



PHP with SQLite on Digi Embedded Linux

SQLite 2.8.17 supported with sql\_..() API

SQLite 3.x support via PDO

## 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, tested on CC9M2443
06/01/2010	V1.3	Retested with php-5.2.13 DEL5.2 toolchain and pdo-sqlite for sqlite3 support on CC9P9215

## Table of Contents

Document History .....	2
Table of Contents .....	2
1 Problem Description .....	2
2 Requirements .....	3
3 Software Setup .....	3
3.1 Download and cross compile PHP .....	5
3.2 Cross compile example application.....	6
3.3 configure Rootfs to contain SQLite pre-compiled binary .....	6
3.4 configure Busybox httpd .....	8
3.5 configure Cherokee webserver.....	9
3.6 add files to your rootfs .....	10
4 Hardware Setup.....	12
5 Testing.....	13
5.1 Testing with sqlite_open() API (SQLite 2.8.17) .....	13
5.2 Testing with PDO interface (SQLite 3.x support).....	14

## 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 ran in the web server.

## 2 Requirements

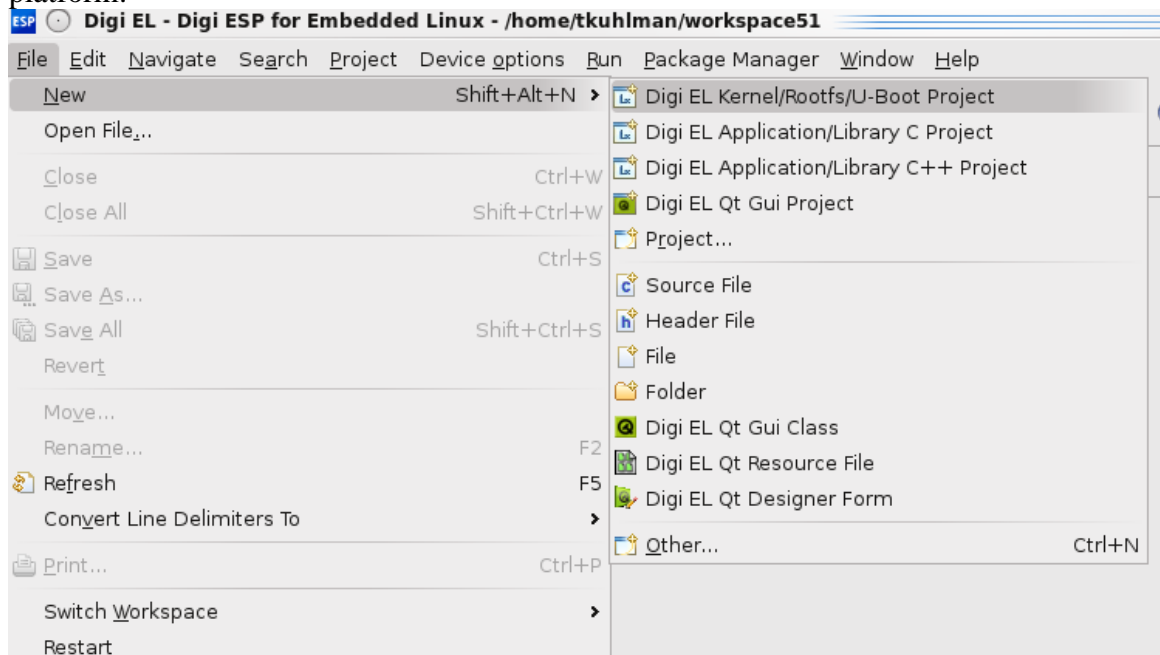
To try the example in this document you need:

- 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 this instructions came with, you might want to skip this section.

- 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:

**Digi EL Kernel/Rootfs/U-Boot Project Wizard**

Create a new Digi EL Kernel/Rootfs/U-Boot project



Project name:

Project contents

Platform:

Use default

Project Path:

**Project components**

- Kernel
- Kernel Modules
- Root File System
- U-Boot
- Applications

**Root File System**

Include the Root File System in this project. This will allow you to create, configure and build new rootfs images.

