Appendix A: User Procedure Calls

This appendix describes the library of user procedure calls provided in `libiwdb.a`. These procedures allow users to write their own interface to the debugger array-server, `iwdb_serv`, without any knowledge or understanding of RPC. All RPC requirements are handled internally. Unless otherwise indicated, all calls return -1 for no error, and 0 for an error.

Initialize Session

This must be the first interface procedure call. It first obtains an exclusive lock on the SIB of the remote server, and then starts the array-server process. *count* is the number of cells on the array connected to *sib*. If the request cannot be satisfied, the procedure exits. There is no return value.

```
void initialize_remote(server, sib, count)
char *server;
int sib;
int count;
```

End Session

Releases the lock on the SIB and terminates the array server process. The argument, *dummy*, is not currently used. Return implies normal completion.

```
void remote_close(dummy)
int dummy;
```

Load Program

Loads the program named onto the cell specified. *program* must be either a full path name, or found in the users current search path.

```
int remote_open(cell, program)
int cell;
char *program;
```

Run

Starts one cell program running on the cell number given, It must be previously loaded using the `load program` command.

```
int remote_run(cell, dummy)
int cell, dummy;
```

Resume

Restart a cell program after a breakpoint. If *step* is greater than 0, then single-step the specified cell. Otherwise, the cell program runs until the next breakpoint or program termination. The argument *signal* is not currently used.

```
int remote_resume(step, signal, cell_id)
int step, cell_id;
int signal;
```

Wait Array

Wait for any cell in the array to stop. The status is reported as either 0 for program termination or the value 0177 | SIGTRAP <<8. *cell_id* is the number of the cell that stopped.

```
int remote_wait(status, cell_id)
int *status;
int *cell_id;
```

Wait Cell

Wait for and report state of specified cell only. *cell_id* is the number of the cell to wait on, and *status* is reported above.

```
int remote_wait_cell(status, cell_id)
int     *status;
int     cell_id;
```

Set Breakpoint

Set a breakpoint on a cell at a specified address.

```
int remote_sbp(addr, cell_id)
unsignedint addr;
int              cell_id;
```

Clear Breakpoint

Clear a breakpoint on a cell at the specified address. An error results if there is not a breakpoint set at the specified address on the cell.

```
int remote_ubp(addr, cell_id)
unsigned int     addr;
int              cell_id;
```

Fetch Registers

Fetches the register context from the cell. The contents of the register context block are described in the include file, `m_iwarp.h`. The size of the context block , `regs`, is the constant `NUM_REGS` (currently 118).

```
int remote_fetch_register(regs, cell_id)
int *regs;
int cell_id;
```

Read Memory

Read local memory starting at address *addr* for length *len* words from the specified cell. Store it in the array *buf*. You must ensure there is enough room for the data.

```
int remote_read_inferior_memory(addr, buf, len, cell_id)
unsigned long    addr;
int              *buf;
int              len;
int              cell_id;
```

Get Instruction

Returns the instruction word at the specified address and cell. This procedure recognizes inserted breakpoints, and returns the original instruction.

```
int remote_get_instruction(addr, cell_id)
unsigned int addr;
int cell_id;
```