I/O CARDS WITH BUILT-IN LINUX AND TCP/IP BECOME TURNKEY SYSTEMS

D.Golob, U.Platise, M.Plesko, M.Sekoranja, Cosylab and J. Stefan Institute, Ljubljana, Slovenia

Abstract

We have developed a family of I/O cards with onboard Linux and Ethernet, which accept Industry Pack modules. Combining those cards into a small casing and adding control software packages such as EPICS or ACS, provides for compact software driven turnkey systems. It has always been our goal to reduce the number of hardware and software interfaces and the flavour of different fieldbusses to a minimum. With the advent of low priced mass produced embedded controllers with high CPU power and an array of standard interfaces such as Ethernet, USB, RS-232, etc., it became possible to get rid of the fieldbus completely, even in highly distributed applications, bringing Ethernet and TCP/IP down to each single I/O card. We have opted for a single board computer (SBC) with a StrongARM CPU and Ethernet interface, for which several low footprint solutions exist on the market. The SBC is positioned as a mezzanine module on our I/O cards. The CPU power and the available RAM allow for normal operating systems and control programs to run. We have customized a standard Linux port for the StrongARM to our SBC. This allowed us to port the CORBA-based Advanced Control System ACS onto the I/O card. An EPICS port is under way, OPC is equally possible. This allows to integrate the I/O card seamlessly into any control system, without the necessitiy for any extra drivers. As both Linux and Ethernet are indeterministic, we need another realtime component to handle timing and I/O synchronisation. This is done with a CPLD, which is built into each of our I/O cards. The CPLD can be programmed from Linux via JTAG from the SBC for all realtime tasks that are not too computing intensive. It serves as an on-board timing module, function generator and general digital interface. It also drives the Industry Pack (IP) standard interface, which allows all commercial standard IP modules to be attached to a slightly modified I/O card of ours. First applications of this approach are a turnkey timing system and a vacuum pump current measuring unit.

INTRODUCTION

Network attached devices (NAD) are here. There are many good reasons why, which we will not go into, but refer to other papers in this session of the ICALEPCS[1]. The main advantage as we see it is that by using NADs we can completely avoid any fieldbus solutions. No matter which fieldbus one chooses, it is far less standardized than Ethernet and, even more important, the TCP/IP protocol. Using a fieldbus usually results in some proprietary protocol, while TCP/IP is of course supported everywhere.

Our goal therefore is twofold:

- To build sufficiently low-cost I/O cards that would have Ethernet on-board.
- To provide a sufficiently strong processor that would allow to run a standard operating system (in our case Linux) that supports not only the pure TCP/IP stack but also allows conventional distributed processes to run directly at the I/O level

In short, our mission is to “Bring Ethernet and TCP/IP to the Masses”. The real usability of such an approach is that the interface from the I/O board to the control system becomes the API of the control system itself – nothing new or proprietary is added.

The development dilemma that we have faced was how to develop what we want but still use as much commercial of the shelf (COTS) components as possible. We discuss the options and our decisions in the following sections.

ASSEMBLE YOUR OWN TURNKEY SYSTEM

The embedded market offers less standards than the VME or cPCI world. There are cheap CPU/Ethernet modules in abundance, but there is no global standard that would allow to interchange those boards. Typically, they are considered for turnkey solutions where the hardware never changes after installation. Due to lack of standards there are of course many ingenious solutions: some cards come in the PC-Card format, others as DIMM modules. Unfortunately, none are pin-compatible, unless one resorts to PC104, which costs typically twice as much.

Other commercially available items, albeit not turn-key, are a plethora of I/O driver chips and Industry Pack (IP) modules that one can get for nearly any type of I/O. Industry Pack modules are used mainly in VME systems, on non-intelligent carrier boards. However, their protocol is very simple and can be therefore easily implemented on a logic chip. IP modules are also reasonably cheap, since they come from competing manufacturers.

It is therefore possible to just combine the concept of low-cost Single Board Controllers (SBC) with the I/O capabilities of I/O drivers or, for more complex requirements, IP modules, by developing a simple board with just some glue logic. To test the concept, we have built some dedicated I/O cards that accept an SBC as described in the following subsection.

The Single Board Computer CEP

We have decided to use the SBC called CEP, which stands for Custom Embedded Platform. Apart from the fact that the company that produces it is only 40 km away from us, it also has the lowest price on the market that we could find.

