DAQ Control with Epics

MBS monitor (FOPI)

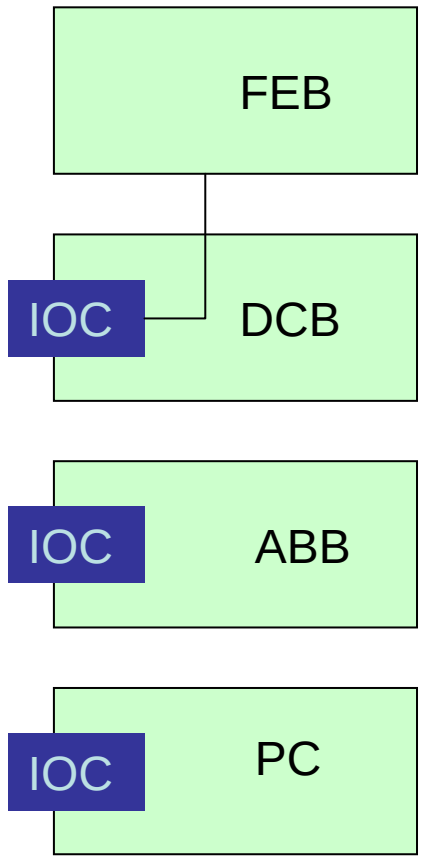
(J.Adamczewski, M.Stockmeier)

DAQ Control

- Control
  - Setup
  - Steering
  - Observing
  - Configuration persistency
- Monitoring
  - Status of all components
  - Performance
  - Logging
- Alarms
  - Messages
  - Conditions
- Diagnosis

EPICS Components

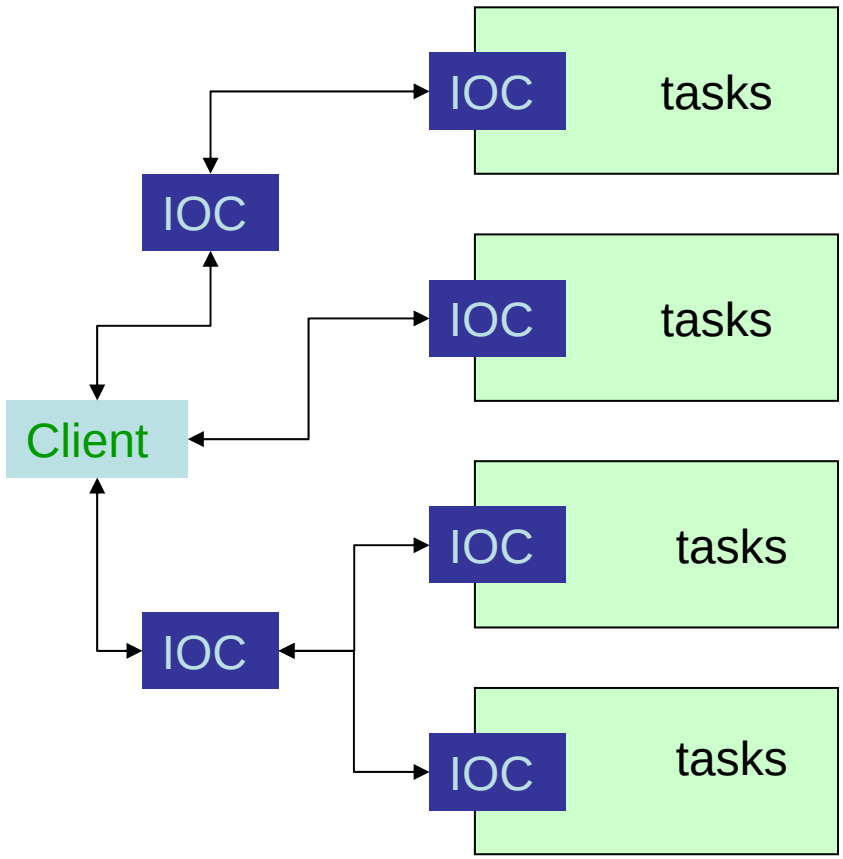
- IOCs
  - One IOC per standard CPU (Linux, Lynx, VxWorks)
- clients
  - on Linux, (Windows)
- Agents
  - Segment IOCs beeing also clients



