PHP with SQlite on Digi Embedded Linux

Document History

Date	Version	Change Description
10/14/2009	V1.0	Initial entry/outline
10/16/2009	V1.1	Switched back to compiling php-5.2.11
11/09/2009	V1.2	Integrated SQlite support

Table of Contents

1 Problem Description

Digi embedded modules running Digi Embedded Linux are delivered with pre-compiled out of the box web servers with PHP capabilities. Also a pre-compiled small embedded data base SQLite is provided. This document describes how to cross compile a third party PHP interpreter and configure the web server(s) to execute it (e.g. with a test.php page). It also describes how to configure the firmware for running SQLite data base and an example to access it from PHP pages run in the web server.

2 Requirements

To try the example in this document you need:

- Digi Connect ME 9210 or ConnectCore module mounted on Digi development board.
- Digi Embedded Linux (DEL) 5.1 or above development environment.
- You can get everything together in a Digi Linux JumpStart Kit

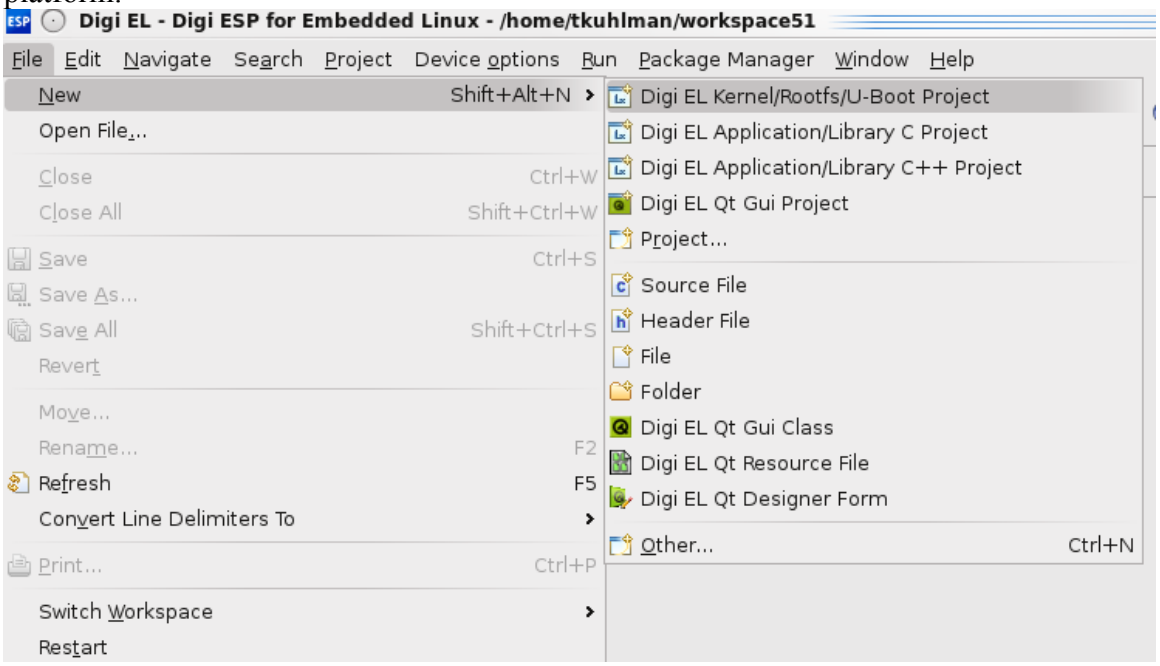
3 Software Setup

For your convenience find all files and images compiled into the archive these instructions came with at this link –

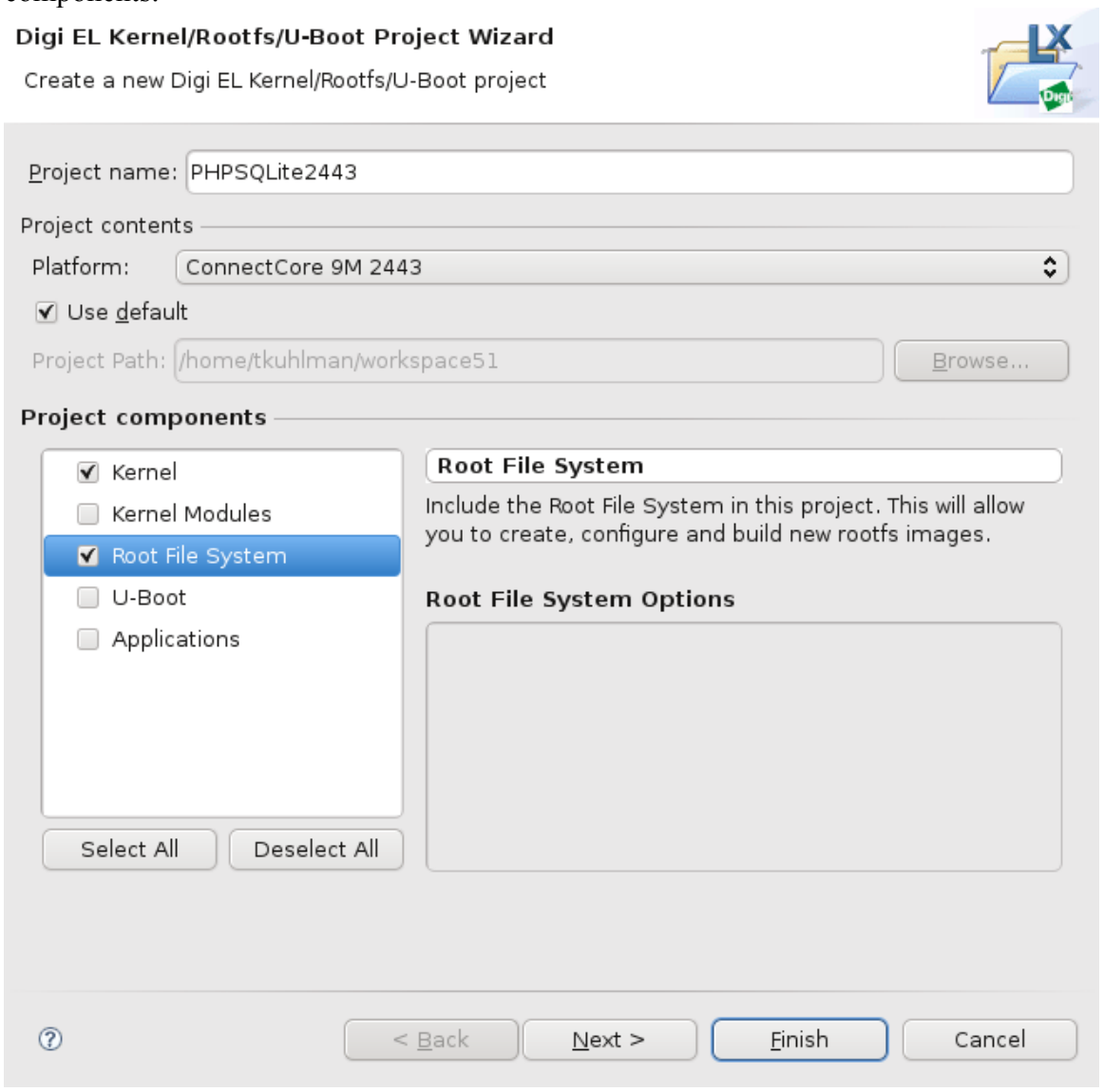
http://ftp1.digi.com/support/documentation/EmbeddedLinuxPHP-SQLite.zip