**Root File System Options**

- Select appropriate NFSROOT/TFTP Configuration

#### NFSROOT/TFTP Configuration

Select the NFSROOT/TFTP Configuration



**NFSROOT Configuration**

Export Root file system to an NFS Directory

Use platform dependant default path (/exports/nfsroot-cc9m2443js)

NFSROOT Directory:

**TFTP Configuration**

Export Images for tftp download

TFTP Directory:

### 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-5.2.13.tar.bz2 and store/deflate it into your project directory:

```
# cd $HOME/$YOURWORKSPACE/$YOURPROJECTNAME
# tar xjf php-5.2.13.tar.bz2
# cd php-5.2.13
```

Cross compile the PHP interpreter like this:

*Note: cross compilation will fail with php-5.3.2! Use php-5.2.13*

```
# export PATH=/usr/local/DigiEL-5.2/usr/bin:$PATH
```

## PHP with Digi Embedded Linux

```
# 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 --with-pdo-sqlite=/usr/local/DigiEL-
5.2/usr
# 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).

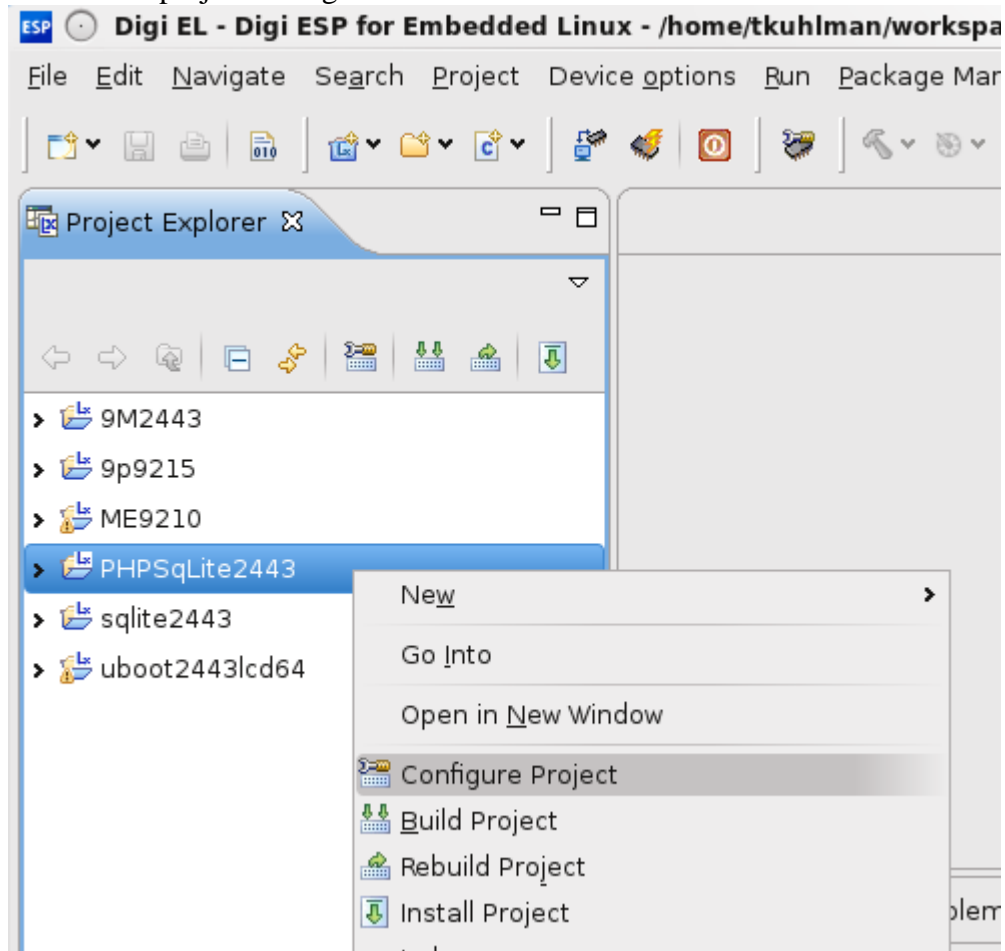
