

## Software Installation Guide

Author: P Milne

Date: 2002 11 27.

<i>Rev</i>	<i>Date</i>	<i>Who</i>	<i>What</i>
1	21127	pgm	issued

## Table of Contents

1	<a href="#">Introduction</a> .....	2
1.1	Aim.....	2
1.2	Overview.....	2
1.3	Exclusions.....	2
2	<a href="#">Prerequisites</a> .....	2
2.1	Build System.....	2
2.2	Target System.....	2
2.3	D-TACQ software deliverables.....	3
3	<a href="#">Software Build Process</a> .....	4
3.1	Inputs.....	4
3.1.1	Driver Source Archive.....	4
3.1.2	Application Source Archive.....	4
3.1.3	Kernel Configuration File.....	4
3.1.4	Documentation Archive.....	4
3.2	Outputs.....	5
3.2.1	Driver Object Archive.....	5
3.2.2	Apps Object Archive.....	5
3.3	Method.....	5
3.3.1	Build a standard kernel.....	5
3.3.2	Build the Software.....	6
4	<a href="#">Installation</a> .....	7
4.1	Install Objects.....	7
4.2	Ensure boot time driver load on target.....	7
4.3	Web interface.....	8
4.4	Update onboard firmware.....	8
5	<a href="#">Testing</a> .....	8
6	<a href="#">2G Driver Considerations</a> .....	9
6.1	Fixed Buffer Allocation.....	9

# 1 Introduction

## 1.1 Aim

The aim of this document is to supply all necessary information for building the D-TACQ driver and example application software, in order to run a dt100 networked data acquisition system

## 1.2 Overview

It is D-TACQ policy to supply all host driver and application software as source code under GPL. D-TACQ aim to demonstrate that the entire dt100 system can be built from source code.

## 1.3 Exclusions

The run time images for the embedded microprocessor and embedded Field Programmable Gate Array(s) are supplied as binary images, which may be updated in the field by reprogramming target flash memory. However the source code for this target resident firmware is D-TACQ proprietary information and is not released.

# 2 Prerequisites

The dt100 system is supported on Red Hat Linux 6.2 and 7.3 for Intel x86.

## 2.1 Build System

A disk based system, with development and kernel development packages installed is required to build the driver. It is necessary that the Build System run the same kernel version as the Target System. D-TACQ recommend building a customised kernel to run the dt100 system; while no special customisation is needed, poor results have been obtained with Red Hat stock kernels, and in any case it is good practise to be able to recreate the kernel from scratch.

A single user account is needed on the Build System for building software  
( D-TACQ default: “pgm” )

## 2.2 Target System

The target system may be disk based or diskless (so far we have only had working Red Hat 6.2 diskless systems; if you do better, please tell us). The main requirement of the Target system is that it have sufficient PCI slots and power supply Watts to support the required acquisition boards.

A single user account “dt100” is required to run the dt100 software.

It is very common that the Build and Target systems are the same unit.

### ***2.3 D-TACQ software deliverables.***

D-TACQ supply software in both source and object form, as described in the next section.

The main deliverable is a “Driver Source Archive”, with sources for the Linux driver, and a number of simple test apps and utilities, as well as current firmware images. It is highly recommended that this archive be build from scratch on a local build machine.

The second deliverable is the “Apps Source Archive”, comprising an embedded server and a remote control GUI test application (dt100rc). The applications are supplied as Java JARS so that the remote control will in principle run on any host computer with JRE1.4 installed. These applications may be used directly from supplied executables in the “Apps Executable Bundle”, or recreated from source. Recreating the Apps from source is not considered to be a critical activity for users, but is possible provided the JDK1.4 is available on the Build Machine.

Software is supplied in tgz archives, identified by type and a date code. In addition, the software will carry internal release numbers:

- Arm Firmware Build # : found from `acqcmd getFwrev`
- Driver Release Numbers : `cat /proc/.acq32`
- Application Release Number. : About box on dt100rc.

## 3 Software Build Process

In a software release, D-TACQ may provide the following:

### 3.1 Inputs

#### 3.1.1 Driver Source Archive

dt100\_src\_dt100\_datecode\_timestamp.tgz

eg:

dt100\_src\_dt100\_021115\_165017.tgz

#### 3.1.2 Application Source Archive

dt100\_apps\_src\_dt100\_datecode\_timestamp.tgz

eg:

dt100\_apps\_src\_dt100\_021115\_165017.tgz

#### 3.1.3 Kernel Configuration File

configs/dt100.datecode

#### 3.1.4 Documentation Archive

File: dt100www.tar.