- Install Digi Embedded Linux (DEL) 5.1 or higher, apply latest patches with the Package Manager.

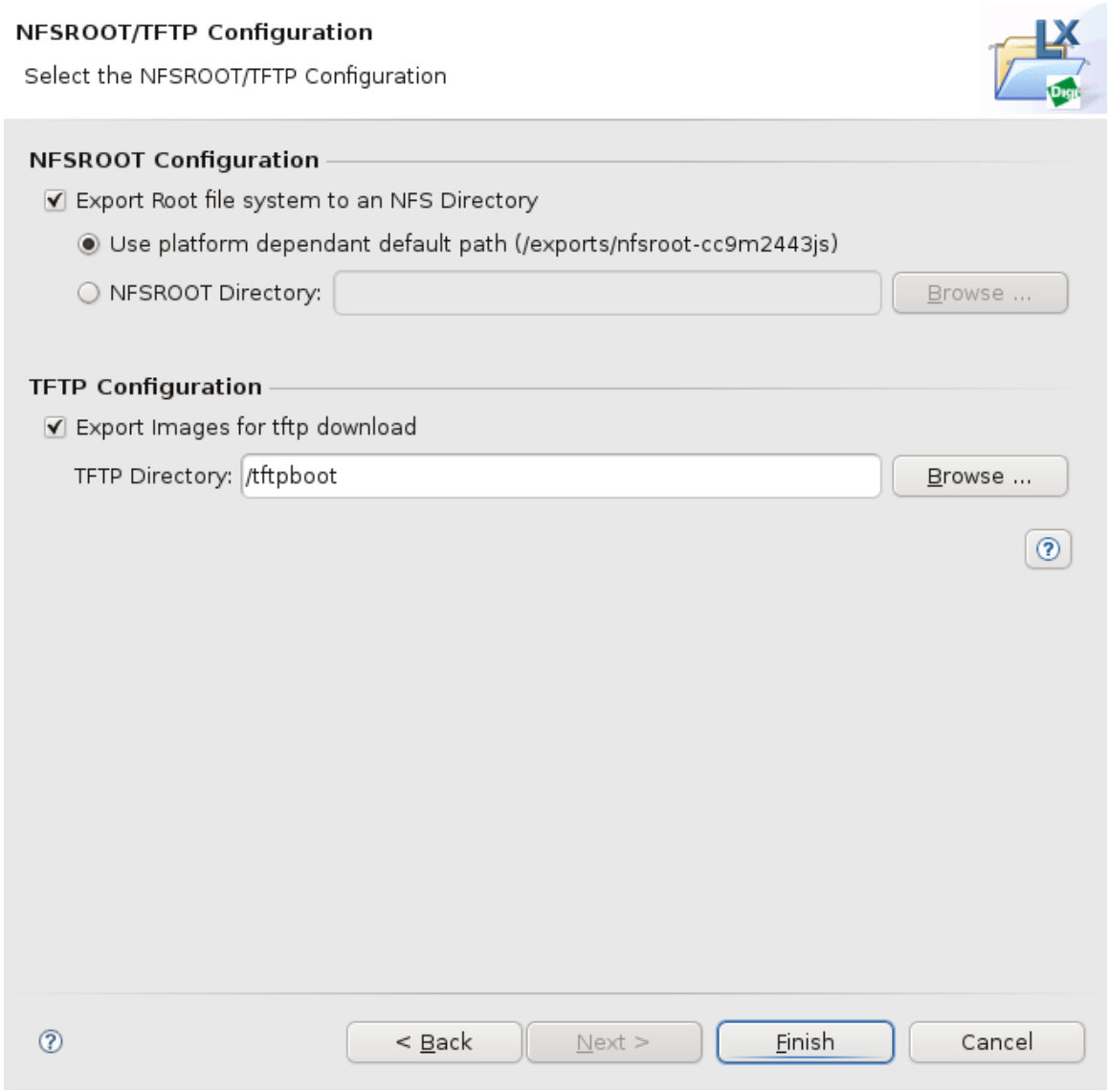
- Create a new Digi EL Kernel/Rootfs/U-Boot Project for your platform:



- Select Kernel and Root File System as project components:



- Select appropriate NFSROOT/TFTP Configuration



3.1 Download and cross compile PHP

Download a recent stable PHP source code package from http://php.net/. This document has been tested with php-5.2.11.tar.bz2. Store/deflate it into your project directory:

```
# cd $HOME/$YOURWORKSPACE/$YOURPROJECTNAME
# tar xjf php-5.2.11.tar.bz2
# cd php-5.2.11
```

Cross compile the PHP interpreter like this:

```
# export PATH=/usr/local/DigiEL-5.1/usr/bin:$PATH
# CC=arm-linux-gcc ./configure --host=arm-linux --prefix=/usr --without-iconv --
disable-xml --without-pear --disable-libxml --disable-dom --disable-simplexml --
disable-xmlreader --disable-xmlwriter --without-pdo-sqlite --enable-fastcgi
```

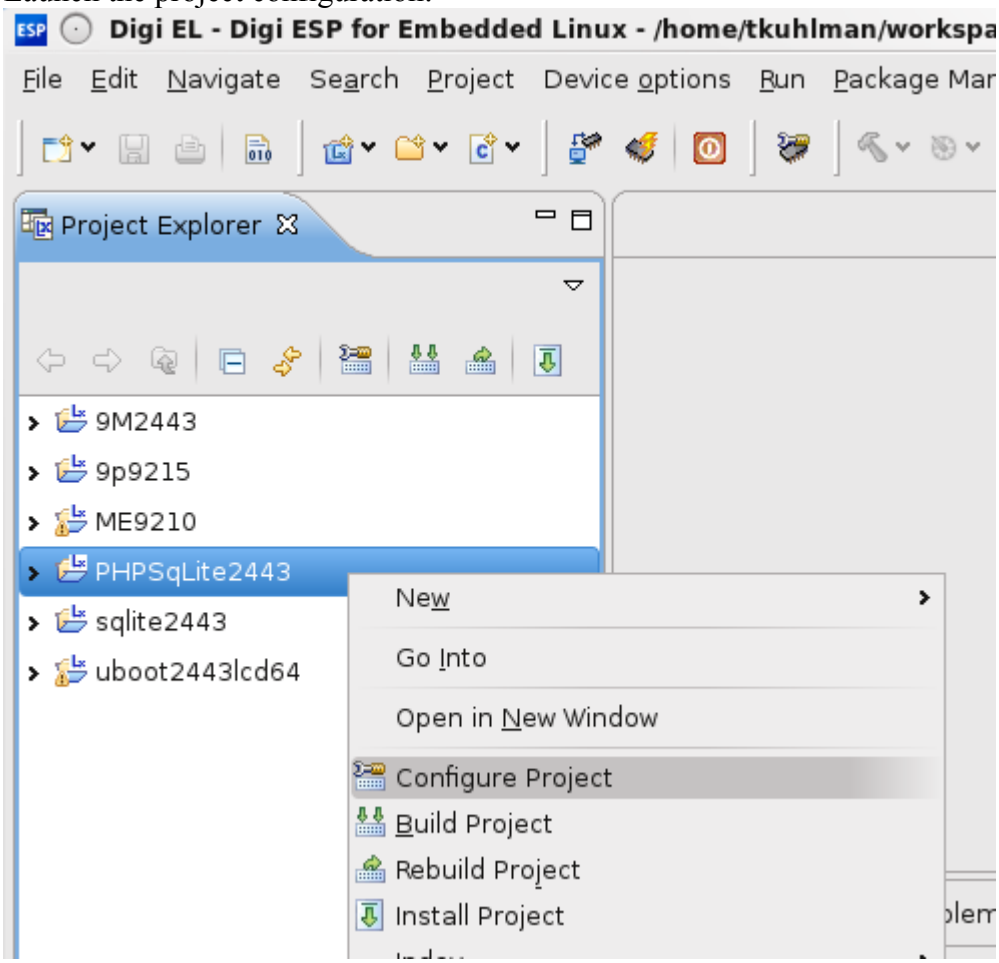
```
# make
# make INSTALL_ROOT=$HOME/$YOURWORKSPACE/$YOURPROJECTNAME install
```

If you don't see php-cgi in your project directory (the make install failed?), copy it manually from built directory (or take the pre-compiled php-cgi from the zip attached).

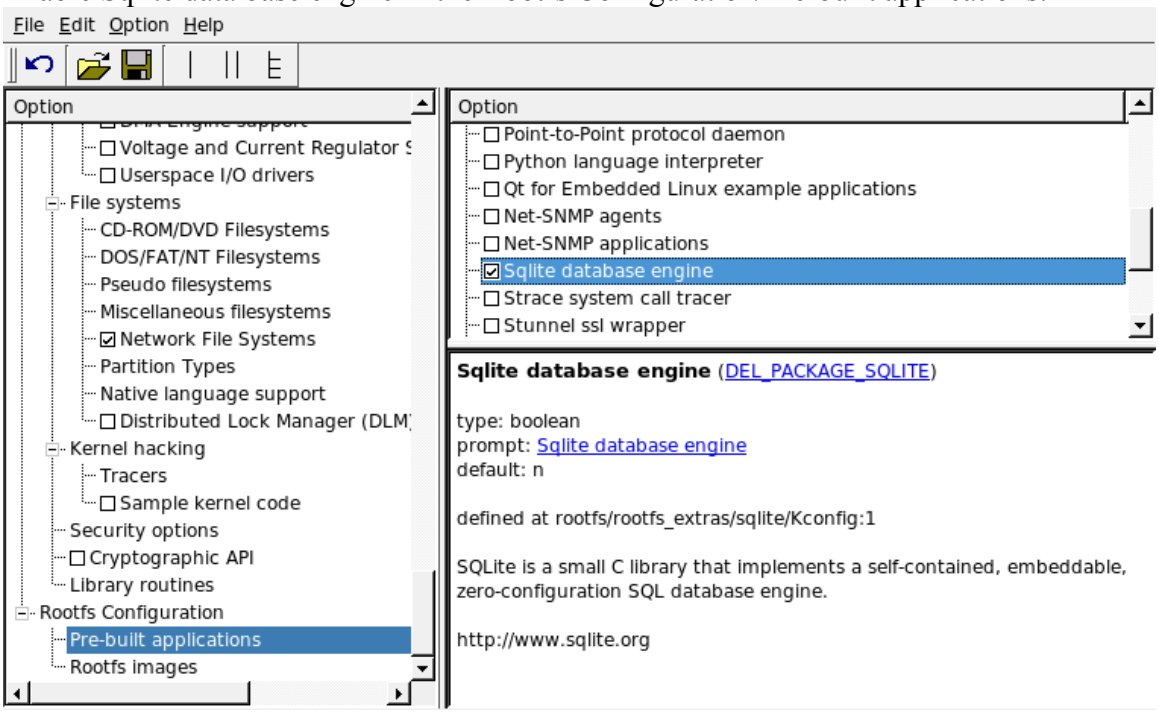
```
# find . -name php-cgi
./sapi/cgi/php-cgi
# cp sapi/cgi/php-cgi ..
```