```
# find . -name php-cgi
./sapi/cgi/php-cgi
# cp sapi/cgi/php-cgi ..
```

### 3.2 Cross compile example application

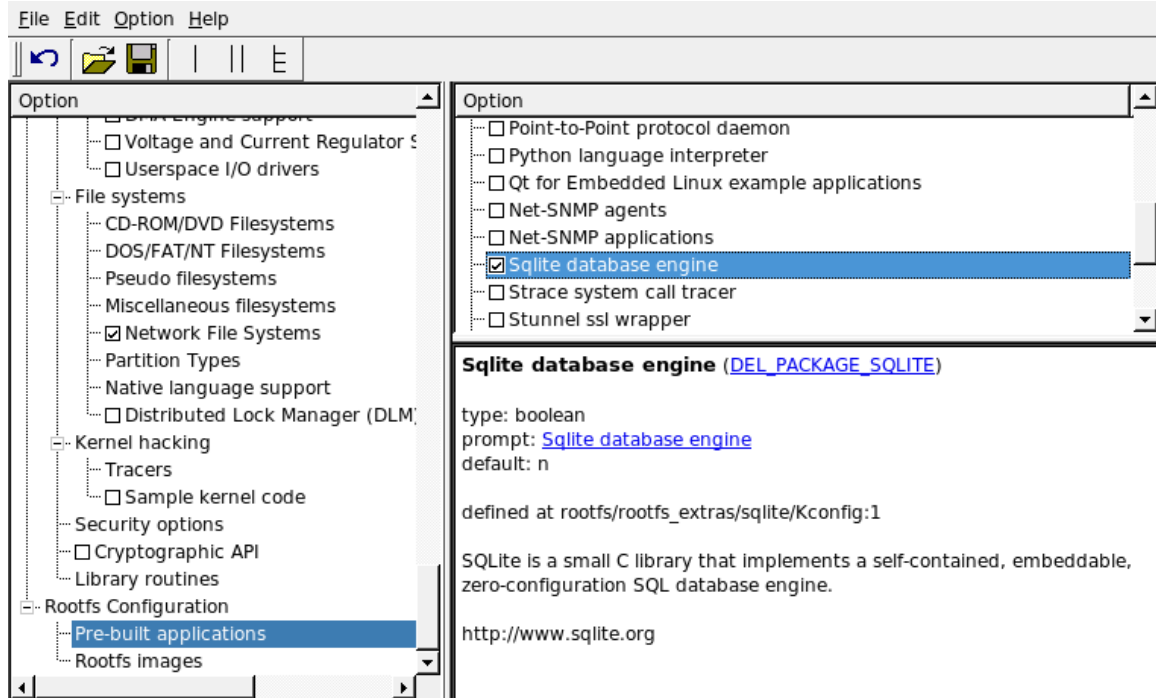
```
# export PATH=/usr/local/DigiEL-5.2/usr/bin:$PATH
# cd $HOME/$YOURWORKSPACE/$YOURPROJECTNAME
# arm-linux-gcc sqlite_test.c -o sqlite_test -lsqlite3
```

### 3.3 configure Rootfs to contain SQLite pre-compiled binary

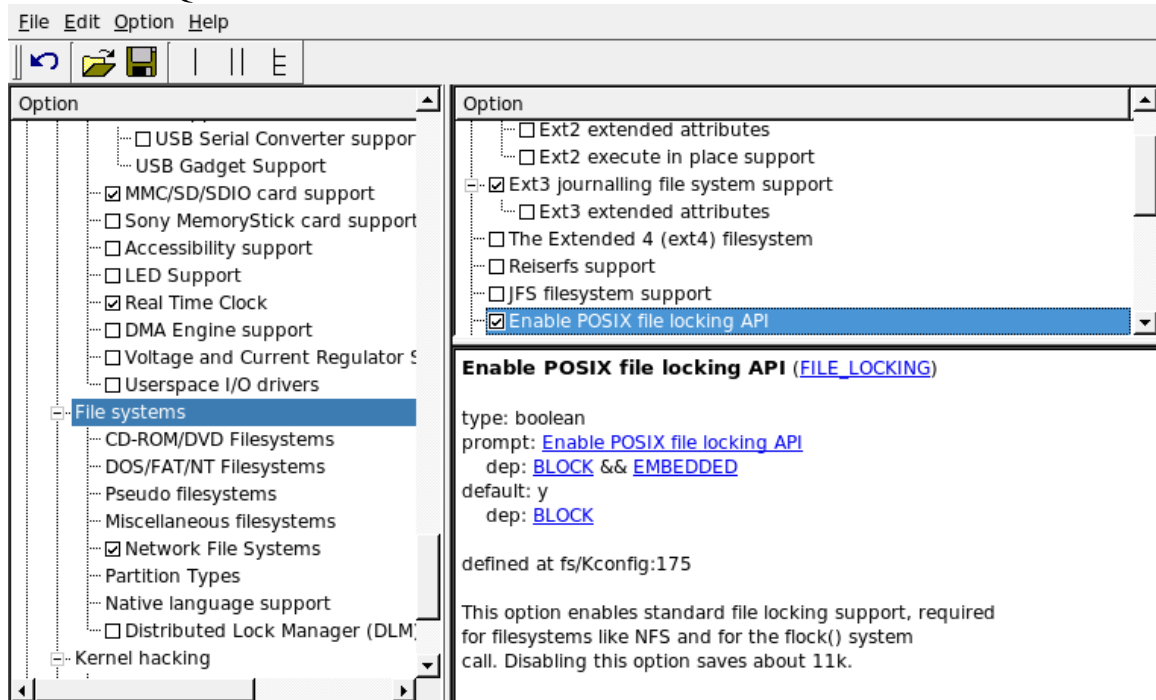
Launch the project configuration:



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:

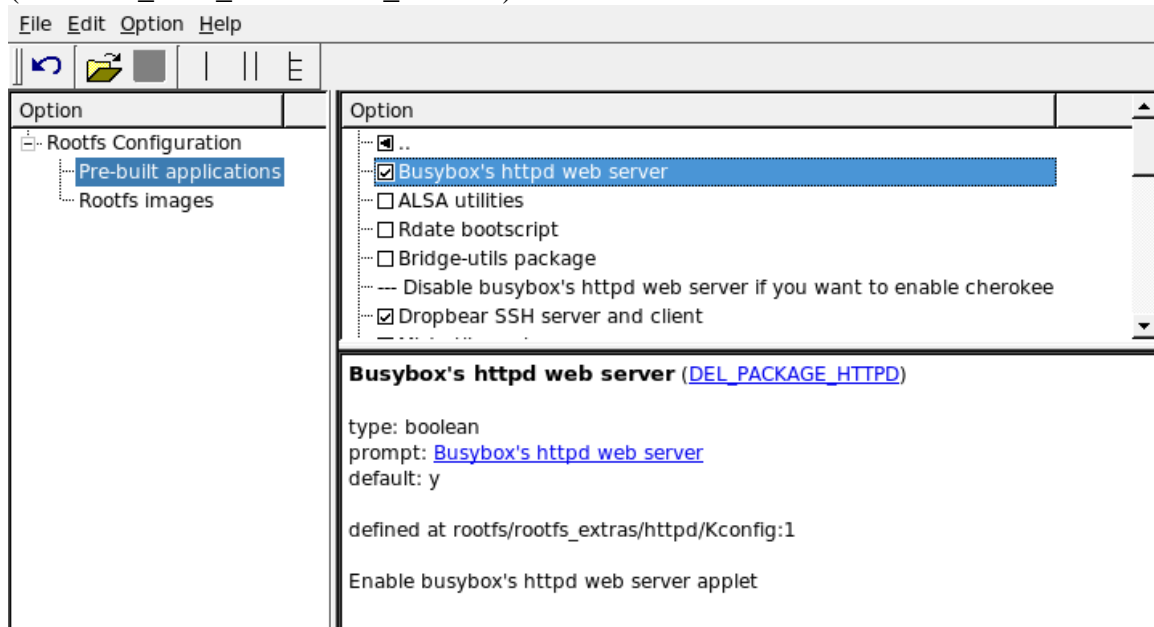


Save the changes.

### 3.4 *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:

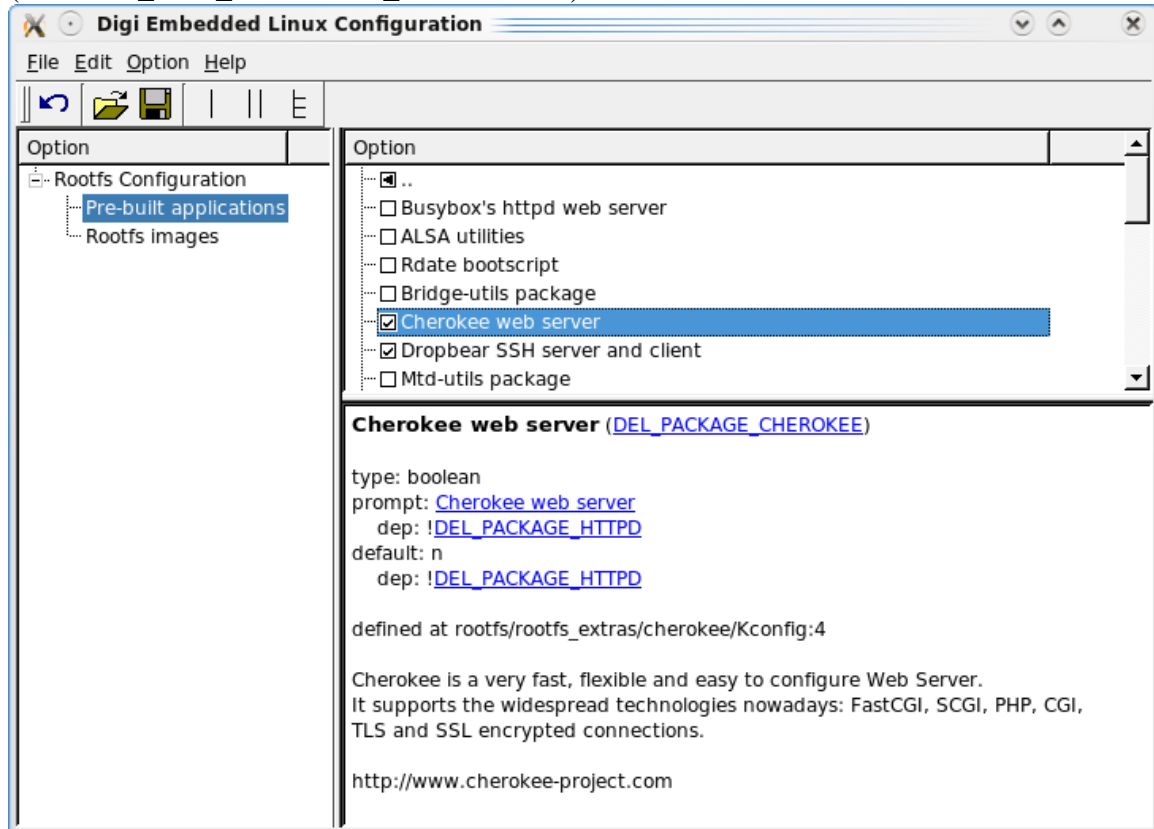
```
*.php:/usr/bin/php-cgi
```

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.



### 3.5 configure Cherokee webserver

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/ Cherokee.conf into your project directory and edit it.

### 3.6 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 webserver) to the rootfs:

```

File Edit Navigate Search Project Device options Run Package Manager Window Help
Project Explorer
9215kernel
9M2443UARTAppKit
9p9215
9P9215UARTAppKit
9P9360Kernel
9P9360UARTAppKit
calclient
cps220
PHPSQLite9215
  Binaries
  Archives
  Includes
  build
    kernel
    rootfs
    scripts
  configs
    add_files.sh
    Kconfig
  images
  rules
  usr
  ..

add_files.sh
### these variables are exported in topgit makefile.
###
### Though you can use all above variables, the recommended way is to customize
### the project rootfs (see ROOTFS_DIR below). Doing this way, your changes
### will be included in rootfs images and in nfs-exported rootfs as well.
###
#####
ROOTFS_DIR="${DEL_PROJ_DIR}/build/rootfs"

## Example: create a custom directory in rootfs etc dir.
# mkdir -p "${ROOTFS_DIR}/etc/myfolder"

# PHP interpreter
# first try prebuild:
cp "${DEL_PROJ_DIR}/php-cgi" "${ROOTFS_DIR}/usr/bin/php-cgi"
# if there is a self build one copy it over:
cp "${DEL_PROJ_DIR}/usr/bin/php-cgi" "${ROOTFS_DIR}/usr/bin/php-cgi"

# PHP test page
cp "${DEL_PROJ_DIR}/test.php" "${ROOTFS_DIR}/usr/share/www/test.php"

# SQLite API test page
cp "${DEL_PROJ_DIR}/sqlite-example.php" "${ROOTFS_DIR}/usr/share/www/sqlite-example.php"

# SQLite PDO test page
cp "${DEL_PROJ_DIR}/sqlite-pdo.php" "${ROOTFS_DIR}/usr/share/www/sqlite-pdo.php"
cp "${DEL_PROJ_DIR}/sqlite_test" "${ROOTFS_DIR}/usr/bin/sqlite_test"

# for busybox httpd:
cp "${DEL_PROJ_DIR}/httpd.conf" "${ROOTFS_DIR}/etc/httpd.conf"

# for Cherokee
mkdir -p "${ROOTFS_DIR}/etc/Cherokee"
cp "${DEL_PROJ_DIR}/Cherokee.conf" "${ROOTFS_DIR}/etc/Cherokee/Cherokee.conf"

exit 0
    
```

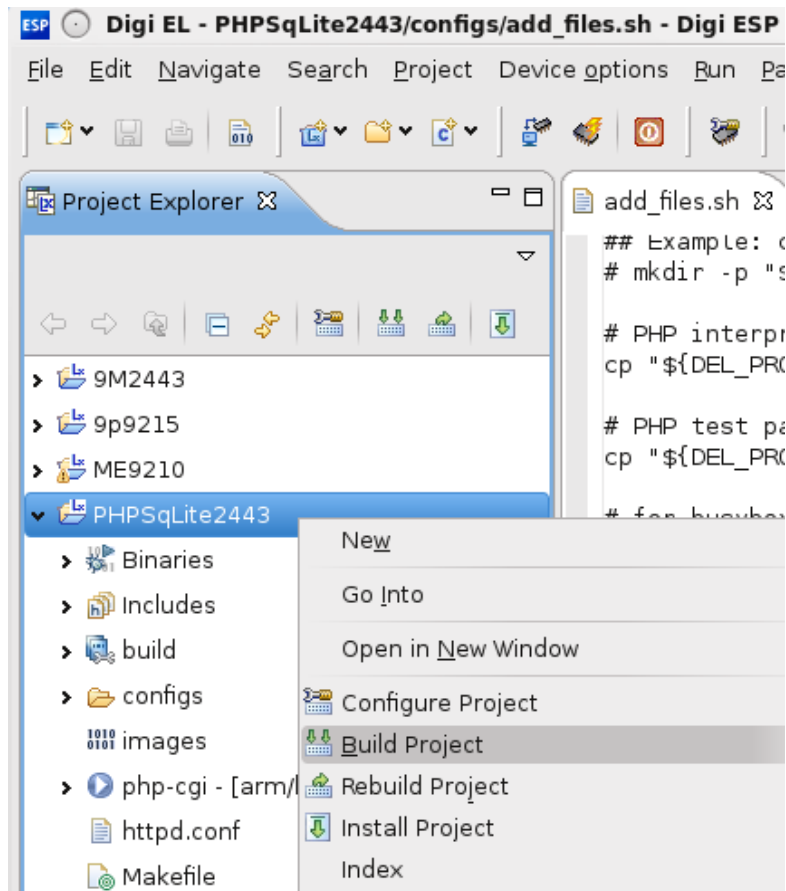
Also add test.php, sqlite-example.php, sqlite-pdo.php to your project directory.

