $@ $(OBJS) $(LDLIBS$(LDLIBS_$@))

romfs:
	$(ROMFSINST) /bin/$(EXEC)

clean:
	-rm -f $(EXEC) *.elf *.gdb *.o
```