The intention is that this tarball is extracted in the top level directory of a convenient webserver; this webserver is best located on each target system, as it allows simple web-based diagnostics on the target system.

Structure of the tarball:

./html/	
./html/index.shtml	Home Page
./html/DT100_Docs.html	Docs Index
./html/dt100KickstartInstall.html [.pdf]	Kickstart install instructions
./html/HitchikersGuideToTheDt100. html [pdf]	Structure and faultfinding guide
./html/OperatingModesSummary.pdf	Summary
./html/ICD/	
./html/ICD/dt100icd.html [pdf]	Command Reference manual
./cgi-bin/index.cgi	Diagnostic script

## **3.2 Outputs**

### **3.2.1 Driver Object Archive**

```
dt100_hostname_kernelversion_datecode_timestamp.tgz
```

eg

```
dt100_lewis_2.4.18-3.dt100.020702_021115_165845.tgz
```

### **3.2.2 Apps Object Archive**

```
dt100apps_hostname_kernelversion_datecode_timestamp.tgz
```

```
dt100apps_hostname_kernelversion_datecode_timestamp.zip
```

eg

```
dt100apps_lewis_2.4.18-3.dt100.020702_021115_165845.tgz
```

## **3.3 Method**

### **3.3.1 Build a standard kernel**

#### **1. Set kernel extension in Makefile**

```
vi /usr/src/linux/Makefile
```

```
#D-TACQ uses convention dt100.datecode to name kernels  
#EXTRAVERSION = -3.dt100.020702
```

#### **2. Clean**

```
cd /usr/src/linux  
make mrproper;
```

#### **3. Add new configuration**

```
copy dt100.020702 /usr/src/linux/.config
```

#### **4. Build, following usual procedure.**

```
make oldconfig;make bzImage  
make modules;make modules_install  
make install
```

### 3.3.2 Build the Software

Log in with user account (D-TACQ default: pgm)

1. Extract Source Archive on Host

```
tar xvzf dt100_src_dt100_021115_165017.tgz
```

2. Optionally Extract and build Applications

*NB: this only to build dt100rc – recommend using pre-built dt100rc.*

```
tar xvzf dt100_apps_src_dt100_021115_165017.tgz
cd BASE/DTACQ/SRC/dt100/Java
make release
```

3. Build Drivers

```
cd BASE/DTACQ/SRC/dt100/Linux
make all
make release
make release_package
```

The output of this process is a Driver Object Archive and optionally an Application Object Archive, in this example:

```
/home/pgm/BASE/DTACQ/SRC/dt100/Linux:
dt100_lewis_2.4.18-3_021114_143519.tgz:
    is the Driver Object Package
dt100apps_lewis_2.4.18-3.dt100.020702_021127_142229.tgz:
    is the Application Object Package.
```

```
[pgm@lewis Linux]$ make release_package
./bin/ship.dt100
ship.dt100 mode regular
dir bin found OK
dir CAL found OK
dir API found OK
dir Drivers found OK
dir mex found OK
dir etc found OK
-----
Making ship file dt100_lewis_2.4.18-3.dt100.020702_021127_142229
cp: omitting directory
`../ArmImages/CVS'/home/pgm/dt100_lewis_2.4.18-
3.dt100.020702_021127_142229.tgz
1224992 Nov 27 14:22 dt100_lewis_2.4.18-
3.dt100.020702_021127_142229.tgz
-----
Making apps file dt100apps_lewis_2.4.18-3.dt100.020702_021127_142229
/home/pgm/dt100apps_lewis_2.4.18-3.dt100.020702_021127_142229.tgz
1110227 Nov 27 14:22 dt100apps_lewis_2.4.18-
3.dt100.020702_021127_142229.tgz
-----
adding: client/ (stored 0%)
adding: client/default-resources (deflated 44%)
adding: client/dt100rc (deflated 2%)
adding: client/DT100RC.BAT (deflated 3%)
adding: client/jars/ (stored 0%)
adding: client/jars/dt100rc.jar (deflated 11%)
adding: server/ (stored 0%)
adding: server/dt100d.sh (deflated 28%)
adding: server/DtacqServer.jar (deflated 9%)
adding: server/shutdown.sh (deflated 17%)
/home/pgm/dt100apps_lewis_2.4.18-3.dt100.020702_021127_142229.zip
1111154 Nov 27 14:22 dt100apps_lewis_2.4.18-
3.dt100.020702_021127_142229.zip
```

