EMBEDDED SOLUTIONS FOR EPICS BASED CONTROL SYSTEMS

M. Dach, G. Marinkovic, Paul Scherrer Institut, 5232 Villigen PSI, Switzerland

Abstract

Embedded systems are the next step in controls and automation evolution. They are powerful enough to compete, in several domains, with conventional control solutions based, for example, on VME technology. This paper describes the generic embedded system solutions for EPICS based control systems. It presents the hardware and software issues when dealing with embedded systems. It shows a concrete example of the embedded system based on FPGA concept which could be used as a generic solution. Finally, usage of the EPICS general purpose driver for memory mapped embedded devices under VxWorks or Linux operating systems, is discussed.

INTRODUCTION

The evolution in computing and automation has a big impact on the control systems architecture and implementation. Most of the control systems, used in large scale experiments and accelerators, refer to the distributed architecture.

The distributed aspect refers to the set of self-dependent computers (or controllers), which are linked together by means of the communication media (i.e. bus or network) used for the data transfer. The computers (controllers) are equipped with software that allow for their inter-communication.

The distributed architecture of the system reveals itself by its decomposition to the functional parts, fulfilling given requirements, which are spread out geographically within a given area. The distributed computers could either be multifunctional (e.g. VME crates) or dedicated to specific task (e.g. network based motion controller).

The multifunctional computers are sufficiently powerful to run several control algorithms in separate tasks. Their functionality could be easily extended by additional hardware modules and software packages. However there is no free lunch and this generic approach sometimes lacks of performance. This is the world of the high performance , application specific controllers. This paper presents the design and implementation issues of such small scale controllers i.e. embedded systems [1].

EMBEDDED SYSTEMS

Typically, an embedded system is a specialized computer system which is built on a single microprocessor board with the program stored in FLASH.

Commercially available embedded systems are designed to perform dedicated functions. Their hardware implementation is based on the highly integrated specialised ASIC chips. The functionality of the embedded system depends on the software and hardware implementation. Software could be relatively easy to modify to extend the functionality of the system. The hardware implementation is however fixed. An alternative to the embedded system with fixed hardware architecture could be the system with the FPGA circuit. In such cases functionality could be extended either by software or hardware. The flexibility of the FPGA based embedded systems makes them highly attractive for new developments.

EMBEDDED SYSTEMS WITH FPGA

There are several FPGA chip vendors. The most popular are ALTERA and XILINX. These offer their chips with FPGA structures together with soft/hard core processors. ALTERA uses soft core Nios and XILINX hard core PowerPC (and soft core MicroBlaze). The PowerPC processor, from the programming point of view, seems to be more attractive since it can operate on one of a number of operating systems, including Linux. At PSI (Paul Scherrer Institute) the XILINX circuit family has been used for many years.

Figure 1 shows a schematic of the general purpose AVNET evaluation board [2] with XILINX Virtex-4 circuit (which contains hard core PowerPC405), FLASH, EEPROM, RAM and peripherals (UART, LAN and three general purpose connectors). Depending on the FPGA configuration, this board could be used for various purposes. The FPGA is configured as a computer system which links the hardcore PowerPC and external components by means of the communication buses. Custom designed (soft core) components could be added to the FPGA in order to extend the functionality of the board. These components can make use of the general purpose connectors in order to interface desired type of hardware.

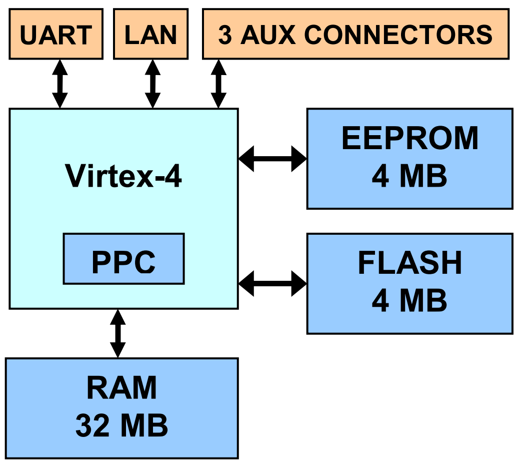


Figure 1: Structure of the Avnet Virtex-4 board.

