ROD Test Stand
Overview

http://www-hep.fzu.cz/~tomasekl/RODTestStandOverview.ppt

Lukáš Tomášek <tomasekl@fzu.cz>
Jan Valenta <valenta@fzu.cz>

LBNL
The University of Wisconsin, Madison
Institute of Physics AS CR, Prague

Test Stand basics

Purpose

- initial testing and debugging a standalone ROD,
- production testing and ROD maintenance.

Software

- programming language ANSI C
- MS Windows 9X, NT, 2000, XP operating system
- NI LabWindows/CVI 5.5 development software

Hardware

- PC with NI PCI-MXI-2 interface card,
- VME crate with NI VME-MXI-2 card

Main TestStand tasks and features

- **ROD initialization and configuration:**
  - loading FPGA code to flash memories,
  - R/W FPGA command/status registers (program/reset manager),
  - loading program for DSPs (MDSP flash, slaves directly).

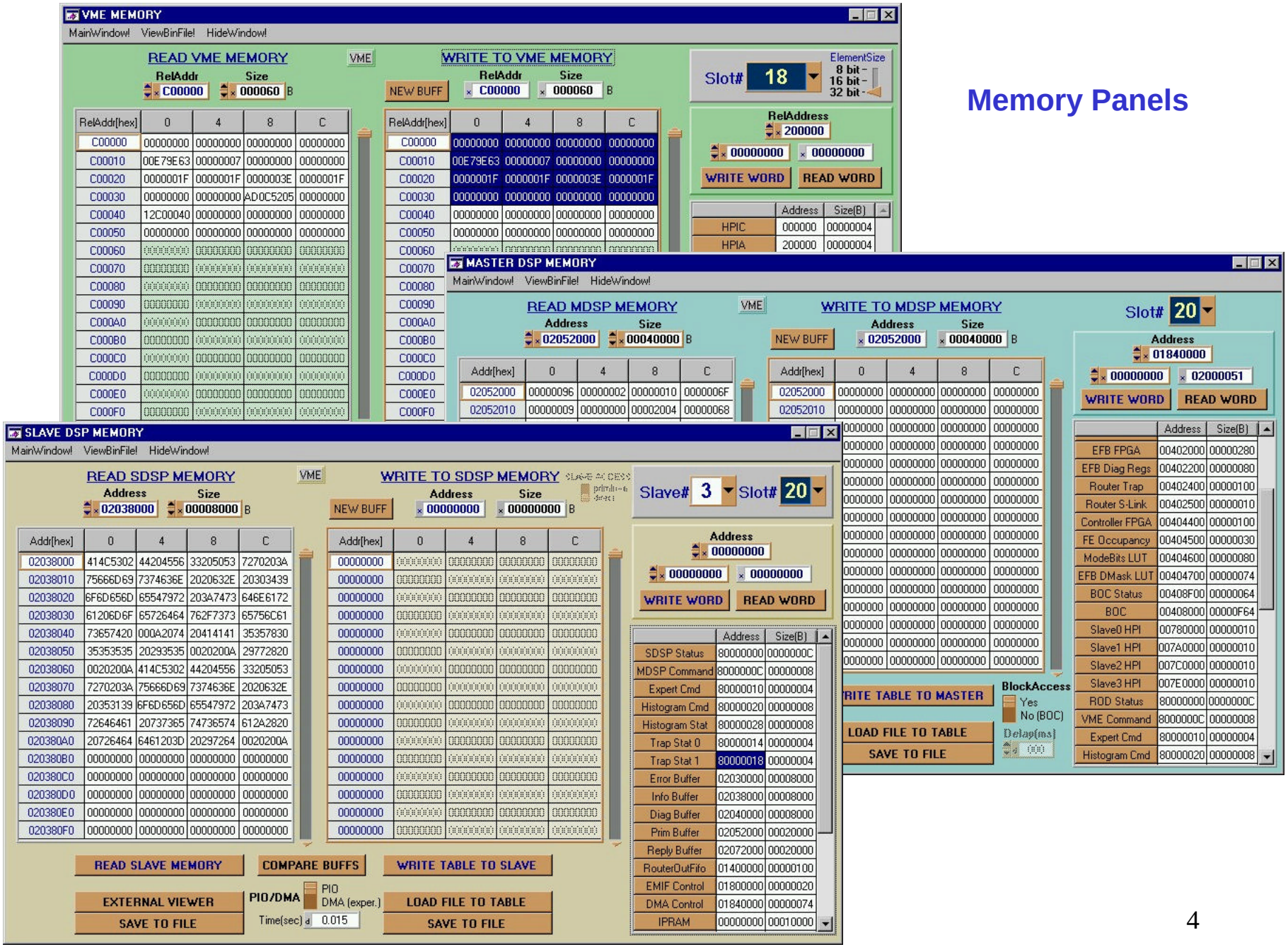
- **ROD control:**
  - by *PrimitiveLists*,
  - direct commands via *VmeCommandRegister* bits (DmaRequest, Pause …),
  - more comprehensive tasks by *CommandLists*.

- **ROD monitoring:**
  - periodic checking of the ROD status registers (ControllerFPGA, DSP status),
  - readout text messages from the *TextBuffers*.

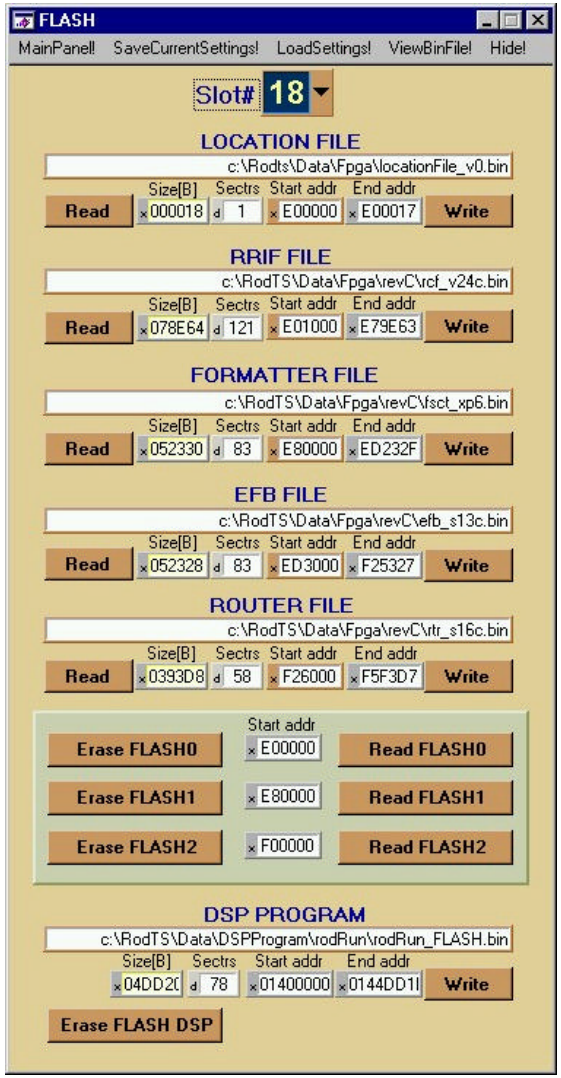
- **ROD utilities and supporting applications:**
  - file format converters (binary-hex, coff, …),
  - binary data creator, binary file viewer (freeware Witched), etc.

