NAME

mgsimdev-rpc − Host-simulation RPC interface in MGSim

DESCRIPTION

The RPC pseudo−device provides access to the filesystem of the simulation environment to programs running on the platform.

An I/O device of this type can be specified in MGSim using the device type **RPC**.

CONFIGURATION

RPCBufferSize1, RPCBufferSize2

Maximum size for the 1st and 2nd RPC procedure arguments fetched from shared memory.

RPCIncomingQueueSize, RPCReadyQueueSize, RPCCompletedQueueSize, RPCNotificationQueueSize

Size of the request queues.

PROTOCOL

The RPC device acts as a service provider: software running on the platform issues *requests* to the RPC interface by writing to its control words; the requests are processed by the RPC device, and their completion is eventually *signalled* back to the originating core(s).

The RPC device is pipelined.

The input to the pipeline is a tuple:

- device ID for DCA (ID of requestor); channel ID for completion notification;
- service procedure ID (read/open/write/seek...);
- completion writeback value (payload in the completion notification);
- address in memory where the arguments are to be fetched from;
- memory size for the argument data;
- address in memory where the results are to be stored.

The completion writeback and memory addresses are passive registers; the request is issued with their current value when control word 0 is written to.

Internally; for each request issued:

1. the argument data is fetched from the I/O core's memory via DCA;
2. the request is served;
3. the results are sent back to memory via DCA;
4. the notification is sent back to signal completion.

The device maintains the following queues:

- incoming queue: commands issued via the device interface;
- ready queue: commands after the argument data has been fetched;
- completed queue: results that need to be communicated back to memory;
- notification queue: for completion notifications.

Supported requests

The procedure IDs are listed as symbolic constants in the file **RPCServiceDatabase.h** shipped together with MGSim's source code.

The following are currently in use:

Procedure ID	Description	Arg. 1	Arg. 2	Arg. 3	Arg. 4	Res. 1	Res. 2
RPC_nop	"Do nothing", used for testing	N/A	N/A	N/A	N/A	N/A	N/A
RPC_open	Open file	path	N/A	flags	mode	vfd+errno	N/A
RPC_read	Read from file	N/A	N/A	vfd	size (bytes)	status+errno	read data
RPC_pread	Read from file at offset	offset	N/A	vfd	size (bytes)	status+errno	read data
RPC_write	Write to file	data to write	N/A	vfd	size (bytes)	status+errno	N/A
RPC_pwrite	Write to file at offset	offset	data to write	vfd	size (bytes)	status+errno	N/A
RPC_lseek	Seek in file	offset (low 32 bits, then high 32 bits)	N/A	vfd	whence	status+errno	new offset
RPC_close	Close file	N/A	N/A	vfd	N/A	status+errno	N/A
RPC_sync	Synchronize storage	N/A	N/A	N/A	N/A	N/A	N/A
RPC_fsync	Synchronize storage for file	N/A	N/A	vfd	N/A	status+errno	N/A
RPC_dup	Duplicate vfd.	N/A	N/A	vfd	N/A	new vfd+errno	N/A
RPC_dup2	Duplicate vfd.	N/A	N/A	vfd	desired vfd	vfd/status+errno	N/A
RPC_getdtablesize	Get max nr. of fds.	N/A	N/A	N/A	N/A	dtable size	N/A
RPC_link	Hard link file	source path	dest. path	N/A	N/A	status+errno	N/A
RPC_unlink	Remove link / delete	path	N/A	N/A	N/A	status+errno	N/A
RPC_rename	Rename file/dir	src path	dst path	N/A	N/A	status+errno	N/A
RPC_mkdir	Create directory	path	N/A	N/A	N/A	status+errno	N/A
RPC_rmdir	Remove directory	path	N/A	N/A	N/A	status+errno	N/A
RPC_fstat	Stat file by fd	N/A	N/A	vfd	N/A	status+errno	struct vstat
RPC_stat / RPC_lstat	Stat file	path	N/A	N/A	N/A	status+errno	struct vstat
RPC_opendir	Open directory	path	N/A	N/A	N/A	vdd+errno	N/A
RPC_fdopendir	Open directory by fd	N/A	N/A	vfd	N/A	vdd+errno	N/A
RPC_readdir	Read directory entry	N/A	N/A	vdd	N/A	status+errno	struct vdirent

RPC_rewinddir	Rewind directory	N/A	N/A	vdd	N/A	errno	N/A
RPC_telldir	Tell position in directory	N/A	N/A	vdd	N/A	pos+errno	N/A
RPC_seekdir	Seek in directory	N/A	N/A	vdd	offset	errno	N/A
RPC_closedir	Close directory	N/A	N/A	vdd	N/A	status+errno	N/A

The status+errno field reported in the 1st response block always follows the following format:

- bytes 0−3: low 32 bits of the procedure's return value
- bytes 4−7: high 32 bits of the procedure's return value
- butes 8−12: errno value

The **struct vstat** returned by the **RPC_*stat** request responses, as well as the **struct vdirent** returned by the **RPC_readdir** request responses, are also defined in **RPCServiceDatabase.h**.

INTERFACE

The RPC device presents itself to the I/O bus as a single device. It must be accessed using 32−bit I/O operations. Its device address space is as follows:

Word	Mode	Description
0	R	0 = idle; 1 = busy/queuing
0	W	start queuing current command
1	R/W	Device ID for DCA (bits 0−15), notification channel (bits 16−31)
2	R/W	Service procedure ID
4	R/W	Low 32 bits of the completion payload value
5	R/W	High 32 bits of the completion payload value
6	R/W	Size in bytes of the 1st procedure argument
7	R/W	Size in bytes of the 2nd procedure argument
8	R/W	Low 32 bits of the 1st procedure argument base address
9	R/W	High 32 bits of the 1st procedure argument base address
10	R/W	Low 32 bits of the 2nd procedure argument base address
11	R/W	High 32 bits of the 2nd procedure argument base address
12	R/W	3rd argument (passed by value)

13	R/W	4th argument (passed by value)
14	R/W	Low 32 bits of the address where to write the 1st result value
15	R/W	High 32 bits of the address where to write the 1st result value
16	R/W	Low 32 bits of the address where to write the 2nd result value
17	R/W	High 32 bits of the address where to write the 2nd result value
64	R	Maximum size for the 1st argument
65	R	Maximum size for the 2nd argument
66	R	Maximum size for the 1st result value
67	R	Maximum size for the 2nd result value
68	R	Current number of requests in the incoming queue
69	R	Capacity of the incoming queue
70	R	Current number of requests in the ready queue
71	R	Capacity of the ready queue
72	R	Current number of responses in the completed queue
73	R	Capacity of the completed queue
74	R	Current number of responses in the notification queue
75	R	Capacity of the notification queue

SEE ALSO

mgsim(1), mgsimdoc(7)

BUGS

Report bugs & suggest improvements to *microgrids@svp−home.org*.

AUTHOR

MGSim was created by Mike Lankamp. MGSim is now under stewardship of the Microgrid project. This manual page was written by Raphael ’kena’ Poss.

COPYRIGHT