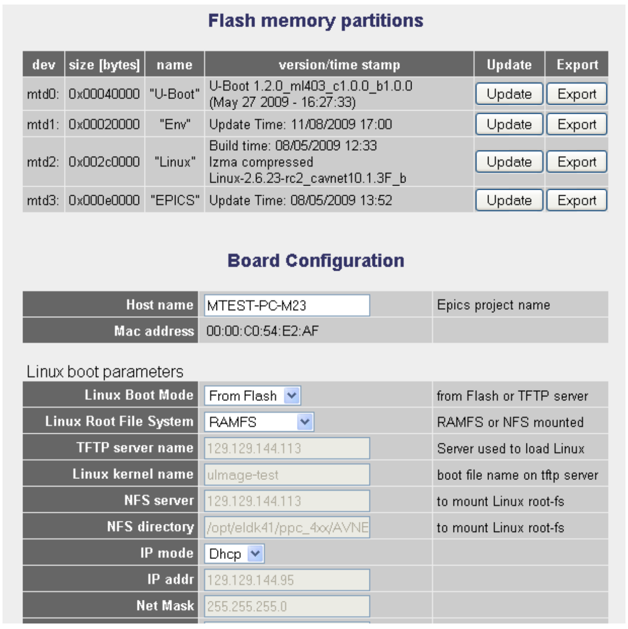
EEPROM memory is used to hold FPGA configuration (bitstream), while FLASH memory is used to preserve data and user programs; user programs are executed in RAM. An initiative was undertaken at PSI to interface the Avnet board to the EPICS control system. The challenge was not only to build EPICS itself but the whole software "back bone" which includes the boot loader, Linux OS and the RootFS (see Appendix).

Finally the software was built and customised to fit onto 4MB of FLASH. It was possible to achieve this through LZMA [3] compression which is 35% more effective than gzip and 15% better then bzip2. FLASH is divided into the following 4 partitions:

• U-boot boot loader	256 KB
• U-boot configuration	128 KB
• Linux OS + RootFS compressed with LZMA	2.62MB
• EPICS DB	1 MB

SYSTEM OPERATION

When the board is powered, the FPGA is configured and the PowerPC processor starts to operate. Linux OS is booted by means of u-boot [4]. The u-boot takes into account configuration parameters and loads Linux either from Flash memory or from the tftp server. The u-boot can pass several parameters to Linux, such as the rootfs location (i.e. initrams or nfs mounted), FLASH partitions configuration and IP addresses etc. All u-boot configuration parameters can be modified through a web browser. An example of a configuration web page is shown in Fig. 2.



Flash memory partitions

dev	size [bytes]	name	version/time stamp	Update	Export
mtd0:	0x00040000	"U-Boot"	U-Boot 1.2.0_ml403_c1.0.0_b1.0.0 (May 27 2009 - 16:27:33)	Update	Export
mtd1:	0x00020000	"Env"	Update Time: 11/08/2009 17:00	Update	Export
mtd2:	0x002c0000	"Linux"	Build time: 08/05/2009 12:33 lzma compressed Linux-2.6.23-rc2_cavnet10.1.3F_b	Update	Export
mtd3:	0x000e0000	"EPICS"	Update Time: 08/05/2009 13:52	Update	Export

Board Configuration

Host name	MTEST-PC-M23	Epics project name
Mac address	00:00:C0:54:E2:AF	

Linux boot parameters

Linux Boot Mode	From Flash	from Flash or TFTP server
Linux Root File System	RAMFS	RAMFS or NFS mounted
TFTP server name	129.129.144.113	Server used to load Linux
Linux kernel name	uImage-test	boot file name on tftp server
NFS server	129.129.144.113	to mount Linux root-fs
NFS directory	/opt/eldk41/ppc_4xx/AVNE	to mount Linux root-fs
IP mode	Dhcp	
IP addr	129.129.144.95	
Net Mask	255.255.255.0	

Figure 2: Configuration web page.

The EPICS server runs on top of Linux. It can access hardware components by means of the GPMM (General Purpose Memory Mapped) driver [5]. The GPMM driver is generic in a sense that it can be setup for any type of board. Taking the case where the board contains an ADC component which is visible as a set of 3 registers in memory as shown in Fig. 3. Here, there are two GPMM functions which setup EPICS to access the registers in the memory: GPMMConfigure and addGPMMRegister.

0xB000000	Status and Control register
0xB000004	ADC offset register
0xB000008	ADC FIFO

Figure 3: ADC memory representation.

These functions are used in the startup script before IocInit. The GPMMConfigure function is used to register the card or hardware component, in the system, under a given base address.

Example:
GPMMConfigure(0,0xB0000000,"ADC","A32","D32")

where:

0	: card (component) number
0xB0000000	: card base address
ADC	: card name
A32	: address access mode
D32	: data access mode

The addGPMMRegister function is used to associate a user defined name with the given register.

Example:
addGPMMRegister(0,0,0x8,4,"X","ADC-FIFO",0)

where:

0	: card (component) number
0	: not used
0x8	: address offset with respect to the base
ADC-FIFO	: register name
0	: user defined function (0-> no function)

The GPMM driver was written originally to support VME cards under the VxWorks kernel and was later extended for Linux OS. In order to keep the EPICS database portable, GPMM uses the same syntax for VxWorks and Linux.

