ROD TestStand User Guide

http://www-hep.fzu.cz/~tomasekl/RODTestStandUserGuide.doc

Lukas Tomasek <tomasekl@fzu.cz>
Jan Valenta <valenta@fzu.cz>

July 18, 2003

Contents:

1. Introduction

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

1.1 System requirements

Hardware: PC with NI PCI-MXI-2 interface card (and NIVXI-2 driver installed), VME crate with NI VME-MXI-2 card, Atlas ROD cards;
Operating system: Windows 9X, NT, 2000, XP;
Developer software: National Instruments LabWindows CVI/5.5 (Measurement Studio).

1.2 Installation

Install the whole RodTS directory on the C drive of your computer, if possible (i.e. C:\RodTS). Include A32 VME address range occupied by RODs (i.e. 0x05000000 – 0x15FFFFFF for slots 5 – 21) into your NI-VXI database using T&M Explorer.

Although the TestStand is designed to be installed virtually anywhere on your computer, on any drive and under any subdirectory, the command and primitive list files supplied with the TestStand contain the absolute paths to all input data files (usually C:\RodTS\Data). Otherwise, the lists must be converted to point to your actual directory. (This "feature" can be changed in the future so that all file paths will be relative inside the RodTS directory. It'd also mean all input data files being restricted to RodTS subdirectories.)

1.3 Starting the TestStand

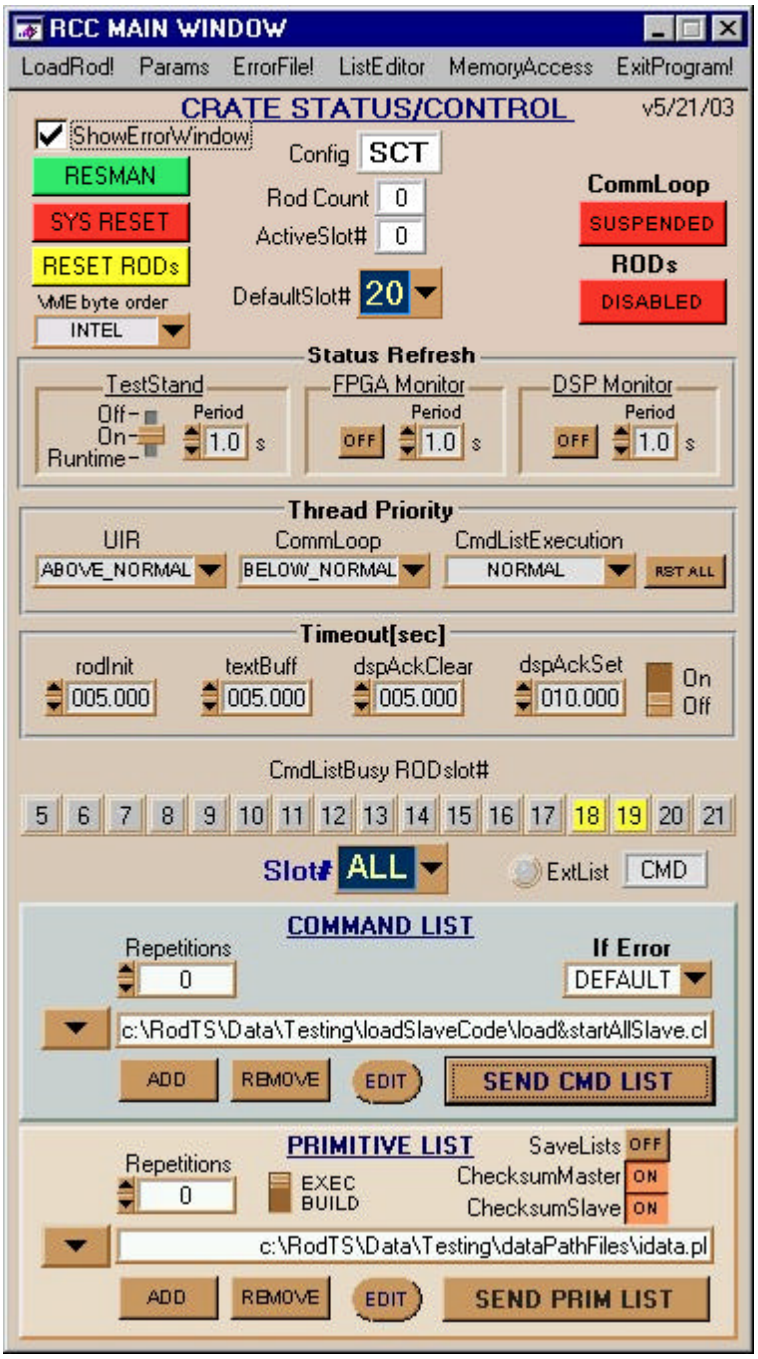
Start executable **RodTS\Rcc\rcc_SCT.exe** or **rcc_PIXEL.exe** (SCT or Pixel ROD version).

A popup window **"Run RESMAN and InitVXIlibrary()?"** will appear. Confirm if want to communicate with the RODs and have all necessary hardware ready → VXI Resource Manager will be executed and VXI library initialized. Choose "No" if you only want to edit primitive and command lists on a computer without VME access (you can also run Resman and initialize VXI libraries anytime later, clicking on RESMAN button on the main TestStand panel).

Next dialog box **"Delete ErrorFile and all RodData files?"** gives you an option to delete all status files from the previous TestStand run - the *ErrorFile* (*RodTS\Data\ErrorFile.txt*) and all Rod output data binary files, text buffer files, command list status files etc. (in *RodTS\Data\Rod\Slot#* directories). If these "old" files aren't deleted, all new error and status text messages (ErrorFile, TextBuffer files, CommandList status) will be simply appended at the end.