*Text example transcript from make release\_package*

## 4 Installation

### 4.1 Install Objects

Copy the Driver Object Package followed by the Application Object Package to ~dt100 on the target machine and extract.

```
cp xxxx.tgz ~dt100;cd ~dt100; tar xvzf xxxx.tgz
```

### 4.2 Ensure boot time driver load on target

```
# you'll need to be root to do this
# this only needs to be done ONCE (ie not required for upgrades)
cd ~dt100
cp etc/rc.d/init.d/dt100 /etc/rc.d/init.d/dt100

# if remote control is required:
cp etc/rc.d/init.d/dt100d /etc/rc.d/init.d/dt100d

set up the sysV runlevel options using tksysv or equivalent.
```

The kickstart install script takes a no-nonsense approach:

```
cp ~dt100/etc/rc.d/init.d/* /etc/rc.d/init.d
(cd /etc/rc.d/rc3.d;ln -s ../init.d/dt100 S98dt100)
(cd /etc/rc.d/rc3.d;ln -s ../init.d/dt100d S99dt100d)
```

### 4.3 Web interface

```
cd /var/www/; tar xvf dt100www.tar
```

```
# NB: on non-kickstart installs, httpd will need to be tweaked:
perl -i.orig -p -e 's/(Options Indexes FollowSymLinks)/$1
+Includes/' /etc/httpd/conf/httpd.conf
```

### 4.4 Update onboard firmware

First reboot and check that driver loads correctly

(check System screen on Web interface, or

```
[pgm@lewis dt100-install]$ cat /proc/acq32
model n chn      s/n      arm-fw      lca-fw      cal date cycle heartbeat
ACQ32 0 32  D22222  boot66c3    acq32-lca    251299  0 98
ACQ32 1 32  D22223  boot66c3    acq32-lca    251299  1 99
ACQ32 2 32  D22224  boot66c3    acq32-lca    251299  2 100
ACQ32 3 32  D22225  boot66c3    acq32-lca    251299  3 101
```

To upgrade the firmware, log into Target as dt100:

```
[dt100@kst dt100]$ cd ArmImages/
[dt100@kst ArmImages]$ ./upgrade.all
```

Upgrade takes a little over a minute, after upgrade, reboot.

## 5 Testing

Follow the steps indicated in “The HitchikersGuideToTheDt100.html”

The Web status page gives a quick indication as to board health: look for non-zero heartbeat.

D-TACQ recommend using the remote control application to verify that the system is working correctly.

## 6 2G Driver Considerations.

The 2G driver is build from the same source directory as 1G.

The 2G driver is loaded via modules/acq200.load

### 6.1 Fixed Buffer Allocation.

Each 2G card requires a 16MB pre-allocated comms buffer in host memory.

Currently this is allocated using the “big buf” allocation method, assigning the memory at boot time. The easiest way to do this is to make an entry in grub.conf:

Set the following parameters:

mem= Limit memory available to Linux.

NB: for Linux 2.4, leave a deadband ~32 MB.

acq32\_big\_buf\_base=BASE-ADDRESS

acq32\_big\_buf\_len=LEN

Choose acq32\_big\_buf\_len = MAX 2G boards x 16MB

eg 2 boards,

2 x 0x01000000 => 0x02000000

Choose acq32\_big\_buf\_base= TOP OF MEMORY - acq32\_big\_buf\_len

eg mem = 512MB (0x20000000)

Choose acq32\_big\_buf\_base=0x1e000000

Finally choose mem= as acq32\_big\_buf – 32MB (0x02000000)

eg mem=448m

Example (all one line in grub.conf):

```
ro root=LABEL=/  
  
mem=320M
```

```
                ;# screen off Linux
```

```
acq32_big_buf_base=0x18000000
```

```
                ;# base addr of buffer
```

```
acq32_big_buf_len=0x08000000
```

```
                ;# length of buffer
```

