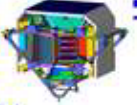
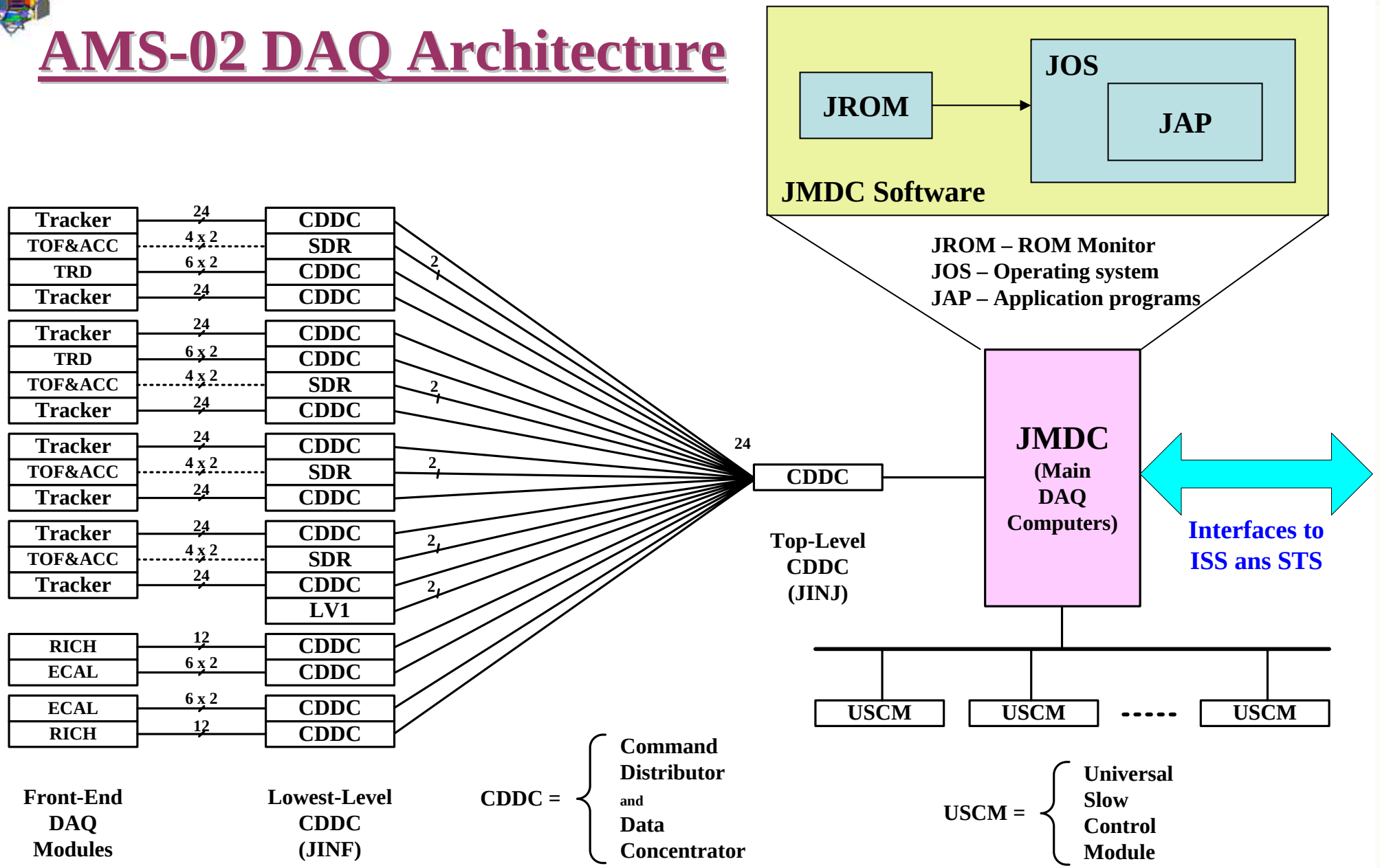