Then the main panel is opened:

2. Main Panel



This is the TestStand global control/status window. It gives you the access to all TestStand functionality and the other function panels.

Main Panel menu:

- **LoadRod** *(see chapter 4.)*

- **Params**
  - Save list rings (saves current set of default primitive and command lists inside  buttons to a cfg text file)
  - Load list rings (load set of default lists from a cfg file to  buttons)

- **Error File** – opens the Error File *(RodTS\Data\ErrorFile\CommonErrorFile.txt)*, which collects all error messages encountered during TestStand operation (error type and the location in the source code by file name and line number). Provide this file if having trouble with TestStand functionality.

- **List Editor**
  - Primitive List *(chapter 5.)*
  - Command List *(chapter 6.)*

- **Memory Access**
  - TIM (opens the TIM panel with basic R/W access to Tim registers)
  - VME *(chapter 3.3)*
  - Master DSP *(chapter 3.4)*
  - Slave DSP *(chapter 3.5)*
  - FPGA Status/Control Regs. *(chapter 3.2)*
  - Flash (*chapter 3.1)*
  - View Bin File - opens the external binary viewer Witched installed in *RodTS\Witched* directory (this tool gives you the option to watch binary file in hex, binary or text representation, compare it with another file etc.)

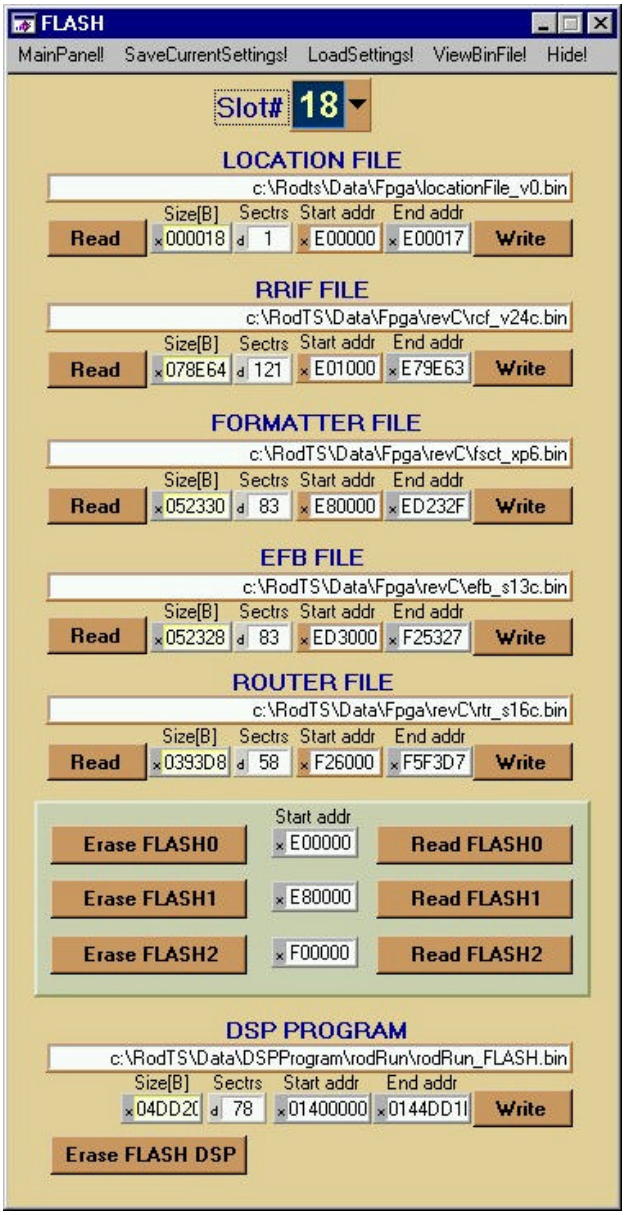
- **Help** – opens the directory *RodTS\TestStandUserGuide* with TestStand documentation (this User Guide, Overview …)

- **Exit** – program exit (Note: The standard Windows "kill" cross button doesn't work on Windows 9X, NT and 2000)

X) More Main Panel details later …..

