ACQ400 Architecture Guide

High Performance Simultaneous Data Acquisition

DOC #XXXXXX-YY

Revision History

Revision	Date	Author(s)	Description
0.1	2022-04-29	PM	Created
0.2	2022-02-17	PM	update to D-TACQ Style
0.3	2022-02-17	PM	appendix FPGA personality description

Contents

1 Introduction **3**
1.1 ACQ400 Architecture . . . 3
1.2 Intended Audience . . . 3
1.3 Scope . . . 3
1.4 Glossary . . . 3
1.5 References . . . 4
1.6 Notation . . . 4

2 Overview **4**
2.1 Single Software Image, Multiple FPGA Personalities . . . 4
2.2 Embedded Linux System . . . 4
2.3 Self Identifying System . . . 5
2.4 ESW Components . . . 5
2.4.1 Bootloader: u-boot . . . 5
2.4.2 uImage: kernel image . . . 5
2.4.3 initrd : initial ramdisk image . . . 5
2.4.4 devicetree . . . 5
2.4.5 rootfs.tgz . . . 5
2.4.6 Filesystem overlays . . . 5
2.4.7 Packages . . . 6
2.5 SD Boot or QSPI boot . . . 6
2.6 Boot Sequence . . . 6
2.6.1 BOOT.BIN . . . 6
2.6.2 u-boot . . . 6
2.6.3 uImage . . . 6
2.6.4 initrd . . . 6
2.6.5 packages . . . 7
2.6.6 /mnt/local/rc.user . . . 7
2.7 Package Sequence . . . 7
2.7.1 03-acq400_common . . . 7
2.7.2 05-machine . . . 7
2.7.3 10-acq420 . . . 7
2.7.4 20-httpd . . . 7
2.7.5 40-acq400ioc . . . 7
2.7.6 Custom Packages . . . 7
2.8 SD File System Layout . . . 7
2.9 SD card top-level files . . . 8

3 User/SystemIntegrator Guide **8**
3.1 u-boot environment . . . 8
3.1.1 z7io . . . 8
3.2 kernel options . . . 8
3.3 userland options . . . 8
3.4 packages . . . 8

4 Developer Guide **8**
4.1 Build ESW . . . 8
4.1.1 ESW . . . 8
4.1.2 Making a Complete Firmware Release . . . 9
4.2 BOOT.BIN . . . 9
4.2.1 z7io . . . 9
4.3 kernel and devicetrees . . . 9

5 APPENDIX **10**
5.1 FPGA PERSONALITIES . . . 10

1 Introduction

1.1 ACQ400 Architecture

The ACQ400 Architecture is a firmware framework used by all D-TACQ “4G” DAQ Appliances, based on the ZYNQ-7000 platform, including

1. Compact DAQ Appliances: ACQ1001, ACQ1002, ACQ1014
2. Mainframe DAQ Appliances: ACQ2106, ACQ2206
3. MTCA: KMCU-Z30, Z7IO
4. Custom: CPSC2

This document describes the structure of the Embedded Software ESW in 3 levels of detail

1. Overview - a brief description
2. User/SystemIntegrator - a detailed at the boot process and structure of the elements
3. Developer - describes how to build the system from scratch.

1.2 Intended Audience

1. Users
2. Developers

1.3 Scope

1. ACQ400 Embedded Software
2. excludes FPGA development.

1.4 Glossary

1. *ACQ400* Embedded Software, excludes FPGA development.
2. *DAQ Appliance*: a freestanding networked data acquisition system
3. *ZYNQ 7000* (Z7, Z7000) : Xilinx system on chip architecture.
4. *PS* : Processor System in Z7000
5. *PL* : Programmable Logic (FPGA part) of Z7000
6. *A9* : Dual Core ARM A9 processor found in Z7000, the processor in PS
7. *ESW* : Embedded SoftWare : software to run on the A9
8. *Gateware*: programmable logic for the PL. This is delivered as an
9. *FPGA* Personality file.
10. *Firmware Release*: combination of all ESW and Gateware needed to run an ACQ400 System..
11. *FMC*: Vita 57 standard for expansion modules
12. *ELF* : D-TACQ extension of FMC : Analog module uses a limited subset of FMC I/O, optional larger size module, to allow a larger analog payload while at the same time using less FPGA pin resource.
13. *Carrier/Motherboard/MB*: Electronic circuit board with Z7000 SOC, DRAM, Networking and one or more FMC or ELF payload sites.
14. *QSPI Flash*: FLASH memory in the Z7000 system, typically used for customization and robust bootloader storage.

