The MONSOON Libraries API for Linux

P. N. Daly
National Optical Astronomy Observatories, 950 N. Cherry Avenue, P. O. Box 26732, Tucson AZ 85726–6732, USA.
pnd@noao.edu

N. C. Buchholz
National Optical Astronomy Observatories, 950 N. Cherry Avenue, P. O. Box 26732, Tucson AZ 85726–6732, USA.
ncb@noao.edu

Abstract. This document defines the application programming interface for all MONSOON utility libraries.

Contents

1. Preamble 5
  1.1. GNU Public License ........................................... 5
  1.2. Copyright Protection ........................................... 5
  1.3. Typographic Conventions .................................... 5

2. Introduction 5
  2.1. Making the Libraries ...................................... 5

3. The cliUtil 1.0.0 Interface 7
  3.1. cliUtil.h ....................................................... 7
  3.2. cliUtilDump ................................................... 7
  3.3. cliUtilInit .................................................... 7
  3.4. cliUtilNull .................................................... 8
  3.5. cliUtilParse .................................................. 8
  3.6. cliUtilUninit .................................................. 9

4. The queUtil 1.0.0 Library 10
  4.1. queUtil.h .................................................... 10
  4.2. queUtilDequeueAdd ....................................... 10
  4.3. queUtilQueueAdd ......................................... 10
  4.4. queUtilStackAdd ......................................... 11
  4.5. queUtilQueueClose ....................................... 11
  4.6. queUtilQueueClose ....................................... 12
4.7. queUtilStackClose ......................................................... 12
4.8. queUtilDequeueDump ..................................................... 12
4.9. queUtilQueueDump ........................................................ 13
4.10. queUtilStackDump ....................................................... 13
4.11. queUtilDequeueEmpty .................................................. 14
4.12. queUtilQueueEmpty .................................................... 14
4.13. queUtilStackEmpty ..................................................... 14
4.14. queUtilDequeueFull .................................................... 15
4.15. queUtilQueueFull ....................................................... 15
4.16. queUtilStackFull ....................................................... 16
4.17. queUtilDequeueNew ..................................................... 16
4.18. queUtilQueueNew ....................................................... 17
4.19. queUtilStackNew ....................................................... 17
4.20. queUtilDequeueOpen ................................................... 18
4.21. queUtilQueueOpen ..................................................... 18
4.22. queUtilStackOpen ..................................................... 19
4.23. queUtilDequeueRemove ............................................... 19
4.24. queUtilQueueRemove ................................................. 20
4.25. queUtilStackRemove ................................................ 20

5. The semUtil 1.0.0 Library ............................................... 21
  5.1. semUtil.h ............................................................. 21
  5.2. semUtilGet .......................................................... 21
  5.3. semUtilGive .......................................................... 21
  5.4. semUtilInit ........................................................... 22
  5.5. semUtilNew ........................................................... 22
  5.6. semUtilRelease ...................................................... 23
  5.7. semUtilTake .......................................................... 23

6. The shmUtil 1.0.0 Library ............................................... 24
  6.1. shmUtil.h ............................................................. 24
  6.2. shmUtilAttach ........................................................ 24
  6.3. shmUtilDetach ........................................................ 25
  6.4. shmUtilInit ........................................................... 25
  6.5. shmUtilSize ........................................................... 25
  6.6. shmUtilUninit ........................................................ 26

7. The sockUtil 1.0.0 Library ............................................... 27
  7.1. sockUtil.h ............................................................. 27
  7.2. sockUtilAccept ...................................................... 27
  7.3. sockUtilBind .......................................................... 27
  7.4. sockUtilClose ....................................................... 28
  7.5. sockUtilConnectTO .................................................. 28
  7.6. sockUtilConnect ..................................................... 29
  7.7. sockUtilCreate ..................................................... 29
  7.8. sockUtilListen ...................................................... 30

DRAFT
### 7. Monsoon Libraries API for Linux

<table>
<thead>
<tr>
<th>Section</th>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.9</td>
<td>sockUtilNew</td>
<td>30</td>
</tr>
<tr>
<td>7.10</td>
<td>sockUtilRead</td>
<td>30</td>
</tr>
<tr>
<td>7.11</td>
<td>sockUtilReadln</td>
<td>31</td>
</tr>
<tr>
<td>7.12</td>
<td>sockUtilWrite</td>
<td>31</td>
</tr>
<tr>
<td>7.13</td>
<td>sockUtilWriteln</td>
<td>32</td>
</tr>
<tr>
<td>7.14</td>
<td>sockUtilWriteString</td>
<td>32</td>
</tr>
</tbody>
</table>

### 8. The comUtil 1.0.0 Library

<table>
<thead>
<tr>
<th>Section</th>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1</td>
<td>comUtil.h</td>
<td>33</td>
</tr>
<tr>
<td>8.2</td>
<td>comUtilClose</td>
<td>33</td>
</tr>
<tr>
<td>8.3</td>
<td>comUtilDump</td>
<td>33</td>
</tr>
<tr>
<td>8.4</td>
<td>comUtilIOctl</td>
<td>34</td>
</tr>
<tr>
<td>8.5</td>
<td>comUtilInit</td>
<td>34</td>
</tr>
<tr>
<td>8.6</td>
<td>comUtilOpen</td>
<td>35</td>
</tr>
<tr>
<td>8.7</td>
<td>comUtilRead</td>
<td>35</td>
</tr>
<tr>
<td>8.8</td>
<td>comUtilSim</td>
<td>36</td>
</tr>
<tr>
<td>8.9</td>
<td>comUtilUninit</td>
<td>36</td>
</tr>
<tr>
<td>8.10</td>
<td>comUtilWrite</td>
<td>36</td>
</tr>
</tbody>
</table>

### 9. The dheUtil 1.0.0 Library

<table>
<thead>
<tr>
<th>Section</th>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1</td>
<td>dheUtil.h</td>
<td>38</td>
</tr>
<tr>
<td>9.2</td>
<td>dheUtilOpen</td>
<td>38</td>
</tr>
<tr>
<td>9.3</td>
<td>dheUtilClose</td>
<td>38</td>
</tr>
<tr>
<td>9.4</td>
<td>dheUtilSend</td>
<td>39</td>
</tr>
<tr>
<td>9.5</td>
<td>dheUtilRecv</td>
<td>39</td>
</tr>
<tr>
<td>9.6</td>
<td>dheUtilGetData</td>
<td>40</td>
</tr>
<tr>
<td>9.7</td>
<td>dheUtilDownloadWF</td>
<td>40</td>
</tr>
<tr>
<td>9.8</td>
<td>dheUtilIOctl</td>
<td>41</td>
</tr>
<tr>
<td>9.9</td>
<td>dheUtilInit</td>
<td>41</td>
</tr>
<tr>
<td>9.10</td>
<td>dheUtilUninit</td>
<td>42</td>
</tr>
<tr>
<td>9.11</td>
<td>readValue</td>
<td>42</td>
</tr>
<tr>
<td>9.12</td>
<td>writeValue</td>
<td>42</td>
</tr>
<tr>
<td>9.13</td>
<td>asyncResponse</td>
<td>43</td>
</tr>
<tr>
<td>9.14</td>
<td>startExp</td>
<td>43</td>
</tr>
<tr>
<td>9.15</td>
<td>abortExp</td>
<td>44</td>
</tr>
<tr>
<td>9.16</td>
<td>pauseExp</td>
<td>44</td>
</tr>
<tr>
<td>9.17</td>
<td>resumeExp</td>
<td>45</td>
</tr>
<tr>
<td>9.18</td>
<td>stopExp</td>
<td>45</td>
</tr>
<tr>
<td>9.19</td>
<td>armExpTrigger</td>
<td>46</td>
</tr>
</tbody>
</table>

### 10. The comHdwr Library

<table>
<thead>
<tr>
<th>Section</th>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1</td>
<td>comHdwr.h</td>
<td>47</td>
</tr>
</tbody>
</table>

### 11. The dheHdwr Library

<table>
<thead>
<tr>
<th>Section</th>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1</td>
<td>dheHdwr.h</td>
<td>50</td>
</tr>
</tbody>
</table>
12. The systranUtil Library 53
12.1. systranUtil.h ................................. 53
12.2. comHdwClose ................................. 53
12.3. comHdwConfig ............................... 53
12.4. comHdwDump ................................. 54
12.5. comHdwIOctl ................................. 54
12.6. comHdwInit ................................. 55
12.7. comHdwOpen ................................. 55
12.8. comHdwRead ................................. 55
12.9. comHdwStatus ............................... 56
12.10. comHdwUninit .............................. 56
12.11. comHdwWrite ............................... 57

13. The simHdw 1.0.0 Library 58
13.1. simHdw.h .................................... 58
13.2. hdwrClose .................................. 58
13.3. hdwrConfig ................................ 58
13.4. hdwrDump .................................. 59
13.5. hdwrIOctl .................................. 59
13.6. hdwrInit .................................... 60
13.7. hdwrOpen .................................. 60
13.8. hdwrRead .................................. 60
13.9. hdwrStatus ................................ 61
13.10. hdwrUninit ................................ 61
13.11. hdwrWrite ................................ 62

14. Document Revision History 63

List of Figures

List of Tables

1 Markup Conventions for this Document .......................... 5

DRAFT
1. Preamble

1.1. GNU Public License

This document and associated, original software is free. You can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation either version 2 of the License, or (at your option) any later version. This document is distributed in the hope that it will be useful, but without any warranty. Without even the implied warranty of merchantability or fitness for a particular purpose. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this document. If not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge MA 02139, USA.