```

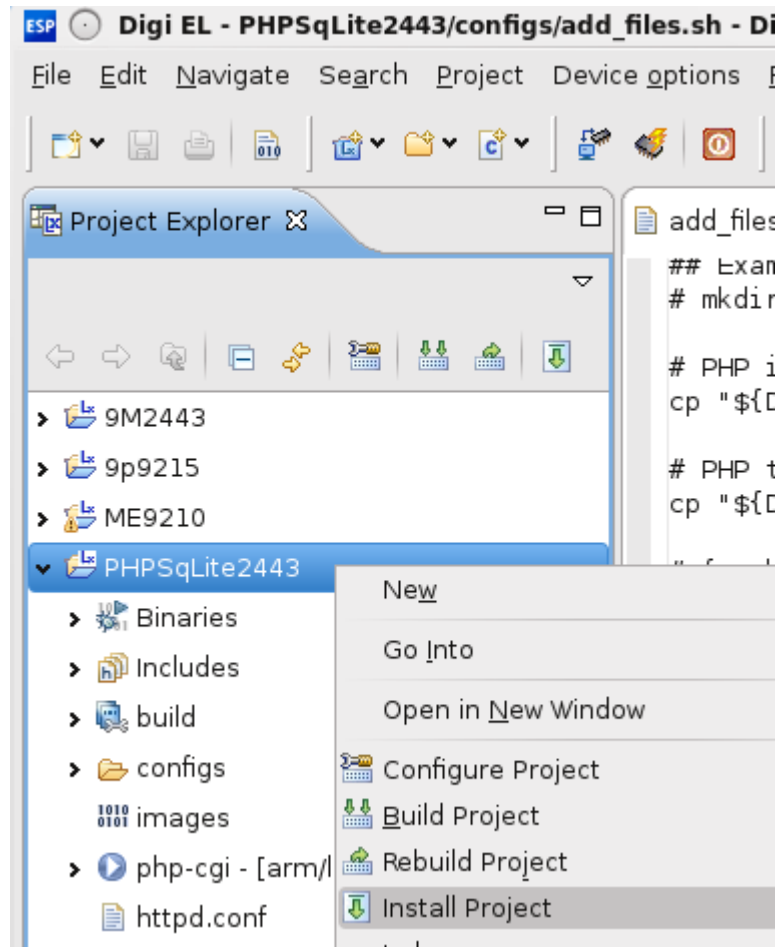
add_files.sh test.php
<html>
<body>
<p>Date: <?php print(Date("r")); ?>
<p>Server Info: <?php phpinfo(); ?>
</body>
</html>
    