The CEP has the following characteristics:

- StrongARM, up to 32 MB Flash and 64 MB RAM (In preparation are modules with the next generation CPUs PXA 210 and PXA 250)
- RISC core, DSP specific instruction set
- High performance: 235 Dhrystone 2.1 MIPS @ 206 MHz
- LCD Driver Interface
- Fast Ethernet IEEE 802.3/802.3u 100 Base-Tx 10 Base-T
- Audio Codec Interface
- PCMCIA Interface
- Serial Interface RS232/RS485
- USB SlaveSingle 3.3V Power, Low power (normal mode)
- —<240 mW @1.55 V/133 MHz
- —<400 mW @1.75 V/206 MHz
- High Integration: 54.0 x 85.6 mm, PCMCIA board format including optional extension connectors

Our Carrier Board

So we bought the CEP and all we had to develop was a carrier board that accepts the CEP as a mezzanine module, provides an Ethernet transceiver, some UARTS for RS232 and a logic chip with glue logic. We have decided to use a CPLD (complex programmable logic device), which appears better suited for such tasks than a FPGA – but the concept is exactly the same. The format of the card is 160mm x 100mm, the Europe standard

The first prototype had several digital communication I/Os just to show the principle:

- 7 x RS-232 Ports
- 8 x RS-485 Ports
- 9 x General Purpose Optically isolated I/Os
- 5 Mbit Fibre Transmitter
- 5 Mbit Fibre Receiver
- 12 MBit USB Slave Port
- 10/100 Mbit Ethernet Port
- I2C Interface
- Cypress 1-Wire interface
- CAN bus interface

We call it Muserio, which stands for MUltiple SERial I/O.

Eight RS-485 ports, six RS-232 ports and the fibre optics receiver and transmitter are attached to the Cypress high-density CPLD and may be configured as required by an application with the VHDL code. The seventh RS-232 port is directly connected to the Intel StrongARM processor. It is used as a terminal port for configuration purposes. Besides that the Muserio provides nine general purpose current limited I/O that can be used to build up a complete timing system as described in a further section.

The I2C, 1-Wire and CAN protocols must also be implemented in the Cypress CPLD.

Linux and TCP/IP

It was very easy to port Linux onto the processor. Currently, we have a version with “normal” Linux and on top of it ACS, our CORBA-based control system. We are preparing a version with EPICS for the Swiss Light Source.

The normal CEP StrongARM board has 64 MByte of memory, which is more than many VME controllers have. So that should be definitely enough for all our needs.

In fact, after some optimizations we need only 2 MB for Linux. Then we need for a free version of CORBA, which comprises ACE (1.8 MB) and TAO (3.3 MB) a total of 5.1 MB. This means that even on the cheapest CEP StrongARM board, which has 16 MBytes, we can run our complete control system and still have plenty of memory left for process data.

DEVELOP ONLY WHAT IS NECESSARY

Once our system was up and running after very little development time, we wanted to develop several dedicated carrier boards. We first developed a board that controls 4 motor axes, completely with encoders and end switches. However, while the hardware was easy, the programming of all necessary motor functions in VHDL on the CPLD turned out to be far too exhaustive for our small team. Originally, we also wanted to develop an analog board with ADCs and DACs, but each customer has his own requirements. That means that we would have to develop and maintain a big number of carrier boards.

Therefore we decided to develop only an Industry Pack (IP) carrier card, and leave it to the user/customer to put there any IP module she likes. There are many IPs with ADCs and DACs of any type (low or high resolution, fast or slow, etc.), and also for motor control, thus making our motor controller obsolete in a flash.

The standard for IP modules suggests that there are 4 IPs on a 6U card (a standard VME card) and 2 IPs on a 3U card (Europe format). This assumes that all IPs are only on one side of the card, because of the standard widths of 1 inch. As we have also the StrongARM card, we lose one slot. We could maybe squeeze one more IP, but there would be already problems with cabling and connectors: each IP module has 50 pins and a normal three row Europe card connector has up to 96 pins.

So our standard Europe IP carrier card has only one IP slot. We can of course develop a custom card that is larger or takes more lateral space than a standard Europe card, but that would be expensive. The problem is that we as a Company can not offer to many options, otherwise the development costs are too high. We can not expect high sales numbers, therefore we should make few simple modules that can be mixed, but not too many independent solutions. This is actually the main reason, why we go for IP modules, because we get flexibility, but not at the cost of extra development.

Should people need more IPs controlled by a single SBC, then we propose a different solution that we plan to provide in the future: put for example 3 carrier cards together in one box. Only one carrier has a StrongARM, the others have only IP modules. They are connected together via a simple local bus. Then the StrongARM

controls all three IP modules. We can stack up to eight carrier boards to one StrongARM.

TURNKEY : MAKE IT AN IOC

Certainly, there are cheaper solutions than our on the market. After all, our processor is not the cheapest. Other microprocessors, such as the Motorola ColdFire, are cheaper and come with a TCP/IP stack. However, we go for a turnkey system, such that our board becomes an IOC (input/output controller, the basic node of a control system) runs the control system core, such as ACS or EPICS and other high level applications. The beauty of such an approach is that the interface is really the control system, not any electric interface or proprietary API. And everything suddenly falls into place:

Form factor, mechanics

The carrier card is exactly Europe format. One can use it either standalone in a neat box, a DIN rail mount, or put several of them into an Europe crate. This is also the reason, why we can not put 3 IP modules on one carrier, because then we would have problems with connectors in an Europe crate.

Power Supply

Our boards accept an external supply of 24V or 12V. There is also a new standard, which is called “Power over Ethernet” (PoE). According to this standard, the electric supply comes directly over normal UTP Ethernet cables. One can buy a switch which provides Ethernet + 48V over UTP cables. The advantage of such setup is that there is no need for local DC supplies. Our boards support this new standard.

Direct AC 110/220V supply is also possible. We can add a commercial AC/DC converter unit onto the card. However, it will add up to the cost of the box and also make it a little larger, because of the needed space. It is usually more cost-effective to have one AC/DC converter for several boxes and then distributed the DC low voltage - either over PoE or locally in a crate.

Performance Tests

A simple scope simulation showed the following times:

4ADC*2bytes*3000values from input over network:
0.3 ms (ADCs)
3 ms (CPLD to Linux) (corresponds to 6 MByte/s)
0.032 ms (Interrupt latency & driver execution)
4 ms (conversion Int-> Double and averaging)
8-20 ms (UDP network transmission)
==========
15 - 27 ms (Total)
Which corresponds to a streaming-video view.

Alternatively, should one need to acquire and store a burst of 30 seconds of data, as in a storage oscilloscope, we get: 100 kHz * 4 * 2bytes * 30s = 24 MByte which we can just dump into RAM for later transfer.

Linux Or Real-time - Not Our Dilemma

We don’t see any problems on porting RT Linux instead of normal Linux or RTEMS, which is also a nice RT OS for control systems. However, there is another, easier way to provide a few simple real-time tasks that are much faster than an OS can ever be.

The CPLD can do much more than glue logic. We use the Cypress CPLD Quantum/Delta 38000/39000 Families, which have several pin-compatible chips. We can take the more powerful one and program RT-tasks in VHDL. For this we have developed a way to download code from Linux via JTAG.

This makes the RT tasks safe against Linux reboots and Linux takes care only of communication and of non-RT applications. A result of this concept is described in the following subsection:

A Turnkey Timing Box

We are currently designing a stand-alone timing system, where each Muserio is a Timing System Unit (TSU) that functions simultaneously as an event generator and event receiver. All is implemented in VHDL on the CPLD. The following features are planned:

- Fibre Optics/Copper Point-to-Point Input/Output.
- Multi-Point connection is possible on copper lines.
- The top-most TSU is synchronized with high-precision reference to 50 Hz.
- Each TSU can synchronize to input and regenerate or generate events on output channels (programmable delay)
- Multiple Events per Transmission Line
- Precision Triggering with Transmission Delay Compensation
- Event confirmation/acknowledgment support for high-reliability systems.
- CPU Interface for Event/Data Packet Exchange, TSU configuration, Event Generation/Spawning, and more

CONCLUSIONS

Despite the nice idea, the main question that remains is: “Is It Really Cost-Effective?”. The price per channel is basically determined by the price of I/O and the price of the crate and supply. If a VME crate is full, the price of the crate and supply become negligible and VME is as cost-effective as our turnkey solution or better.

So the potential of our approach is for cases, where I/O is highly distributed and few channels are located together. It may be a niche, but it is definitely there.

REFERENCES

[1] See for example L.R.Doolittle, Embedded Networked Front Ends – Beyond the Crate, this conference.