AMS-02 JMDC Software

Xudong CAI @ MIT

CERN, 22 July 2004

AMS-02 DAQ Architecture

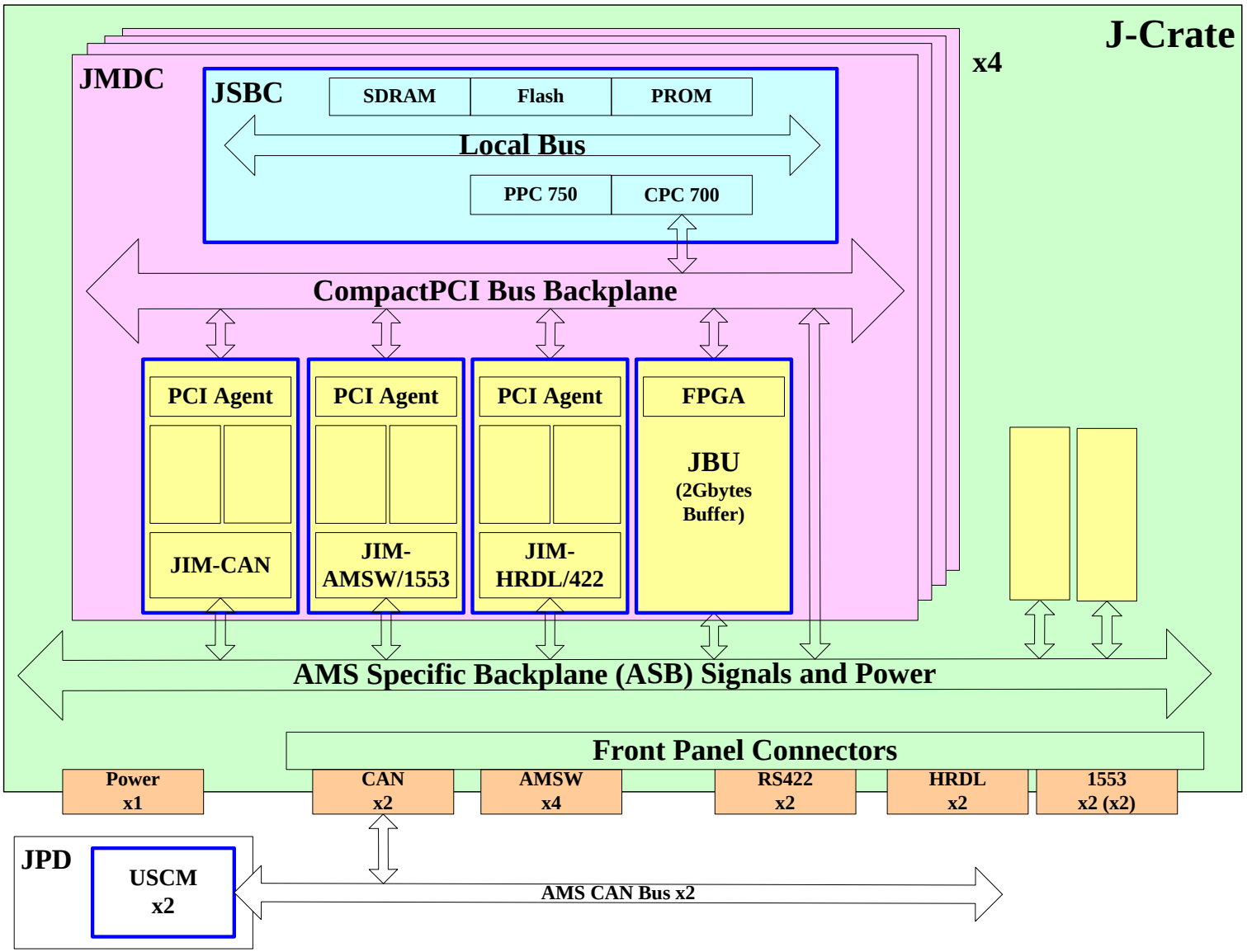


Working Team