3.2 Configure Rootfs to contain SQLite pre-compiled binary

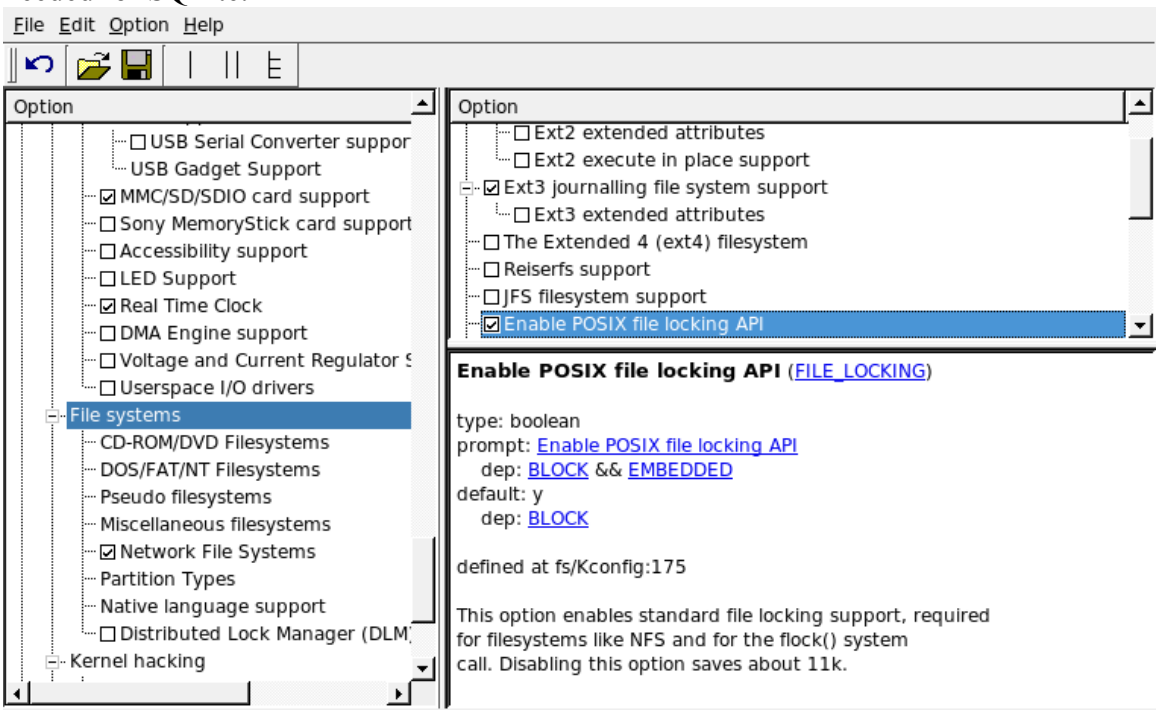
Launch the project configuration:



Enable Sqlite data base engine in the Rootfs Configuration/Pre-built applications:



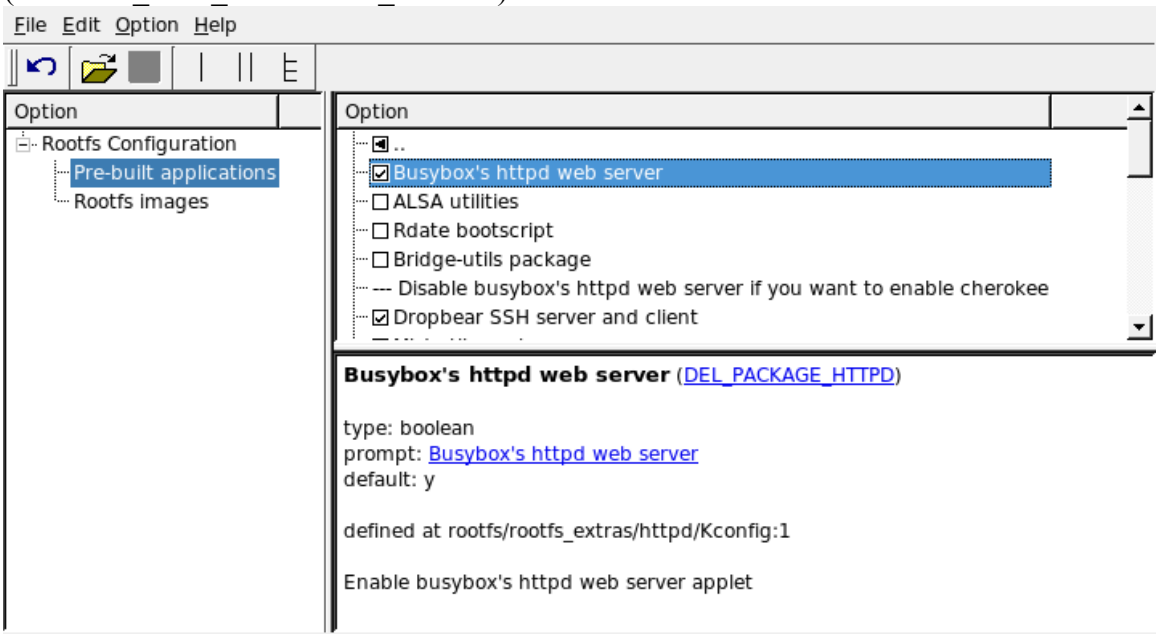
configure the kernel for POSIX file locking (CONFIG_FILE_LOCKING) which is needed for SQLite:



Save the changes.