The example below shows how to associate the EPICS analog input record to the ADC-FIFO register:

```
record(ai,"TEST:ADC-readout"){
        field(DESC,"ADC FIFO register")
        field(DTYP,"GPMM")
        field(INP,"#C0 S0 @ADC-FIFO")}
```

where:

C0	: card (component) number
S0	: sift the readout by number of bytes
@ADC-FIFO	: register name to be read out

REAL TIME CONSTRAINS

Some nodes of a distributed control system are expected to react on time for external events. Such real time constrains should be taken into account when designing the distributed control system. Embedded systems with FPGA chips are well suited to fulfill hard real time requirements because the FPGA operation is inherently concurrent. Therefore, time dependent parts of the system may be implemented in the FPGA structure with the remainder implemented by software under Linux.

Linux as a general purpose operating system can, nevertheless, be tuned to fulfill soft real time constrains. Two approaches are: preemption improvement and interrupt abstraction [6].

Preemption Improvement

In general, the strategy is to reduce the length of the longest section of non preemptible code in order to minimize the latency of interrupts or real-time task scheduling in the system. The Linux kernel 2.6 onwards can be configured as a preemptible kernel.

Interrupt Abstraction Concept

The main strategy is to make use of the RTHAL (Real Time Hardware Abstraction Layer). The RTHAL intercepts all hardware interrupts and routes them to either the standard linux kernel or to the real-time task. This approach provides a way to preserve the Linux kernel "as is" with the additional RT-linux module (see Fig. 4) in the RTHAL layer.

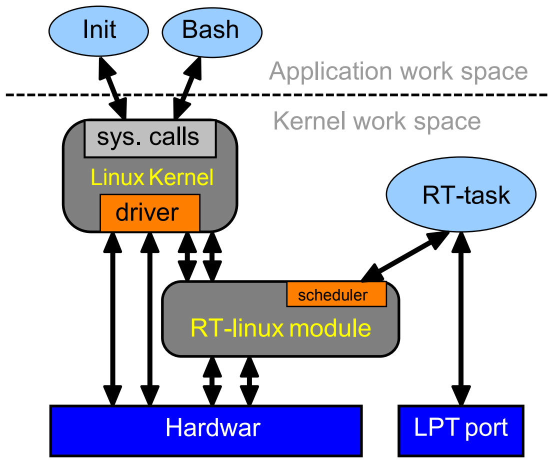


Figure 4: Linux RTAI system layout.

Interrupts which are meant for a scheduled real-time task are sent directly to that task, while those interrupts which are not required by any scheduled real-time task are sent directly to the standard Linux kernel where they are handled according to normal needs. One of the most popular implementation of the RTHAL concept is the RTAI (Real Time Application Interface) [7]. The RTAI performance is very competitive with the best commercial Real Time Operating Systems (such as VxWorks or QNX), offering typical context switch times of 4 usec, 20 usec interrupt response, 100 KHz periodic tasks. Test results obtained in SLS/PSI confirm RTAI performance. Linux with RTAI extension fulfils much better real time requirements than Linux OS with the preemption improvement concept.

CONCLUSION

Embedded systems with FPGA circuits are very powerful solutions which could be easily customized to perform given tasks and extendable for future requirements. Some modern FPGAs contain hard core processors. They can be configured as computer systems and run fully featured operating systems like Linux OS. Embedded systems with FPGAs are well suited to fulfill real-time constrains. The real time aspect of the system could be achieved both on the hardware level and also software wise. A generic board comprising a Xilinx FPGA (Virtex-4 with built-in PowerPC) interfaced to the EPICS control system, has been integrated at PSI. New electronics is currently being developed making use of this new approach.

APPENDIX

Generic board (used in PSI) specification:
Hardware:

- Xilinx Virtex-4 FX12 (with built-in PPC405)
- RAM 32 MB
- FLASH 4 MB
- EEPROM 4 MB
- UART, LAN, 3 general purpose sockets

Software:

- U-boot 1.2.0
- Linux 2.6.23
- Busybox 1.4.0 rootfs + goahead webserver
- EPICS 3.14.8.2
    - GPMM driver with interrupt sup.
    - Asyn device driver
    - Stream device driver

REFERENCES

[1] Embedded systems: http://en.wikipedia.org/wiki/Embedded_system
[2] AVNET evaluation board: http://www.em.avnet.com/ctf_shared/evk/df2df2usa/Xilinx_Virtex-4_FX12_Evaluation_Kit_-_Product_Brief.pdf
[3] LZMA compression: http://www.7-zip.org
[4] Das U-boot bootloader: http://www.denx.de/wiki/U-Boot
[5] GPMM driver: http://epics.web.psi.ch/software/GPMM
[6] RTLinux versus RTAI: http://www.linuxfordevices.com/files/misc/ripoll-rtl-v-rtai.html
[7] DIAPM RTAI Programming Guide: http://www.rtai.org