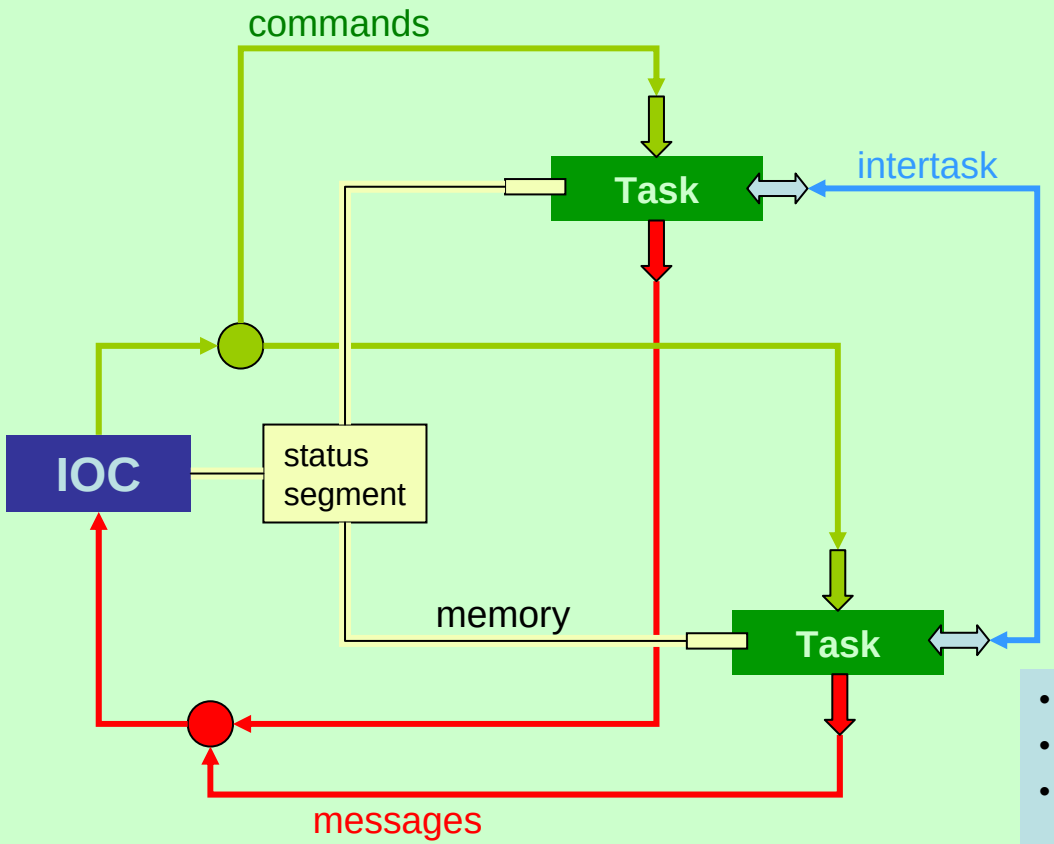
Hierarchy in a flat system

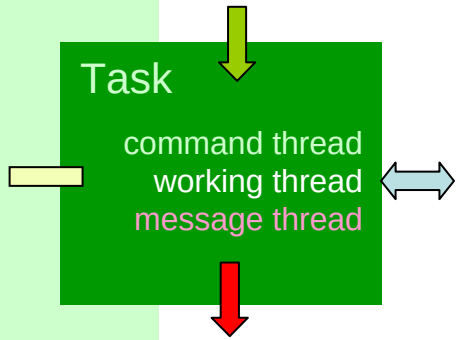
- IOCs
  - One IOC per standard CPU (Linux, Lynx, VxWorks)
- clients
  - on Linux, (Windows)
- Agents
  - Segment IOCs beeing also clients

Name space architecture!



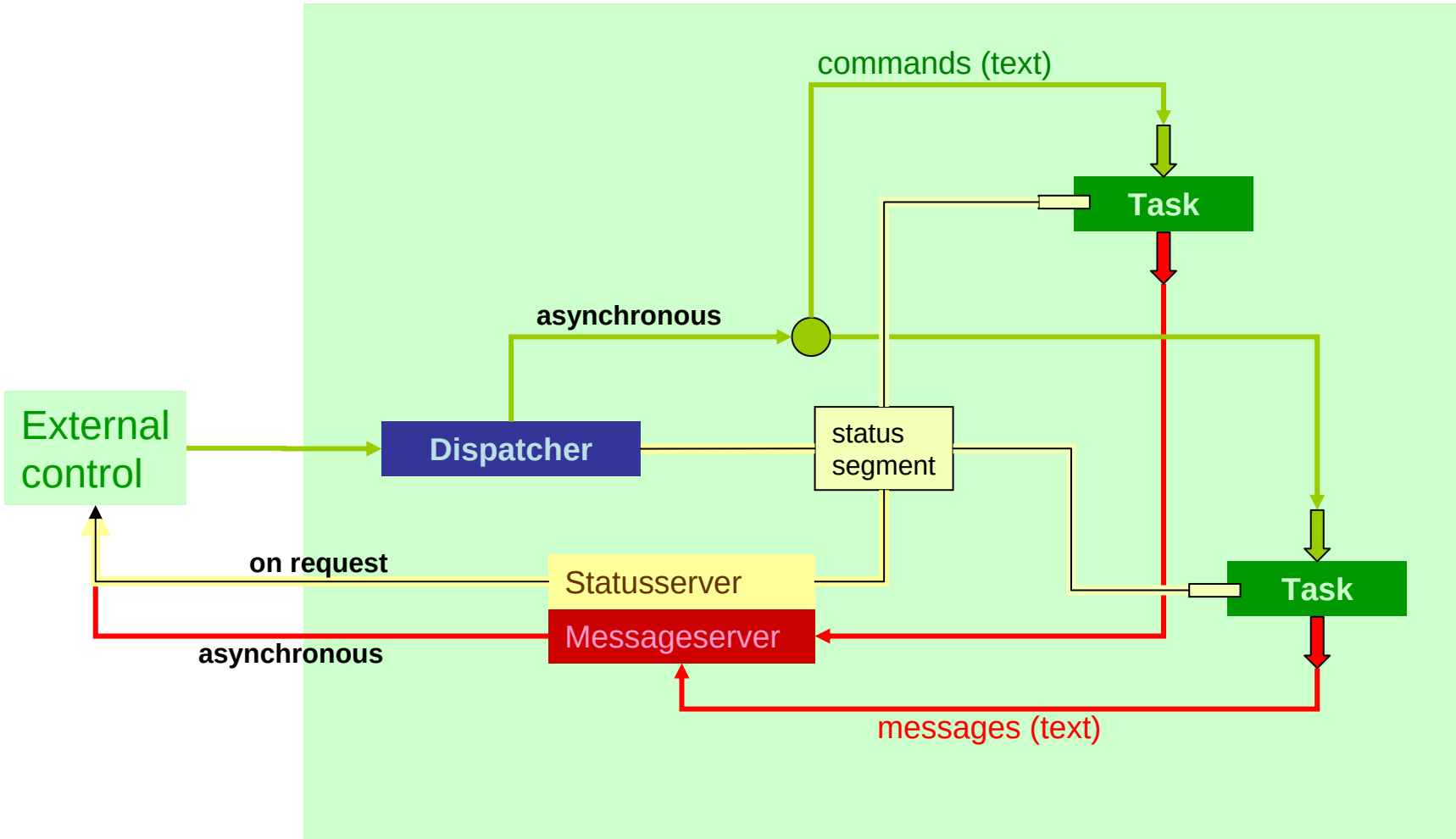
Local communication



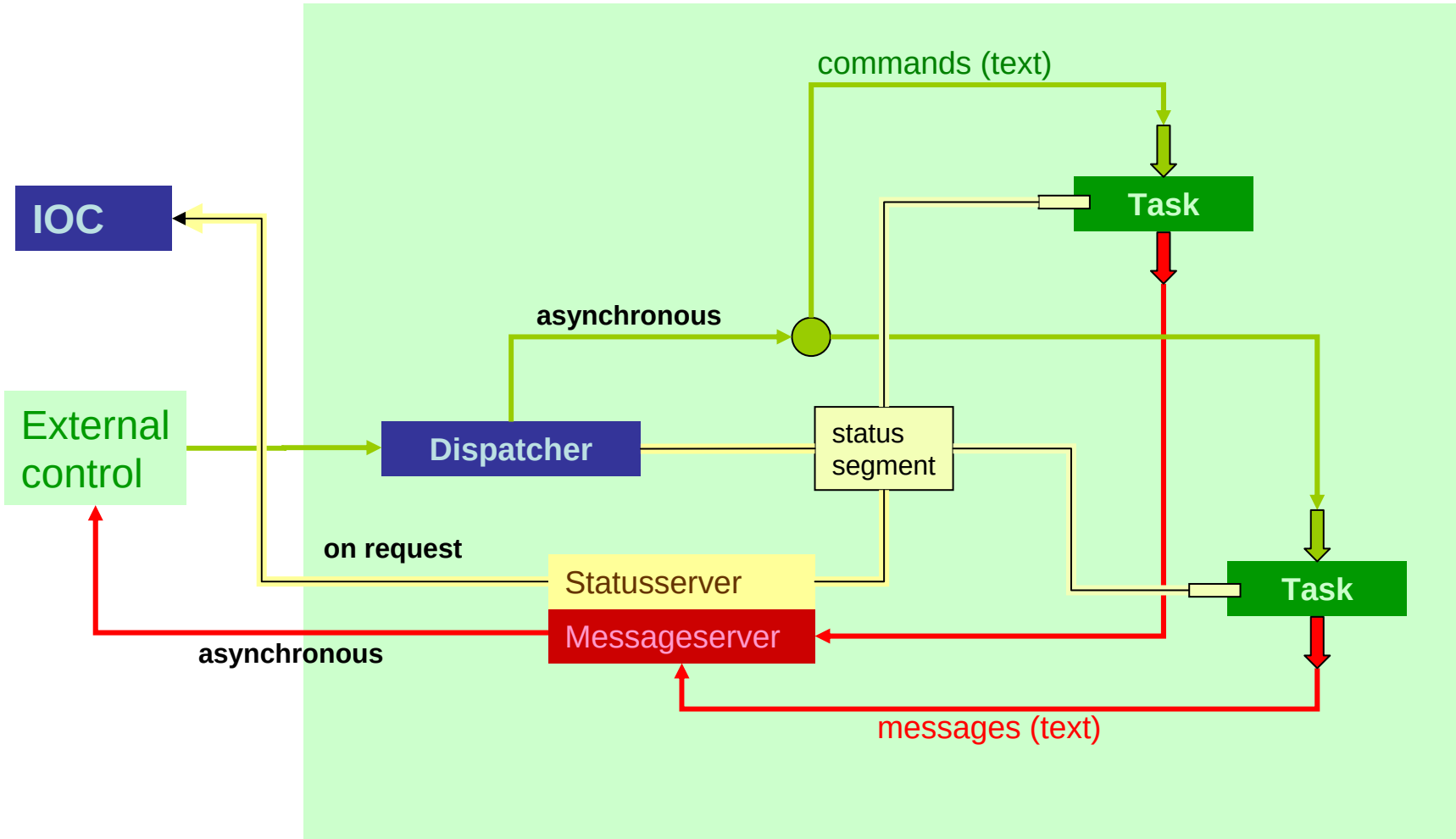


- Commands handled by threads
- Execution maybe in working thread
- Message thread maybe not needed

MBS node


commands (text)
Task
asynchronous
External control
Dispatcher
status segment
on request
Statusserver
Task
asynchronous
Messageserver
messages (text)

MBS node and monitor IOC



MBS monitor IOC

After one week work from scratch

- new MBS record type
- one record instance per MBS node
- Device support gets status of MBS node and fills MBS record
- Fan out to MBS node process variables
- IOC started on any Linux
- IOC could also run on each MBS node
- Display with medm

Starting script IOC

Listings not complete!

```
#!epics/mbs/bin/linux-x86/mbsmon

dbLoadDatabase("dbd/mbsmon.dbd") # Load all record definitions
mbsmon_registerRecordDeviceDriver(pdbbase) # Register all support components

# Load record instances, one line for each MBS node to be monitored.
dbLoadRecords("db/dbMbsMonitor.db","user=FOPI, mbsnode=mbs1, mbsname=lxi010, scan=1 second, evhirange=30000, kbhirange=1500")
dbLoadRecords("db/dbMbsMonitor.db","user=FOPI, mbsnode=mbs2, mbsname=lxi011, scan=1 second, evhirange=30000, kbhirange=1500")
dbLoadRecords("db/dbMbsMonitor.db","user=FOPI, mbsnode=mbs3, mbsname=lxi012, scan=1 second, evhirange=30000, kbhirange=1500")
dbLoadRecords("db/dbMbsMonitor.db","user=FOPI, mbsnode=mbs4, mbsname=lxi013, scan=1 second, evhirange=30000, kbhirange=1500")
dbLoadRecords("db/dbMbsMonitor.db","user=FOPI, mbsnode=mbs5, mbsname=r3-2, scan=1 second, evhirange=30000, kbhirange=1500")
dbLoadRecords("db/dbMbsSums.db","user=FOPI, mbsnode1=mbs1,,,,,mbsnode2=mbs2, scan=2 second ,evhirange=300000, kbhirange=15000")

cd ${TOP}/iocBoot/${IOC}
iocInit()
```

Starting script IOC

Listings not complete!

mbsmon.dbd

```
recordtype(mbsStatus) {
   include "dbCommon.dbd"
      field(VAL,DBF_DOUBLE) {
            prompt("dummy value")
            asl(ASL0)
            pp(TRUE)
      }
      field(ISRUN,DBF_ENUM) {
            prompt("Acquisition running")
            asl(ASL0)
            pp(TRUE)
      }
      field(ISAVAIL,DBF_ENUM) {
            prompt("Mbs node available")
            asl(ASL0)
            pp(TRUE)
      }.....}
device(mbsStatus,CONSTANT,devmbsStatusSoft,"SoftChannel")
```

```
#!epics/mbs/bin/linux-x86/mbsmon

dbLoadDatabase("dbd/mbsmon.dbd") # Load all record definitions
mbsmon_registerRecordDeviceDriver(pdbbase) # Register all support components

# Load record instances, one line for each MBS node to be monitored.
dbLoadRecords("db/dbMbsMonitor.db","user=FOPI, mbsnode=mbs1, mbsname=lxi010, scan=1 second, evhirange=30000, kbhirange=1500")
dbLoadRecords("db/dbMbsMonitor.db","user=FOPI, mbsnode=mbs2, mbsname=lxi011, scan=1 second, evhirange=30000, kbhirange=1500")
dbLoadRecords("db/dbMbsMonitor.db","user=FOPI, mbsnode=mbs3, mbsname=lxi012, scan=1 second, evhirange=30000, kbhirange=1500")
dbLoadRecords("db/dbMbsMonitor.db","user=FOPI, mbsnode=mbs4, mbsname=lxi013, scan=1 second, evhirange=30000, kbhirange=1500")
dbLoadRecords("db/dbMbsMonitor.db","user=FOPI, mbsnode=mbs5, mbsname=r3-2, scan=1 second, evhirange=30000, kbhirange=1500")
dbLoadRecords("db/dbMbsSums.db","user=FOPI, mbsnode1=mbs1,,,,,mbsnode2=mbs2, scan=2 second ,evhirange=300000, kbhirange=15000")

cd ${TOP}/iocBoot/${IOC}
iocInit()
```

Starting script IOC

Listings not complete!

mbsmon.dbd

```
recordtype(mbsStatus) {
   include "dbCommon.dbd"
      field(VAL,DBF_DOUBLE) {
            prompt("dummy value")
            asl(ASL0)
            pp(TRUE)
      }
      field(ISRUN,DBF_ENUM) {
            prompt("Acquisition running")
            asl(ASL0)
            pp(TRUE)
      }
      field(ISAVAIL,DBF_ENUM) {
            prompt("Mbs node available")
            asl(ASL0)
            pp(TRUE)
      }.....}
device(mbsStatus,CONSTANT,devmbsStatusSoft,"SoftChannel")
```

```
#!epics/mbs/bin/linux-x86/mbsmon

dbLoadDatabase("dbd/mbsmon.dbd") # Load all record definitions
mbsmon_registerRecordDeviceDriver(pdbbase) # Register all support components

# Load record instances, one line for each MBS node to be monitored.
dbLoadRecords("db/dbMbsMonitor.db","user=FOPI, mbsnode=mbs1, mbsname=lxi010, scan=1 second, evhirange=30000, kbhirange=1500")
dbLoadRecords("db/dbMbsMonitor.db","user=FOPI, mbsnode=mbs2, mbsname=lxi011, scan=1 second, evhirange=30000, kbhirange=1500")
dbLoadRecords("db/dbMbsMonitor.db","user=FOPI, mbsnode=mbs3, mbsname=lxi012, scan=1 second, evhirange=30000, kbhirange=1500")
dbLoadRecords("db/dbMbsMonitor.db","user=FOPI, mbsnode=mbs4, mbsname=lxi013, scan=1 second, evhirange=30000, kbhirange=1500")
dbLoadRecords("db/dbMbsMonitor.db","user=FOPI, mbsnode=mbs5, mbsname=r3-2, scan=1 second, evhirange=30000, kbhirange=1500")
dbLoadRecords("db/dbMbsSums.db","user=FOPI, mbsnode1=mbs1,,,,,mbsnode2=mbs2, scan=2 second ,evhirange=300000, kbhirange=15000")

cd ${TOP}/iocBoot/${IOC}
iocInit()
```

dbMbsMonitor.db

```
record(mbsStatus, "$(user):mbsStatus@$(mbsnode)")
{   field(DESC, "mbsStatus@$(mbsname)")
    field(SCAN, "$(scan)")
    field(FLNK, "$(user):masterfanout@$(mbsnode)")
}
record(fanout, "$(user):masterfanout@$(mbsnode)")
{    field(DESC, "Trigger all")
     field(LNK1, "$(user):runfanout@$(mbsnode)")
}
record(fanout, "$(user):runfanout@$(mbsnode)")
{    field(DESC, "Trigger runstates")
     field(LNK1, "$(user):runningstate@$(mbsnode)")
     field(LNK6, "$(user):available@$(mbsnode)")
}
```

Starting script IOC

Listings not complete!

mbsmon.dbd

```
recordtype(mbsStatus) {
    include "dbCommon.dbd"
        field(VAL,DBF_DOUBLE) {
                prompt("dummy value")
                asl(ASL0)
                pp(TRUE)
        }
        field(ISRUN,DBF_ENUM) {
                prompt("Acquisition running")
                asl(ASL0)
                pp(TRUE)
        }
        field(ISAVAIL,DBF_ENUM) {
                prompt("Mbs node available")
                asl(ASL0)
                pp(TRUE)
        }.....}
device(mbsStatus,CONSTANT,devmbsStatusSoft,"SoftChannel")
```

```
#!epics/mbs/bin/linux-x86/mbsmon

dbLoadDatabase("dbd/mbsmon.dbd") # Load all record definitions
mbsmon_registerRecordDeviceDriver(pdbbase) # Register all support components

# Load record instances, one line for each MBS node to be monitored.
dbLoadRecords("db/dbMbsMonitor.db","user=FOPI, mbsnode=mbs1, mbsname=lxi010, scan=1 second, evhirange=30000, kbhirange=1500")
dbLoadRecords("db/dbMbsMonitor.db","user=FOPI, mbsnode=mbs2, mbsname=lxi011, scan=1 second, evhirange=30000, kbhirange=1500")
dbLoadRecords("db/dbMbsMonitor.db","user=FOPI, mbsnode=mbs3, mbsname=lxi012, scan=1 second, evhirange=30000, kbhirange=1500")
dbLoadRecords("db/dbMbsMonitor.db","user=FOPI, mbsnode=mbs4, mbsname=lxi013, scan=1 second, evhirange=30000, kbhirange=1500")
dbLoadRecords("db/dbMbsMonitor.db","user=FOPI, mbsnode=mbs5, mbsname=r3-2, scan=1 second, evhirange=30000, kbhirange=1500")
dbLoadRecords("db/dbMbsSums.db","user=FOPI, mbsnode1=mbs1,,,,,mbsnode2=mbs2, scan=2 second ,evhirange=300000, kbhirange=15000")

cd ${TOP}/iocBoot/${IOC}
iocInit()
```