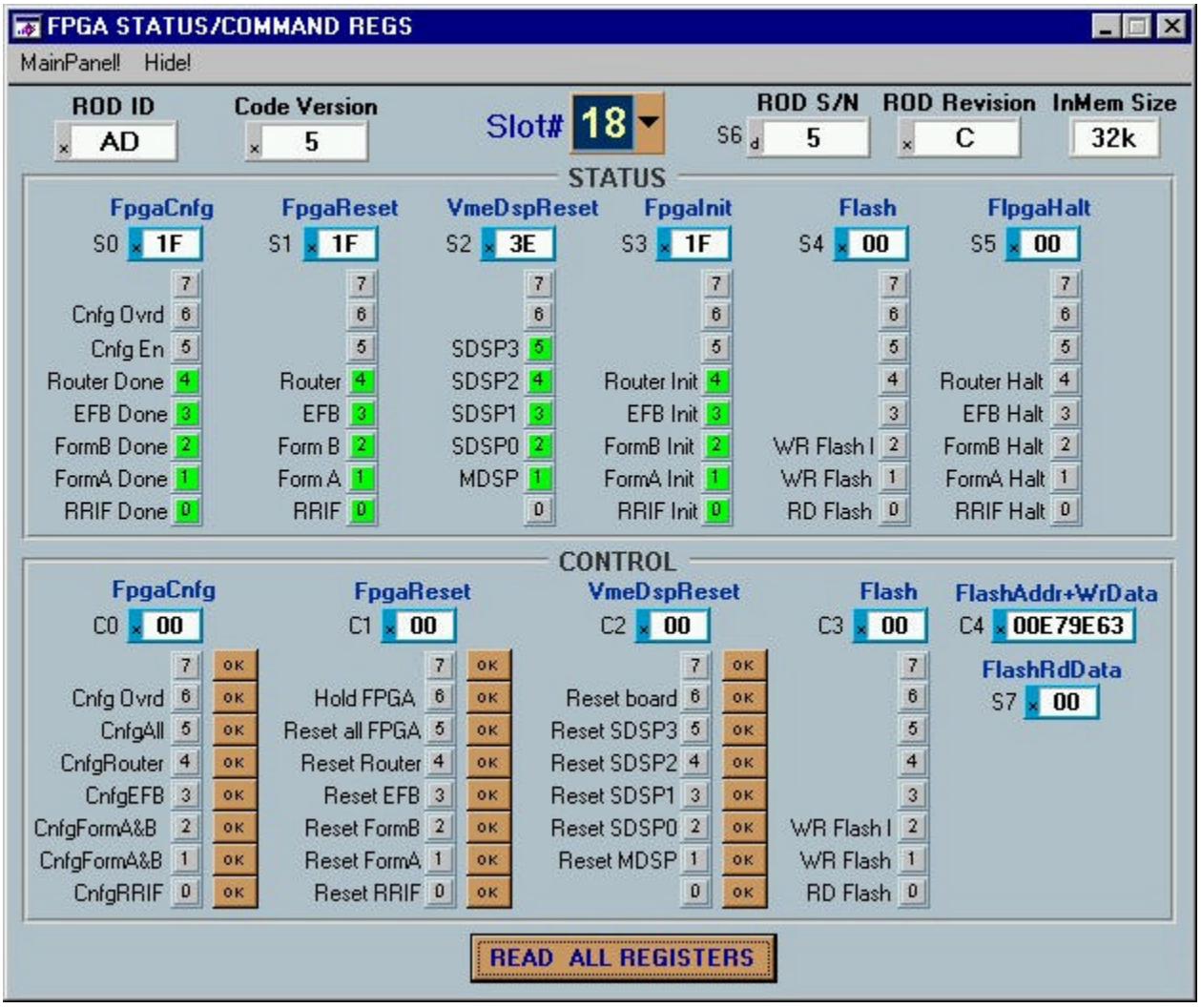
3. Memory Panels

3.1 FLASH Panel



This panel is a utility for writing and reading all Flash memories on the Rod. The Flashes 0, 1 and 2 contain FPGA code, the Master DSP code is loaded in the DSP Flash. To write the selected file to Flash, press **WRITE** button (click on the file name box to select another file). Read Flash content using **READ** buttons (and enter the file name for storage). You can also erase flashes by **Erase FLASHx**. If you want to read Masted DSP Flash, do it from the *Master DSP Memory Panel* (chapter 3.4).
To save and retrieve different flash configurations, use **SaveCurrentSettings** and **LoadSettings** from the panel menu bar.
Note: Writing and reading flashes can be also done via *CommandList* (chapter 6.).

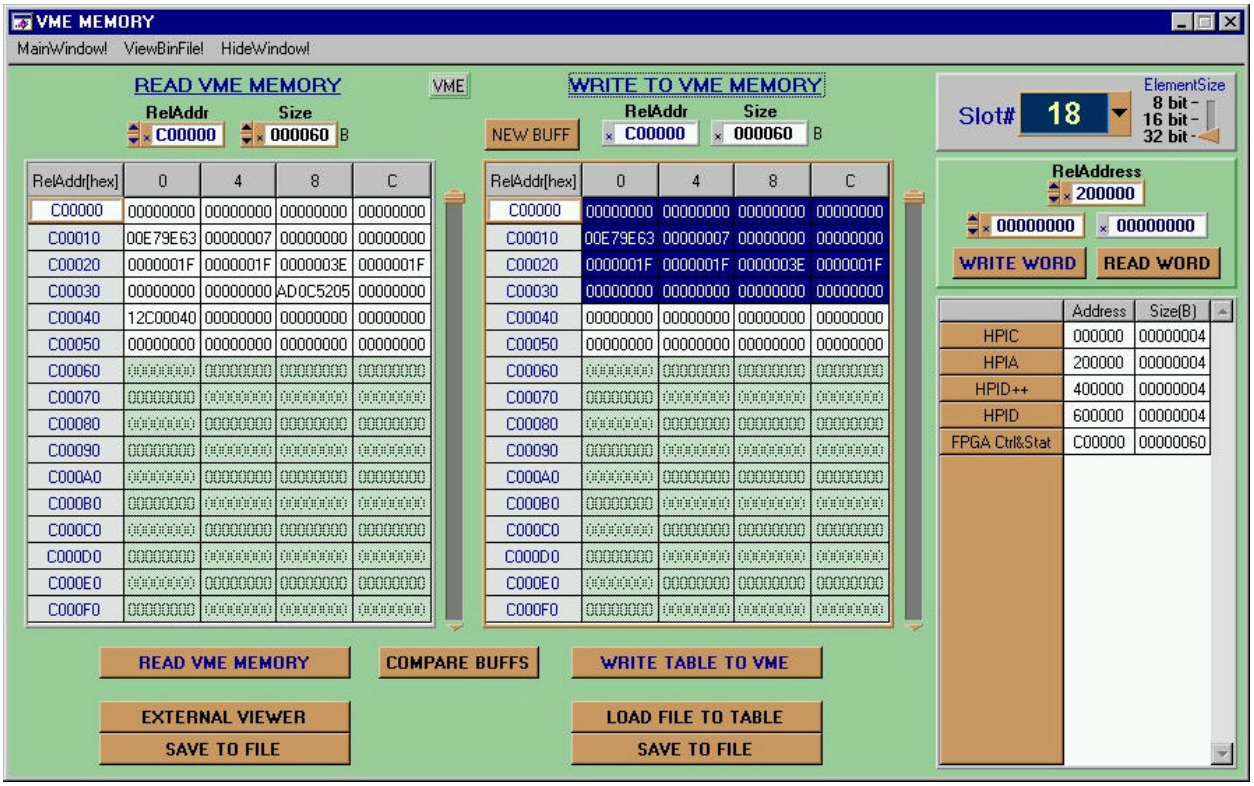
3.2 FPGA Program/Reset Panel



This panel provides basic FPGA status information - press **READ ALL REGISTERS** button to refresh it. The control registers bits are used to reset and configure main functional parts of the Rod (pressing OK button sets immediately this bit in the appropriate register on the Rod).

Note: You can optionally use VME Memory Panel (chapter 3.3) or *CommandList* with the R/W VME commands instead (chapter 6.).

3.3 VME Memory Panel



This panel can be used for simple read/write access to any VME memory space on Rod or any VME accessible device in the crate. You can read or write one register using the “word” controls in the upper right corner below slot number or memory block using block VME transfer (left table and controls for readout, right table and controls for writing). The table on the right side of the panel contains the default memory locations – name, address and size for each.

For any type of access first select the slot number in the upper right corner and element size (ROD uses 32bit wide words).

Register Access

Edit the address (i.e. relative lower 24 bits, the upper 8 bits are set by slot number) and press **READ WORD** button to read the data from register (the retrieved value is printed in the control above the button). To write value to register, edit the data field to be written and press **WRITE WORD** button.

If you want to copy any address from the default “address table”, select the address value and copy it to any address control using standard Windows copy (***Ctrl+C***) and paste (***Ctrl+V***) functions.

Block Read

Edit manually the desired starting address and the block size in bytes or use predefined values from the address table. Just double click on the desired field, the address and size values will be

automatically copied to the appropriate fields. Then press the **READ VME MEMORY**, the readout data will be loaded into the table. If the data block is larger than the table size, you can scroll up or down using the slider on the right side (pressing the up or down "arrows" will move just one page). Or edit the address you want to see in the first address box (address just below Addr[hex] label) and press enter. For a larger block of data it's more convenient to open the data in the external binary viewer Witched – use **EXTERNAL VIEWER** button.
The loaded data can be saved with **SAVE TO FILE** button.

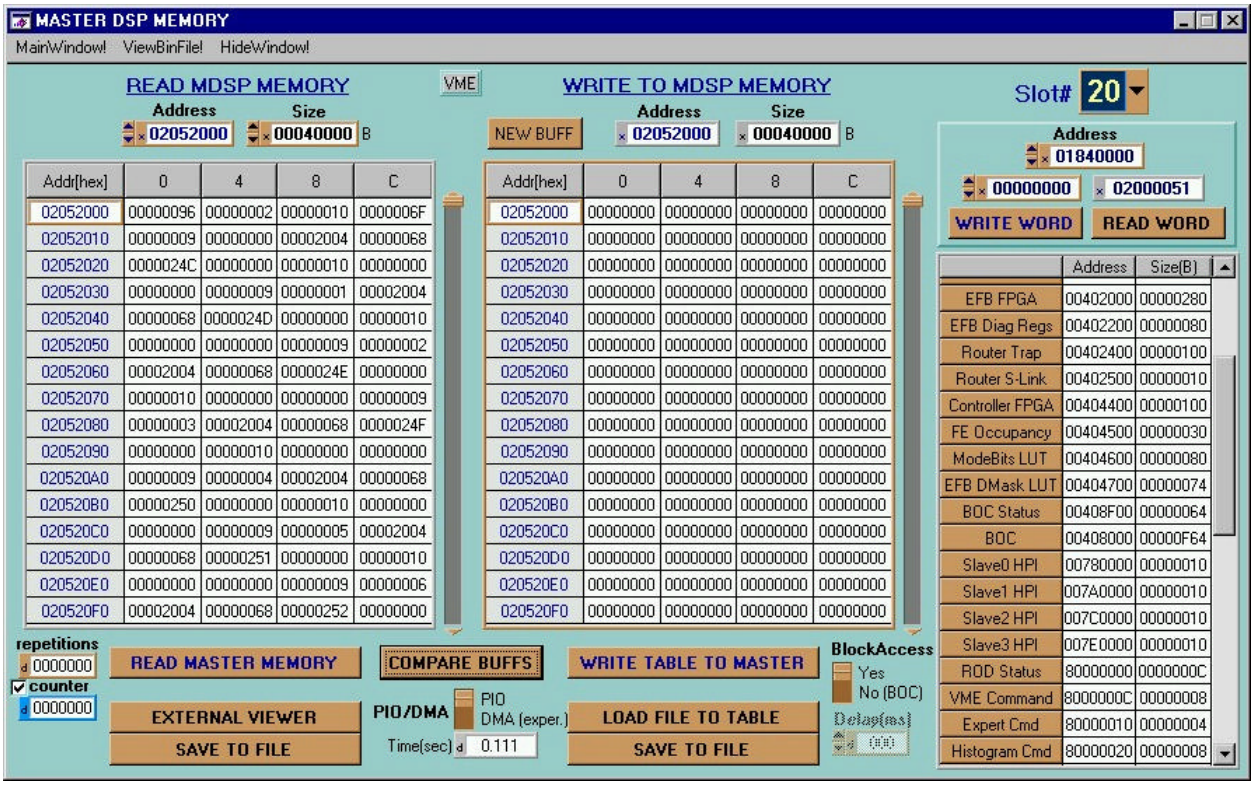
Block write

First you have to load the "write table" with data to be written.
Press **NEW BUFF**, the window where you can edit the desired starting address, block length and default value for all words will appear. If you want to select the default address and length from the address table, double click on the desired address field BEFORE clicking on **NEW BUFF**.
Or you can load the table with data from any binary file using **LOAD FILE TO TABLE** button instead.
After the memory buffer was initially loaded, you can edit any data word in the table (double click on the word you want to edit and change the value). You can also use standard copy and paste functions (***CTRL+C, CTRL+V***). To copy a block of data from the read table to write table select the data block you want to copy, ***Ctrl+C***, select a BLOCK (not only first word!) where you want to paste it, ***Ctrl+V***.
To save the data from table to binary file use **SAVE TO FILE** button (i.e. the write table can be used as a simple binary editor*****).
Whenever you want to write data from the table to the VME destination press **WRITE TABLE TO VME**.

The **COMPARE BUFFS** button quickly compares the data *word by word* in the read and write table (both buffers must be the same size!).
Optionally you can use Witched viewer for file comparisons.

*****_Note:_ Another binary file editor utility is also attached (*RodTS\DataFileCreator\ DataFileCreator.exe;* can be also opened directly from PrimListEditor and CommandListEditor menus).

3.4 Master DSP Memory Panel

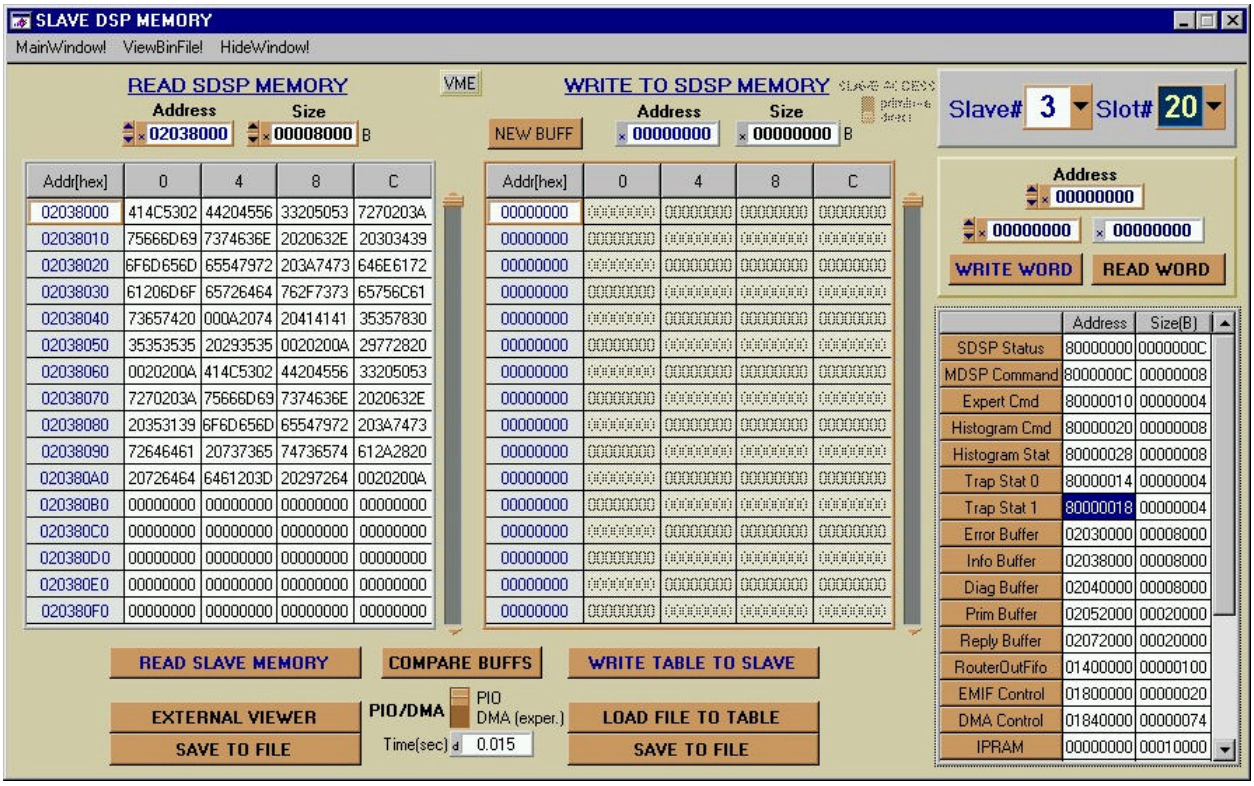


Access to Master DSP memory. Similar user interface to VME panel, but addressing is within the Master DSP memory space.

Additions:

Time[sec] box - the last block R/W access time to MasterDSP memory (in seconds).
Repetitions - any block transfer can be repeated more times (the number in this box means additional repetitions, so 0 reads or writes block once, 1 twice etc.), the counter update can be disabled (for timing measurements).
PIO/DMA option - selection between programmed I/O and DMA access to VME (DMA access is not reliable at this moment!!).
Block Access option "**Yes**"- MasterDSP memory space accessed by fast VME block transfers.
Block Access option "**No**"- the data block from the table is written to MasterDSP memory space "*word by word*" with additional delay between words (e.g. setting BOC registers).

3.5 Slave DSP Memory Panel

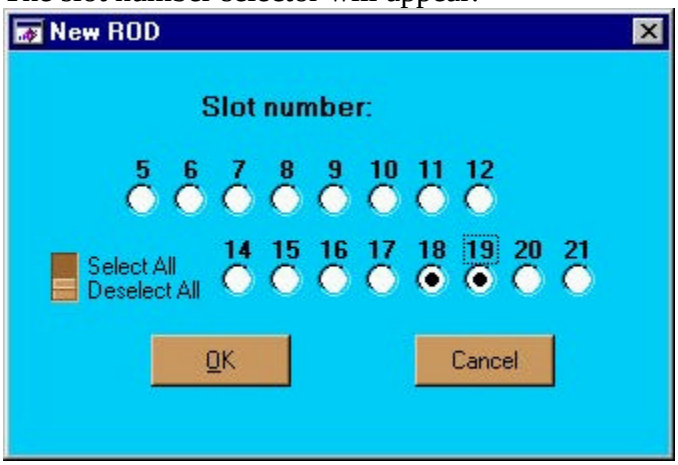


Similar to the Master DSP Panel, but the memory space is within the selected Slave DSP memory (you have to select the slave number 0 to 3).

Note: If the slave is running (i.e. the program has been loaded and started) and there is any inter MasterDSP-SlaveDSP communication going on, you can access the slave memory using this panel only **AFTER** the Master-Slave communication has been stopped by setting *DmaRequest* bit in MasterDSP *CommandRegister* (from ROD Panel or using primitive *ConfigSlave*)!

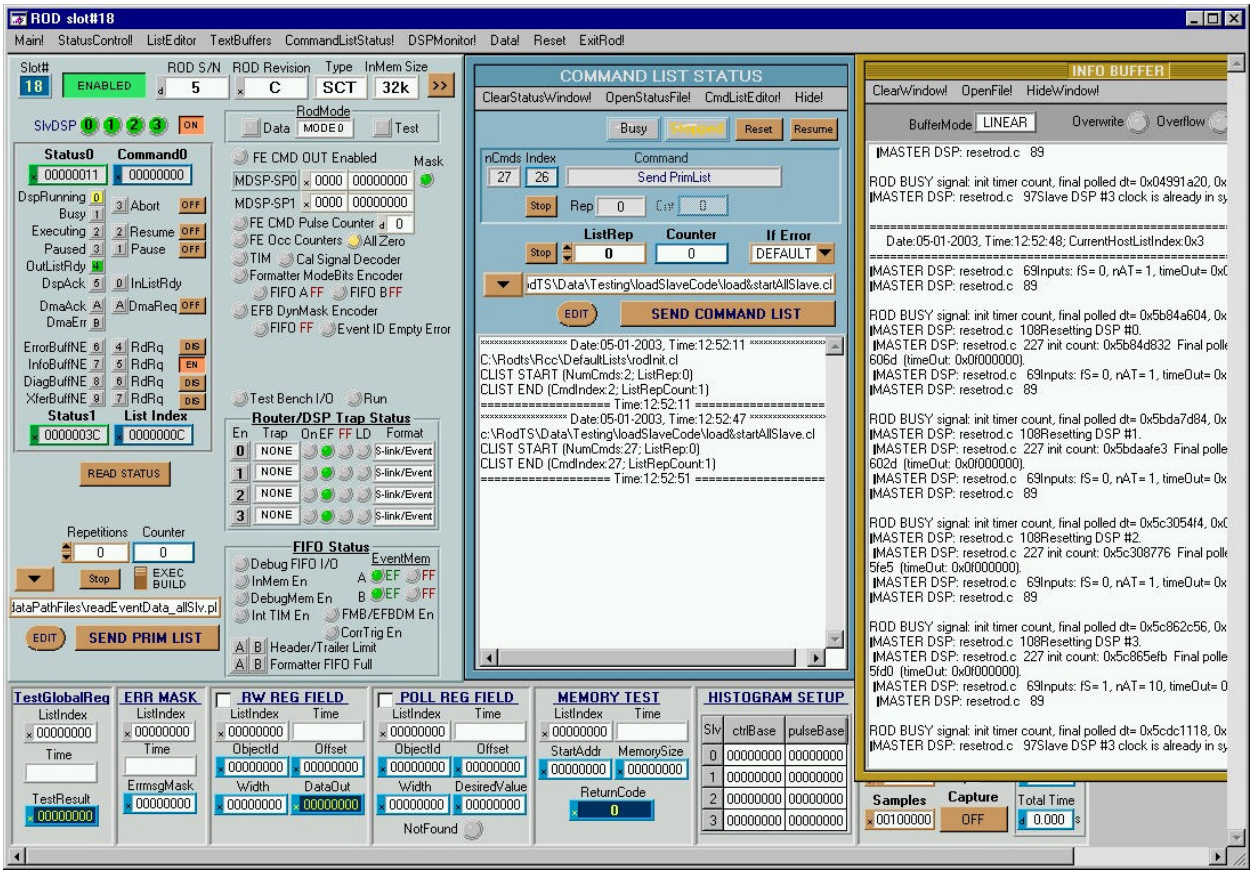
4. ROD[n] Panel

If you want to control any Rod using command and primitive lists and periodically check its' status, the Rod must be loaded into TestStand and started.
Load the *Rod status/control window* using **LoadRod** menu item in the *Main Panel*.
The slot number selector will appear:



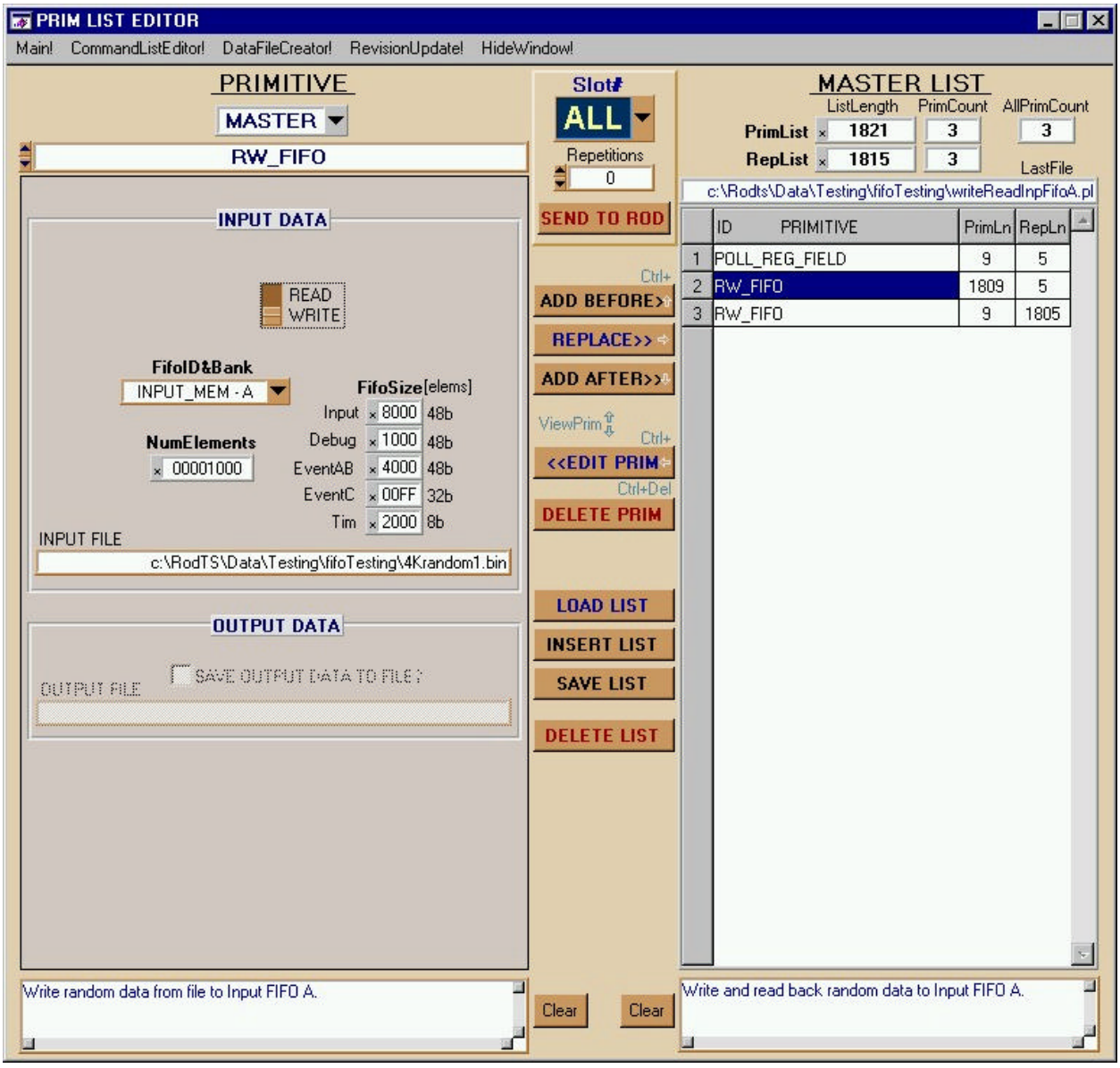
Select all slots you want to use (naturally, the Rod board should be loaded in the corresponding slot). A new Rod can be added to the system anytime, you don't have to load all at once.

For each selected slot the *RODStatus/Control window* is loaded:



XX) More info about Rod Panel later …..

5. Primitive List Editor



PrimitiveList Editor menu:

- **Main** - moves the *Main Panel* to the front.
- **Command List Editor** - opens *Command List Editor*.
- **Data File Creator** - opens *Data File Creator*.
- **Revision Update** – Converts old saved primitive lists (i.e. lists with outdated list or primitive revision number) to the new revision. Just select the directory with the lists for conversion (usually C:\RodTS) and all lists in this directory and all SUBDIRECTORIES will be automatically converted. After the conversion the status report file

PrimListRevisionUpdate_ERRORS.txt will appear. This text file lists only the files that couldn't be updated to the new revision and reason why. More detailed information about the last conversion is also available in *PrimListRevisionUpdate_STATUS.txt* (revision changes for each converted list). Both files are saved in *RodTS\Rcc\PrimListRevisionUpdate* directory.

Primitive List Editing

The right table on the PrimListEditor window is the edited *PrimitiveList*. The panel on the left shows the parameters for the selected primitive.

To add primitive to the list, select the desired primitive in the popup box first and then set the parameters for this primitive. A comment for this primitive can be entered into the bottom left text box (Enter = new line).
Then click (only **ONCE**!!) in the list window on the position (index) where you want to include the primitive. The position is highlighted by blue background. Add the primitive to the list using **ADD BEFORE, REPLACE, ADD AFTER** buttons (or use shortcut keys ***Ctrl+*** - ® ¯ instead).

To see the parameters for any primitive already included in the list, select the required primitive (one click on its' position in the list) and then press **EDIT** button (or ***Ctrl+*** ¬)**,** or better yet just simply **double click on the primitive** instead.
Another way to "look through" the list is using the keyboard up and down arrows (*View prim* - ¯). Set focus to the list table (click on the list) and then scroll up or down using - ¯ keys, the parameter window on the left side is updated automatically for the currently specified primitive.

To include a slave list into the master list, select primitive **SEND_SLAVE_LIST** , then the slave number, add this primitive to the list and then click on the **EDIT SLAVE LIST** button. At this moment the right list table shows only the included slave list. When your slave list is done, press **DONE** button to return back to the main Master DSP list.

To delete primitive from the list, use **DELETE PRIM** button (**one** strong click or shortcut key *Ctrl+Delete*), to delete the list and clear the table **DELETE LIST**.

Before you save the list by **SAVE LIST** button, enter the list comment to the text box below the list (Enter = new line).

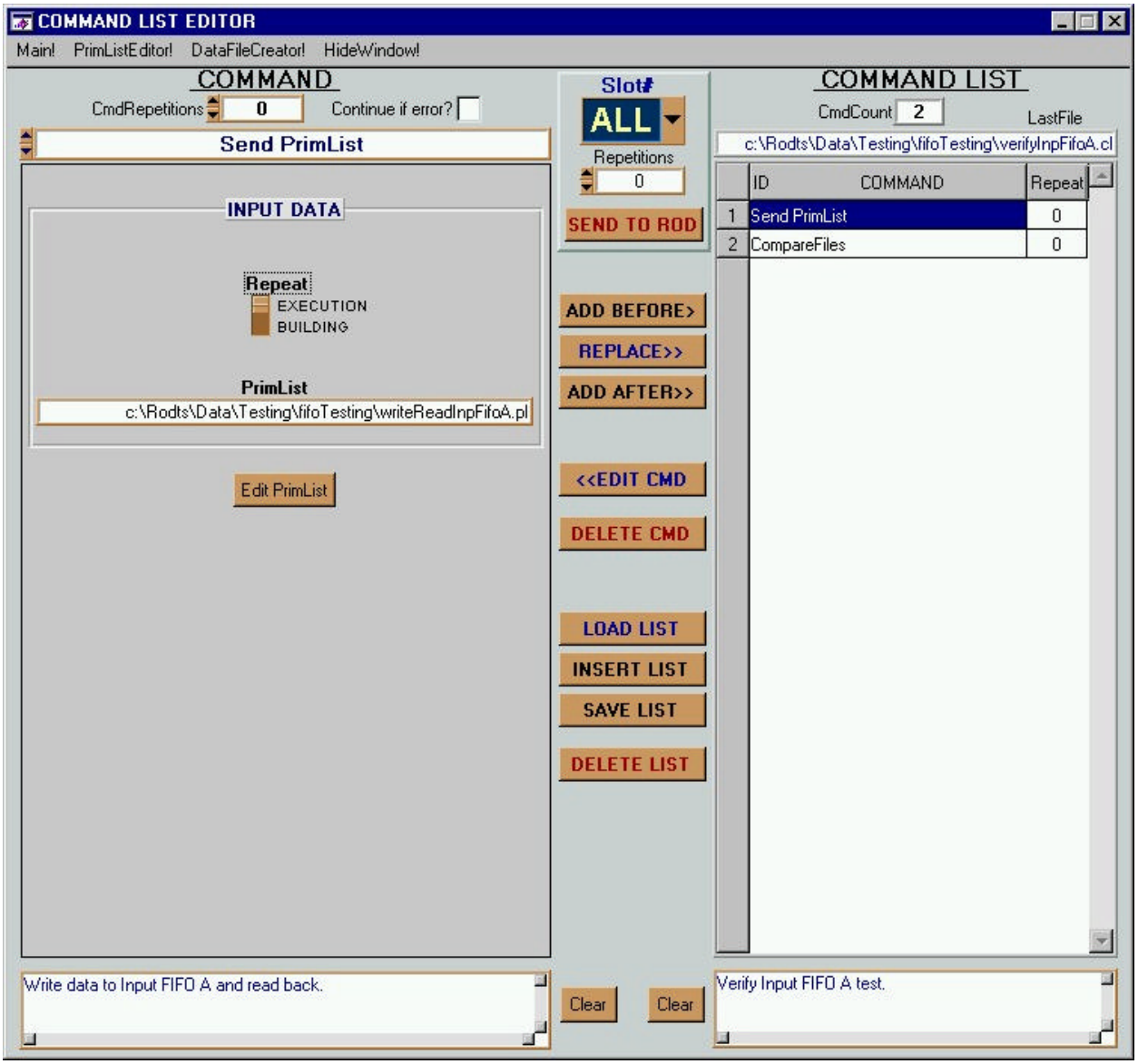
If you want to load already saved list, use **LOAD LIST** or **INSERT LIST** button. When using **LOAD LIST**, you are asked if the current list in the list table should be deleted. If you say yes, the current list will be deleted and replaced by the loaded list. If you say no, the loaded list will be included after the highlighted index in the current list (**INSERT LIST** does this directly).
The name of the last saved or loaded primitive list file is showed in the **LastFile** box above the list table. The saved lists are simple text files, which can be also viewed in any text editor. However, editing outside the PrimListEditor is not recommended, except adding or changing the comments (enter the comment between the - - - - - - and ====== lines, number of comment lines is not strictly limited).

Primitive List example:

```
This is comment for the primitive list.
The list contains one primitive (ECHO).
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
length: 0x10A, numMsgs= 0x1, primListRevision= 111
allPrimCount: 1, replyLength: 0x10A, replyCount: 0x1
============================================================
This is comment for the first primitive (index 0).
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
index: 0, id: 0x0, primRevision: 100, length: 0x104, replyLength:
0x104
ECHO
InputDataOption= 0
testPattern= 0xABCDEF, testDataLength= 0x100
```

The PrimitiveList can be sent directly from the PrimListEditor table to any Rod for execution (the Rod must be loaded and initialized in TestStand, see chapter 3.).
Select the slot number, number of list repetitions (0 is executed once) and press **SEND TO ROD** button. The option Slave# ALL means the list will be sent to all loaded Rods.

6. Command List Editor



The user interface very similar to *PrimList Editor*. The table on the right is the CommandList, the panel on the left shows the parameters for the selected command.

Command List Editing

The function of all buttons is the same as in *PrimitiveListEditor*. No shortcut keys here, hovewer. Above that you also have to enter the number of repetitions for each command in the command list (**CmdRepetitions** box: 0 means that the command will be executed once, 1 twice etc.) and select/deselect the option **Continue if error?**. Check this box if you want the command list execution to continue even if this command was not successful (this option can be overridden by the global *ContinueIfError* option, see chapters 2. and 3.).

Command List example:

```
This is comment for the command list.
The list contains one command (SEND_PRIM_LIST).
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
commandCount: 1
============================================================
This is comment for the first command (index 0).
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
index: 0, id: 3, numRepetitions: 0, continueIfError: 0
SEND_PRIM_LIST_COMMAND_ID
RepeatExecNotBuild= 1
FileName[]= c:\Rodts\Data\Testing\echo2.pl
```