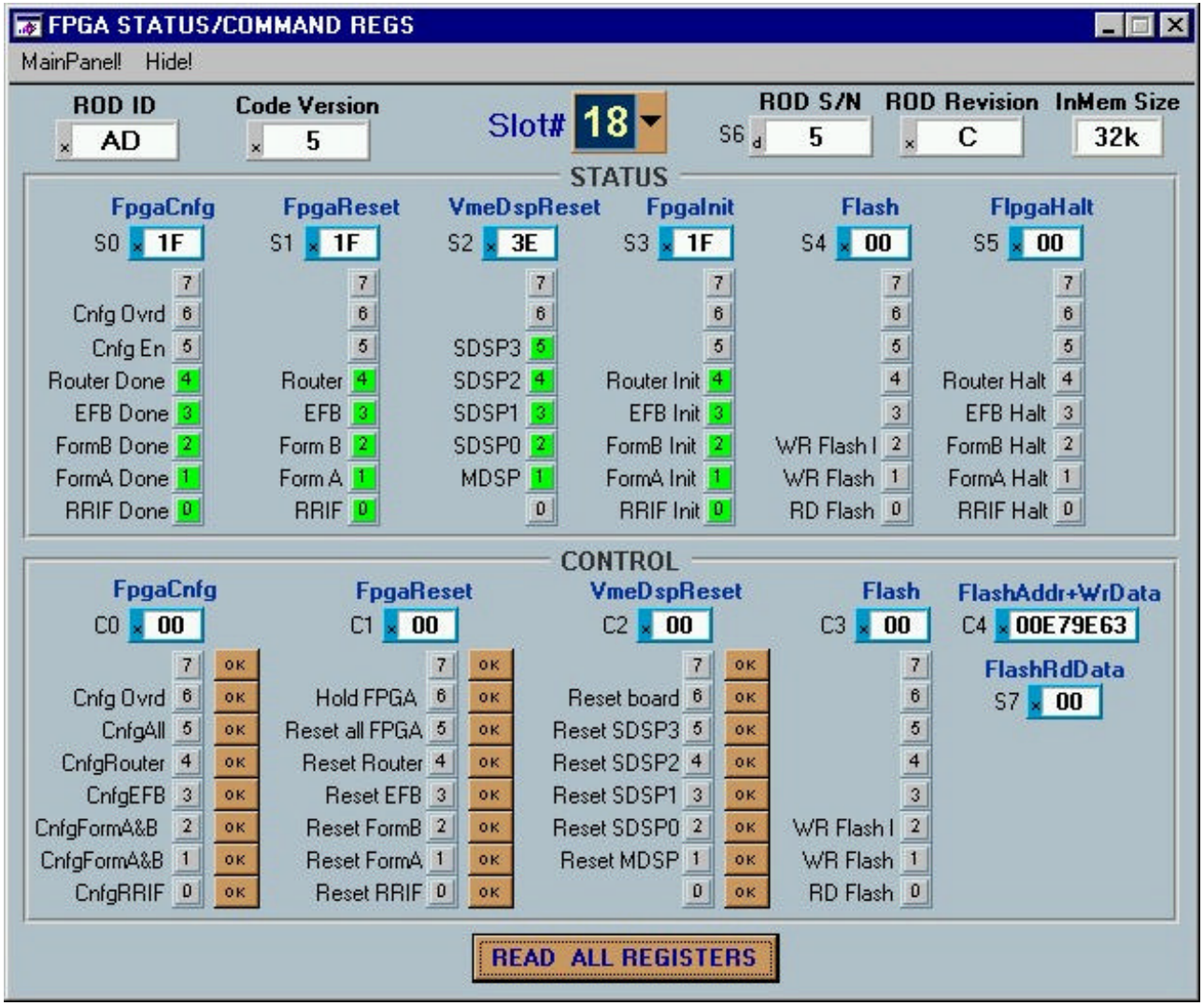
TestStand supports one VME crate - up to 16 Rods can run simultaneously.