1.2. Copyright Protection

©2002, AURA Inc., this document and original software. All rights reserved.

1.3. Typographic Conventions

<table>
<thead>
<tr>
<th>Effect</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>blue type-face</td>
<td>user input</td>
</tr>
<tr>
<td>magenta sans-serif</td>
<td>machine output</td>
</tr>
<tr>
<td>black times-roman</td>
<td>normal text</td>
</tr>
<tr>
<td>teal italic</td>
<td>margin notes</td>
</tr>
</tbody>
</table>

The typographical conventions used in this document are described in table 1 on page 5.

2. Introduction

In order to generalize the interface between MONSOON and its component parts, a suite of utility libraries has been created to deal with attribute value pairs, shared memory, semaphores, sockets, queues and stacks. All these libraries have the same generic form and follow the MONSOON coding convention. One part of that convention is the ability to produce documentation ‘on-the-fly’. This document is created from the source code of these utility libraries so that any change in the code can be reflected here.

2.1. Making the Libraries

The software can be located anywhere as it is a self-contained unit. You may wish to edit the file Makefile.Linux, however, to reflect your preferred installation directory. To make the and install the libraries, type:
% ln -s ./Makfile.Linux ./makefile
% make everything
3. The cliUtil 1.0.0 Interface

3.1. cliUtil.h

USE: #include "cliUtil.h"

DESCRIPTION: this file contains all common code required by the functions needed to build the static and dynamic cliUtil libraries. These libraries abstract the command line interface to the system.

ARGUMENT(S): not applicable

RETURN(S): not applicable

LAST MODIFIED: Monday, 16 July 2003

3.2. cliUtilDump

USE: void cliUtilDump(long *istat, char *resp, unsigned long argc, char **argv[]);

DESCRIPTION: this function dumps memory buffers. The inherited status is updated.

ARGUMENT(S):

- *istat the inherited status value.
- *resp a user supplied buffer for message strings.
- argc the number of memory buffers to dump.
- argv[] the returned pointers to the memory buffers.

RETURN(S): void.

LAST MODIFIED: Monday, 16 July 2003

AUTHOR(S): Phil Daly (pnd), Nick Buchholz (ncb)

LICENSE: (c) 1996-2002 AURA Inc. All rights reserved, Released under the GPL.

3.3. cliUtilInit

USE: void cliUtilInit(long *istat, char *resp, unsigned long argc, char **argv[]);

DESCRIPTION: this function mallocs argv[argc] memory buffers. The inherited status is updated.

ARGUMENT(S):

- *istat the inherited status value.
- *resp a user supplied buffer for message strings.
- argc the number of memory buffers to malloc.
- argv[] the returned pointers to the memory buffers.

RETURN(S): void.

LAST MODIFIED: Monday, 16 July 2003

AUTHOR(S): Phil Daly (pnd), Nick Buchholz (ncb)

LICENSE: (c) 1996-2002 AURA Inc. All rights reserved, Released under the GPL.

DRAFT
3.4. cliUtilNull

USE: void cliUtilNull(long *istat, char *resp, unsigned long argc, char *argv[]);

DESCRIPTION: this function initializes (nulls out) argv[argc] memory buffers. The inherited status is updated.

ARGUMENT(S):

- long *istat the inherited status value.
- char *resp a user supplied buffer for message strings.
- unsigned long argc the number of memory buffers to initialize.
- char *argv[] the pointers to the memory buffers to null.

RETURN(S): void.

LAST MODIFIED: Monday, 16 July 2003

AUTHOR(S): Phil Daly (pnd), Nick Buchholz (ncb)

LICENSE: (c) 1996-2002 AURA Inc. All rights reserved, Released under the GPL.

3.5. cliUtilParse

USE: void cliUtilParse(long *istat, char *resp, char *inCmd, size_t inSize, unsigned long *argc, char *argv[]);

DESCRIPTION: this function parses a command line into separate argv[argc] memory buffers. The inherited status is updated.

ARGUMENT(S):

- long *istat the inherited status value.
- char *resp a user supplied buffer for message strings.
- char *inCmd the incoming command string.
- size_t inSize the size of the incoming command string.
- unsigned long *argc the returned number of memory buffers used.
- char *argv[] the pointers to the memory buffers.

RETURN(S): void.

LAST MODIFIED: Monday, 16 July 2003

AUTHOR(S): Phil Daly (pnd), Nick Buchholz (ncb)

LICENSE: (c) 1996-2002 AURA Inc. All rights reserved, Released under the GPL.
3.6. cliUtilUninit

USE: `void cliUtilUninit(long *istat, char *resp, unsigned long argc, char **argv[]);`

DESCRIPTION: this function frees `argv[argc]` memory buffers. The inherited status is updated.

ARGUMENT(S):
- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `unsigned long argc` the number of memory buffers to free.
- `char **argv[]` the pointers to the memory buffers to free.

RETURN(S): `void`.

LAST MODIFIED: Monday, 16 July 2003

AUTHOR(S): Phil Daly (pnd), Nick Buchholz (ncb)

LICENSE: (c) 1996-2002 AURA Inc. All rights reserved, Released under the GPL.
4. The queUtil 1.0.0 Library

4.1. queUtil.h

USE: #include “queUtil.h”

DESCRIPTION: this file contains all common code required by the functions needed to build the static and dynamic queUtil libraries. These libraries abstract the queue, dequeue and stack interface to the system.

ARGUMENT(S): not applicable

RETURN(S): not applicable

LAST MODIFIED: Monday, 4 November 2002

4.2. queUtilDequeueAdd

USE: void queUtilDequeueAdd(long *istat, char *resp, deque *deQ, char toHead, long value);

DESCRIPTION: this routine adds a long value to the head or tail of a Dequeue. The parameter toHead steers the value as required. The inherited status is updated.

ARGUMENT(S):
- long *istat: the inherited status value.
- char *resp: a user supplied buffer for message strings.
- deque *deQ: the dequeue structure.
- char toHead: position in queue.
- long value: the value to write.

RETURN(S): void.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

4.3. queUtilQueueAdd

USE: void queUtilQueueAdd(long *istat, char *resp, queue *Que, long value);

DESCRIPTION: this routine adds a long value to the head or tail of a queue. The parameter toHead steers the value as required. The inherited status is updated.

ARGUMENT(S):
- long *istat: the inherited status value.
- char *resp: a user supplied buffer for message strings.
- deque *Que: the queue structure.
**char toHead**  position in queue.

**long value**  the value to write.

RETURN(S):  void.

LAST MODIFIED:  Monday, 4 November 2002

AUTHOR(S):  Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE:  (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

### 4.4. queUtilStackAdd

**USE:**  void queUtilStackAdd(long *istat, char *resp, stack *Stk, long value);

**DESCRIPTION:**  this routine adds a long value to the head or tail of a stack. The parameter toHead steers the value as required. The inherited status is updated.

**ARGUMENT(S):**

- **long *istat**  the inherited status value.
- **char *resp**  a user supplied buffer for message strings.
- **stack *Stk**  the stack structure.
- **char toHead**  position in queue.
- **long value**  the value to write.

RETURN(S):  void.

LAST MODIFIED:  Monday, 4 November 2002

AUTHOR(S):  Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE:  (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

### 4.5. queUtilDequeueClose

**USE:**  void queUtilDequeueClose(long *istat, char *resp, char *name, deque **deQ);

**DESCRIPTION:**  this routine closes a dequeue. The inherited status is updated.

**ARGUMENT(S):**

- **long *istat**  the inherited status value.
- **char *resp**  a user supplied buffer for message strings.
- **char *name**  the queue name,
- **deque **deQ**  the dequeue structure.

RETURN(S):  void.

LAST MODIFIED:  Monday, 4 November 2002

AUTHOR(S):  Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE:  (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

---

*DRAFT*
4.6. queUtilQueueClose

USE: void queUtilQueueClose(long *istat, char *resp, char *name, queue **Que);

DESCRIPTION: this routine closes a queue. The inherited status is updated.

ARGUMENT(S):

long *istat  the inherited status value.
char *resp  a user supplied buffer for message strings.
char *name  queue name.
queue **Que  the queue structure.

RETURN(S): void.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

4.7. queUtilStackClose

USE: void queUtilStackClose(long *istat, char *resp, char *name, stack **Stk);

DESCRIPTION: this routine closes a stack. The inherited status is updated.

