D-TACQ 2G Programmer's Guide

Prepared By: Peter Milne

Date: 14 June 2005

Rev	***Date***	***Description***
1	01/03/05	First issue
2	14/06/05	Update kernel version

Table of Contents

Document created using OpenOffice.Org www.openoffice.org.
PDF rendition by extendedPDF, www.jdisoftware.co.uk

This document and D-TACQ Software comprising platform Linux port, Linux kernel modules and most applications are released under GNU GPL/FDL:

Document:

Software:

1 Introduction

1.1 Scope

This document gives an overview of the structure of the D-TACQ 2G Firmware and supporting host side software, and describes the build procedure. The overview includes a discussion of the various application/driver software interfaces; This complements the external interface definition in [1], [2]; in addition, it does not attempt to be a complete and definitive definition of these internal interfaces, for which the ultimate reference is the source code. In addition, while D-TACQ will attempt to maintain these interfaces between releases, this is not guaranteed.

1.2 Audience

Users of this document will be programmers wishing to extend the firmware capability of the 2G product. In order to use the standard product, it is not necessary to write any firmware, and users are referred to [1].

1.3 References

1. D-TACQ 2G User Guide (2GUG) .
2. Dt100 Interface Control Document (ICD).
3. Building Embedded Linux Systems – Yaghmour (O'Reilly).
4. Linux Device Drivers – Rubini & Corbet (O'Reilly).
5. Www.lwn.net - Corbet's website – subscription recommended.

1.4 Coding Standards

Coding style is that used by the Linux Kernel.

2 Firmware Structure

2.1 Flash Disk Images

Images marked RW may be updated from Linux, others are not user-updateable, with the exception of Env, the u-boot enviroment which may be updated from within u-boot.

Images are stored as:

- binary (B)
- u-boot bootlable image (U)
- compressed ram disk image (Z)
- compressed FPGA rbt image (RBT)
- flash file system FFS

```
f0000000-f001ffff : /dev/mtd0 Bootldr           RO B
f0020000-f003ffff : /dev/mtd1 Env               RO B
f0040000-f01bffff : /dev/mtd2 SafeKrn           RO U
f01c0000-f01fffff : /dev/mtd3 FPGA              RW RBT
f0200000-f03fffff : /dev/mtd4 SafeRd            RO U
f0400000-f05fffff : /dev/mtd5 FieldKrn          RW U
f0600000-f07fffff : /dev/mtd6 FieldRd           RW U
f0000000-f0bbffff : /dev/mtd7 extra             RW Z
f0bc0000-f0f3ffff : /dev/mtd8 home              RW Z
f0f40000-f0ffffff : /dev/mtd9 cal               RW F
```

2.1.1 /dev/mtd0 – Bootloader image u-boot – not user modifiable

2.1.2 /dev/mtd1 – Env – u-boot environment, modified from within u-boot

2.1.3 /dev/mtd2 – Safe Krn – backup kernel image

2.1.4 /dev/mtd3 – FPGA – default FPGA image

2.1.5 /dev/mtd4 – Safe RD – backup initrd disk image

2.1.6 /dev/mtd5 – Field Krn – field (current) kernel image

2.1.7 /dev/mtd6 – Field Rd – field (current) initrd image

2.1.8 /dev/mtd7 – extra – field (current) extra file system image

2.1.9 /dev/mtd8 – home – spare (user) file system image

2.1.10 /dev/mtd2 – cal – jffs2 flash file system image for non volatile file store.

2.2 Ramdisk Structure

2.3 Possibilities for Extension

- NFS mount
- Store in /ffs
- Store in existing ramdisk image
- Build new ramdisk image.

3 Tools

3.1 Host System

Host System: Linux, x86, kernel 2.2.4 or better.

You will need root access, and a large amount of disk space.

A high spec system will give better results. The D-TACQ reference system is:

3GHz P-IV, 1GB memory, 100GB disk, Fedora Core 1.

The build system user should have unlimited root capablity using sudo.

3.2 Tool Chain

All arm executables are built using gcc, and linked to standard libraries glibc etc.

To build a cross compiler, follow the excellent auto build procedure at:

http://www.kegel.com/crosstool/

The tool chain currently in use is:

```
[pgm@islay pgm]$ arm-xscale-linux-gnu-gcc --version
arm-xscale-linux-gnu-gcc (GCC) 3.4.1
Copyright (C) 2004 Free Software Foundation, Inc.
```

3.3 Browsing Sources

D-TACQ uses the Source Navigator tool extensively to aid understanding complex project structures.

4 Host Computer File Structure:

4.1 Kernel

Please patch standard kernel.org software with latest patch fromwww.d-tacq.com.

Current version is Linux 2.6.11

Path	**description**
/arm-linux/	
/arm-linux/bin	Build tools
/arm-linux/acqX00	Kernel build tree
/arm-linux/acqX00/ arch/arm/mach-iop3xx/acq200/	Most machine specific files to be found here

4.2 Main Project

The project is set up approximately according to Yaghmour [3].

~/PROJECTS/ACQ200/

.	
./bin	Host tools here
./bootldr/u-boot-0.3.0	Bootloader sources
./debug	Gdb sources
./project	Applications found here
./rootfs	Anchor point for root file systems build
./sysapps	
./tmp	
./tools	
./images	Images stored here
./quick-release	

4.3 File System Images

~/PROJECTS/ACQ200/rootfs

./rootfs/extra	Extra file system image
./rootfs/cal	JFFS2 file system image
./rootfs/initrd	Root file system image
./rootfs/mdsplus	Example user system image.

4.3.1 initrd

./rootfs/initrd	
./rootfs/initrd/acq200	Acq200 specific part
./rootfs/initrd/bin	
./rootfs/initrd/cgi-bin	
./rootfs/initrd/dev	
./rootfs/initrd/etc	
./rootfs/initrd/etc/init.d	
./rootfs/initrd/extra	
./rootfs/initrd/home	
./rootfs/initrd/lib	
./rootfs/initrd/lib/modules	
./rootfs/initrd/mnt	
./rootfs/initrd/proc	
./rootfs/initrd/root	
./rootfs/initrd/sbin	
./rootfs/initrd/sys	Mount point sysfs
./rootfs/initrd/tmp	
./rootfs/initrd/usr	
./rootfs/initrd/usr/local	
./rootfs/initrd/usr/sbin	
./rootfs/initrd/var	

4.3.2 extra

./rootfs/extra	
./rootfs/extra/home	Mounted at /home
./rootfs/extra/home/dt100	
./rootfs/extra/home/pgm	
./rootfs/extra/home/rtmdds	
./rootfs/extra/local	Mounted at /usr/local
./rootfs/extra/local/bin	
./rootfs/extra/local/fpga	
./rootfs/extra/local/lib	
./rootfs/extra/local/sbin	
./rootfs/extra/local/CARE	/usr/local/CARE – self test scripts.
./rootfs/extra/local/cgi-bin	

4.3.3 cal

./rootfs/cal	
./rootfs/cal/cal	
./rootfs/cal/safe	
./rootfs/cal/tmp	
./rootfs/cal/dropbear	Copied to / at boot time.
./rootfs/cal/dropbear/home	
./rootfs/cal/dropbear/root	

5 Build Procedure

5.1 Build a stand alone ARM application

This is very simple. Applications are standard Linux applications, they will run in a regular VM linux environment, linking to standard GLIBC, and, importantly Name Server Switch libraries. This is a full Linux implementation, as opposed to uClinux or uCLibc, diet LibC etc. It should also be possible to run C++ applications, however the C++ run time library is not loaded by default for space reasons.

Build sources using a Makefile. Ideally this application can be built to run on the Host computer as follows.

- Build and test on Host

```
make clean;make all;
```

```
# run test (as far as it goes)
```

- Cross Compile for ARM:

```
make clean;CC=arm-xscale-linux-gnu-gcc make all;
```

- Test on ARM

```
scp my-app root@target:/tmp
```

- [optionally] Store in /ffs for non volatile storage
- [optionally] Incorporate in a customised version of the extra ramdisk image.
- [optionally] Incorporate in a custom ramdisk image

A gdbserver implementation is available on the ARM system, however in practise this is not very useful, and a debug style using switchable debug printouts has been found to be more useful.

5.2 Bootloader

The bootloader is u-boot, customised for this application.

It is not possible for users to modify the u-boot sector, however sources are supplied under GPL rules.

5.3 Kernel

```
cd /arm-linux/acqX00;make vmlinux; ../bin/mkimage.uboot
# produces vmlinux.img
```

5.4 Kernel Modules

```
cd /arm-linux/acqX00;make modules;
sudo ../bin/deploy.modules
```

5.5 Disk Images

```
cd PROJECTS/ACQ200;./bin/wrap.initrd initrd
# produces initrd.boot

cd PROJECTS/ACQ200;./bin/wrap.initrd extra
# produces extra.img.gz

cd PROJECTS/ACQ200;./bin/wrap.initrd cal
# produces cal.img (images for default ffs)

cd PROJECTS/ACQ200;./bin/wrap.initrd mdsplus
# produces mdsplus.img.gz - example User file system image.
```

5.6 Applications

(typical).

```
cd PROJECTS/ACQ200/project/{MYPROJECT};make all deploy
```

6 Interfaces

6.1 /dev/dtacq_drv/

6.2 /dev/dtacq/

6.3 /dev/acq200/data

6.4 /dev/acq32/

6.5 /proc/driver/acq200/