1.5 References

1. 4GUG

1.6 Notation

1. command : indicates name of a program (command)
2. preformatted text : literal input or output from terminal session.
3. *Defined Term* : some term or acronym specific to this domain (perhaps referenced in the glossary)

2 Overview

2.1 Single Software Image, Multiple FPGA Personalities

The *ACQ400* firmware system runs on about 10 carrier platforms 1.1, supporting a wide range (>10) analog payload modules. Each hardware combination of modules and sites, is supported by a unique FPGA personality 5.1. A single software image may handle all platforms (excepting CPSC2, which has a forked image).

2.2 Embedded Linux System

The *ACQ400* system runs Linux. A single Linux image supports all variants. This is the standard Linux kernel, and a fairly conventional userland, with two exceptions:

1. Kernel DMA subsystem is non-standard.
2. The system root file system rootfs is a RAMDISK.

The RAMDISK rootfs is chosen for

1. Robustness: the file system is volatile, and the entire system may be reset or powered off at any time during operation.
2. Realtime performance: the DAQ appliance is a real time system, and must not be subject to any unpredictable delay eg disk access, or swapping.

The ZYNQ7000 has strictly limited DRAM: 1GB. In addition, *ACQ400* systems, being aimed at large scale data collection, typically assign at least half the DRAM to data buffers. So this is a restricted memory environment, and the size of the RAMDISK must therefore be kept in bounds. The systems are typically fitted with an SD card, and the system boots from the SD card.

1. The SD card is treated as a ROM. There's no writeable access to the SD card at runtime on a production system.
2. However, the SD card is writeable, this is convenient for system maintenance and for firmware upgrade. After deployment, there should be no files opened for write while the unit is deployed, so it's still safe to cut the power at any time.

The RAMDISK isn't ideal for a limited memory system, and a pure ramdisk system (such as D-TACQ ACQ200) has a very limited userland. *ACQ400* has a rich user-land, based on Buildroot. Each *ACQ400* system also carries a large inventory (can be 100+) of FPGA personalities. *ACQ400* handles this by mounting additional read-only file systems from file images on the SD card. The OS then has access to a wide range of software images, loaded on demand. In a typical system, most of this functionality is swapped out at runtime, preserving DRAM. Most systems are SD based, however two types of netboot systems are also supported:

1. Pure TFTP : the entire system boots from a RAMDISK image loaded via TFTP. This has a limited userland, and it does depend on a TFTP server for boot, but once running is self-standing and extremely robust. CPSC2 is the main example of this method.
2. NFS : the pure TFTP system boots a limited userland, then brings in the rest of the userland as an NFS mount. This allows high functionality with limited stored state, but it does totally depend on a reliable NFS server during run time.

2.3 Self Identifying System

1. On boot, *ESW* will identify which motherboard it is running on.
2. Then it will use SMB (i2c) to identify the current payload ie which analog modules are present in the module sites
3. The system will then select the first available FPGA personality, and load the FPGA
4. The system then proceeds to load kernel modules, instantiate device drivers and all services needed to build the system.

=> The same *ESW* image can run any supported personality, based on runtime identification. Z7IO: currently, the *ESW* is not able to complete system indentification before loading the FPGA, so this is selected at boot time.

2.4 ESW Components

2.4.1 Bootloader: u-boot

The system bootloader. u-boot will read local customization from the u-boot environment, then boot the system.

2.4.2 uImage: kernel image

The kernel image contains the Linux kernel executable code.

2.4.3 initrd : initial ramdisk image

The initial filesystem image, with limited functionality, this provides the first userland after boot.

2.4.4 devicetree

This is a data structure that defines the architecture of the particular motherboard, and any customisations.

1. eg ACQ1001 or ACQ2106, dtb.d/acq1001.dtb, dtb.d/acq2106.dtb
2. eg ACQ2106 (no sfp) or ACQ2106sfp (with sfp)., dtb/acq2106.dtb, dtb.d/acq2106sfp.dtb

The boot-time dveicetree may be augmented later in the boot-sequence by ”devicetree-overlays”, that define additional devices once more is known about the system.