ARGUMENT(S):

long *istat  the inherited status value.
char *resp  a user supplied buffer for message strings.
char *name  stack name.
stack **Stk  the stack structure.

RETURN(S): void.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

4.8. queUtilDequeueDump

USE: void queUtilDequeueDump(long *istat, char *resp, deque *deQ);

DESCRIPTION: this routine dumps information on the given queue. The inherited status is updated.

ARGUMENT(S):

long *istat  the inherited status value.
char *resp  a user supplied buffer for message strings.
deque *deQ the dequeue structure.

RETURN(S): void.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

4.9. queUtilQueueDump

USE: void queUtilQueueDump(long *istat, char *resp, queue *Que);

DESCRIPTION: this routine dumps information on the given queue. The inherited status is updated.

ARGUMENT(S):
- long *istat the inherited status value.
- char *resp a user supplied buffer for message strings.
- deque *Que the queue structure.

RETURN(S): void.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

4.10. queUtilStackDump

USE: void queUtilStackDump(long *istat, char *resp, stack *Stk);

DESCRIPTION: this routine dumps information on the given stack. The inherited status is updated.

ARGUMENT(S):
- long *istat the inherited status value.
- char *resp a user supplied buffer for message strings.
- stack *Stk the stack structure.

RETURN(S): void.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

DRAFT
4.11. queUtilDequeueEmpty

USE:  int queUtilDequeueEmpty(long *istat, char *resp, deque *deQ);

DESCRIPTION:  this routine checks to see if a Dequeue is empty. It returns 0 if not empty, 1 if empty. The inherited status is updated.

ARGUMENT(S):

  long *istat  the inherited status value.
  char *resp  a user supplied buffer for message strings.
  deque *deQ  the dequeue structure.

RETURN(S):  0=not empty, 1=empty otherwise an error.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S):  Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE:  (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

4.12. queUtilQueueEmpty

USE:  int queUtilQueueEmpty(long *istat, char *resp, queue *Que);

DESCRIPTION:  this routine checks to see if a queue is empty. It returns 0 if not empty, 1 if empty. The inherited status is updated.

ARGUMENT(S):

  long *istat  the inherited status value.
  char *resp  a user supplied buffer for message strings.
  queue *Que  the queue structure.

RETURN(S):  0=not empty, 1=empty otherwise an error.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S):  Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE:  (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

4.13. queUtilStackEmpty

USE:  int queUtilStackEmpty(long *istat, char *resp, queue *Stk);

DESCRIPTION:  this routine checks to see if a stack is empty. It returns 0 if not empty, 1 if empty. The inherited status is updated.

ARGUMENT(S):

  long *istat  the inherited status value.
  char *resp  a user supplied buffer for message strings.
4.14. **queUtilDequeueFull**

**USE:**
```
int queUtilDequeueFull(long *istat, char *resp, deque *deQ);
```

**DESCRIPTION:**
This routine checks to see if a Dequeue is full. It returns 0 if not full, 1 if full. The inherited status is updated.

**ARGUMENT(S):**
- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `deque *deQ` the dequeue structure.

**RETURN(S):**
0=not full, 1=full otherwise an error.

**LAST MODIFIED:** Monday, 4 November 2002

**AUTHOR(S):** Nick Buchholz (ncb), Phil Daly (pnd)

**LICENSE:** (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

4.15. **queUtilQueueFull**

**USE:**
```
int queUtilQueueFull(long *istat, char *resp, queue *Que);
```

**DESCRIPTION:**
This routine checks to see if a queue is full. It returns 0 if not full, 1 if full. The inherited status is updated.

**ARGUMENT(S):**
- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `queue *Que` the queue structure.

**RETURN(S):**
0=not full, 1=full otherwise an error.

**LAST MODIFIED:** Monday, 4 November 2002

**AUTHOR(S):** Nick Buchholz (ncb), Phil Daly (pnd)

**LICENSE:** (c) 2002 AURA Inc. All rights reserved, Released under the GPL.
4.16. queUtilStackFull

USE:  int queUtilStackFull(long *istat, char *resp, stack *Stk);

DESCRIPTION: this routine checks to see if a stack is full. It returns 0 if not full, 1 if full. The inherited status is updated.

ARGUMENT(S):

  long *istat  the inherited status value.
  char *resp   a user supplied buffer for message strings.
  stack *Stk   the stack structure.

RETURN(S): 0=not full, 1=full otherwise an error.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

4.17. queUtilDequeueNew

USE:  void queUtilDequeueNew(long *istat, char *resp, char *name, deque *deQ);

DESCRIPTION: this routine either creates space for a new dequeue or clears existing space passed in. If deQ == NULL new space is created otherwise the existing pointer is used as a pointer to a dequeue structure and it is cleared. The inherited status is updated.

ARGUMENT(S):

  long *istat  the inherited status value.
  char *resp   a user supplied buffer for message strings.
  char *name   the queue name,
  deque *deQ   the dequeue structure.

RETURN(S): void.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

DRAFT
4.18. queUtilQueueNew

USE:  void queUtilQueueNew(long *istat, char *resp, char *name, queue *Que);

DESCRIPTION: this routine either creates space for a new queue or clears existing space passed in. If Que == NULL new space is created otherwise the existing pointer is used as a pointer to a queue structure and it is cleared. The inherited status is updated.

ARGUMENT(S):

  long *istat  the inherited status value.
  char *resp  a user supplied buffer for message strings.
  char *name  queue name.
  queue *Que  the queue structure.

RETURN(S):  void.

LAST MODIFIED:  Monday, 4 November 2002

AUTHOR(S):  Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE:  (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

4.19. queUtilStackNew

USE:  void queUtilStackNew(long *istat, char *resp, char *name, stack *p Stk);

DESCRIPTION: this routine either creates space for a new stack or clears existing space passed in. If Stk == NULL new space is created otherwise the existing pointer is used as a pointer to a queue structure and it is cleared. The inherited status is updated.

ARGUMENT(S):

  long *istat  the inherited status value.
  char *resp  a user supplied buffer for message strings.
  char *name  stack name.
  stack *p Stk  the stack structure.

RETURN(S):  void.

LAST MODIFIED:  Monday, 4 November 2002

AUTHOR(S):  Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE:  (c) 2002 AURA Inc. All rights reserved, Released under the GPL.
4.20. queUtilDequeueOpen

USE: void queUtilDequeueOpen(long *istat, char *resp, char *name, deque **deQ);

DESCRIPTION: this routine either creates space for a new dequeue or clears existing space passed in. If deQ == NULL new space is created otherwise the existing pointer is used as a pointer to a dequeue structure and it is cleared. The inherited status is updated.

ARGUMENT(S):

- long *istat  the inherited status value.
- char *resp  a user supplied buffer for message strings.
- char *name  the queue name,
- deque **deQ  the dequeue structure.

RETURN(S): void.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

4.21. queUtilQueueOpen

USE: void queUtilQueueOpen(long *istat, char *resp, char *name, queue **Que);

DESCRIPTION: this routine either creates space for a new queue or clears existing space passed in. If Que == NULL new space is created otherwise the existing pointer is used as a pointer to a queue structure and it is cleared. The inherited status is updated.

ARGUMENT(S):

- long *istat  the inherited status value.
- char *resp  a user supplied buffer for message strings.
- char *name  queue name.
- queue **Que  the queue structure.

RETURN(S): void.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.
4.22. **queUtilStackOpen**

**USE:**
```c
void queUtilStackOpen(long *istat, char *resp, char *name, stack **Stk);
```

**DESCRIPTION:** this routine either creates space for a new stack or clears existing space passed in. If Stk == NULL new space is created otherwise the existing pointer is used as a pointer to a queue structure and it is cleared. The inherited status is updated.

**ARGUMENT(S):**
- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `char *name` stack name.
- `stack **Stk` the stack structure.

**RETURN(S):** void.

**LAST MODIFIED:** Monday, 4 November 2002

**AUTHOR(S):** Nick Buchholz (ncb), Phil Daly (pnd)

**LICENSE:** (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

4.23. **queUtilDequeueRemove**

**USE:**
```c
void queUtilDequeueRemove(long *istat, char *resp, deque *deQ, char fromHead, long *value);
```

**DESCRIPTION:** this routine removes a long value from the head or tail of a Dequeue. The parameter fromHead steers the value as required which is returned in value. The inherited status is updated.

**ARGUMENT(S):**
- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `deque *deQ` the dequeue structure.
- `char fromHead` position in queue.
- `long *value` the address of the returned value.

**RETURN(S):** void.

**LAST MODIFIED:** Monday, 4 November 2002

**AUTHOR(S):** Nick Buchholz (ncb), Phil Daly (pnd)

**LICENSE:** (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

---

*Draft*
4.24. queUtilQueueRemove

USE: void queUtilQueueRemove(long *istat, char *resp, queue *Que, long *value);

DESCRIPTION: this routine adds a long value to the head or tail of a queue. The parameter toHead
steers the value as required. The inherited status is updated.

ARGUMENT(S):

  long *istat  the inherited status value.
  char *resp  a user supplied buffer for message strings.
  queue *Que  the queue structure.
  long *value  the returned value.