3.3 Configure Busybox httpd

If you decide to use the pre-compiled httpd included in the busybox, you need to select it into your project config (which is the default). Right click on your project and select "Configure project". Select Rootfs Configuration, Pre-built applications and Busybox's httpd web server
(CONFIG_DEL_PACKAGE_HTTPD)



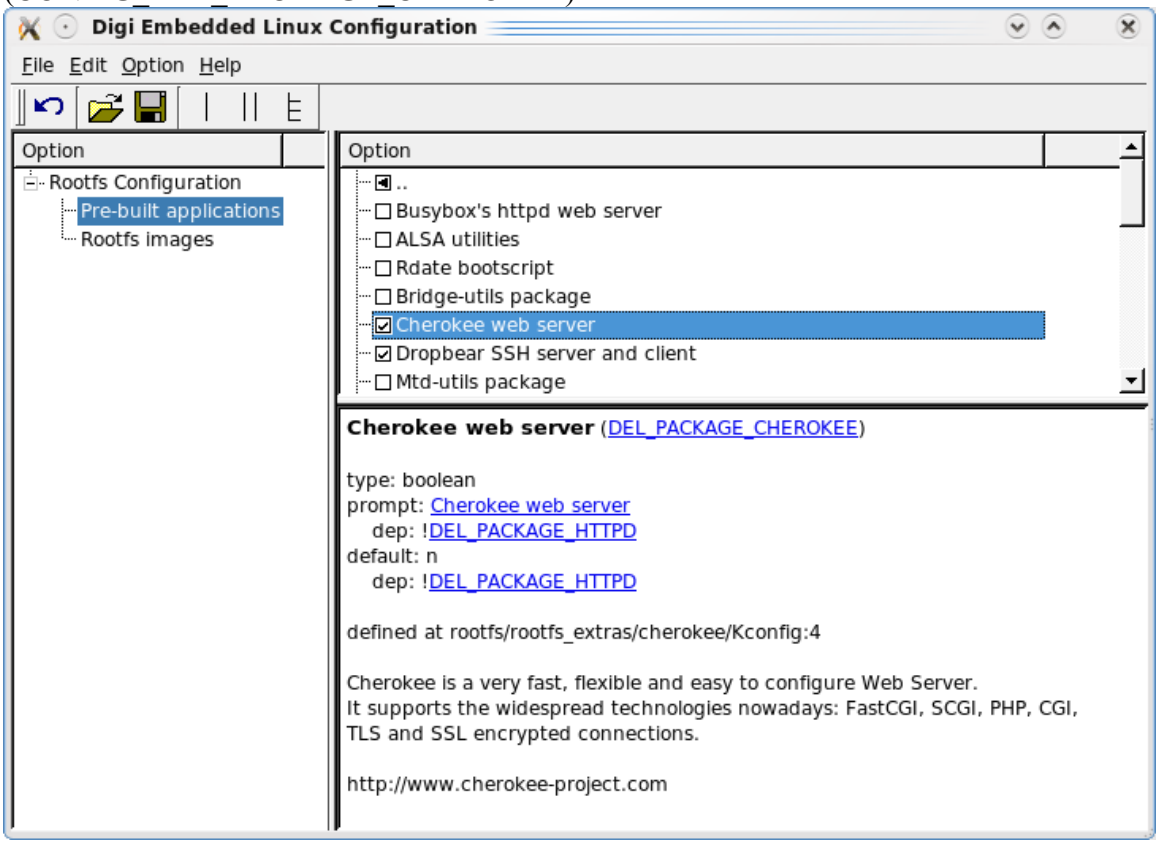
To enable the busybox httpd to launch the PHP interpreter when called with .php in the URL, add the following line to the /etc/httpd.conf of your rootfs:

```
*.php:/usr/bin/php-cgi
```

E.g. copy the default /usr/local/DigiEL-5.1/rootfs/rootfs_extras/httpd/rootfs/etc/httpd.conf into your project directory and edit it.

3.4 Configure Cherokee web server

If you decide to use the pre-compiled Cherokee web server, you need to select it into your project config (disable busybox httpd for this first). Right click on your project and select "Configure project". Select Rootfs Configuration, Pre-built applications deselect Busybox's httpd web server and select Cherokee (CONFIG_DEL_PACKAGE_CHEROKEE)