2.4.5 rootfs.tgz

This is the main userland. This is created using Buildroot. This is expanded onto the SD card at install time, then mounted onto the ramdisk at run time, with the goal that most of the userland will NOT be resident in memory at runtime.

2.4.6 Filesystem overlays

These are read-only file system images mounted onto the ramdisk, and most of these images are NOT resident in memory at runtime. The most prominent overlays are:

1. FPGA stock: a “stock” of all known FPGA personalities; one of these files may be selected to load the FPGA on boot.
2. ko stock : a stock of kernel objects (device drivers), and appropriate modules will be loaded on boot.
3. EPICS7 : a full EPICS7 implementation is quite large, so it is shipped as an overlay rather than loading everything to RAMDISK.

2.4.7 Packages

ACQ400 uses a sequenced set of packages to configure the system; packages are tarballs that are expanded onto the ramdisk in sequence, together with initialisation files that are used to instantiate the entire system. Packages may reference overlays to save RAM, and they may load devicetree-overlays to instantiate new devices.

2.5 SD Boot or QSPI boot

Z7 systems typically have a jumper setting to boot from SD (certainly, for all newly assembled units), or from QSPI (soldered flash). On D-TACQ motherboards, initial boot is from SD, then the QSPI is programmed, and the jumper set to QSPI boot. The QSPI contains:

1. (minimum) : u-boot environment ie per-motherboard customization.
2. (ideally) : the bootloader, then a motherboard can boot to a prompt even with no SD card present, and remote recovery is possible.

2.6 Boot Sequence

2.6.1 BOOT.BIN

First to load is BOOT.BIN, a file created with Xilinx tools containing:

1. (minimum) : FSBL: First Stage Bootloader, a fixed boot kernel, and a motherboard specific IO configuration.
2. (ideally) : u-boot : the main bootloader.

As per 2.5 XREF HOWTO?, in a production system this image is held in QSPI for robustness

2.6.2 u-boot

The u-boot environment includes a BOOTCMD macro that specifies:

1. module ID (serial number, mac address) :: (ACQx0xx)
2. devicetree
3. kernel image
4. initrd

BOOTCMD loads the three images, and boots uImage

2.6.3 uImage

This is a standard Linux ARM kernel boot up, the Linux system is configured, and all the devices in the devicetree are instantiated, most importantly:

1. Ethernet eth0 (or eth1)
2. All i2c proms in the self-identification system

Finally uImage sets up the initrd and hands over to userland.

2.6.4 initrd

The init system runs /etc/init.d/rcS, a highly customized script that will

1. mount the rootfs file system
2. perform any custom network configuration
3. load packages in sequence.

2.6.5 packages

This is the real DAQ Appliance application userland, discussed in more detail in the next section.

2.6.6 /mnt/local/rc.user

Last to run is the file /mnt/local/rc.user, this contains final factory and/or user customisation. rc.user can be configured to make a completely turnkey system.

2.7 Package Sequence

2.7.1 03-acq400_common

Basic setup common to all platforms.

2.7.2 05-machine

Machine-specific setup eg 05-acq1001-yymmdd.tgz In particular, this will enumerate the site payload modules, select an FPGA personality and load the FPGA 5.1.

2.7.3 10-acq420

This is the main device driver and operating code of the *ACQ400* system. Once the FPGA has been loaded, then the core device driver acq420fmc.ko is loaded, and this instantiates a Linux device per site, bringing up all site services, and loading associated kernel modules, notably to instantiate DMA controllers. The package also provides acq400stream, the main data-moving application.

2.7.4 20-httpd

Configures and starts the embedded webserver. The embedded webserver provides useful system observability and diagnostics.

2.7.5 40-acq400ioc

This package loads and configures the embedded EPICS IOC, responsible for running most of the “business logic” of the DAQ appliance, and providing a rich set of control and monitor points for external clients.