RETURN(S): void.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

4.25. queUtilStackRemove

USE: void queUtilStackRemove(long *istat, char *resp, stack *Stk, long *value);

DESCRIPTION: this routine adds a long value to the head or tail of a stack. The parameter toHead
steers the value as required. The inherited status is updated.

ARGUMENT(S):

  long *istat  the inherited status value.
  char *resp  a user supplied buffer for message strings.
  stack *Stk  the stack structure.
  long *value  the returned value.

RETURN(S): void.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.
5. The semUtil 1.0.0 Library

5.1. semUtil.h

USE: 

```
#include "semUtil.h"
```

DESCRIPTION: this file contains all common code required by the functions needed to build the static and dynamic semUtil libraries. These libraries abstract the semaphore interface to the system.

ARGUMENT(S): not applicable

RETURN(S): not applicable

LAST MODIFIED: Monday, 4 November 2002

5.2. semUtilGet

USE: 

```
void semUtil(long *istat, char *resp, char *semName, int *semId, int semAccess);
```

DESCRIPTION: this function gets a semaphore with the specified name and access. If the semaphore doesn’t exist it is created. The inherited status is updated.

ARGUMENT(S):

- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `int semId` the returned semaphore identifier.
- `int semAccess` the semaphore access.

RETURN(S): void.

LAST MODIFIED: Monday, 20030210

AUTHOR(S): Nick Buchholz (ncb)

LICENSE: (c) 2003 AURA Inc. All rights reserved, Released under the GPL.

5.3. semUtilGive

USE: 

```
void semUtilGive(long *istat, char *resp, int semId);
```

DESCRIPTION: this function gives the specified semaphore. The inherited status is updated.

ARGUMENT(S):

- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `int semId` the semaphore identifier.

RETURN(S): void.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

DRAFT
5.4. **semUtilInit**

**USE:** `void semUtilInit(long *istat, char *resp, int semId);`

**DESCRIPTION:** this function initializes the specified sempahore with the value. The inherited status is updated.

**ARGUMENT(S):**
- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `int semId` the semaphore identifier.
- `int semVal` the semaphore value.

**RETURN(S):** `void`.

**LAST MODIFIED:** Monday, 4 November 2002

**AUTHOR(S):** Nick Buchholz (ncb), Phil Daly (pnd)

**LICENSE:** (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

5.5. **semUtilNew**

**USE:** `void semUtilNew(long *istat, char *resp, int *semId, int semAccess);`

**DESCRIPTION:** this function gets a new sempahore with the specified access. The inherited status is updated.

**ARGUMENT(S):**
- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `int semId` the returned semaphore identifier.
- `int semAccess` the semaphore access.

**RETURN(S):** `void`.

**LAST MODIFIED:** Monday, 4 November 2002

**AUTHOR(S):** Nick Buchholz (ncb), Phil Daly (pnd)

**LICENSE:** (c) 2002 AURA Inc. All rights reserved, Released under the GPL.
5.6. **semUtilRelease**

**USE:**  
```c
void semUtilRelease(long *istat, char *resp, int semId);
```

**DESCRIPTION:**  
this function releases the specified semaphore. The inherited status is updated.

**ARGUMENT(S):**
- `long *istat`  the inherited status value.
- `char *resp`  a user supplied buffer for message strings.
- `int semId`  the semaphore identifier.

**RETURN(S):**  
`void`

**LAST MODIFIED:**  
Monday, 4 November 2002

**AUTHOR(S):**  
Nick Buchholz (ncb), Phil Daly (pnd)

**LICENSE:**  
(c) 2002 AURA Inc. All rights reserved, Released under the GPL.

5.7. **semUtilTake**

**USE:**  
```c
void semUtilTake(long *istat, char *resp, int semId);
```

**DESCRIPTION:**  
this function takes the specified semaphore. The inherited status is updated.

**ARGUMENT(S):**
- `long *istat`  the inherited status value.
- `char *resp`  a user supplied buffer for message strings.
- `int semId`  the semaphore identifier.

**RETURN(S):**  
`void`

**LAST MODIFIED:**  
Monday, 4 November 2002

**AUTHOR(S):**  
Nick Buchholz (ncb), Phil Daly (pnd)

**LICENSE:**  
(c) 2002 AURA Inc. All rights reserved, Released under the GPL.
6. The shmUtil 1.0.0 Library

6.1. shmUtil.h

USE: #include “shmUtil.h”

DESCRIPTION: this file contains all common code required by the functions needed to build the static and dynamic shmUtil libraries. These libraries abstract the shared memory interface to the system.

ARGUMENT(S): not applicable

RETURN(S): not applicable

LAST MODIFIED: Monday, 4 November 2002

6.2. shmUtilAttach

USE: void *shmUtilAttach(long *istat, char *resp, char *libName, int *create, int size);

DESCRIPTION: this function initializes the shared memory segment. The inherited status is updated.

ARGUMENT(S):

long *istat  the inherited status value.
char *resp  a user supplied buffer for message strings.
char *libName  the library name.
int *create  TRUE if we need to create the segment first (returns shmid).
int size  the size of the desired segment.

RETURN(S): pointer to memory address or (void *) NULL.

LAST MODIFIED: Wednesday, 12 March 2003

AUTHOR(S): Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

6.3. shmUtilDetach

USE: void shmUtilDetach(long *istat, char *resp, char *libName, void *address);

DESCRIPTION: this function detaches from the shared memory segment for global library variables. A flag indicates that the memory should also be destroyed. The inherited status is updated and returned along with a message response.

ARGUMENT(S):

long *istat  the inherited status value.
char *resp  a user supplied buffer for message strings.
char *libName  the library name.
void *address the address of the segment to detach from.

RETURN(S): void

LAST MODIFIED: Thursday, 21 November 2002

AUTHOR(S): Phil Daly (pnd), Nick Buchholz (ncb)

LICENSE: (c) 2002 AURA Inc. All rights reserved. Released under the GPL.

6.4. shmUtilInit

USE: key_t shmUtilInit(long *istat, char *resp, char *libName, int *create);

DESCRIPTION: this function initializes the shared memory segment file lock. The inherited status is updated.

ARGUMENT(S):
- long *istat the inherited status value.
- char *resp a user supplied buffer for message strings.
- char *libName the library name.
- int *create TRUE if we need to create the segment first (returns shmid).

RETURN(S): a key_t related to the lock file to be used in creating the shared memory

LAST MODIFIED: Monday, 12 March 2003

AUTHOR(S): Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

6.5. shmUtilSize

USE: u_long shmUtilSize(long *istat, char *resp, u_long *inSize, u_long inUnit);

DESCRIPTION: this function resizes inSize so that it is an exact multiple of of inUnit. It is used to ensure boundary integrity prior to memory allocation. The inherited status is updated.

ARGUMENT(S):
- long *istat the inherited status value.
- char *resp a user supplied buffer for message strings.
- u_long *inSize input size (modified on return).
- u_long inUnit the base unit for re-sizing.

RETURN(S): number of elements in new inSize.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Phil Daly (pnd)

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.
6.6. **shmUtilUninit**

**USE**: `void shmUtilUninit(long *istat, char *resp, char *libName);`

**DESCRIPTION**: this function un-initializes the shared memory segment. The inherited status is updated.

**ARGUMENT(S):**
- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `char *libName` the library name.

**RETURN(S)**: `void`.

**LAST MODIFIED**: Monday, 4 November 2002

**AUTHOR(S)**: Nick Buchholz (ncb), Phil Daly (pnd)

**LICENSE**: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.
7. The sockUtil 1.0.0 Library

7.1. sockUtil.h

USE: `#include "sockUtil.h"

DESCRIPTION: this file contains all common code required by the functions needed to build the static and dynamic sockUtil libraries. These libraries abstract the socket interface to the system.

ARGUMENT(S): not applicable

RETURN(S): not applicable

LAST MODIFIED: Monday, 4 November 2002

7.2. sockUtilAccept

USE: `void sockUtilAccept(long *istat, char *resp, int sFd, struct sockaddr_in *clientAddr, int *newFd);

DESCRIPTION: this function accepts new socket connections. The inherited status is updated.

ARGUMENT(S):

- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `int sFd` socket file descriptor.
- `struct sockaddr_in *clientAddr` information on client socket connection.
- `int *newFd` socket file descriptor for new connection.

RETURN(S): void.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Tad Morgan (tmorgan), Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 1996-2002 AURA Inc. All rights reserved, Released under the GPL.

7.3. sockUtilBind

USE: `void sockUtilBind(long *istat, char *resp, int sFd, int port);

DESCRIPTION: this function attempts to bind the socket to the requested port. The inherited status is updated.

ARGUMENT(S):

- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `int sFd` socket file descriptor.
7.4. sockUtilClose

USE: `void sockUtilClose(long *istat, char *resp, int *sFd);` The inherited status is updated.

DESCRIPTION: this function closes the given file descriptor.