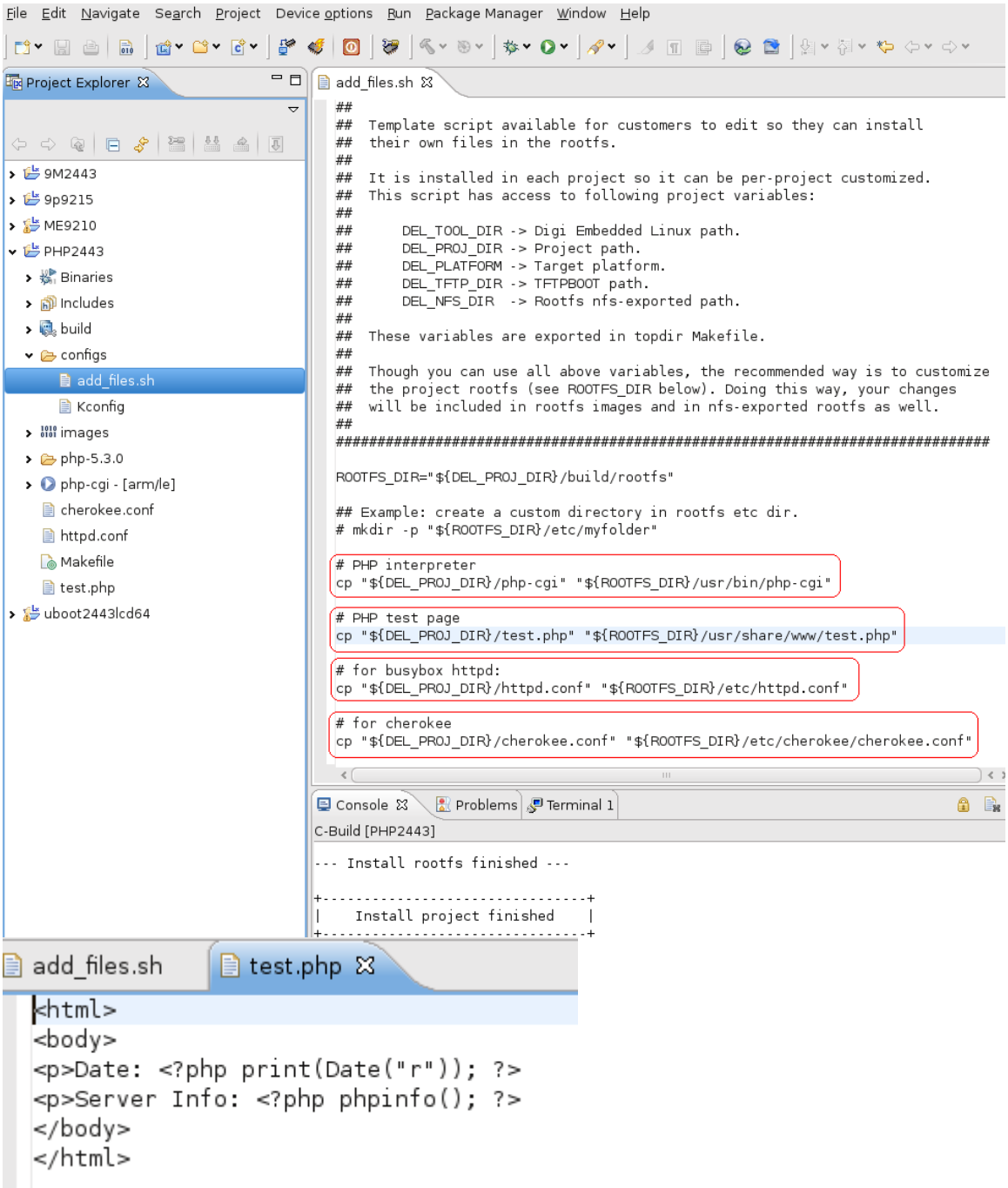
To configure Cherokee to launch the PHP interpreter, add the following lines to your cherokee.conf:

```
vserver!10!rule!600!encoder!deflate = 0
vserver!10!rule!600!encoder!gzip = 1
vserver!10!rule!600!handler = fcgi
vserver!10!rule!600!handler!balancer = round_robin
vserver!10!rule!600!handler!balancer!source!1 = 1
vserver!10!rule!600!handler!change_user = 0
vserver!10!rule!600!handler!check_file = 1
vserver!10!rule!600!handler!error_handler = 1
vserver!10!rule!600!handler!pass_req_headers = 1
vserver!10!rule!600!handler!xsendfile = 0
vserver!10!rule!600!match = extensions
vserver!10!rule!600!match!extensions = php
vserver!10!rule!600!only_secure = 0
source!1!host = localhost:50000
source!1!interpreter = /usr/bin/php-cgi -b localhost:50000
source!1!nick = PHP Interpreter
source!1!type = interpreter
```

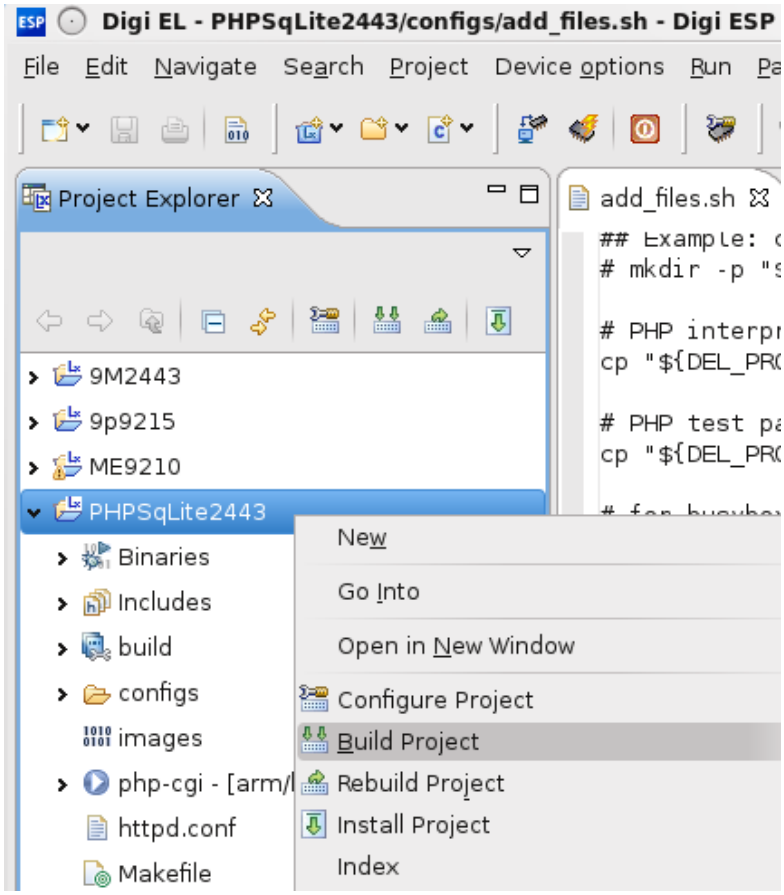
E.g. copy the default /usr/local/DigiEL-5.1/rootfs/rootfs_extras/cherokee/rootfs/etc/Cherokee/cherokee.conf into your project directory and edit it.

