

Emittance Device Specifications:

Sasha (Diagnostics AP Requirements, 10/20/2000):

Emittance accuracy +/- 10%, pulse length 0.3 to 10 μ s.

John Staples (FE-PH-039):

Beam Parameters

Beam energy	2.5	MeV
Beam current	52	mA
Pulse length	100	μ s
Number of pulses	50	
Beam σ_x	0.5	cm
Beam σ_y	0.3	cm
Peak power density	500	kW/cm ²

Preslit and Precision Slit

Slit width	0.02	cm
Slit width variation	<5	%
Slit bevel	30	mrad
Slit length	4.3	cm, for 45° insertion angle
Slit travel	4.3	cm, each slit across aperture
Energy Deposited	650	Joules in one scan
Maximum speed	3	cm/sec
Minimum step size	0.010	cm
Minimum actuator step size	0.014	cm, for 45° insertion angle
Number of positions	50	during one scan

Collector Array

Array height	3	cm
Array width	3	cm
Number of elements	64	
Element c-c spacing	0.47	mm
Max energy deposited	25	Joules in one scan
Peak current per element	1	mA
Bias capability	300	V
Input referred noise	<10	μ A

Emittance Device Status:

Problems moving from conceptual to preliminary phase. Instantaneous temperature rise on the slit is a limiting factor, it's clear we can't meet the original goal of $100 \mu\text{S}$, full current. We assert (but have not proven) that there is a pulse size that is both short enough to not destroy the slits, and long enough to give meaningful information about the beam.

Component design:

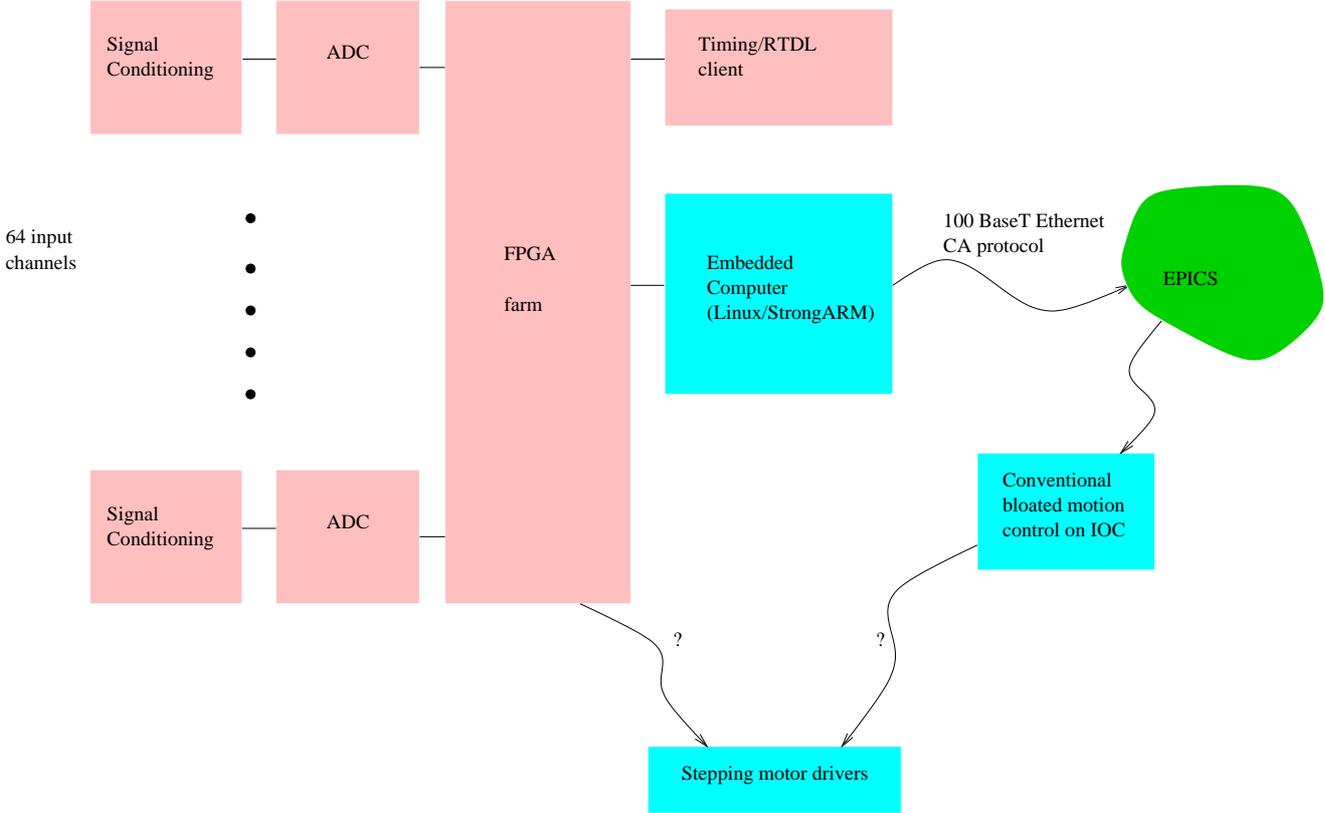
- 64 channel front end and acquisition electronics design is at the schematic and board layout phase.
- Beam box and actuator space allocation and topology drawings made

Schedule:

Dec-Jan	01:	Physics and Engineering (Quantify and verify assertion that enough information will be measured to justify the effort)
Jan-Feb	01:	Mechanical Design
Mar	01:	Preliminary Design Review

We are uncertain what the conclusions of the PDR will be. Certainly it is difficult to imagine a well engineered device, meeting space constraints, installed and operational during the commissioning at LBNL in Feb-Apr, 2002, especially with the existing manpower.

MEBT Emittance Scanner Electronics



Embedded computer performance tests:

Bright Star Engineering nanoEngine running Linux.

Hardware:

- 200 MHz StrongARM SA-1110
- 32 M RAM
- 4 M Flash
- 100 BaseT Ethernet

Development kit (GNU):

- binutils-2.9.5.0.22
- gcc-2.95.2
- glibc-2.1.2
- Linux-2.4.0-test10-rmk1-bse
- BusyBox-0.48
- net-tools-1.57

Questions:

- 1. Does it work?*
- 2. How fast is it?*
- 3. Is it software developer-friendly?*
- 4. Is it hardware developer-friendly?*
- 5. Is it network friendly?*

1. *Does it work?*

Of course!

Example PCAS cross-built and serves channels

External trigger pulse can trigger PCAS to update “record”
(kernel driver needed, mostly copied from existing samples)

2. How fast is it?

Time delay from rising logic level on a digital I/O pin to second level kernel interrupt routine: 6 μ s (first level only demultiplexes 17 possible GPIO pins)

User level program stalled on read-wait: 29 μ s

Simple user level program using select() or poll(): 62 μ s

Entry to EPICS PCAS callback routine, after decoding select() response: 100 μ s

Data transfer from the off-board address space, if performed from user level with caching turned off, can happen at about 7 M transfers/second.

Same transfer within PCAS, (badly) converting 16-bit ints to float, 2 M words/second.

1000 point array updated by external trigger at 60 Hz runs pretty well, with only 20% of CPU time accounted. Something in the EPICS network stack stalls occasionally, resulting in dropped updates. The problem could be anywhere in PCAS, its interaction with Linux, the Linux network stack, network hardware, and the client readout.

3. Is it software developer-friendly?

Absolutely. The same toolkit that is used for conventional EPICS development applies here, although in a different configuration. Your development directory can be NFS mounted by the embedded device, you can telnet in, version control “just works.” There are license managers to get in the way. Supported languages by the Gnu Compiler Collection include c, c++, Fortan, and Java (I have not tested the latter two on this platform). Normal Unix-style development “just works.” gdb is not tested, strace and gprof worked on their first try. Anything that can be compiled by gcc should build easily. Python anyone? All code can be tested for functionality using standard GNU tools locally on your favorite workstation (Linux, Solaris, NT, ...). Since this is a true, MMU enabled Unix-alike system, it shows traditional rock solid stability. This contrasts with the VxWorks or uCLinux style flat memory model, that has a fragile response to misbehaved application code.

4. Is it hardware developer-friendly?

Absolutely. With one 160-pin Molex connector, you get the complete embedded computer, ready for 3.3 V power (2 Watts max.), and four wires to an RJ-45 Jack for the Ethernet. Ten GPIO pins make for easy JTAG configuration of FPGAs. External bus options are multiplexed (32-bit), non-multiplexed (16-bit), and PCI.

5. Is it network friendly?

Absolutely. The system will autoboot from Flash, ending with a fully running Linux system in 8 seconds. The only network traffic to get that far is one exchange to set the system time (it has no on-board CMOS clock). At that point you can log in remotely, or have scripts run that mount NFS disks and run programs either locally or over the net. The PCAS fits on Flash, so the full network functionality can be brought up that way. Like VxWorks, its fail-safe boot capability supports the serial console only. For testing, or to recover from corrupted Flash, an RS-232 link is needed.

Embedded computer conclusions:

For US\$700, 2 Watts, and 1 cubic inch, an instrument designer gets an embedded computer that is in many ways superior to what was available in an EPICS VME computer of five years ago.

The hardware and software infrastructure to do application development is in place now.

LBNL will use this device to support the emittance scanner. A PCAS driver for that gear will be written in parallel with the hardware development.

MEBT Emittance Scanner Electronics