ARGUMENT(S):

- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `int *sFd` the file descriptor to close.

RETURN(S): void.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Tad Morgan (tmorgan), Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 1996-2002 AURA Inc. All rights reserved, Released under the GPL.

7.5. sockUtilConnectTO

USE: `void sockUtilConnectTO(long *istat, char *resp, int *sFd, int port, char *saver_inet_addr, int tout);`

DESCRIPTION: this function creates and connects to a new socket with a timeout. The inherited status is updated.

ARGUMENT(S):

- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `int *sFd` the returned file descriptor.
- `int port` the port number to connect to.
- `char *saver_inet_addr` IP (‘dot’) address of saver server, as a string.
- `int tout` the timeout on the connection.

RETURN(S): void.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Tad Morgan (tmorgan), Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 1996-2002 AURA Inc. All rights reserved, Released under the GPL.
7.6. **sockUtilConnect**

**USE:**
```c
t void sockUtilConnect(long *istat, char *resp, int *sFd, int port, char *saver_inet_addr);
```

**DESCRIPTION:** this function creates and connects to a new socket. The inherited status is updated.

**ARGUMENT(S):**
- `long *istat` - the inherited status value.
- `char *resp` - a user supplied buffer for message strings.
- `int *sFd` - the returned file descriptor.
- `int port` - the port number to connect to.
- `char *save_inet_addr` - IP (‘dot’) address of saver server, as a string.

**RETURN(S):** `void`.

**LAST MODIFIED:** Monday, 4 November 2002

**AUTHOR(S):** Tad Morgan (tmorgan), Nick Buchholz (ncb), Phil Daly (pnd)

**LICENSE:** (c) 1996-2002 AURA Inc. All rights reserved, Released under the GPL.

7.7. **sockUtilCreate**

**USE:**
```c
t void sockUtilCreate(long *istat, char *resp, int *sFd, int *port);
```

**DESCRIPTION:** this function creates a new socket and returns a descriptor to it. The inherited status is updated.

**ARGUMENT(S):**
- `long *istat` - the inherited status value.
- `char *resp` - a user supplied buffer for message strings.
- `int *sFd` - returned socket file descriptor.
- `int *port` - returned port to bind socket to.

**RETURN(S):** `void`.

**LAST MODIFIED:** Monday, 4 November 2002

**AUTHOR(S):** Tad Morgan (tmorgan), Nick Buchholz (ncb), Phil Daly (pnd)

**LICENSE:** (c) 1996-2002 AURA Inc. All rights reserved, Released under the GPL.
7.8. **sockUtilListen**

**USE:** `void sockUtilListen(long *istat, char *resp, int sFd, int maxconnections);`

**DESCRIPTION:** this function listens for client connection requests. The inherited status is updated.

**ARGUMENT(S):**
- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `int sFd` socket file descriptor to listen on.
- `int maxconnections` maximum number of connections allowed.

**RETURN(S):** OK or ERROR.

**LAST MODIFIED:** Monday, 4 November 2002

**AUTHOR(S):** Tad Morgan (tmorgan), Nick Buchholz (ncb), Phil Daly (pnd)

**LICENSE:** (c) 1996-2002 AURA Inc. All rights reserved, Released under the GPL.

7.9. **sockUtilNew**

**USE:** `int sockUtilNew(long *istat, char *resp, int *sFd);`

**DESCRIPTION:** this function creates a new socket. The inherited status is updated.

**ARGUMENT(S):**
- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `int *sFd` the returned socket file descriptor.

**RETURN(S):** OK or ERROR.

**LAST MODIFIED:** Monday, 4 November 2002

**AUTHOR(S):** Tad Morgan (tmorgan), Nick Buchholz (ncb), Phil Daly (pnd)

**LICENSE:** (c) 1996-2002 AURA Inc. All rights reserved, Released under the GPL.

7.10. **sockUtilRead**

**USE:** `void sockUtilRead(long *istat, char *resp, int *sFd, char *buffer, int sz);`

**DESCRIPTION:** this function reads the socket into a buffer. The inherited status is updated.

**ARGUMENT(S):**
- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `int *sFd` the file descriptor to read from.
char *buffer  the buffer to write the data to.
int sz  the number of bytes to read from socket.

RETURN(S): void.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Peter Ruckle (pruckle), Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 1996-2002 AURA Inc. All rights reserved, Released under the GPL.

7.11. sockUtilRead

USE: int sockUtilRead(long *istat, char *resp, int *sFd, char *buffer, int sz);

DESCRIPTION: this function reads the socket into a buffer. The inherited status is updated.

ARGUMENT(S):

long *istat  the inherited status value.
char *resp  a user supplied buffer for message strings.
int *sFd  the file descriptor to read from.
char *buffer  the buffer to write the data to.
int sz  the number of bytes to read from socket.

RETURN(S): number of bytes read.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Peter Ruckle (pruckle), Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 1996-2002 AURA Inc. All rights reserved, Released under the GPL.

7.12. sockUtilWrite

USE: void sockUtilWrite(long *istat, char *resp, int *sFd, char *buffer, int sz);

DESCRIPTION: this function writes to the socket from a buffer. The inherited status is updated.

ARGUMENT(S):

long *istat  the inherited status value.
char *resp  a user supplied buffer for message strings.
int *sFd  the file descriptor to read from.
char *buffer  the buffer to read the data from.
int sz  the number of bytes to write to the socket.

RETURN(S): OK or ERROR.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Peter Ruckle (pruckle), Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 1996-2002 AURA Inc. All rights reserved, Released under the GPL.

DRAFT
7.13. sockUtilnWrite

USE: int sockUtilnWrite(long *istat, char *resp, int *sFd, char *buffer, int sz);

DESCRIPTION: this function writes to the socket from a buffer. The inherited status is updated.

ARGUMENT(S):

- long *istat: the inherited status value.
- char *resp: a user supplied buffer for message strings.
- int *sFd: the file descriptor to read from.
- char *buffer: the buffer to read the data from.
- int sz: the number of bytes to write to the socket.

RETURN(S): OK or ERROR.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Peter Ruckle (pruckle), Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 1996-2002 AURA Inc. All rights reserved, Released under the GPL.

7.14. sockUtilWriteString

USE: int sockUtilWriteString(long *istat, char *resp, int *sFd, char *buffer, int sz);

DESCRIPTION: this function writes to the socket from a buffer after adding a terminating NULL byte. The inherited status is updated.

ARGUMENT(S):

- long *istat: the inherited status value.
- char *resp: a user supplied buffer for message strings.
- int *sFd: the file descriptor to read from.
- char *buffer: the buffer to read the data from.
- int sz: the number of bytes to write to the socket.

RETURN(S): OK or ERROR.

LAST MODIFIED: Monday, 4 November 2002

AUTHOR(S): Peter Ruckle (pruckle), Nick Buchholz (ncb), Phil Daly (pnd)

LICENSE: (c) 1996-2002 AURA Inc. All rights reserved, Released under the GPL.
8. The comUtil 1.0.0 Library

8.1. comUtil.h

USE: `#include "comUtil.h"

DESCRIPTION: this file contains all common code required by the functions needed to build the static and dynamic comUtil libraries. These libraries abstract the communications interface to the system.

ARGUMENT(S): not applicable

RETURN(S): not applicable

LAST MODIFIED: Monday, 4 November 2002

8.2. comUtilClose

USE: `void comUtilClose(long *istat, char *resp, ulong unit);

DESCRIPTION: this function closes the DHE communications device referenced by the fd argument. The inherited status is updated and returned.

ARGUMENT(S):

- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `ulong unit` unit number.

RETURN(S): void.

LAST MODIFIED: Friday, 13 September 2002

8.3. comUtilDump

USE: `void comUtilDump(long *istat, char *resp, ulong unit);

DESCRIPTION: this function reports the status comUtil library by dumping the contents of the shared memory segment. The inherited status is updated and returned along with a message response.

ARGUMENT(S):

- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for response strings.
- `ulong unit` unit number.

RETURN(S): void.

LAST MODIFIED: Wednesday, 10 October 2002
8.4.  comUtilIOctl

USE:  void comUtilIOctl(long *istat, char *resp, ulong unit, ulong iocfunc, ...);

DESCRIPTION:  this function reads from the DHE communications device referenced by the fd argument. The inherited status is updated and returned and the number of bytes read is returned by the function. The variable argument list allows hardware specific flags, timeouts etc to be passed on.

ARGUMENT(S):

  long *istat  the inherited status value.
  char *resp  a user supplied buffer for message strings.
  ulong unit  unit number.
  ulong iocfunc  the ioctl number.
  ...  variable argument list.

VARIABLE ARGUMENT(S):

  u_long *devFlags  device specific flags passed by address.
  u_long timeout  a timeout for the read operation (in 0.01s increments).

RETURN(S):  u_long number of bytes read.

LAST MODIFIED:  Friday, 13 September 2002

8.5.  comUtilInit

USE:  void comUtilInit(long *istat, char *resp, ulong unit);

DESCRIPTION:  this function initializes the comUtil library and creates the shared memory segment for global library variables. The inherited status is updated and returned along with a message response.