3.5 Add files to your rootfs

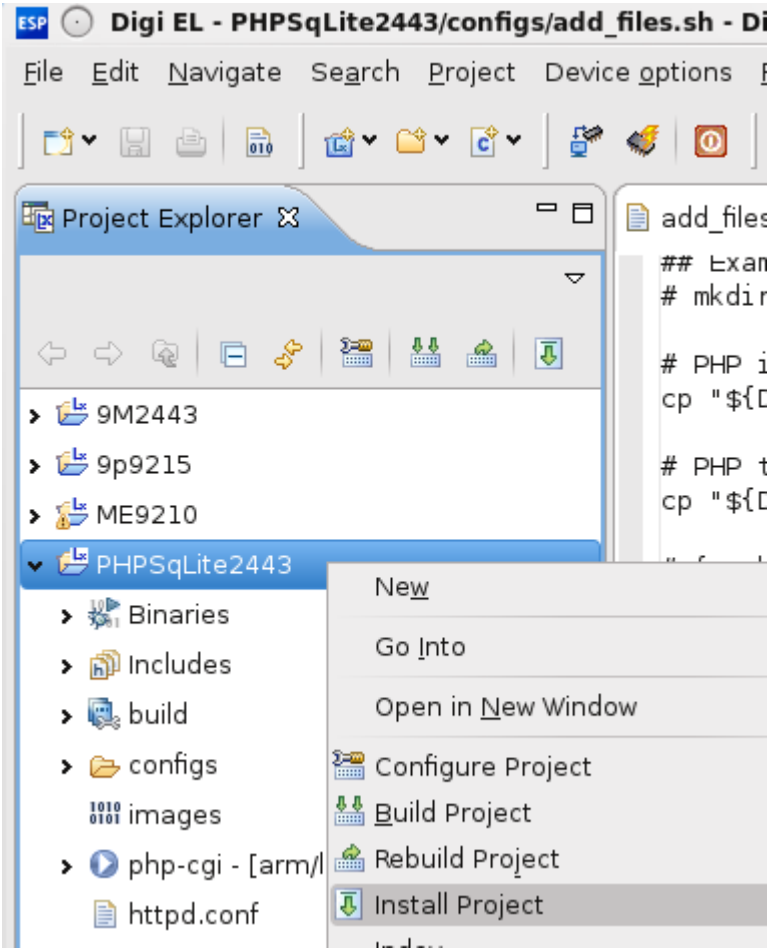
Unfold your project, select configs and open add_files.sh. Add the php-cgi, the test.php and the httpd.conf or cherokee.conf (according to your selected web server) to the rootfs:



- Built the project:



- Install the project:



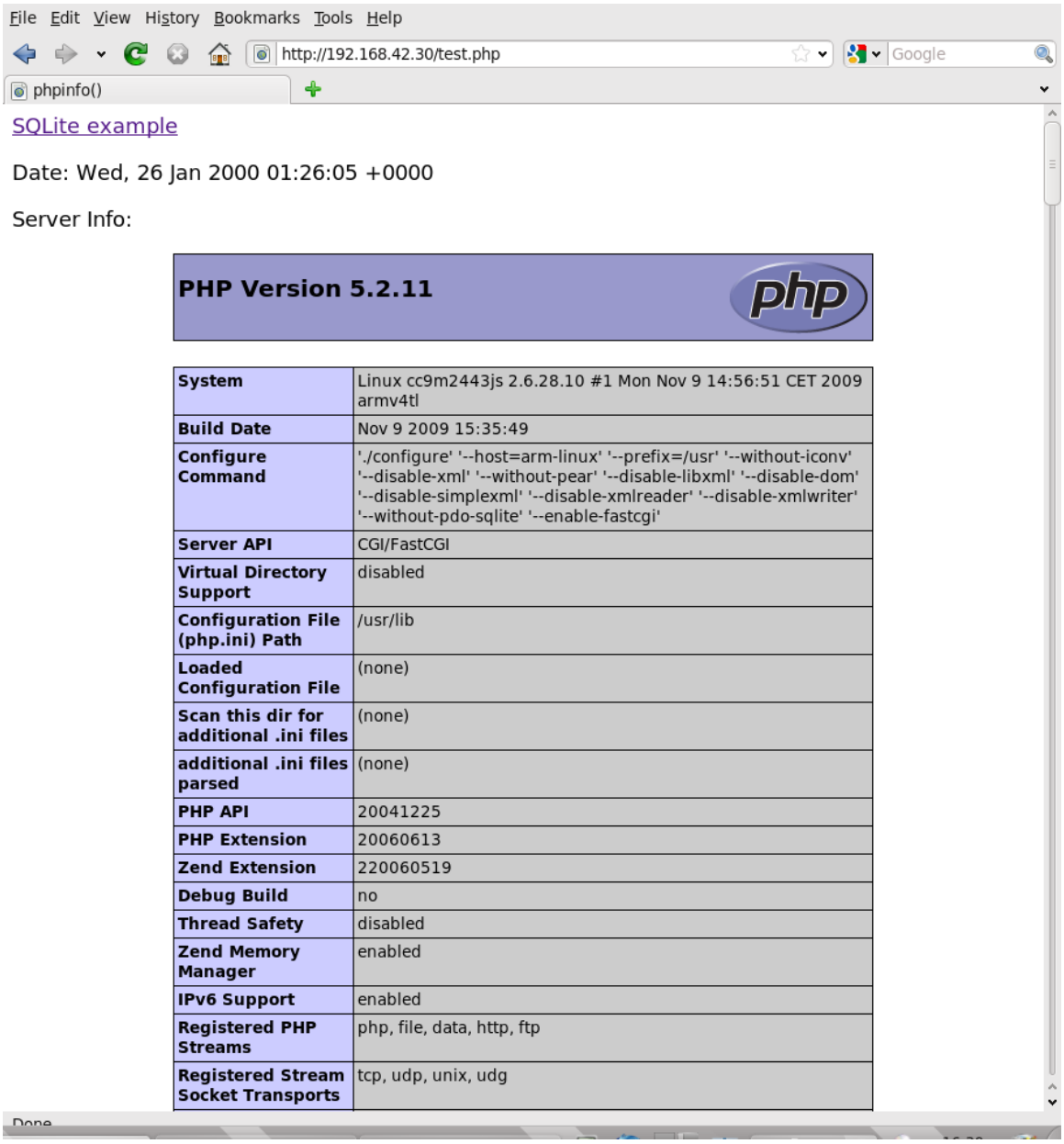
4 Hardware Setup

- Turn off the development board.
- Connect the power cable.
- Plug the Connect ME 9210 or ConnectCore module to the development board.
- Connect Ethernet between development board and your development PC or switch (for updating firmware).
- Connect a serial null modem cable (pins 2 and 3 crossed) to your host computer (e.g. COMA is the CONSOLE). Plug the cable into Serial Port A of the Digi development board. On ConnectCore9P9215 development board use Serial Port D (CONSOLE).

5 Testing

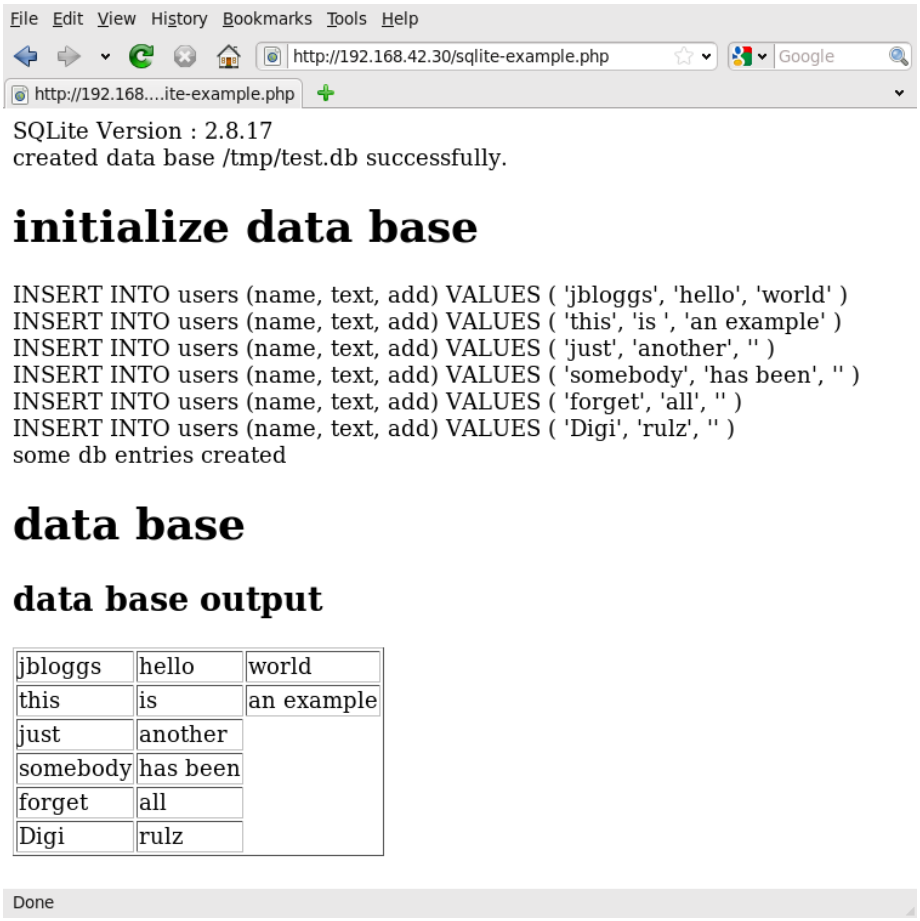
Run the newly built kernel and rootfs on module (e.g. update the images in flash built in section 3 “update linux tftp;rootfs tftp;dboot linux flash”, or boot with NFS-root: “dboot linux tftp”).

Start a web browser on your development host and point it to the IP of your module to check the php test page: http://192.168.42.30/test.php



System	Linux cc9m2443js 2.6.28.10 #1 Mon Nov 9 14:56:51 CET 2009 armv4tl
Build Date	Nov 9 2009 15:35:49
Configure Command	'./configure' '--host=arm-linux' '--prefix=/usr' '--without-iconv' '--disable-xml' '--without-pear' '--disable-libxml' '--disable-dom' '--disable-simplexml' '--disable-xmlreader' '--disable-xmlwriter' '--without-pdo-sqlite' '--enable-fastcgi'
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/lib
Loaded Configuration File	(none)
Scan this dir for additional .ini files	(none)
additional .ini files parsed	(none)
PHP API	20041225
PHP Extension	20060613
Zend Extension	220060519
Debug Build	no
Thread Safety	disabled
Zend Memory Manager	enabled
IPv6 Support	enabled
Registered PHP Streams	php, file, data, http, ftp
Registered Stream Socket Transports	tcp, udp, unix, udg

If you click on the SQLite example link (or enter URL: http://192.168.42.30/sqlite-example.php) you will see the output of the SQL statements executed in this PHP page.



If you execute this a second time, you will get an error message, since the data base is already existing: