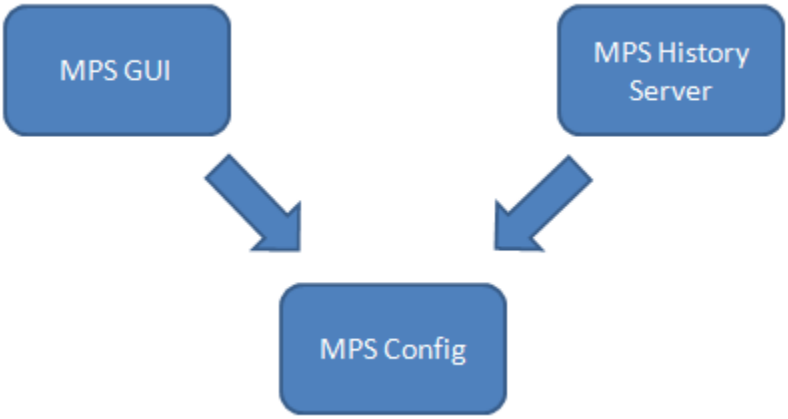
MPS

MPS Config is a client library for interacting with the Config/Logic DBs as well as relevant MPS PVs. Written in Java.
MPS GUI is a desktop application that displays the state of the MPS. Written in Java.
MPS History Server is a Unix daemon that saves MPS messages to Oracle
MPS Editors page for MPS Logic editor and MPS Config editor.
MPS Stats is a prototype that can correlate MPS History data with the data from the Channel Archiver.

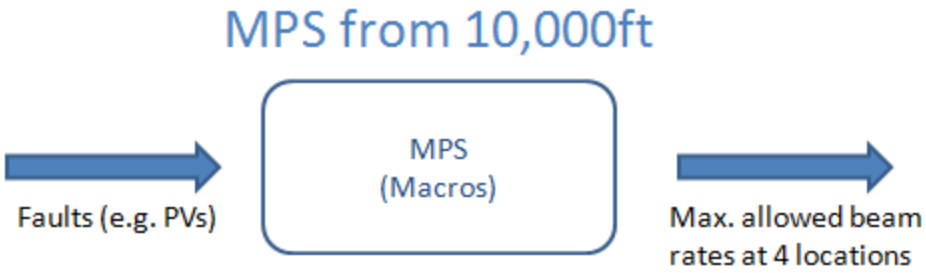
General Remarks

- MPS GUI and MPS History Server are built on top of the module MPS Config



- All SQL queries are stored in .properties files in corresponding packages
- (Almost) all constants/properties for each MPS "module" are located in the class edu.stanford.slac.module.ModuleProperties

Glossary



Fault

- aka (binary) **input**, signal
- The basic event in the MPS
  - Belongs to an MPS device; usually a PV
- Has two states: OK (true or 1) and Faulted (false or 0)
  - Current fault states is what drives the MPS
- Stored in the Config DB
- 4 types: EPICS, LinkNode, LinkNodeChannel, LinkProcessor
  - Classes in MPS Config edu.stanford.slac.mps.fault
- Fault numbers are keys; fault names may be changed by MPS engineers

Macro

- aka logic, **truth table**
- Defines 2-4 #MacroStates for 1 or 2 faults
    - The order of faults matters (there is a column "position" in the DB schema)
    - May need support for more than 2 faults in the future (major redesign needed!)
- Can be bypassed (set) to a #MacroState for a period of time
    - Bypassed fault numbers (ids) are stored in IOC:BSY0:MP01:BYPASS_LIST.VALA
    - Corresponding (absolute) end times are stored in IOC:BSY0:MP01:BYPASS_LIST.VALB (using EPICS epoch)
    - Operators want to know when bypasses expire
- Macro numbers are keys; macro names may be changed by MPS engineers
- Stored in the Logic DB