2.7.6 Custom Packages

Systems may be customized by installing “nn-custom_PACKAGE modules. ”install” copy the package to the active package directory: /mnt/packages

2.8 SD File System Layout

The SD card appears in the booted linux system as /mnt

File	Description
/mnt/ko	mountable file system images eg
fpga-511-yymmdd.img	FPGA STOCK
packageko-4.14.0-acq400-xilinx-yymmdd.img	kernel object STOCK
/mnt/packages	live packages are found here
/mnt/packages.opt	optional (inactive) packages
/mnt/bin	scripts for firmware upgrade
/mnt/boot.d	BOOT.bin instances for supported carriers.
/mnt/dtb.d	stock of devicetree instances
/mnt/fpga.d	FPGA stock mounted here
/mnt/local	user/system customisation..
/mnt/local/cal	calibration files for current payload
/mnt/local/sysconfig	system customization.

2.9 SD card top-level files

BOOT.BIN	The specific bootloader for this carrier (for SD boot)
uImage	The kernel image
uramdisk.image.gz	The initial ramdisk initrd
rootfs.ext2	The root file system, to be mounted onto the ramdisk
Optionally	•
ACQ2106_TOP_04_ff_ff_ff_ff_ff_9011_32B_UDP.bit.gz:	Promoted FPGA Image

This is a “PROMOTED” fpga image - provided this image is compatible with the module payload, it will be loaded in preference to the first compatible image in the fpga stock. This is useful for:

1. Development: load a test image in preference to the release image.
2. Patch: when the FPGA image doesn’t exist in the release yet.
3. Promoted: when the stock contains multiple compatible images, and we want to force a particular personality to load eg: DEC10 filter.

3 User/SystemIntegrator Guide

3.1 u-boot environment

3.1.1 z7io

z7io has NO unit-specific customization in the u-boot environment. Instead, the z7io u-boot reads the eth0 MAC address from a soldered PROM.

3.2 kernel options

3.3 userland options

3.4 packages

4 Developer Guide

4.1 Build ESW

4.1.1 ESW

The entire *ACQ400 ESW* software suite is Open Source, and is presented by a single git top level archive with submodules: ACQ400_ESW_TOP and may be built from scratch by following these build instructions

4.1.2 Making a Complete Firmware Release

A complete bootable SD card image may be assembled as foloows

1. Build *ESW*
2. Include appropriate BOOT.BIN if needed
3. Include appropriate FPGA filesystem image.
4. Package and deploy the release

4.2 BOOT.BIN

4.2.1 z7io

z7io BOOT.BIN is built from DESY sources.

4.3 kernel and devicetrees

5 APPENDIX

5.1 FPGA PERSONALITIES

Let's try explain some personalities. Each Module has a unique "MTYPE", so that we tie the module in the site to the FPGA personality. ACQ400_ModuleIDs.pdf

For example:

1. ACQ435 has M=02
2. ACQ465 has M=0A
3. AO424 has M=41
4. DIO482 has M=6B

You can see the current stock of personalities on any *ACQ400* unit, in /mnt/fpga.d

```
# 162 personalities in stock!
acq2206_001> find /mnt/fpga.d -type f -print | wc
162       162      8296
```

```
# 85 of them for ACQ2106
acq2206_001> find /mnt/fpga.d -type f -print | grep ACQ2106 | wc
85        85      5103
```

```
# what support for ACQ465 do we already have?
acq2206_001> find /mnt/fpga.d -type f -print | grep ACQ2106 | grep 0A
/mnt/fpga.d/ACQ2106_TOP_0A_0A_0A_0A_0A_0A_9011_32B.bit.gz
# format
/mnt/fpga.d/CARRIER_____S1_S2_S3_S4_S5_S6_COMMS_FEATURES
```

```
# ZERO released personalities so far for ACQ2206
acq2206_001> find /mnt/fpga.d -type f -print | grep ACQ2206 | wc
0         0         0

# So how does this unit work?. We patched in an appropriate personality to load preferentially:

acq2206_001> cat /tmp/fpga_status
load.fpga loaded /mnt/ACQ2206_TOP_02_02_02_02_02_02_9011_32B.bit.gz
xiloader r1.01 (c) D-TACQ Solutions
eoh_location set 0
Xilinx Bitstream header.
built with tool version   : 48
generated from filename   : ACQ2206_TOP_02_02_02_02_02_02_9011_32B
part                      : 7z030ffg676
date                      : 2023/02/09
time                      : 17:45:05
bitstream data starts at  : 134
bitstream data size       : 5979916

ERROR: loaded FPGA image NOT in RELEASE at all

# this is compatible with the current payload: M=02 in S1, and the COMMS module M=90

CARRIER
SITE      MANUFACTURER           MODEL                                PART          SERIAL
0     D-TACQ Solutions      acq2206sfp                          acq2206sfp       CE4260001
-------------------------------------------------------------------------------------------
build detail: root@rpi-008 R1010 Fri Jan 27 14:29:53 UTC 2023
eth0 macaddr: 00:21:54:14:00:01    eth0  ipaddr: 10.12.197.98
eth1 macaddr: 00:21:54:24:00:01    eth1  ipaddr:
-------------------------------------------------------------------------------------------

MODULES
SITE      MANUFACTURER           MODEL                                PART          SERIAL
1     D-TACQ Solutions       ACQ435ELF         ACQ435ELF-32FF-5V N=32 M=02       E43510193
C     D-TACQ Solutions          MGT483               MGT483-SFP4 N=3 M=90       AM4831001

# If a site has no module, then it's a wildcard match.
```

```
# eg what support for ACQ435 do we already have?
acq2206_001> find /mnt/fpga.d -type f -print | grep ACQ2106 | grep 02
/mnt/fpga.d/ACQ2106_TOP_02_02_02_02_02_02_9011.bit.gz
/mnt/fpga.d/ACQ2106_TOP_02_02_02_02_02_02_9011_32B-D37.bit.gz
/mnt/fpga.d/ACQ2106_TOP_02_02_02_02_02_02_9011_32B.bit.gz
/mnt/fpga.d/ACQ2106_TOP_02_02_02_02_02_02_9011_32B_WR-D37.bit.gz
/mnt/fpga.d/ACQ2106_TOP_02_02_02_02_02_02_9011_32B_WR.bit.gz
/mnt/fpga.d/ACQ2106_TOP_02_02_02_02_02_41_61_9011.bit.gz
/mnt/fpga.d/ACQ2106_TOP_02_ff_02_ff_02_ff_9011_32B_WR_UDP-FFC.bit.gz
/mnt/fpga.d/ACQ2106_TOP_02_ff_02_ff_02_ff_9011_WR_32B-FFC.bit.gz
```

```
# and for AO424 and DIO482 ?  ... quite a lot of options, usually with 09 (ACQ423), common in PCS applications.

acq2206_001> find /mnt/fpga.d -type f -print | grep ACQ2106 | grep 41 | grep 6B
/mnt/fpga.d/ACQ2106_TOP_09_09_09_09_41_6B_9011_32B_WR-PG.bit.gz
/mnt/fpga.d/ACQ2106_TOP_09_09_09_09_41_6B_9011_32B_WR.bit.gz
/mnt/fpga.d/ACQ2106_TOP_09_09_09_09_41_6B_9011_32B_WR_UDP.bit.gz
/mnt/fpga.d/ACQ2106_TOP_09_09_41_7B_7B_6B_9011_32B_WR_UDP.bit.gz
/mnt/fpga.d/ACQ2106_TOP_41_41_41_41_61_6B_9011-PWM.bit.gz
/mnt/fpga.d/ACQ2106_TOP_41_41_41_41_61_6B_9011-PWMFAST.bit.gz
/mnt/fpga.d/ACQ2106_TOP_41_41_41_41_61_6B_9011-PWMPROG.bit.gz
/mnt/fpga.d/ACQ2106_TOP_41_41_41_41_61_6B_9011_32B_WR-PWMPROG.bit.gz
/mnt/fpga.d/ACQ2106_TOP_41_41_41_41_61_6B_9011_WR-PG.bit.gz


Let's assume we pick two personalities:
1. ACQ465 in sites 1,2,3,4 AO424 in 5, DIO482 in 6
2. ACQ435 in sites 1,2,3,4 AO424 in 5, DIO482 in 6

The files would be:
/mnt/fpga.d/ACQ2206_TOP_02_02_02_02_41_6B_COMMS_EXTRA
and
/mnt/fpga.d/ACQ2206_TOP_0A_0A_0A_0A_41_6B_COMMS_EXTRA
```