Memory Panels

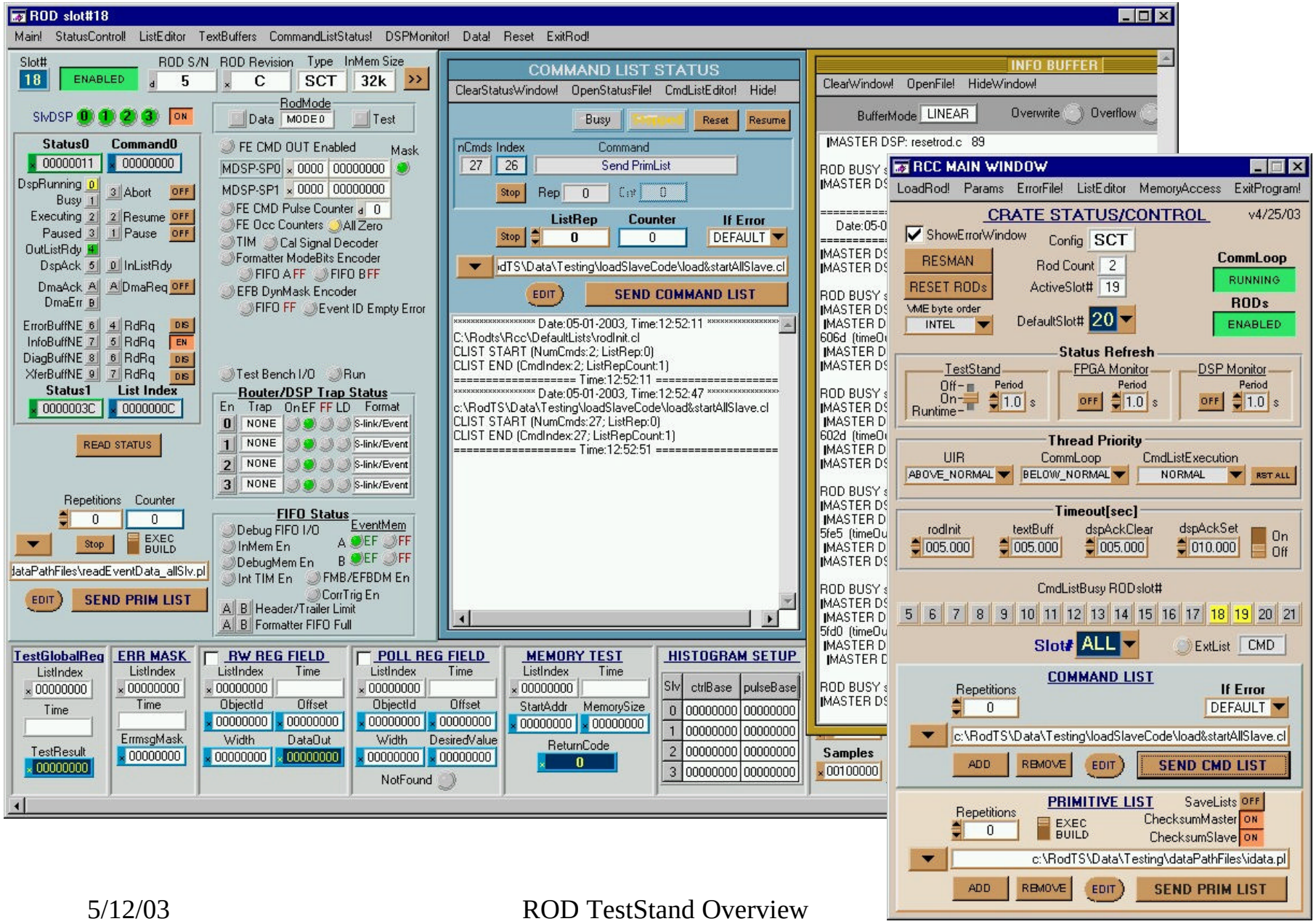


Flash Panel and FPGA Program/Reset Manager





ROD Control/Status[n] and Main TestStand Control/Status Panels



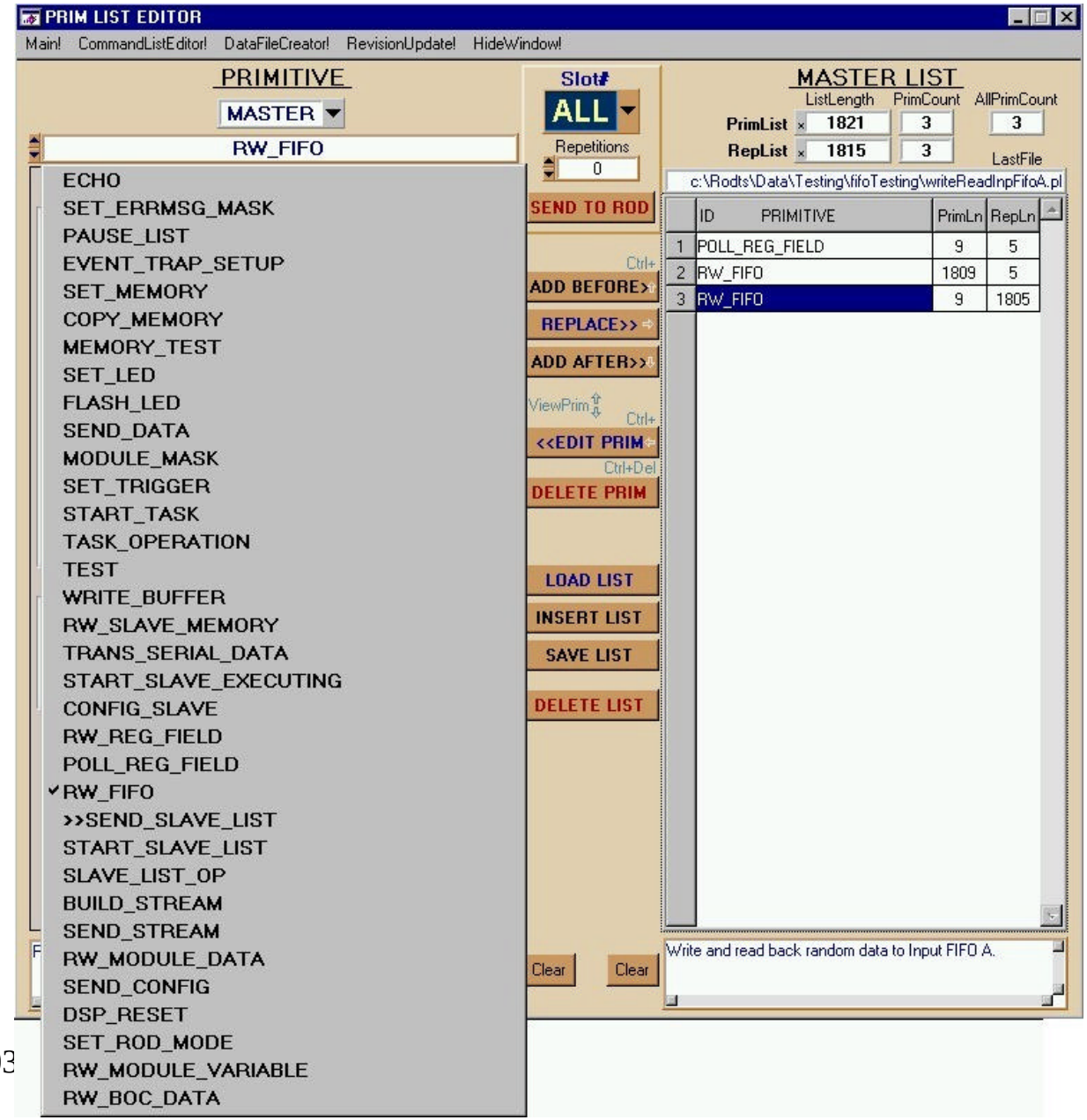
Primitive List

- **List of *primitives* executed by master/slave DSP; prepared and sent by Host.**

- The primitives, input and output data formats, communication registers and communication scheme are described in ***"Communication Protocol":***
  http://sctpixel.home.cern.ch/sctpixel/MessageProtocol/MessageProtocolV25.pdf

- The TestStand user can create different PrimitiveLists in the **PrimitiveList Editor** and save them in files for the next use or directly execute. The PrimitiveListEditor uses more complex version of the PrimitiveList (***PrimitiveListTable)***, since it has to contain information not only about the input data but also variable parametes for the reply data (save to file, print on screen etc.).

Primitive List Editor:



Primitive List Internal Structure (PrimListTable):

primListTableDefinitions.h

```
/*----------------------------- LIST TABLE DEFINITIONS ----------------------*/
#define MAXPRIMS_IN_LIST 100
#define COMMENT_LENGTH   600

struct PRIM_TABLE{
   char comment[COMMENT_LENGTH];  /* primitive comment */
   struct MSG_HEAD inPrimHeader;  /* primitive header */
   UINT32 replyLength;  /* predicted length of reply data */
   union PRIM_PARAMS_UNION params; /* primitive parameters */
};
struct LIST_TABLE{
   char comment[COMMENT_LENGTH];  /* list comment */
   UINT32 allPrimCount;    /* number of all primitives(master+slave) in list */
   UINT32 masterToTablePrimIndex[MAXPRIMS_IN_LIST];
         /* the array contains the real index in PrimTable for each master primitive;
            array index=master index */
   struct MSG_LIST_HEAD inListHeader; /* primitive list header */
   UINT32 replyLength;  /* predicted length of reply list */
   UINT32 replyCount;   /* number of prims with reply data */
   struct PRIM_TABLE_UNION primTable[MAXPRIMS_IN_LIST];
};
```

---

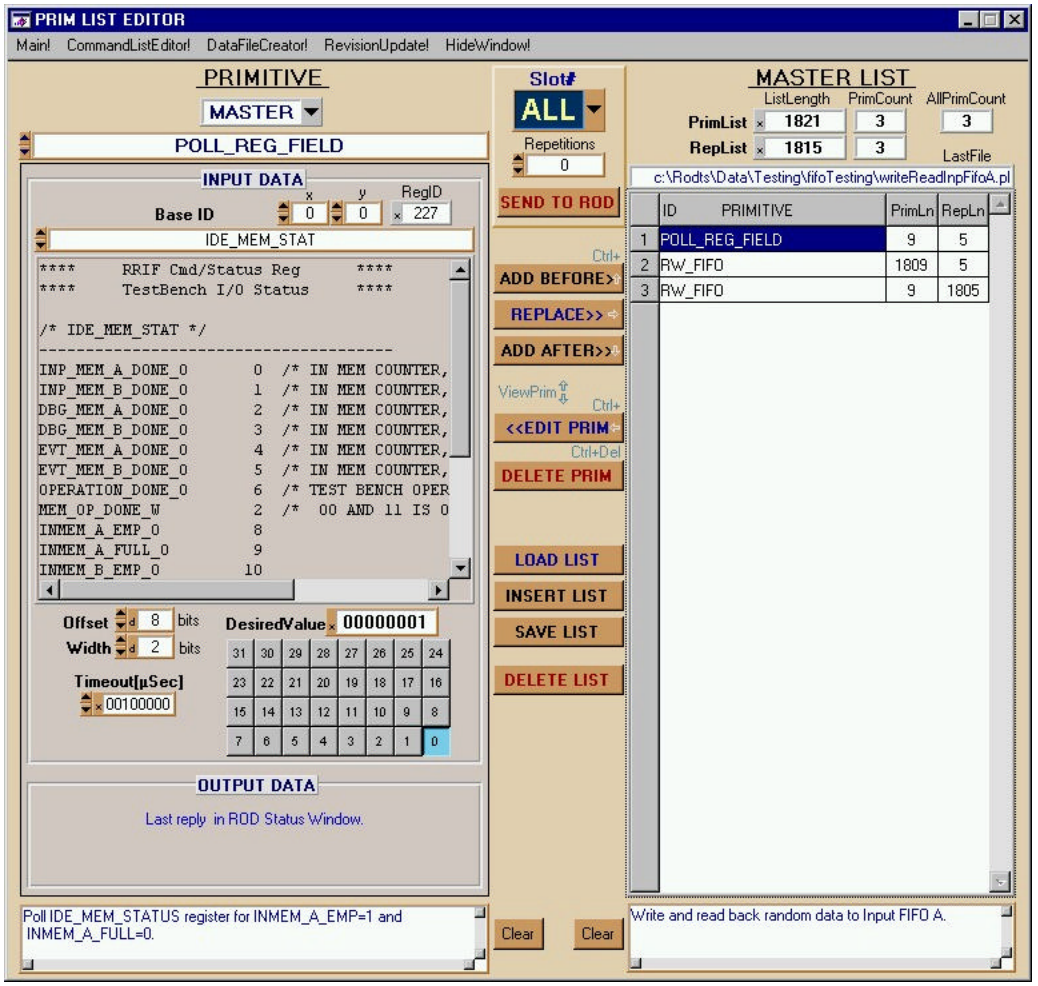
primFunc_RWFifo.h

```
/* RW_FIFO primitive params example - included in PRIM_PARAMS_UNION */

struct PRIM_RWFIFO_PARAMS{ /* params for host */
   struct RW_FIFO_IN inputParams;
   char dataFileName[PRIM_PATHNAME_LENGTH];
};
```

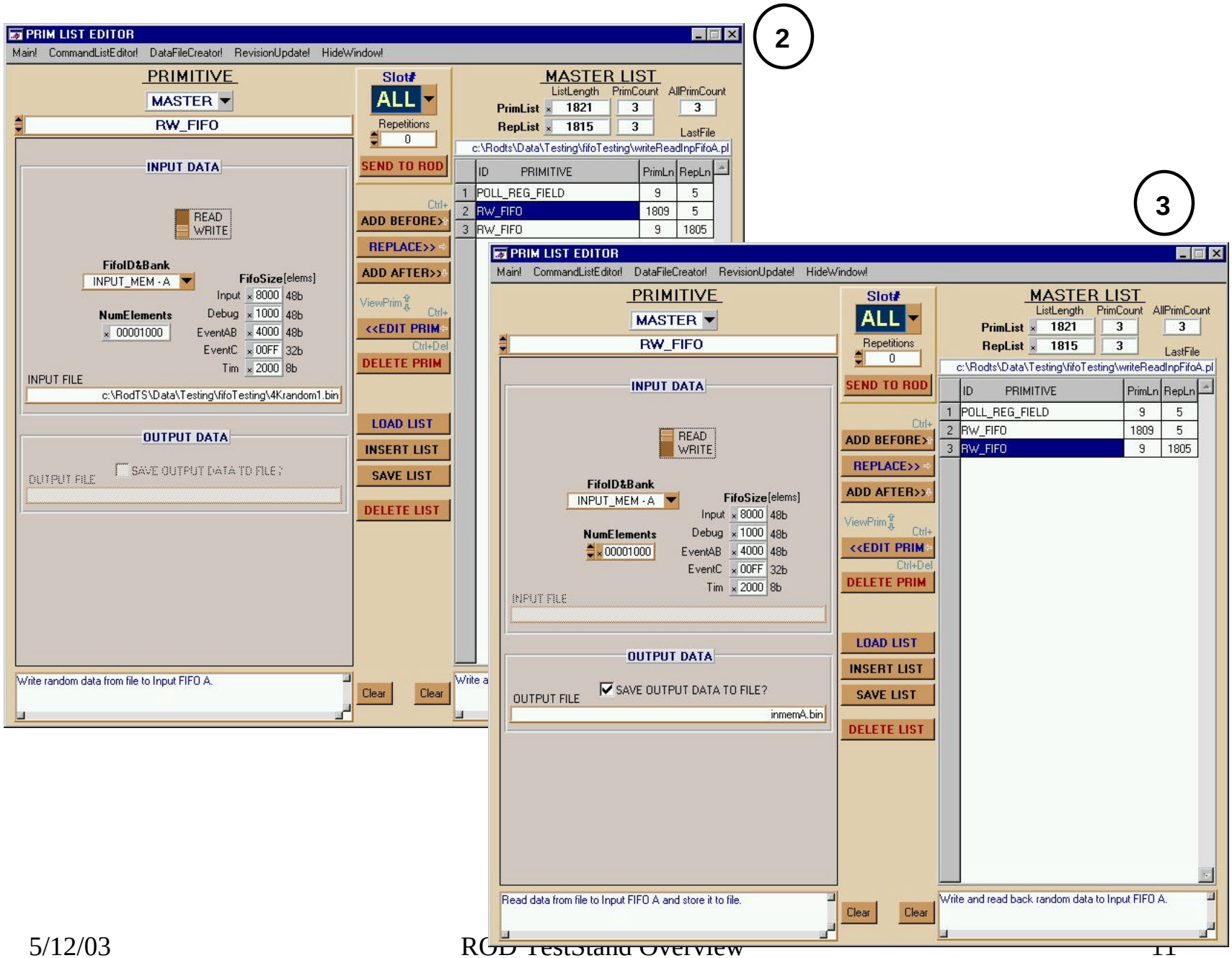
Note: The max. number of *primitives* in the *PrimListTable* is limited for simplicity and to avoid possible problems with dynamic memory allocation (current limit is 100).
However this number can be easily changed if necessary. The same applies to the comment length field “artificially” limited to 600 characters.

Primitive List example:



1

continued on next page


2
PRIM LIST EDITOR
Main! CommandListEditor! DataFileCreator! RevisionUpdate! HideWindow!
PRIMITIVE
MASTER
RW_FIFO
INPUT DATA
READ
WRITE
FifoID&Bank
INPUT_MEM - A
NumElements
00001000
FifoSize[elems]
Input 8000 48b
Debug 1000 48b
EventAB 4000 48b
EventC 00FF 32b
Tim 2000 8b
INPUT FILE
c:\RodTS\Data\Testing\fifoTesting\4Krandom1.bin
OUTPUT DATA
SAVE OUTPUT DATA TO FILE?
OUTPUT FILE
Write random data from file to Input FIFO A.
Slot#
ALL
Repetitions
0
SEND TO ROD
ADD BEFORE>>
REPLACE>>
ADD AFTER>>
ViewPrim
<<EDIT PRIM
DELETE PRIM
LOAD LIST
INSERT LIST
SAVE LIST
DELETE LIST
Clear
Clear
MASTER LIST
ListLength PrimCount AllPrimCount
PrimList 1821 3 3
RepList 1815 3
LastFile
c:\Rodts\Data\Testing\fifoTesting\writeReadInpFifoA.pl
ID PRIMITIVE PrimLn RepLn
1 POLL_REG_FIELD 9 5
2 RW_FIFO 1809 5
3 RW_FIFO 9 1805
3
PRIM LIST EDITOR
Main! CommandListEditor! DataFileCreator! RevisionUpdate! HideWindow!
PRIMITIVE
MASTER
RW_FIFO
INPUT DATA
READ
WRITE
FifoID&Bank
INPUT_MEM - A
NumElements
00001000
FifoSize[elems]
Input 8000 48b
Debug 1000 48b
EventAB 4000 48b
EventC 00FF 32b
Tim 2000 8b
INPUT FILE
OUTPUT DATA
SAVE OUTPUT DATA TO FILE?
OUTPUT FILE
inmemA.bin
Read data from file to Input FIFO A and store it to file.
Slot#
ALL
Repetitions
0
SEND TO ROD
ADD BEFORE>>
REPLACE>>
ADD AFTER>>
ViewPrim
<<EDIT PRIM
DELETE PRIM
LOAD LIST
INSERT LIST
SAVE LIST
DELETE LIST
Clear
Clear
MASTER LIST
ListLength PrimCount AllPrimCount
PrimList 1821 3 3
RepList 1815 3
LastFile
c:\Rodts\Data\Testing\fifoTesting\writeReadInpFifoA.pl
ID PRIMITIVE PrimLn RepLn
1 POLL_REG_FIELD 9 5
2 RW_FIFO 1809 5
3 RW_FIFO 9 1805
Write and read back random data to Input FIFO A.

Primitive List File (*.pl) example:

(created and saved using *PrimitiveListEditor*)

writeReadInpFifoA.pl

```
Write and read back random data to Input FIFO A.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
length: 0x1821, numMsgs= 0x3, primListRevision= 111
allPrimCount: 3, replyLength: 0x1815, replyCount: 0x3
============================================================
Poll IDE_MEM_STATUS register for INMEM_A_EMP=1 and
 INMEM_A_FULL=0.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
index: 0,  id: 0x2005, primRevision: 104, length: 0x9, replyLength: 0x5
POLL_REG_FIELD
registerID= 0x227, offset= 8, width= 2, desiredValue= 0x1, timeoutInUsec= 0x100000
baseId= 0x227, x= 0, y= 0
============================================================
Write random data from file to Input FIFO A.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
index: 1,  id: 0x2006, primRevision: 104, length: 0x1809, replyLength: 0x5
RW_FIFO
fifoId= 0, bank= 0, readNotWrite= 0, numElements= 0x1000, *dataBaseAdr= 0xFFFFFFFF
dataFileName[]= c:\RodTS\Data\Testing\fifoTesting\4Krandom1.bin
============================================================
Read data from file to Input FIFO A and store it to file.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
index: 2,  id: 0x2006, primRevision: 104, length: 0x9, replyLength: 0x1805
RW_FIFO
fifoId= 0, bank= 0, readNotWrite= 1, numElements= 0x1000, *dataBaseAdr= 0xFFFFFFFF
dataFileName[]= inmemA.bin
```

1st primitive

2nd primitive

3rd primitive

← *PrimList comment*

- PrimListTable header

← *primitive comment*

- primitive header
- primitive name
- primitive params

Note: The comments can be edited also outside the CommandList Editor using any text editor. Number of comment lines is not explicitly limited.

Command List

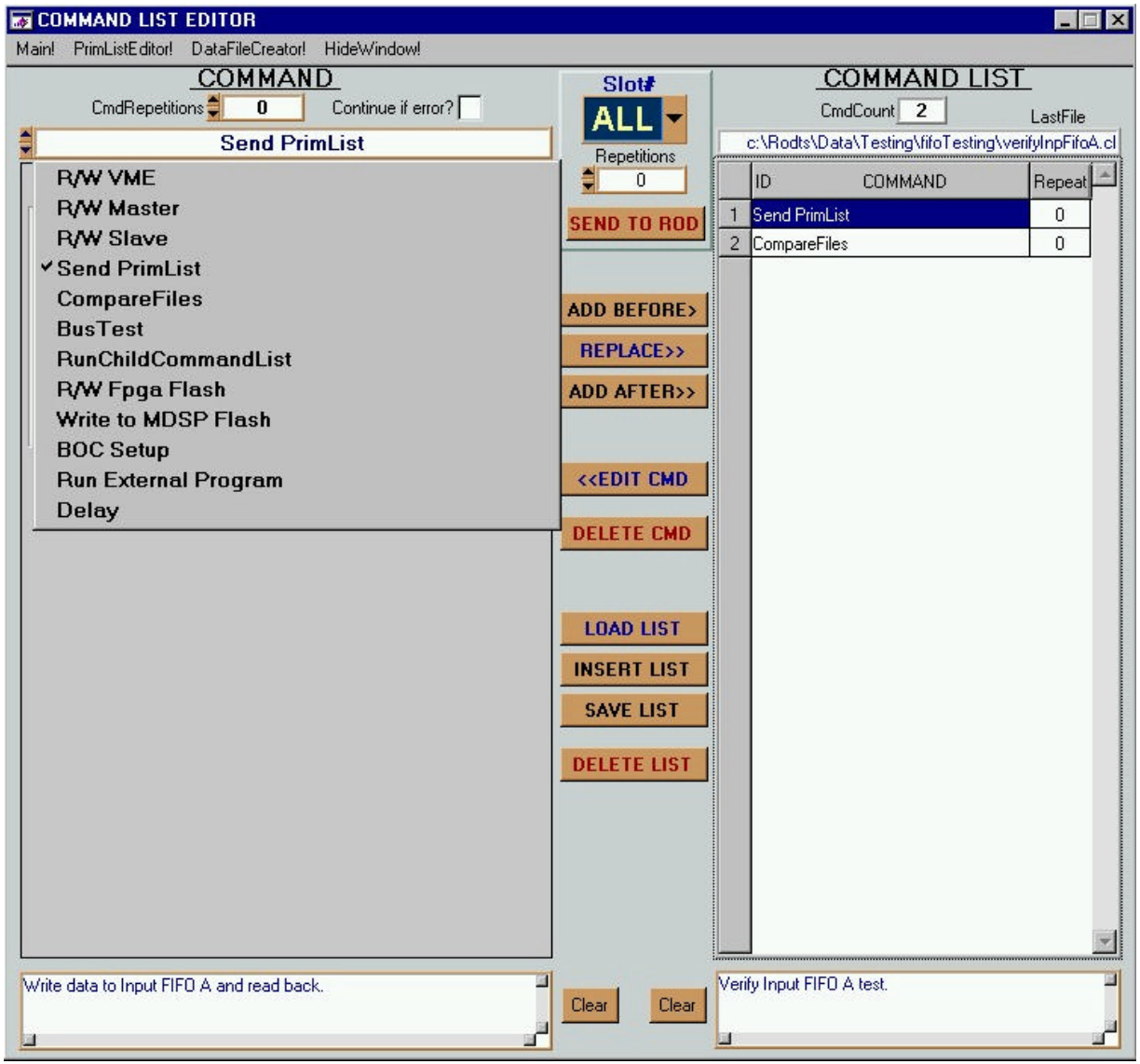
In general any Rod task is defined in the TestStand environment as a sequence of the basic commands executed by the host - ***CommandList***, which can be edited and saved in the interactive *CommandListEditor*. The implementation and the structure of the *CommandList* is very similar to the *PrimitiveList* (CommandList is the host parallel to the Rod PrimitiveList). => TestStand as a "list processor" is ready to run most of possible user defined tasks **without recompiling the code**.

Currently implemented commands:

- R/W Master DSP memory
- R/W Slave DSP memory
- **Send PrimitiveList**
- Compare Files
- Bus Test
- Run Child Command List (the parent command list can execute other command lists)
- R/W FPGA Flashes
- Write to Master DSP Flash
- BOC Setup (used temporarily, PrimitiveList should be used instead)
- Run External Program - user can (create and) call any executable (so far used mainly for data processing and conversion)
- Delay.

New commands can be easily added in the TestStand if the current set is not sufficient for some tests (for example if some variable primitive lists must be built "on the fly") or if it's useful to move more complicated and often repeated tasks into separate commands.
TestStand is also able to execute *CommandLists* and *PrimitiveLists* (in "raw" or *PrimListTable* format) sent from another application.

Command List Editor:



Command List Internal Structure:

commandListDefinitions.h

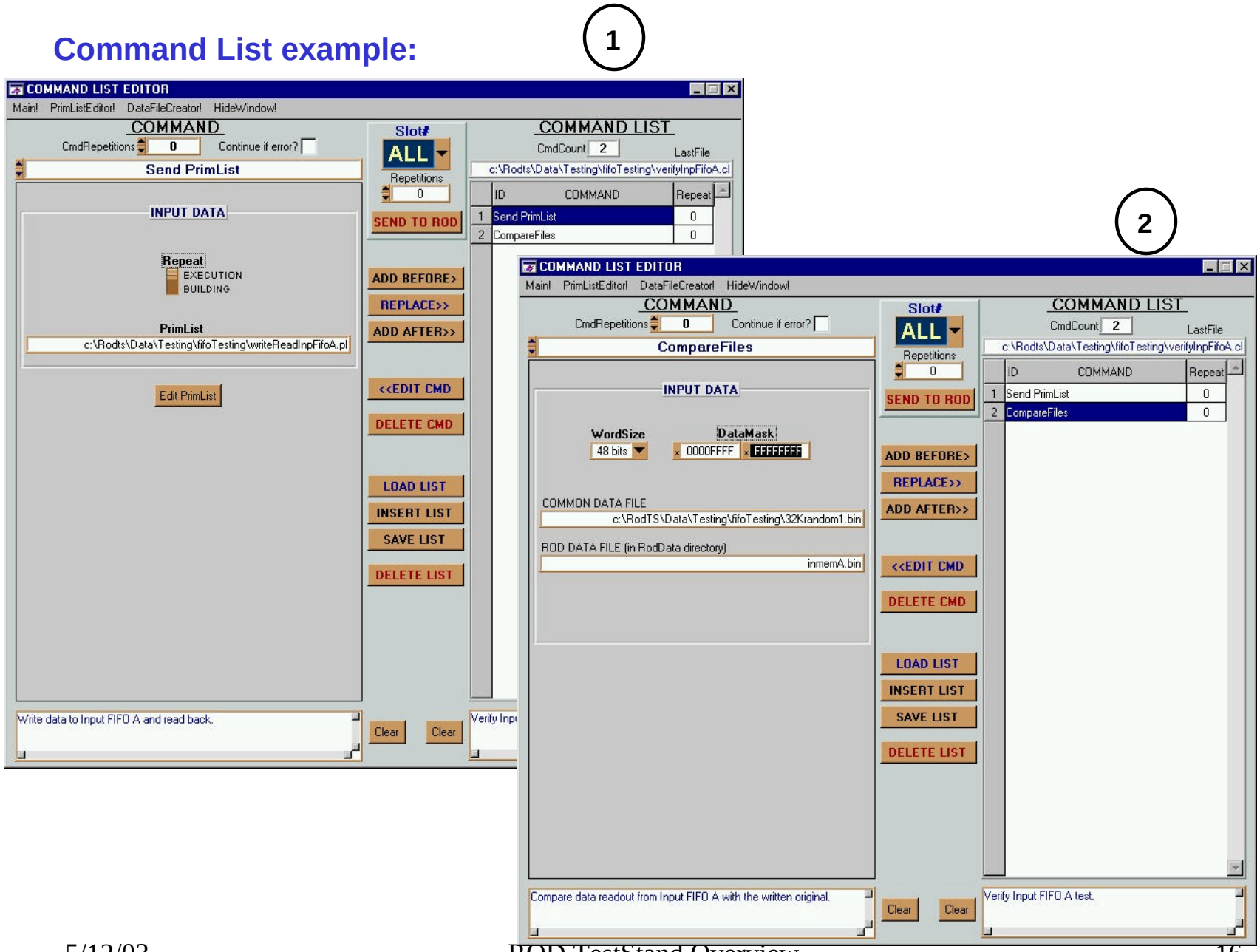
```
/*-------------------------- COMMAND LIST DEFINITIONS -------------------------*/
#define MAXCOMMANDS_IN_LIST 100
#define CMD_COMMENT_LENGTH  600

struct COMMAND{
   char comment[CMD_COMMENT_LENGTH];  /* command comment */
   int id;                          /* command id */
   UINT32 numRepetitions; /* number of command repetitions; 0 means execute once, don't repeat */
   int continueIfError;   /* if set, continue to the next command in the list even
                                                        if this command returned error  */
   union COMMAND_PARAMS_UNION params;   /* command parameters */
};
struct COMMAND_LIST{
   char comment[CMD_COMMENT_LENGTH];  /* cmd list comment */
   int commandCount;       /* number of commands in the list */
   struct COMMAND command[MAXCOMMANDS_IN_LIST]; /* array of commands */
};

-----------------------------------------------------------------------------
commandFunc_sendPrimList.h
/* SEND PRIM LIST command params example - included in COMMAND_PARAMS_UNION */
struct SEND_PRIM_LIST_PARAMS{
   int repeatExecNotBuild;  /* repeat only execution of the list not building */
   char fileName[PATHNAME_LENGTH]; /* primitive list file */
};
```

Note: Again, the number of *Commands* in the *CommadList* is limited (current max is 100), but can be easily redefined if necessary, as well as This number can be redefined if necessary as well as the comment array size (now max 600characters).

Command List example:



Command List File (*.cl) example:

(created and saved using *CommandListEditor*)

verifyInpFifoA.cl

```
Verify Input FIFO A test.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
commandCount: 2
============================================================
Write data to Input FIFO A and read back.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
index: 0,  id: 3, numRepetitions: 0, continueIfError: 0
SEND_PRIM_LIST_COMMAND_ID
repeatExecNotBuild= 1
fileName[]= c:\Rodts\Data\Testing\fifoTesting\writeReadInpFifoA.pl
============================================================
Compare data readout from Input FIFO A with the written original.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
index: 1,  id: 4, numRepetitions: 0, continueIfError: 0
COMPARE_FILES_COMMAND_ID
wordSizeInBytes= 6, dataMask[]={ 0xFFFFFFFF, 0xFFFF }
commonFile[]= c:\RodTS\Data\Testing\fifoTesting\32Krandom1.bin
rodFile[]= inmemA.bin
```

1st command

2nd command

← *CommandList comment*

- *CommandList header*

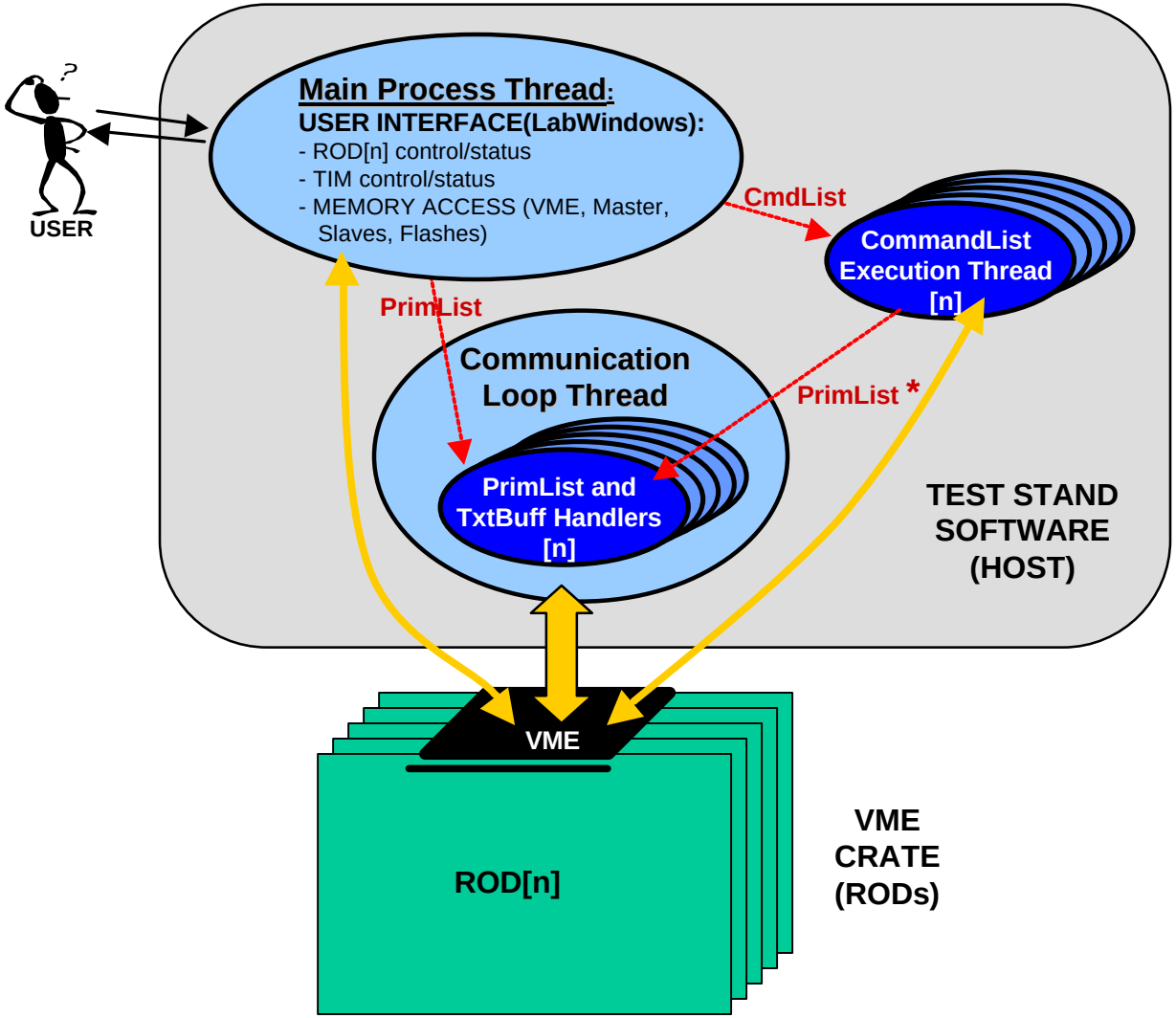
← *command comment*

- *command header*
- *command name*
- *command params*

Note: The comments can be edited also outside the CommandList Editor using any text editor. Number of comment lines is not explicitly limited.

Test Stand Structure

• ***Single process multithreaded*** application:



[n] .... Rod index {0,1…15}

* Commands with VME access are synchronized/executed via CommunicationLoop.

Threads

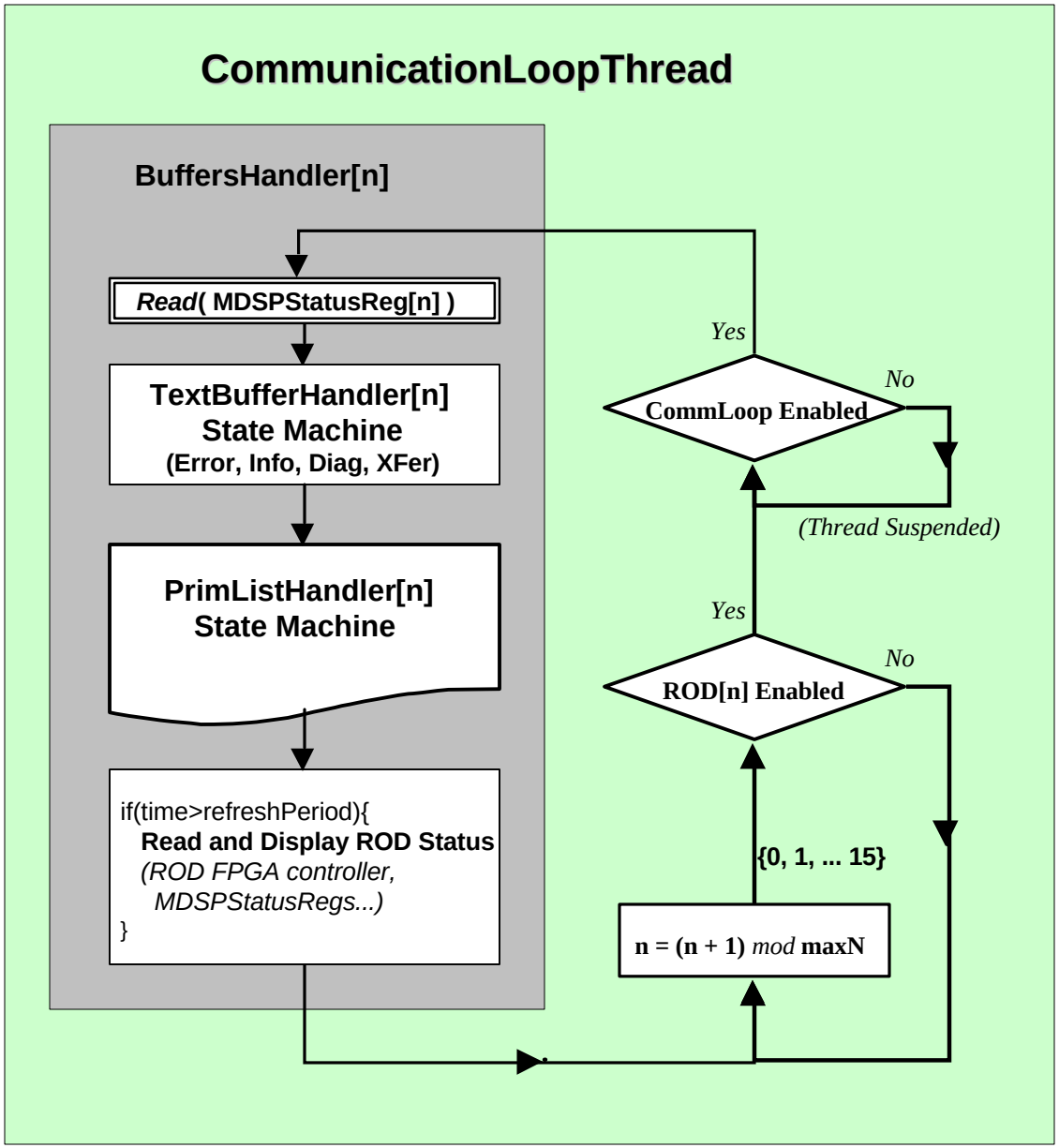
- **Main Thread**

  - **interaction with the User** (LabWindows GUI),

    - Main Panel (program options, thread priorities...),
    - RW Memory Panels(VME, MDSP, SDSPs, Flashes, FPGA Control/Status),
    - PrimList Editor, CommandList Editor,
    - ROD Status/Control Panel;

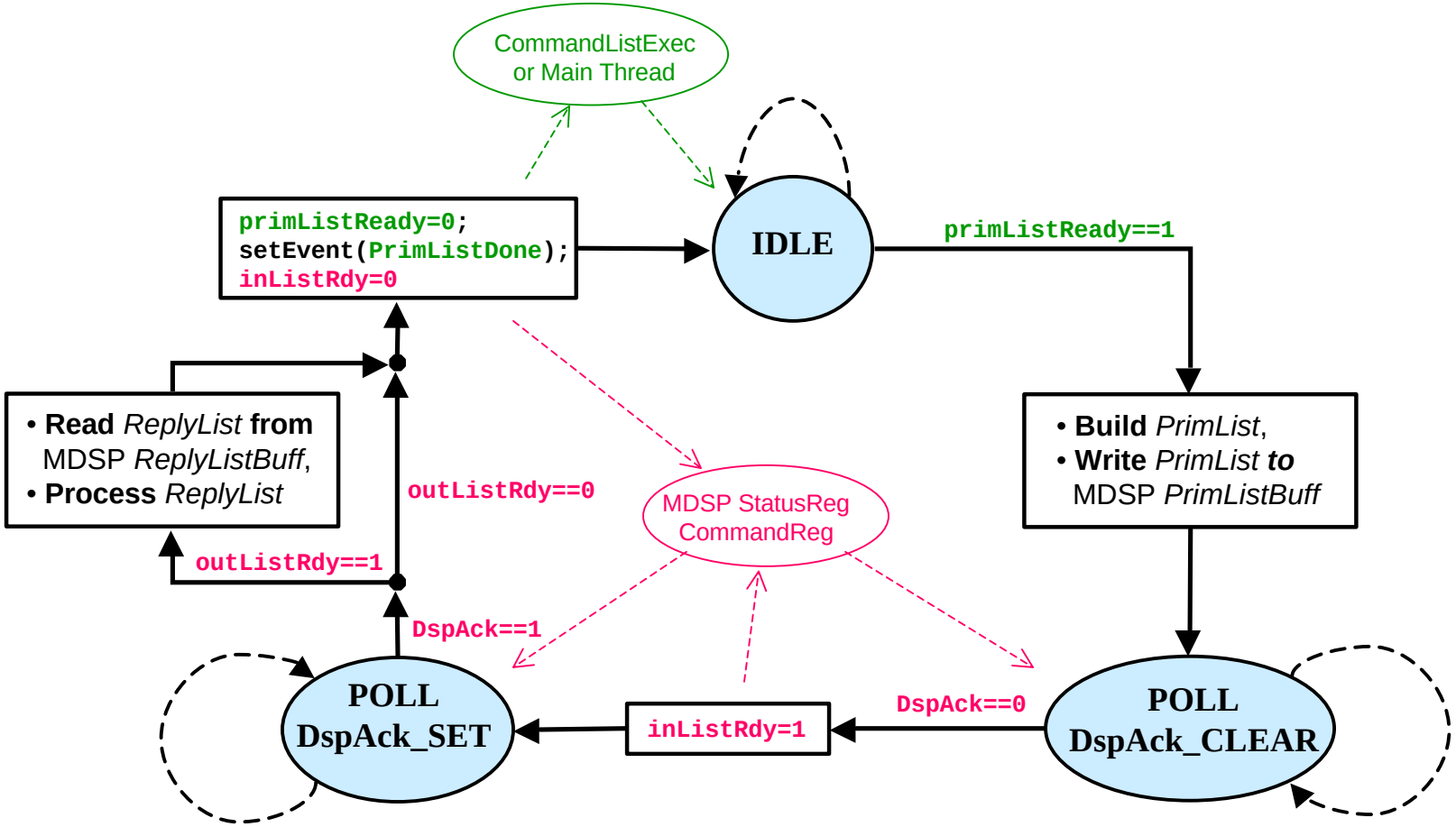
- **CmdListExecution Thread[n]** - *CommandList* execution.

- **CommunicationLoop**
  - common to all RODs,
  - distributes an access to the shared VME bus between all RODs in the crate on ***"round robin"*** basis,
  - executes PrimitiveLists and processes TextBuffer messages;
  - periodically monitors the status of all Rods in the crate.


CommunicationLoopThread
BuffersHandler[n]
Read(MDSPStatusReg[n])
TextBufferHandler[n]
State Machine
(Error, Info, Diag, XFer)
PrimListHandler[n]
State Machine
if(time>refreshPeriod){
Read and Display ROD Status
(ROD FPGA controller,
MDSPStatusRegs...)
}
Yes
No
CommLoop Enabled
(Thread Suspended)
Yes
No
ROD[n] Enabled
{0, 1, ... 15}
n = (n + 1) mod maxN

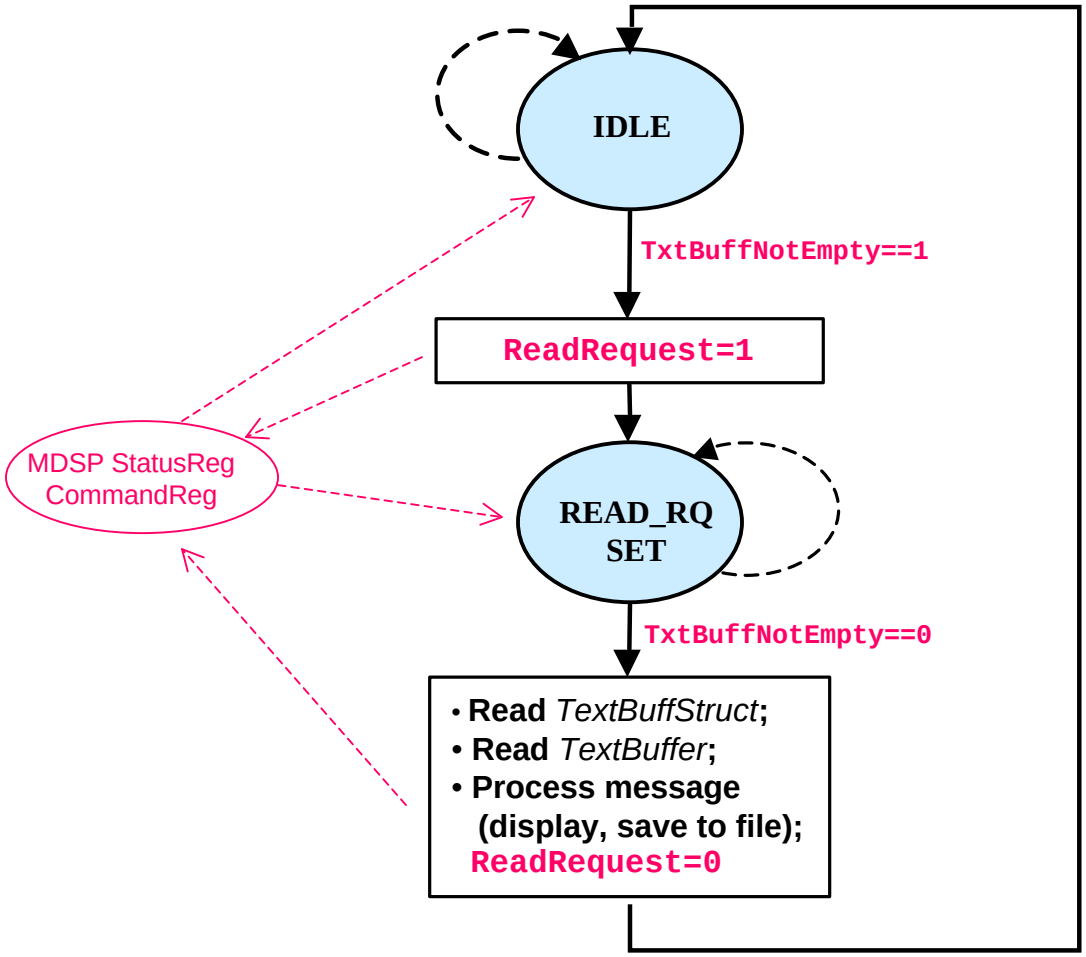
PrimListHandler[n] State Machine

- executes *PrimititiveListTable* (builds *PrimitiveList*, send it to Rod, processes *ReplyData*).



TextBufferHandler[n] State Machine

- reads and processes Text Messages from MasterDSP (*Error*, *Info*, *Diagnostic*, *XFer*).



Thread priorities

- ***MainProcessThread(UserInterface) > the other threads***

  - the user interface must be responsive at any moment!!

- ***CommunicationThread <= CmdListExecutonThreads***

  When CommunicationLoop is not suspended, the processor is always 100% busy (no additional delays included).

  The *CmdListExecutonThreads* are active only for a limited period of time when executing a *CommandList*, so they can have a higher priority than the *CommunicationLoop*.