```

## PHP with Digi Embedded Linux

- 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 ConnectCore 9P9215 development board use Serial Port D (CONSOLE).

## 5 Testing

Run the new 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”).


### 5.1 Testing with `sqlite_open()` API (SQLite 2.8.17)


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>

[SQLite example](#)

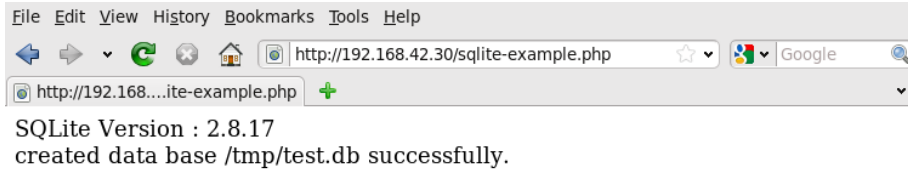
Date: Thu, 01 Jan 1970 00:01:24 +0000

Server Info:

PHP Version 5.2.13 	
<b>System</b>	Linux cc9p9215js 2.6.28.10 #1 Tue Jun 1 14:48:08 CEST 2010 armv5tej
<b>Build Date</b>	May 28 2010 16:16:49
<b>Configure Command</b>	'./configure'
<b>Server API</b>	CGI
<b>Virtual Directory Support</b>	disabled
<b>Configuration File (php.ini) Path</b>	/usr/lib
<b>Loaded Configuration File</b>	(none)
<b>Scan this dir for additional .ini files</b>	(none)
<b>additional .ini files parsed</b>	(none)
<b>PHP API</b>	20041225
<b>PHP Extension</b>	20060613
<b>Zend Extension</b>	220060519
<b>Debug Build</b>	no
<b>Thread Safety</b>	disabled
<b>Zend Memory Manager</b>	enabled
<b>IPv6 Support</b>	enabled
<b>Registered PHP Streams</b>	php, file, data, http, ftp
<b>Registered Stream Socket Transports</b>	tcp, udp, unix, udg
<b>Registered Stream Filters</b>	string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed

This program makes use of the Zend Scripting Language Engine:  
 Zend Engine v2.2.0, Copyright (c) 1998-2010 Zend Technologies 

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.



## initialize data base

```
INSERT INTO users (name, text, add) VALUES ( 'jbloggs', 'hello', 'world' )  
INSERT INTO users (name, text, add) VALUES ( 'this', 'is ', 'an example' )  
INSERT INTO users (name, text, add) VALUES ( 'just', 'another', '' )  
INSERT INTO users (name, text, add) VALUES ( 'somebody', 'has been', '' )  
INSERT INTO users (name, text, add) VALUES ( 'forget', 'all', '' )  
INSERT INTO users (name, text, add) VALUES ( 'Digi', 'rulz', '' )  
some db entries created
```

## data base

### data base output

jbloggs	hello	world
this	is	an example
just	another	
somebody	has been	
forget	all	
Digi	rulz	

Done

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

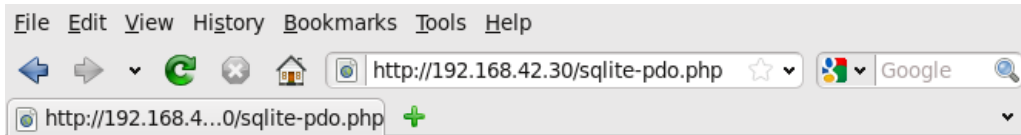


The generated test data base /tmp/test.db should be in SQLite 2.8.17 format.

## 5.2 Testing with PDO interface (SQLite 3.x support)

Point your browser to sqlite-pdo.php example, to create a SQLite 3 example data base with PHP and PDO interface:

## PHP with Digi Embedded Linux



Id	Breed	Name	Age
1	Labrador	Tank	2
2	Husky	Glacier	7
3	Golden-Doodle	Ellie	4
4	Labrador	Tank	2
5	Husky	Glacier	7
6	Golden-Doodle	Ellie	4
7	Labrador	Tank	2
8	Husky	Glacier	7
9	Golden-Doodle	Ellie	4

Done

On the serial console, run the `sqlite_test` program to access the PHP created SQLite 3 data base:

```
/ # sqlite_test /tmp/dogs_pdo.sqlite 'select * from Dogs'  
Id = 1  
Breed = Labrador  
Name = Tank  
Age = 2  
  
Id = 2  
Breed = Husky  
Name = Glacier  
Age = 7  
  
Id = 3  
Breed = Golden-Doodle  
Name = Ellie  
Age = 4  
  
Id = 4  
Breed = Labrador  
Name = Tank  
Age = 2  
  
Id = 5  
Breed = Husky  
Name = Glacier  
Age = 7  
  
Id = 6  
Breed = Golden-Doodle  
Name = Ellie  
Age = 4
```