ARGUMENT(S):

  long *istat  the inherited status value.
  char *resp  a user supplied buffer for message strings.
  ulong unit  unit number to initialize.

RETURN(S):  void

LAST MODIFIED:  Wednesday, 12 March 2003
8.6. comUtilOpen

USE: void comUtilOpen(long *istat, char *resp, ulong unit);

DESCRIPTION: this function opens the DHE communications device referenced by the unit argument. If the unit argument is above MNSN_MAGIC_NUMBER, the device is opened in simulation mode. The inherited status is updated and returned along with a handle in fd.

ARGUMENT(S):
   long *istat  the inherited status value.
   char *resp  a user supplied buffer for message strings.
   ulong unit  the unit number to open.

RETURN(S): void.

LAST MODIFIED: Friday, 13 September 2002

8.7. comUtilRead

USE: void comUtilRead(long *istat, char *resp, ulong unit, uchar *addr, ulong *nBytes, ...);

DESCRIPTION: this function reads from the DHE communications device referenced by the fd argument. The inherited status is updated and returned and the number of bytes read is returned by the function. The variable argument list allows hardware specific flags, timeouts etc to be passed on.

ARGUMENT(S):
   long *istat  the inherited status value.
   char *resp  a user supplied buffer for message strings.
   ulong unit  unit number.
   uchar *addr a user supplied buffer to write the data to.
   ulong *nBytes the number of bytes to read from the link.
   ... variable argument list.

VARIABLE ARGUMENT(S):
   u_long *devFlags  device specific flags passed by address.
   u_long timeout  a timeout for the read operation (in 0.01s increments).

RETURN(S): void

RETURN ARGUMENT(S):
   ulong *nBytes  the number of bytes remaining to be read.
   u_long *devFlags  device specific flags passed back by fxsl_recv.

LAST MODIFIED: Friday, 13 September 2002
8.8. comUtilSim

USE: void comUtilSim(long *istat, char *resp, ulong unit, ulong state);

DESCRIPTION: this function sets the DHE communications device into simulation mode. It can be invoked at any time. The inherited status is updated and returned.

ARGUMENT(S):
- long *istat the inherited status value.
- char *resp a user supplied buffer for message strings.
- ulong unit unit number.
- ulong state the required simulation state.

RETURN(S): void

LAST MODIFIED: Friday, 13 September 2002

8.9. comUtilUninit

USE: void comUtilUninit(long *istat, char *resp, ulong unit);

DESCRIPTION: this function de-initializes the comUtil library and destroys the shared memory segment for global library variables. The inherited status is updated and returned along with a message response.

ARGUMENT(S):
- long *istat the inherited status value.
- char *resp a user supplied buffer for message strings.
- ulong unit unit number.

RETURN(S): void

LAST MODIFIED: Wednesday, 10 October 2002

8.10. comUtilWrite

USE: void comUtilWrite(long *istat, char *resp, ulong unit, uchar *addr, ulong *nBytes, ...);

DESCRIPTION: this function writes to the DHE communications device referenced by the fd argument. The inherited status is updated and the number of bytes not written (i.e. remaining in the buffer) is returned by the function, Thus a function return of 0 indicates all bytes were written successfully. The variable argument list allows hardware specific flags, timeouts etc to be passed on.

ARGUMENT(S):
- long *istat the inherited status value.
- char *resp a user supplied buffer for message strings.
- ulong unit unit number.
uchar *addr a user supplied buffer to read the data from.
ulong *nBytes the number of bytes to write to the link.
... variable argument list.

ARGUMENT(S):
ulong *devFlags device specific flags passed by address.
ulong timeout a timeout for the write operation (in 0.01s increments).

RETURN(S): ulong number of bytes left un-written

LAST MODIFIED: Friday, 13 September 2002
9. The dheUtil 1.0.0 Library

9.1. dheUtil.h

USE: #include "dheUtil.h"

DESCRIPTION: this file contains all common code required by the functions needed to build the static and dynamic dheUtil libraries. These libraries abstract the DHE interface to the system.

ARGUMENT(S): not applicable

RETURN(S): not applicable

LAST MODIFIED: Monday, 4 November 2002

9.2. dheUtilOpen

USE: void dheUtilOpen(long *istat, char *resp, ulong unit, int *fd, ulong (**cbfunc()));

DESCRIPTION: this function opens the DHE and communications device referenced by the unit argument, a Handle to the dhe is returned in the fd argument. The routine runs the dheHdwrInit and dheHdwrOpen routine to get the dhe ready for use The inherited status is updated and returned.

ARGUMENT(S):
- long *istat  the inherited status value.
- char *resp  a user supplied buffer for message strings.
- ulong unit  the unit number to open.
- int *fd     the handle to the open DHE device.
- ulong (**cbfunc()) a call back function.

RETURN(S): void.

AUTHOR(S): Nick Buchholz (ncb), Phil Daly (pnd)

LAST MODIFIED: Tuesday, 19 November 2002

LICENSE: (c) 2002 AURA Inc. All rights reserved. Released under the GPL.

9.3. dheUtilClose

USE: void dheUtilClose(long *istat, char *resp, int fd);

DESCRIPTION: this function closes the DHE communications device referenced by the fd argument. The inherited status is updated and returned.

ARGUMENT(S):
- long *istat  the inherited status value.
- char *resp  a user supplied buffer for message strings.
- int fd      the handle to the open DHE device.
RETURN(S): void.

AUTHOR(S): Nick Buchholz (ncb), Phil Daly (pnd)

LAST MODIFIED: Tuesday, 19 November 2002

LICENSE: (c) 2002 AURA Inc. All rights reserved. Released under the GPL.

9.4. dheUtilSend

USE: `void dheUtilSend(long *istat, char *resp, int fd, char *msg, u_long *len );`

DESCRIPTION: this function closes the DHE communications device referenced by the fd argument. The inherited status is updated and returned.

ARGUMENT(S):

- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for response strings.
- `int fd` the handle to the open DHE device.
- `char *msg` a user supplied buffer for messages.
- `u_long *len` the length of the msg buffer.

RETURN(S): void.

AUTHOR(S): Nick Buchholz (ncb), Phil Daly (pnd)

LAST MODIFIED: Tuesday, 19 November 2002

LICENSE: (c) 2002 AURA Inc. All rights reserved. Released under the GPL.

9.5. dheUtilRecv

USE: `void dheUtilRecv(long *istat, char *resp, int fd, u_long *msg, u_long *len, ulong cmd );`

DESCRIPTION: this function receives a message from the DHE. It uses the comUtil communications device previously opened by dhwUtilOpen referenced by the dheId argument. The routine waits for len bytes or a timeout on the link. The received message is stored in msg[0 thru len-1] The inherited status parameter and response string are updated and returned.

ARGUMENT(S):

- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for response strings.
- `int fd` the handle to the open DHE device.
- `u_long *msg` a user supplied buffer for messages.
- `u_long *len` the length of the msg buffer in bytes.
- `u_long *len` the length of the msg buffer in bytes.

RETURN(S): void.
9.6. dheUtilGetData

USE:  

```c
void dheUtilGetData(long *istat, char *resp, int fd, u_long *addr, u_long *nPixels, u_Long pSize, u_long delay, u_long timeout);
```

DESCRIPTION: this function closes the DHE communications device referenced by the fd argument. The inherited status is updated and returned.

ARGUMENT(S):
- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `int fd` the handle to the open DHE device.
- `u_long *addr` the user supplied address.
- `u_long *nPixels` the updated number of pixels.
- `u_long pSize` the pixel size.
- `u_long delay` the delay.
- `u_long timeout` the timeout.

RETURN(S): void.

9.7. dheUtilDownloadWF

USE:  

```c
void dheUtilDownloadWF(long *istat, char *resp, int fd, u_short *waddr, u_short *saddr, u_long nWords);
```

DESCRIPTION: this function closes the DHE communications device referenced by the fd argument. The inherited status is updated and returned.

ARGUMENT(S):
- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `int fd` the handle to the open DHE device.
- `u_short *waddr` the address of the waveform.
- `u_short *saddr` the start address.
\textit{ulong nWords} the number of words.

\textbf{RETURN(S):} void.

\textbf{AUTHOR(S):} Nick Buchholz (ncb), Phil Daly (pnd)

\textbf{LAST MODIFIED:} Tuesday, 19 November 2002

\textbf{LICENSE:} (c) 2002 AURA Inc. All rights reserved. Released under the GPL.

\section{dheUtilIOctl}

\textbf{USE:} \texttt{void dheUtilIOctl(long \*istat, char \*resp, int fd, ulong iofunc, ... );}

\textbf{DESCRIPTION:} this function closes the DHE communications device referenced by the \texttt{fd} argument. The inherited status is updated and returned.