dbMbsMonitor.db

```
record(mbsStatus, "$(user):mbsStatus@$(mbsnode)")
{   field(DESC, "mbsStatus@$(mbsname)")
    field(SCAN, "$(scan)")
    field(FLNK, "$(user):masterfanout@$(mbsnode)")
}
record(fanout, "$(user):masterfanout@$(mbsnode)")
{    field(DESC, "Trigger all")
     field(LNK1, "$(user):runfanout@$(mbsnode)")
}
record(fanout, "$(user):runfanout@$(mbsnode)")
{    field(DESC, "Trigger runstates")
     field(LNK1, "$(user):runningstate@$(mbsnode)")
     field(LNK6, "$(user):available@$(mbsnode)")
}
```

dbMbsSums.db

```
record(calc, "$(user):datarate@sums")
{       field(DESC, "Sum of Mbs datarates")
        field(SCAN,"$(scan)")
        field(FLNK, "$(user):eventrate@sums.PROC")
        field(CALC, "A+B+C+D+E")
        field(INPA, "$(user):datarate@$(mbsnode1).VAL  NPP NMS")
        field(INPB, "$(user):datarate@$(mbsnode2).VAL  NPP NMS")
        field(INPC, "$(user):datarate@$(mbsnode3).VAL  NPP NMS")
        field(INPD, "$(user):datarate@$(mbsnode4).VAL  NPP NMS")
        field(INPE, "$(user):datarate@$(mbsnode5).VAL  NPP NMS")
        field(EGU, "kB/s")
}
```

Device support

Listings not complete!

devmbsStatusSoft.c

```
struct {
        long            number;
        DEVSUPFUN       report;
        DEVSUPFUN       init;
        DEVSUPFUN       init_record;
        DEVSUPFUN       get_ioint_info;
        DEVSUPFUN       read_mbsStatus;
}devmbsStatusSoft={
        5,
        NULL,
        NULL,
        init_record,
        NULL,
        read_mbsStatus,
};
epicsExportAddress(dset,devmbsStatusSoft);      // Macro!

static long read_mbsStatus(mbsStatusRecord *pmbsStatus) // called for each mbsStatus instance
{
    s_daqst daqstatus;                          //daqstatus
    curs = strstr(pmbsStatus->desc,"@");        // to get MBS node name from descriptor
    strncpy(node,++curs,63);                    // copy MBS node name
    status=f_mbs_status(node,&daqstatus);       // request status block from MBS node
    pmbsStatus->isrun=daqstatus.bh_running;     // copy variables
    pmbsStatus->udf = FALSE;                    // nothing undefined
    return(0);
}
```

function table

Record support

Listings not complete!

mbsStatusRecord.c

```
#include "mbsStatusRecord.h"
static long process(void *precord)                    // overloaded!
{
   mbsStatusRecord              *pmbsStatus     = (mbsStatusRecord *)precord;
   mbsStatusdset                *pdset          = (mbsStatusdset *)(pmbsStatus->dset);
   status=(*pdset->read_mbsStatus)(pmbsStatus); // from devmbsStatusSoft.c
   recGblGetTimeStamp(pmbsStatus);
   checkAlarms(pmbsStatus);                           // check for alarms
   monitor(pmbsStatus);                               // check event list
   recGblFwdLink(pmbsStatus);                         // process the forward scan link record
   return(status);
}
static void monitor(mbsStatusRecord *pmbsStatus)
{
   unsigned short               monitor_mask;
   monitor_mask = recGblResetAlarms(pmbsStatus);
   // send out monitors connected to the value field
   if (monitor_mask) db_post_events(pmbsStatus,&pmbsStatus->val,monitor_mask);
return;
}
static void checkAlarms(mbsStatusRecord *pmbsStatus)
{
.........
}
```

Screen shot FOPI