Ignore Condition

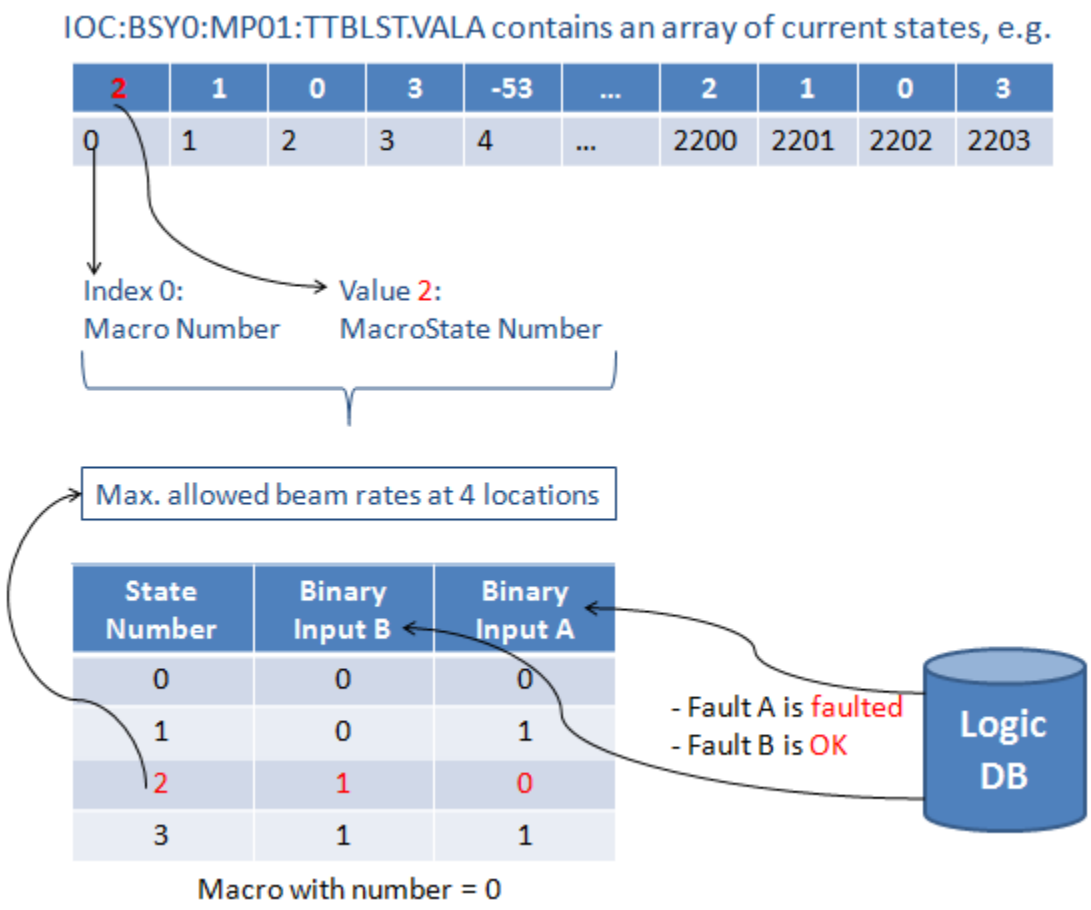
- aka ignoring macro (overload- sigh!), ignore logic
- Sort-of "meta"-macros that can ignore other macros
    - Useful, for instance, when the fault hardware misbehaves
    - Can be set active or inactive (by MPS engineers, not operators)
    - If active, operators want to know the minimum rate the MPS would allow, if the condition became inactive
    - Note: Some macros are always evaluated
- Stored in the Logic DB

MacroState

- aka (just) state
- Defines the maximum allowed beam rates for a combination of fault states
    - Rate names are hard-coded in edu.stanford.slac.mps.jdbc.logic.Rate
- Characterized by a state number
    - A negative state number has a predefined meaning (see method getMacroState() in the class edu.stanford.slac.mps.jdbc.logic.LogicDB in MPS Config)
    - A non-negative state number limits beam rates
- The binary representation of the (non-negative) state number reflects the states of the corresponding faults; example for 2 faults: A (position=0) and B (position=1):

State Number	Binary Representation	B*	A*	Rate Limits at 4 Locations
0	00	0	0	0Hz 10Hz 0Hz 0Hz
1	01	0	1	0Hz 10Hz 120Hz 120Hz
2	10	1	0	120Hz 10Hz 0Hz 0Hz
3	11	1	1	120Hz 10Hz 120Hz 120Hz

    - Note: In A and B columns, you may also see T (for True) and F (for False)
- Current state numbers of every macro are stored in IOC:BSY0:MP01:TTBLST.VALA



- Stored in the Logic DB