\textbf{ARGUMENT(S):}

\begin{itemize}
  \item \texttt{long \*istat} the inherited status value.
  \item \texttt{char \*resp} a user supplied buffer for message strings.
  \item \texttt{int fd} the handle to the open DHE device.
  \item \texttt{ulong iofunc} the ioctl function to service.
  \item ... the variable argument list.
\end{itemize}

\textbf{RETURN(S):} void.

\textbf{AUTHOR(S):} Nick Buchholz (ncb), Phil Daly (pnd)

\textbf{LAST MODIFIED:} Tuesday, 19 November 2002

\textbf{LICENSE:} (c) 2002 AURA Inc. All rights reserved. Released under the GPL.

\section{dheUtilInit}

\textbf{USE:} \texttt{void dheUtilInit(long \*istat, char \*resp);}

\textbf{DESCRIPTION:} this function initializes the dheUtil library and creates the shared memory segment for global library variables. The inherited status is updated and returned along with a message response.

\textbf{ARGUMENT(S):}

\begin{itemize}
  \item \texttt{long \*istat} the inherited status value.
  \item \texttt{char \*resp} a user supplied buffer for message strings.
\end{itemize}

\textbf{RETURN(S):} void

\textbf{LAST MODIFIED:} Wednesday, 10 October 2002
9.10. dheUtilUninit

USE: void dheUtilUninit(long *istat, char *resp);

DESCRIPTION: this function initializes the dheUtil library and creates the shared memory segment for global library variables. The inherited status is updated and returned along with a message response.

ARGUMENT(S):
- long *istat  the inherited status value.
- char *resp  a user supplied buffer for message strings.

RETURN(S): void

LAST MODIFIED: Wednesday, 10 October 2002

9.11. readValue

USE: void readValue(long *status, char *response, dheHandle dheId, ulong attribAddr, long *attribValue);

DESCRIPTION: this function closes the DHE communications device referenced by the fd argument. The inherited status is updated and returned.

ARGUMENT(S):
- long *status  the inherited status value.
- char *response  a user supplied buffer for response strings.
- dheHandle dheId  the handle to the open DHE device.
- ulong attribAddr  dhe address for the request attribute.
- long *attribValue  the returned value of the attribute.

RETURN(S): void.

AUTHOR(S): Nick Buchholz (ncb), Phil Daly (pnd)

LAST MODIFIED: Monday, 20030203

LICENSE: (c) 2002 AURA Inc. All rights reserved. Released under the GPL.

9.12. writeValue

USE: void writeValue(long *status, char *response, dheHandle dheId, ulong attribAddr, long attribValue);

DESCRIPTION: this function closes the DHE communications device referenced by the fd argument. The inherited status is updated and returned.

ARGUMENT(S):
- long *status  the inherited status value.
char *response  a user supplied buffer for response strings.
dheHandle dheId  the handle to the open DHE device.
ulong attribAddr  the address for the requested attribute.
long attribValue  value of the attribute to write.

RETURN(S):  void.

AUTHOR(S):  Nick Buchholz (ncb), Phil Daly (pnd)

LAST MODIFIED:  Monday, 20030203

LICENSE:  (c) 2002 AURA Inc. All rights reserved. Released under the GPL.

9.13.  asyncResponse

USE:  void asyncResponse(long *status, char *response, dheHandle dheId, ulong asyncMessage );

DESCRIPTION:  this function sends an Asynchronous response message to the DHE device referenced by the dheId argument. The inherited status is updated and returned.

ARGUMENT(S):
   long *status  the inherited status value.
   char *response  a user supplied buffer for response strings.
   dheHandle dheId  the handle to the open DHE device.
   ulong asyncMessage  30 bit value written with the asyncResponse bits set.

RETURN(S):  void.

AUTHOR(S):  Nick Buchholz (ncb), Phil Daly (pnd)

LAST MODIFIED:  Monday, 20030303

LICENSE:  (c) 2002 AURA Inc. All rights reserved. Released under the GPL.

9.14.  startExp

USE:  void startExp(long *status, char *response, dheHandle dheId, long *dheState, ulong expCode );

DESCRIPTION:  this function sends a start Exposure message to the DHE device referenced by the dheId argument. The inherited status is updated and returned.

ARGUMENT(S):
   long *status  the inherited status value.
   char *response  a user supplied buffer for response strings.
   dheHandle dheId  the handle to the open DHE device.
   ulong expCode  8 bit value written to the DHE.
9.15. abortExp

**USE:**
```c
void abortExp(long *status, char *response, dheHandle dheId, long *dheState, ulong aAddr, long aValue);
```

**DESCRIPTION:**
this function sends a message containing the Value aValue to the DHE address aAddr. This should result in the DHE aborting any active exposure. The inherited status and response message are updated and returned.

**ARGUMENT(S):**
- `long *status` the inherited status value.
- `char *response` a user supplied buffer for response strings.
- `dheHandle dheId` the handle to the open DHE device.
- `ulong aAddr` the DHE address to write the value to.
- `long aValue` The abort Value to write.

**RETURN(S):** `void`.

9.16. pauseExp

**USE:**
```c
void pauseExp(long *status, char *response, dheHandle dheId, ulong expCode);
```

**DESCRIPTION:**
this function sends an Asynchronous response message to the DHE device referenced by the dheId argument. The inherited status is updated and returned.

**ARGUMENT(S):**
- `long *status` the inherited status value.
- `char *response` a user supplied buffer for response strings.
- `dheHandle dheId` the handle to the open DHE device.
- `ulong expCode` 30 bit value written with the pauseExp bits set.

**RETURN(S):** `void`.

**AUTHOR(S):** Nick Buchholz (ncb), Phil Daly (pnd)

**LAST MODIFIED:** Monday, 20030808

**LICENSE:** (c) 2002 AURA Inc. All rights reserved. Released under the GPL.
9.17. resumeExp

USE: void resumeExp(long *status, char *response, dheHandle dheId, ulong expCode);

DESCRIPTION: this function sends an Asynchronous response message to the DHE device referenced by the dheId argument. The inherited status is updated and returned.

ARGUMENT(S):
- long *status the inherited status value.
- char *response a user supplied buffer for response strings.
- dheHandle dheId the handle to the open DHE device.
- ulong expCode 30 bit value written with the resumeExp bits set.

RETURN(S): void.

AUTHOR(S): Nick Buchholz (ncb), Phil Daly (pnd)

LAST MODIFIED: Monday, 20030303

LICENSE: (c) 2002 AURA Inc. All rights reserved. Released under the GPL.

9.18. stopExp

USE: void stopExp(long *status, char *response, dheHandle dheId, ulong expCode);

DESCRIPTION: this function sends an Asynchronous response message to the DHE device referenced by the dheId argument. The inherited status is updated and returned.

ARGUMENT(S):
- long *status the inherited status value.
- char *response a user supplied buffer for response strings.
- dheHandle dheId the handle to the open DHE device.
- ulong expCode 30 bit value written with the stopExp bits set.

RETURN(S): void.

AUTHOR(S): Nick Buchholz (ncb), Phil Daly (pnd)

LAST MODIFIED: Monday, 20030303

LICENSE: (c) 2002 AURA Inc. All rights reserved. Released under the GPL.

DRAFT
9.19. **armExpTrigger**

**USE:** void armExpTrigger(long *status, char *response, dheHandle dheId, long *dheState, ulong expCode);

**DESCRIPTION:** this function sends a start Exposure message to the DHE device referenced by the dheId argument. The inherited status is updated and returned.

**ARGUMENT(S):**
- long *status  the inherited status value.
- char *response  a user supplied buffer for response strings.
- dheHandle dheId  the handle to the open DHE device.
- ulong expCode  8 bit value written to the DHE.

**RETURN(S):** void.

**AUTHOR(S):** Nick Buchholz (ncb), Phil Daly (pnd)

**LAST MODIFIED:** Monday, 20030808

**LICENSE:** (c) 2002 AURA Inc. All rights reserved. Released under the GPL.
10. The comHdwr Library

10.1. comHdwr.h

USE: 

```
#include "comHdwr.h"
```

DESCRIPTION: this file contains the common definitions required to abstract the communications hardware API. In particular, this file defined the structures used to access hardware and the prototypes of the access functions. Each communications hardware library must implement the full set of functions.