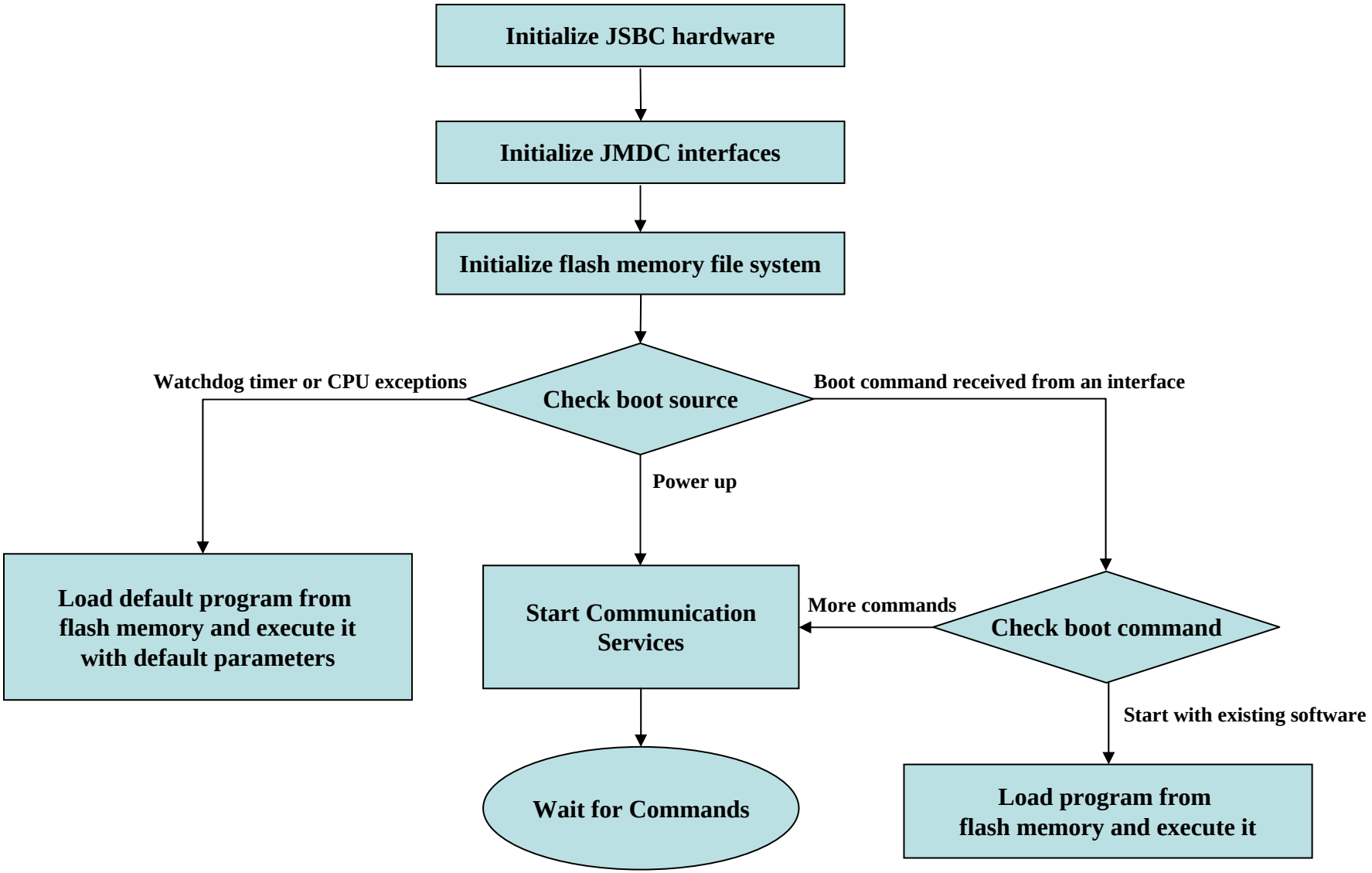
- **A.Kounine (JROM, JAP)**
- **X.Cai (JOS-Linux, JAP)**
- **S.Xu (JOS-eCos, JAP)**
- **Y.Zhou (JAP)**
- **P.Dennett (JAP)**

- **Contributed also from:**
  - M.Capell
  - A.Lebedev
  - V.Koutsenko

JMDC Hardware



JROM – ROM Monitor Functions

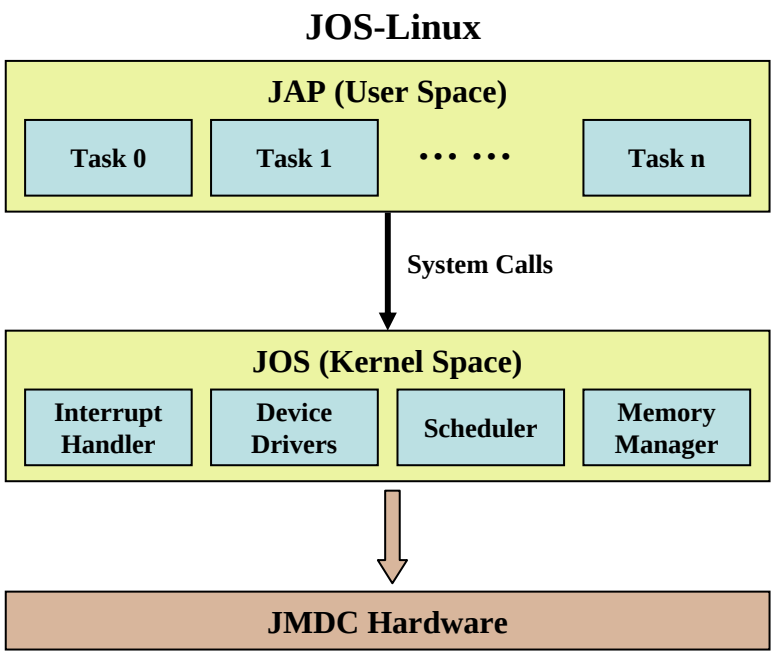


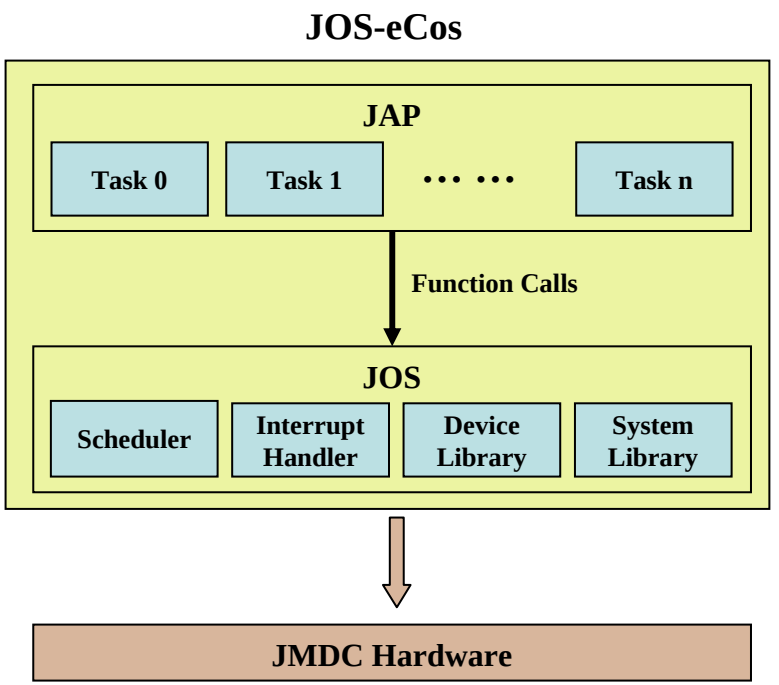
JROM Status

- **The first release will be finished by the end of July with following functions:**
  - Preliminary flash file system is supported
  - Communication over serial port, HRDL and AMSWire links are supported
  - File transfer protocol is supported
  - Files can be loaded via supported links
  - JOS can be loaded from flash memory
  - Boot command from interfaces are handled
- **The rest of interfaces are not supported yet.**
- **Watchdog timer and JSBC SDRAM ECC mode are not supported yet**
- **Continue development and tests are needed.**

JOS - Operating System

- **We have two parallel developments based on two different operating systems.**
  - Linux Hardhat 2.0 with kernel 2.4.2
  - RedHat eCos 2.0
- **Main differences:**
  - Linux is a complicated general-purpose operating system (multiple processes)
  - eCos is simple embedded operating system (single process and multiple threads)



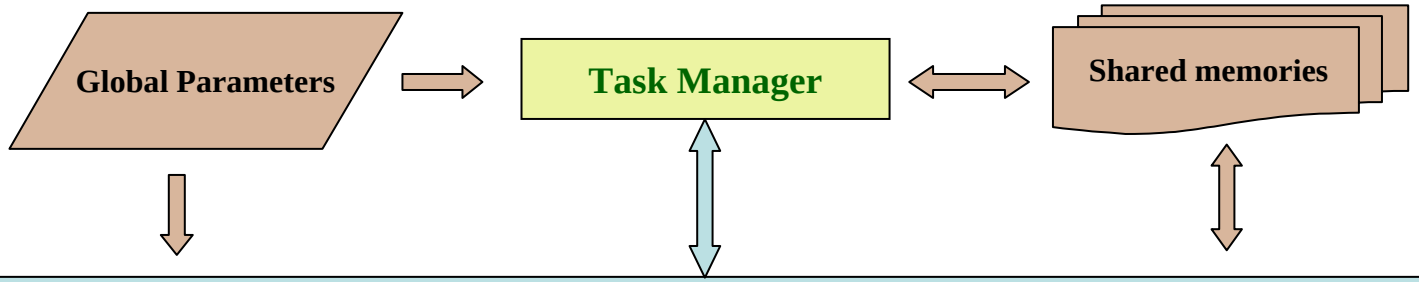


JOS - Status

- **Linux version is based on Hardhat Linux 2.0**
  - Kernel 2.4.2 modified for JSBC support in 2002 and no change since then
  - Device drivers for all JMDC devices (JSBC, JIM-HRDL/422, JIM-AMSW/1553, JIM-CAN and JBU) are supported
  - DMA transfer over PCI bus is supported (default: read using DMA and write not)
  - JAP will be automatically started by JOS after boot

  - JSBC watch-dog timer support is not included yet
  - JSBC flash file system is not included yet
  - JSBC memory ECC mode is not used yet
  - JBU file system is not included yet
- **eCos version 2.0 development was started from this year only. It is not in running status. Hopefully we will have something running in August.**

JAP - Tasks



Basic Tasks

- Display Server
- Status Monitor
- Command Handler
- HRDL TX/RX Servers
- 1553 TX/RX Servers
- AMSW-JINJ Server
- AMSW-Master Server
- AMSW-Slave Server
- CAN-Master Server
- CAN-Slave Server

DAQ Tasks

- Event Builder
- Level-3 Trigger
- Data Compress

Manual Tasks

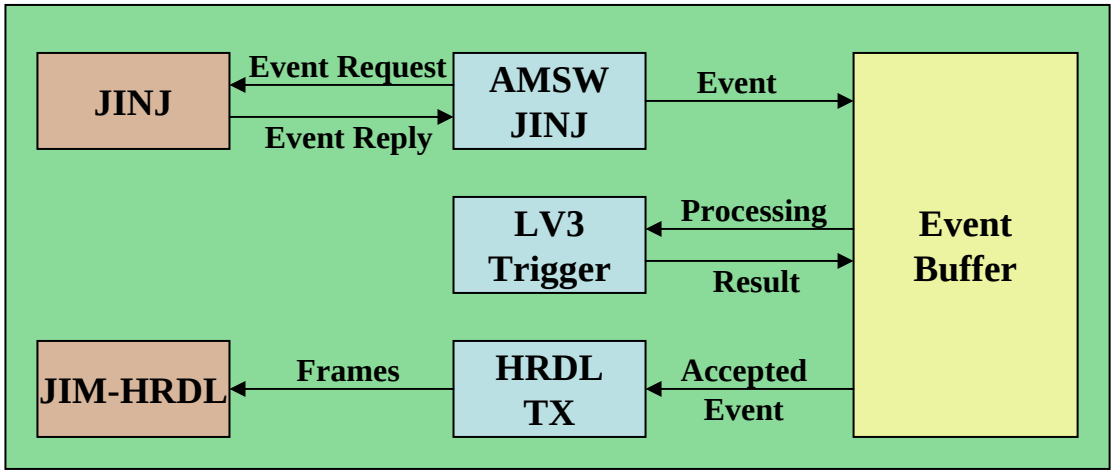
- Calibration
- ⋮

MCT Tasks

- TRD Gas Control
- Tracker Temp. Control
- Magnet Monitor
- Power Control
- Q-List
- ⋮

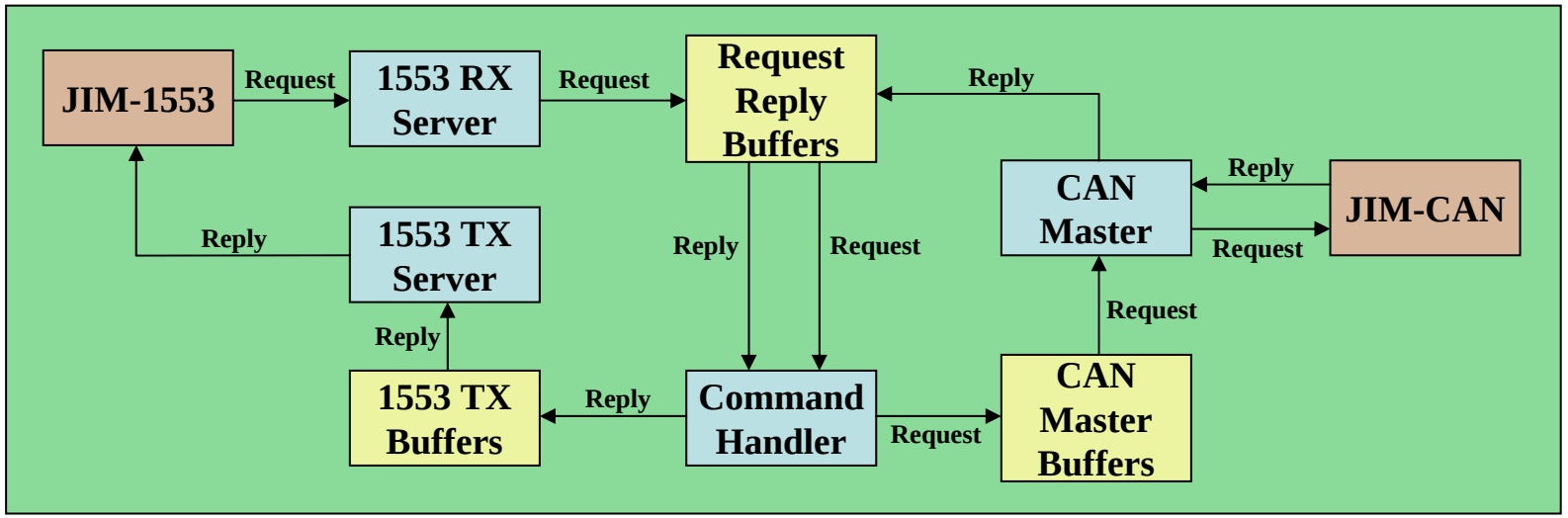
Task Table

JAP - Event Data Flow



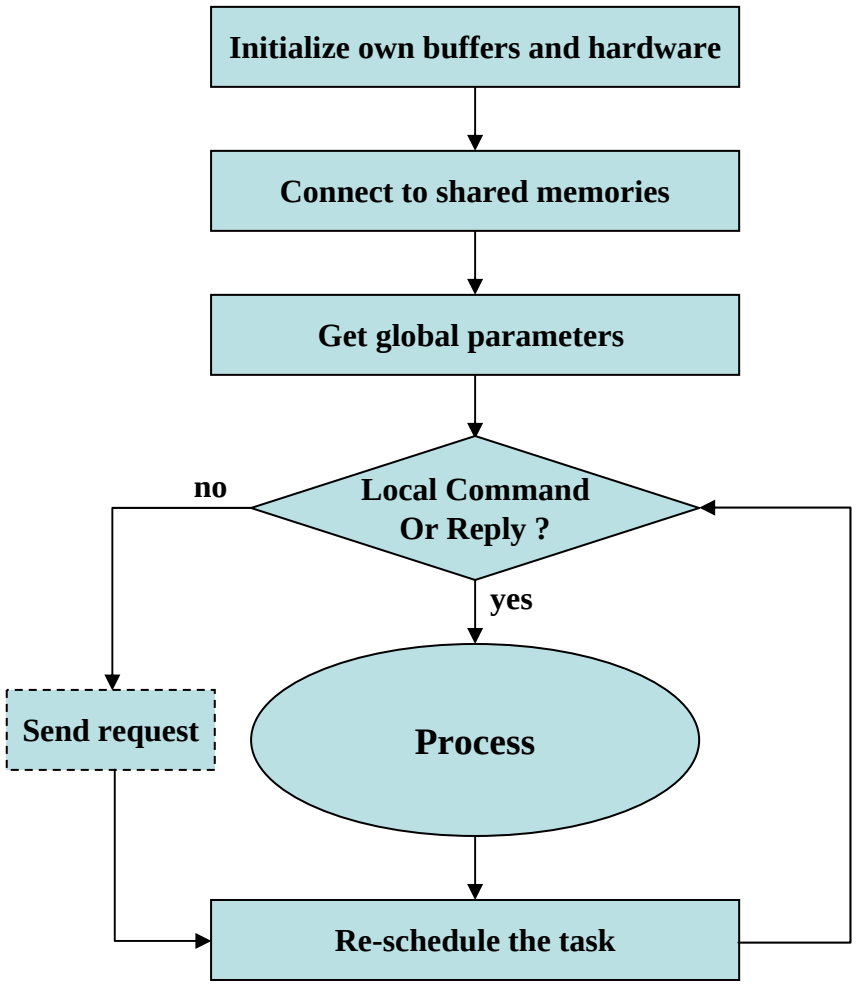
- Event request is a special request from task “AMSW-JINJ”. According to global parameters, AMSW-JINJ will generate event request if no other requests received from its request buffer and DAQ is started.
- Received events will be write into event buffers if there is space left.
- Event headers are reserved at the beginning of the event.
- Level-3 trigger takes event written by AMSW-JINJ and processes them. The result will be written into the same buffer. Event header will be updated.
- HRDL TX task will transfer accepted events over HRDL link and discard the rejected events from the buffer.
- The buffer is freed for new event after the event is transferred or discarded.

JAP - Command Data Flow



- This is a example of commanding from 1553 link to a CAN device.
- “1553 RX Server” gets the request and pass to “Command Handler” via “Request & Reply Buffers”.
- “Command Handler” processes the request and finds out it is for a CAN node. It will send the request to “CAN Master” via its own request buffers.
- “CAN Master” sends the request via JIM-CAN module to the CAN bus and waits for a reply.
- When reply received, “CAN Master” sends this reply back to “Command Handler” via “Request & Reply Buffers”.
- Finally the reply is sent out to 1553 link by “1553 TX Server” when it receives from its TX stream buffer.

JAP - Task Interface



- Task exchange data with others via shared memories.
- AMS block is the only format which is supported for data exchange.
- Each task should divided into states and managed by a state machine.
- One state should not be longer than 500 µsec.
- Only one state will be executed each time when the task become active.
- When task is waiting for data input, it should re-schedule itself. It means that one state is ended.
- All tasks are managed by scheduler. Currently they have equal priority.

JAP - Status

- **The baseline JAP under JOS-Linux was running with event building and processing from AMSWire to HRDL link, CAN master for slow control and commanding interface via HRDL link.**
- **DAQ chain has tested (may be improved):**
  - Without level-3 trigger, it can handle 2 kBytes * 2.5 kHz
  - With level-3 trigger (dummy) < 200 μsec, it can handle 2 kBytes * 2.0 kHz
  - With level-3 trigger (dummy) > 200 μsec, the rate is reduced depend on the time spent.
- **HRDL receiving can accept about 40 Mbits/sec (with 8 receive buffers).**
- **AMSWire master for JMDC, AMSWire slave for JMDC and CAN slave have not finished.**
- **File transfer protocol is not yet supported.**

JMDC Tests

- **Space qualification tests:**
  - Thermal Stress
  - Random vibration
  - EMI/EMC
  - Magnetic field
  - Thermal Vacuum Test (TVT)
- **Preliminary Interface Test (PIT)**
  - HRDL interface test
  - ISS 1553 interface test
  - STS 1553 interface test
- **Functional Integration Test (FIT)**
  - ISS system
  - STS system
  - PAD system
  - T0 system
- **AMS-02 Integration Test**
- **AMS-02 Beam Test**
- **AMS-02 TVT Test**

Photo during PIT in Houston, June 2003

Summary

- **We all work on the preliminary flight software now.**
- **Our target is the functional integration test (FIT) in Jan. 2005. Part of them may be tested together with trigger and tracker systems during beam test at CERN in Sep. – Oct. 2004.**
- **Functions for FIT:**
  - Event building, processing and send to HRDL links
  - Communication support via all interfaces
  - Slow control task for JPD
- **We will continue the parallel developments and try to make selection this year. Afterwards we will concentrate on a single approach for flight software.**