ACQ400 Architecture Guide

Peter Milne
Contact: peter.milne@d-tacq.co.uk

May 29, 2022

Contents

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: ACQ2006, 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
2. excludes FPGA development. • DAQ Appliance: a freestanding networked data acquisition system
3. ZYNQ 7000 (Z7) : Xilinx system on chip architecture.
4. PS : Processor System in ZYNQ7000
5. PL : Programmable Logic (FPGA part) of ZYNQ7000
6. A9 : Dual Core ARM A9 processor found in ZYNQ7000, 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/Motherbord: Electronic circuit board with Z7000 SOC, DRAM, Networking and one or more FMC or ELF payload sites.
14. QPSI Flash: flash memory in the Z7 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. 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 run-time 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 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.