• Instrumentation Hardware and Measurements
• Instrumentation Embedded Software/Firmware
• Communications/Data Translation/Low-level Control
• Calibration and Constants
• System Control Tools
• User Interface/Control System Interface
• Data Handling and Analysis
• System Visualization Tools
• Infrastructure for Building/Maintaining Codes
Comments

- **Timescale**
  - Maximize operational capability by next CLEO-c run
  - 5-6 month results are *critical*
  - Flexibility for use until end of CESR (mid-2008 for HEP, 2012? for CHESS) is *important*

- **Manpower**
  - Draw on experience base for design and supervision
    - Specifications
    - Templates
    - Examples
  - CLEO offering student/postdoc support to gain hardware experience (*and luminosity*)
  - CESR staff support

- **Goals**
  - Provide code for *all standard* operating scenarios
  - Provide DAQ *skeleton* that can support important features
  - Implement features that are rated *critical*
  - Allow for modular construction so that other features can be added when/if they become necessary (*enable students/staff to implement specific applications*)
  - *The most critical feature is a detailed plan capable of supporting short-term implementation and long-term operation/evolution of systems*
Key Feature: SERIAL-XBUS protocol provides 10K word/sec of transmission bandwidth. All major processing must occur in DSPs, particularly when servicing many BPMs.
Hardware/Measurements

• Beam Position Monitors
  – Continuous beam position
    • Orbit monitoring
    • Differential orbit monitoring
  – Orbit requests
  – Phase requests
  – Turn-by-turn trajectory requests
  – Bunch tunes (continuous or by request)
  – Calibration
    • Pedestals
    • Bunch Timing
    • Turn Timing Logic
  – Real-time tune monitor

• Beam Profile Monitors
  – Real-time Differential Vertical Size
    • Interferometer analysis
    • Gaussian profile analysis
  – Calibration
    • Pedestal
    • Bunch Timing
    • Channel cross-talk
    • Turn Timing Logic
  – Integrate into beam dynamics measurements

• Control System Integration/Machine Studies Measurements
  – Grad student projects
  – Accelerator group support

• Who?
  – Dobbins (FEs, timing, test, support hdw)
  – Strohman (dig, test, PS, timing dist)
  – Palmer (requirements/DSP soft/CTL soft)
  – Codner (CTL applications)
  – Tanke (testing/DSP software)
  – Watkins (interferometer algos)
  – Taylor (xray profile monitor algos)
Embedded Software/Firmware

• **General Requirements**
  - Test suite for each DSP
  - Support for standard setup/command/trace/debug operations
  - Calibration and analysis algorithms by detector type
  - Support routines by DSP type
  - Firmware for new FEs with algos running in PLDs
  - General library structure for stability and reusability of code
  - Coding standard to support shared code between different architectures
  - Source code maintained as part of CESR CVS repository on Unix

• **Status**
  - Already have a fairly complete suite of code for 1st generation CBPM modules
  - Preliminary test suite for 2nd generation modules in use and being expanded
  - First algorithms for 2nd generation modules under development (installation of FLM module soon)
  - Visual DSP++ supports makefiles and code libraries. Working on switchover from project by project builds

• **Who**
  - Palmer/Tanke for support codes
  - Palmer/Watkins/Taylor for applications
Communications/Low-Level Control

• General Requirements
  – Standard control/status/debugging/trace interface to ALL module types
  – Mirroring of DSP data structures for debugging and data transfer
  – Automated tools to set up definitions of mirrored data structures and the communications protocol for each

```c
typedef struct {
  int cmd;
  int cmd_status;
  int error[MX_ERROR_WORDS];
  int handshake; // handshake MUST BE LAST!
} CBPM_XBUS_CMD;
```

// CBPM_XBUS_CMD configuration
static const COMM_KEY_CONFIG xbus_cmd_cfg = {
  "XBUS_CMD", // name
  XBUS_CMD_TAG, // ctl_tag
  XBUS_CMD_TAG, // dsp_tag
  GENERIC_EXE, // exe_allowed
  1, // num_pkts
  4, // num_vars
  CTL_READ_WRITE, // protection
  FIXED_REC_LENGTH, // rec length flag
  {T_INT, T_INT, T_INT, T_INT}, // data types
  {1, 1, MX_ERROR_WORDS, 1}, // data counts
  {1, 1, 1, 1}, // element sizes
  {cbpm_int_convert, cbpm_int_convert,
   cbpm_int_convert, cbpm_int_convert}, // conv function ptrs
  NULL, // custom copy/check data function
  cbpm_struct_io // IO function
};

• Status
  – DSP communications interface developed for 1st generation CBPM system
    • Transfer C data structures as XBUS packets (image DSP memory)
    • Control/handshaking methods
    • Debug/Trace methods
    • Input parameter validation methods
    • Communications protocol
      – Data translation
      – Data mapping

  – Parser tool
    • Automated setup of matched DSP/Control headers and generation of communications initialization code

• Who
  – Strohman/Palmer
  – Taylor (Isaacman)
Calibration and Constants

• General Requirements
  – Need to support a large range of constants for each module
    • Gain calibration
    • Pedestal calibration
    • Timing calibration
    • Turn logic info by bunch
    • Basic detector information
    • MPM node/element mapping
  – 1st generation system saves approx. 1000 pieces of information per module
  – Visualization, history tools, and searching would be nice

• Status
  – 1st generation implementation
    • Tagged configuration file saved upon request when calibrations run
    • Files manually checked and rotated into default constants file when needed

• Who
  – Palmer
### System Control Tools

#### General Requirements
- Ability to operate multiple hardware and software combinations
  - BPM system will have 2-3 hardware types and 2-3 (or more) DSP software versions
  - Synchronous measurements between system
- Uniform control interface

#### Status
- Present control loop can handle single hardware/software type
  - Readily expandable
- Communication and low-level control tools already have been developed to support this (in testing phase)

#### Who
- Palmer, Taylor
User/Control System Interaction

**General Requirements**
- Individual systems available for requests for standard event data via server
- Provide (some) streaming data for direct control system monitoring (also see following slide)
- Provide interface for users to run customized data requests (also see following slide for data issues)
  - Need to specify interaction with server
  - Custom requests may require significant re-initialization when control returned to server
- Monitoring/Config/Control GUI would expedite maintenance and user interactions
- Standard user input tools to avoid “crashing” system

**Status**
- Standard CBPM server available for orbit, phase, turn-by-turn trajectory measurements
- Does not offer full configuration capability
- Needs better implementation of interrupts and failure recovery
- User interface through command line menu. Can disable server for long periods (probably need to add interrupt capability)
- Standardized user input methods with parameter checking to protect low-level processors

**Who**
- Palmer (Rock)
Data Handling/Analysis

- **General Requirements**
  - Support single layer of thinned out data to MPM
  - Allow access to detailed data at requesting program level
    - Shared memory on server node
    - User analysis directly in requesting program
  - Allow for new communications path (ethernet) for large bandwidth limited data
  - Support event-like and streaming data
  - Interface routines for Fortran analysis

- **Status**
  - Present methods
    - Buttons/phase data written to MPM for standard requests
    - Trajectory data written to file for user read/analysis/display
    - Differential orbit data written to file for user read/analysis/display
    - Calibration data written to file for user read/analysis/display

- **Who**
  - Palmer
System Visualization Tools

• General Requirements
  – Built-in package for visualization has been requested
    • Large data size
    • Range of data to examine
    • Desire for standard, easily accessible tools

• Status
  – At present, no built-in visualization tools provided
  – Accelerator group tools for plotting MPM data
  – User tools for plotting saved data

• Who
  – Codner
**Infrastructure**

**Code Building/Maintenance**

- **General Requirements**
  - Code maintained in CESR CVS repository on Unix
  - OSF testing of many tools (on CESR machines) is acceptable
    - Compilers on cesrxx OSF machines match VMS compilers
    - More development CPU cycles available
    - Porting tools to move code to VMS and test
  - Multiple source library implementation to support shared source among several applications
    - Control codes
    - DSP codes
    - Control/DSP codes
  - Suitable coding standard to handle code that can be run across multi-DSP and CTL architectures
  - Restrict development to tools available (or portable) to VMS
  - Versioned releases for stability and problem tracking

- **Status**
  - Repository structure in place
  - Porting structure in place
  - CesrBPM and utility libraries (1st generation system) in place with versioning checks
  - Range of new codes ready for import into repository

- **Who**
  - CESR Librarian: Valeriu Smiricinschi
  - Palmer
Library Hierarchy

- General Utilities
- DSP Parser
- DSP Test/Utilities
- Comm/Control

- CesrBPM II
- CesrBSM
- CesrFLM
- CesrXBSM

- Server(s)