PROTOTYPE(S):

```
void hdwrClose ( long *istat, char *resp, int fd );
void hdwrDump ( long *istat, char *resp );
void hdwrConfig ( long *istat, char *resp, uchar flag, hConfig *p config );
void hdwrStatus ( long *istat, char *resp, hStatus *p status );
void hdwrInit ( long *istat, char *resp );
void hdwrIOctl ( long *istat, char *resp, ulong iofunc, va_list nArgs );
void hdwrOpen ( long *istat, char *resp, ulong unit, int *fd );
void hdwrUninit ( long *istat, char *resp );
void hdwrRead ( long *istat, char *resp, int fd, uchar *addr, ulong *nBytes, va_list nArgs );
void hdwrWrite ( long *istat, char *resp, int fd, uchar *addr, ulong *nBytes, va_list nArgs );
```

ARGUMENT(S):

```
long *istat the inherited status.
char *resp the pre-defined response string.
int fd a input file descriptor.
int *fd a output file descriptor.
uchar flag a direction flag (set, get).
hConfig *p config a pointer to a configuration structure.
hStat *p status a pointer to a status structure.
ulong iofunc an ioctl value to select a particular function.
va_list nArgs variable argument list.
ulong unit] the communications link unit number.
uchar *addr the supplied address to read from or write to.
ulong *nBytes the number of bytes to read or write (updated).
ulong flags any hardware flag passed to a read or write routine via variable argument list.
ulong timeout any timeout value passed to a read or write routine via variable argument list.
```
STRUCTURE(S):

typedef struct {
    hUnion_t bits[HDWR_ITEMS];
} hBits_t, *hBits_p, **hBits_s;

typedef struct {
    hUnion_t flags[HDWR_ITEMS];
} hFlags_t, *hFlags_p, **hFlags_s;

typedef struct {
    hUnion_t regs[HDWR_ITEMS];
} hRegs_t, *hRegs_p, **hRegs_s;

typedef struct {
    char msg[HDWR_MAXSTR];
    char driver[HDWR_MAXSTR];
    int status;
    int nBoard;
    int nBus;
    int nSlot;
    int nMajor;
    int nMinor;
    int linkCondition;
    int linkError;
    hUnion_t linkControl;
    hUnion_t linkStatus;
    hUnion_t linkFlags;
    hUnion_t linkThreshold;
    hUnion_t reserved[HDWR_ITEMS];
} hStatus_t, *hStatus_p, **hStatus_s;

typedef struct {
    char msg[HDWR_MAXSTR];
    char driver[HDWR_MAXSTR];
    int sendTimeout;
    int recvTimeout;
    int linkMode;
    int linkHaltOnError;
    int linkQueueOnError;
}
int useFlowControl;
int useCRC;
int useByteSwap;
    hUnion_t reserved[HDWR_ITEMS];
} hConfig_t, *hConfig_p, **hConfig_s:

RETURN(S): void, inherited status is updated.

LAST MODIFIED: Thursday, 14 November 2002.

AUTHOR(S): Phil Daly (pnd), Nick Buchholz (ncb).

LICENSE: (c) 2002 AURA Inc. All rights reserved. Released under the GPL.
11. The dheHdwr Library

11.1. dheHdwr.h

USE: #include “dheHdwr.h”

DESCRIPTION: this file contains the common definitions required to abstract the DHE hardware API. In particular, this file defines the structures used to access hardware and the prototypes of the access functions. Each DHE hardware library must implement the full set of functions.

PROTOTYPE(S):

void dheHdwrOpen ( long *status, char *response, ulong unitNum, dheHandle *dheId );
void dheHdwrClose ( long *status, char *response, dheHandle dheId );
void dheHdwrSend ( long *status, char *response, dheHandle dheId, uchar *dheMsg, long *length, ulong cmd);
void dheHdwrRecv ( long *status, char *response, dheHandle dheId, uchar *dheMsg, long *length, ulong cmd);
void dheHdwrGetData ( long *status, char *response, dheHandle dheId, ulong *pxlAddr , ulong *numPixels, long pixelSize, ulong delay, ulong timeout);
void dheHdwrDownLoadWF ( long *status, char *response, dheHandle dheId, uchar *wfBuffer , ushort startAddr, long numWords);
void dheHdwrIOctl ( long *status, char *response, dheHandle dheId, ulong cmdFunction, va_list args);
void dheHdwrInit ( long *status, char *response );
void dheHdwrUninit ( long *status, char *response, dheHandle dheId );
void dheHdwrDump ( long *status, char *response, dheHandle dheId );

ARGUMENT(S):

long *status the inherited status.
char *response the pre-defined response string.
int dheId a hardware driver file descriptor.
int *dheId an output file descriptor.
ulong cmdFunction an ioctl value to select a particular function.
ulong nArgs the number of argument to follow (for variable argument lists).
va_list vArgs variable argument list.
ulong unitNum] the communications link unit number.
ulong *pxlAddr the address to store the pixels into.
ulong *numPixels - on input the number of pixels to capture , on return the number of pixels captured.
long pixelSize - a set of values designating the size of the pixels expected.
ulong delay the number of seconds to wait before doing the comHdwRead for the pixels.
ulong timeout a timeout value passed to the comHdwRead routine for pixel reads.
uchar *startAddr the supplied address to store the waveforms to.
ulong *numWords the number of shorts to write when updating the waveform table.

STRUCTURE(S):

typedef struct dheHdwrStatus {
    char msg[HDWR_MAXSTR];
    char hdwrID[HDWR_MAXSTR];
    long state;
    dHUnion_t reserved[HDWR_ITEMS];
} dheHdwrStat_t, *dheHdwrStat_p, **dheHdwrStat_s;
typedef struct dheHdwrConfig {
    char cfgID[HDWR_MAXSTR];
    int sendTimeout;
    int recvTimeout;
    long state;
    dHUnion_t reserved[HDWR_ITEMS];
} dheHdwrCfg_t, *dheHdwrCfg_p, **dheHdwrCfg_s;
typedef struct dheHdwrControl {
    int handle;
    long unitNum;
    int comHandle;
    ulong comFlags;
    ulong comTimeOut;
    long numProc;
    long state;
    long dheHdwState;
    long dheSftwrState;
    long simulation;
    dheHdwrCfg_t hdwrConfig;
    dheHdwrStat_t hdwrStatus;
    dheHdwrFlags_t hdwrFlags;
    dheHdwrRegs_t hdwrRegisters;
    ulong (*callback)();
dHUnion_t reserved[HDWR_ITEMS];
}
dheHdwrCtrl_t, *dheHdwrCtrl_p, **dheHdwrCtrl_s;

RETURN(S): void, inherited status is updated.

LAST MODIFIED: Thursday, 14 November 2002.

AUTHOR(S): Phil Daly (pnd), Nick Buchholz (ncb).

LICENSE: (c) 2002 AURA Inc. All rights reserved. Released under the GPL.
12. The systranUtil Library

12.1. systranUtil.h

USE: 
#include “systranUtil.h”

DESCRIPTION: this file contains all common code required by the functions needed to build the static and dynamic systranUtil libraries. These libraries abstract the Systran interface to the system.

ARGUMENT(S): not applicable

RETURN(S): not applicable

LAST MODIFIED: Monday, 4 November 2002

12.2. comHdwClose

USE: void comHdwClose(long *istat, char *resp, int fd);

DESCRIPTION: this function closes the given file descriptor. The inherited status is updated.

ARGUMENT(S):
- long *istat the inherited status value.
- char *resp a user supplied buffer for message strings.
- int fd the file descriptor to close.

RETURN(S): void.

LAST MODIFIED: Monday, 4 November 2002.

AUTHOR(S): Phil Daly (pnd).

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

12.3. comHdwConfig

USE: void comHdwConfig(long *istat, char *resp, int fd, u_char flag, hConfig *p config);

DESCRIPTION: this function get/sets the hardware configuration in the generic structure. The inherited status is updated. This function is deprecated: use hdwrIOctl instead.

ARGUMENT(S):
- long *istat the inherited status value.
- char *resp a user supplied buffer for message strings.
- int fd the file descriptor to read from.
- u_char flag FALSE=get, TRUE=set.
- hConfig *p config the updated configuration structure.

RETURN(S): void.
12.4. comHdwDump

USE: void comHdwDump(long *istat, char *resp, int fd);

DESCRIPTION: this function dumps hardware information onto stdout. The inherited status is updated.

ARGUMENT(S):

- long *istat the inherited status value.
- char *resp a user supplied buffer for message strings.
- int fd the file descriptor to read from.

RETURN(S): void.

12.5. comHdwIOctl

USE: void hwtrConfig(long *istat, char *resp, int fd, ulong iofunc, va_list nArgs);

DESCRIPTION: this function performs IOCTL style i/o on the device. The inherited status is updated.

ARGUMENT(S):

- long *istat the inherited status value.
- char *resp a user supplied buffer for message strings.
- int fd the file descriptor to read from.
- ulong iofunc the IOCTL code to use.
- va_list nArgs the variable argument list.

RETURN(S): void.
12.6. **comHdwInit**

**USE:** `void comHdwInit(long *istat, char *resp);`

**DESCRIPTION:** this function initializes the hardware. The inherited status is updated.

**ARGUMENT(S):**

- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.

**RETURN(S):** `void`.

**LAST MODIFIED:** Monday, 4 November 2002.

**AUTHOR(S):** Phil Daly (pnd).

**LICENSE:** (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

12.7. **comHdwOpen**

**USE:** `void comHdwOpen(long *istat, char *resp, ulong unit, int *fd);`

**DESCRIPTION:** this function opens the given unit and returns a file descriptor. The inherited status is updated.

**ARGUMENT(S):**

- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `ulong unit` the unit number.
- `int *fd` the returned file descriptor.

**RETURN(S):** `void`.

**LAST MODIFIED:** Monday, 4 November 2002.

**AUTHOR(S):** