GMT in the Global Trigger Crate

1 GMT Logic Board

FPGA firmware ready

3 Pipeline Sync. Boards

6-channel prototype available

Global Trigger Crate (top view)

4 DT/CSC + 8 RPC muons

Special wide input board parallel to front panel
GMT Logic Board Status

PCB received last week

M. Padra
Overview

- Just received GMT Logic Board
  - Board will be assembled over next few weeks
  - No specific online software written for this board, yet
  - Will develop online software over the next weeks and months while testing the GMT Board
  - Use the same framework (e.g. VME64 board configuration, ...) as for GT

- This presentation …
  - Plans / Issues for GMT Online Software
  - JTAG Access Library
    - C++ library for accessing JTAG chains for operations such as upgrading firmware
    - Common component used by GMT, GT, DTTF, …
**GMT Configuration Concept**

1) Edit and register configuration (before run)
2) Tag a configuration (e.g. RUN1567) (before run)
3) Configure with tag RUN1567 Just before starting run
4) Retrieve configuration RUN1567 & configure hardware

**Online SW on PC in rack (XDAQ)**
- C++ classes for Chips, Boards
- can be used standalone (testing, …)
- can be used via SOAP interface (e.g. by Run Control or by private interface)

**Configuration Editor GUI**

**Configuration Selection GUI**

**Run Control**

**Trigger Supervisor**

**Config DB**

**Rack PC**

**XDAQ**

**VME**

**Crate**
Thoughts on GMT Configuration Database

- Have to decide whether it is possible to update the configuration during a run
- If configuration update is **not** possible ...
  - Split GMT configuration in several parts (tables)
  - Each record in a table has a revision number
  - A tag collects all the revision numbers for a certain run
- If configuration update is possible …
  - A) create new tag whenever the configuration changes
  - B) provide a mechanism to **update the tag**
    - Could split configuration into
      - Parts, that cannot change during run (main part)
      - Parts that may change (mask bits, thresholds in GT, …)
    - Need extra table to store Intervals of Validity for the parts that may change

- CMS DB working group is working on proposal for Intervals of Validity Service
  - Will follow these developments, then design GMT configuration DB
GMT Interface to Trigger Supervisor

- **Commands needed**
  - **Lock**: obtain exclusive write access to GMT
    - Other clients may read, only
  - **Configure** with certain TAG (from Configuration DB)
  - **Update** sub-set of configuration
    - Done during run
    - Have to define subset of parameters that may be changed during run
    - This command has to register the event/orbit number at which the configuration was changed in the database
  - **Self-Test**
  - **Unlock**
  - (Enable / Disable)
    - Not really needed as this is done via TCS
JTAG Access Library (JAL)
JTAG Access Library (JAL)

- **C++ library to access JTAG chains**
  - Library of basic JTAG commands for use in online software
    - SPY / local DAQ (if foreseen through JTAG)
    - Boundary Scan
  - Complete solution for re-programming of firmware (flash PROMs)
    - Sequencer for Serial Vector Format (SVF) files

- **Hardware Access through the Hardware Access Library (HAL)**

- **Works with Xilinx and Altera PROMs**

- **Common component for …**
  - Global Trigger (Xilinx and Altera PROMs)
  - Global Muon Trigger (Xilinx PROMs)
  - Drift Tube Track Finder (Altera PROMs)
  - Also works for CSC Sector Processor (Xilinx PROMs)
    - JTAG state machine handling in JAL is based on code by National Semiconductor with modifications by Florida CMS group
JTAG Access Library (JAL)

PC running XDAQ

PCI-VME

JTAG Controller
Device (e.g. ScanPSC100F) or firmware mapped into VME address space

JTAG Chain
Containing Altera or Xilinx Devices

Monitoring / Test GUI / Run Ctrl / Config DB

Hardware Access Library

JTAG Access Library as a component of the online software

XDAQ Application running on PC in Rack
JAL Features

- **C++ library to access JTAG chains in hardware**
  - JTAG Flash Programming / General easy access to JTAG Chains
  - Supports simultaneous access to multiple JTAG controllers, multiple chains
  - Multithreading support foreseen (multiple chains on one controller used concurrently)

- **Supports multiple ways to access JTAG chains**
  - JAL ⇒ Hardware Access Library (HAL) ⇒ PCI-VME ⇒ JTAG controller chip (default)
  - JAL ⇒ Parallel port cable (e.g. ByteBlaster)
    - for testing
  - JAL ⇒ other access methods (e.g. PCI; easy to extend)

- **Supports different JTAG Controllers**
  - National Semiconductor ScanPSC100
  - JTAG controller firmware (e.g. developed by J. Erö for DTTF)
  - … other controllers can easily be added
JAL Structure

JAL is organized in three layers (libraries)

- **JTAGController** (with different implementations)
  - send TMS sequence
  - scan bits through chain
- **JTAGChain**
  - Scan Instruction Registers
  - Scan Data Registers
  - Change state …
- **JTAGSVFSequencer**
  - Sequence a Serial Vector Format File (SVF)
  - Format supported by both Xilinx and Altera tools
  - Program/Verify PROMs
JAL Example

JTAG Access Library example using a HAL VMEDevice

```
// create a HAL VME Device
VMEDevice outcard(addressTable, busAdapter, VMEbase);

// create a HAL adapter for a ScanPSC type JTAG Controller
HALScanPSC100Adapter adapter (outcard, "PSC100BASE");

// create the JTAG Controller using the adapter
ScanPSC100JTAGController controller( adapter );

// open JTAG Chain for chain 0 on the controller
JTAGChain chain( controller, 0 );

// instantiate a JTAG SVF sequencer
JTAGSVFSequencer seq;

// load an SVF file
seq.loadSVFFile("firmware_xyz.svf");

// sequence the file on the chain
bool status = seq.execute( chain );
```
JAL Example

- Access method to JTAG is easy to change …

```cpp
// open JTAG Chain for chain 0 on the controller
JTAGChain chain( controller, 0 );

// instantiate a JTAG SVF sequencer
JTAGSVFSequencer seq;

// load an SVF file
seq.loadSVFFile("firmware_xyz.svf");

// sequence the file on the chain
bool status = seq.execute( chain );
```
• Same Example using a ByteBlaster cable …

```cpp
// create the ByteBlaster JTAG Controller
ByteBlasterJTAGController controller;

// open JTAG Chain for chain 0 on the controller
JTAGChain chain( controller, 0 );

// instantiate a JTAG SVF sequencer
JTAGSVFSequencer seq;

// load an SVF file
seq.loadSVFFile("firmware_xyz.svf");

// sequence the file on the chain
bool status = seq.execute( chain );
```
JTAGController Class Reference

JTAG Controller abstract base class. More...

#include <JTAGController.h>

Inheritance diagram for JTAGController:

```
JTAGController
    
ByteBlasterJTAGController

ScanPSC100JTAGController

DTPFJTAGController
```

List of all members.

Public Methods

- `JTAGController()`
- `virtual ~JTAGController()`
- `virtual unsigned int numberOfChains() = 0` returns the number of JTAG chains supported by the JTAG controller. More...
- `virtual void selectChain (int chain=0)` select one of the chains of the JTAG controller as active chain for subsequent commands. More...
JAL Tests

✓ Test 1: Outcard (Drift Tube Track Finder test card)
  ⇒ A) JAL ⇒ HAL ⇒ VME ⇒ ScanPSC100 controller ⇒ Altera EPC2
    ✓ Program / Verify EPC2
  ⇒ B) JAL ⇒ parallel port ⇒ ByteBlaster Cable ⇒ Altera EPC2
    ✓ Program / Verify EPC2
  ⇒ Multiple EPC2 devices programmed in parallel (big time saving)
  ❖ Tests done with a preliminary version of JAL at Universidad Autónoma Madrid
    (Thanks to J. F. Trocóniz, M. Fernandez, J. Erö, Ch. Deldicque)

✓ Test 2: Final Decision Logic board (Global Trigger)
  ⇒ JAL ⇒ HAL ⇒ VME ⇒ JTAG Controller Firmware (J. Ero) ⇒ Xilinx XC18V04
    ✓ Program / Verify XC18V04
  ❖ Test done in Vienna: thanks to H. Bergauer, S. Kostner

✓ Test 3: CSC Sector Processor (CSC Track Finder)
  ⇒ JAL ⇒ HAL ⇒ VME ⇒ ScanPSC100 controller ⇒ Xilinx XC18V04
    ✓ Program / Verify XC18V04
  ❖ Test done at CERN: thanks to H. Stoeck, D. Acosta
JAL Status and Plans

➢ Status
  ➔ Experimental release JAL_0_0_3
  ➔ see: cmsdoc.cern.ch/~hsakulin/jal
    ▪ Tar-ball
    ▪ Brief documentation
  ➔ If you are interested, contact me: Hannes.Sakulin@cern.ch

➢ Plans
  ➔ Support other systems (if there is interest)
Summary

- GMT Online software will be developed in parallel with testing of hardware
- Same approach used as for GT online software
- So far developed JTAG Access Library (JAL)
  - Common component for GT, GMT, DTTF, …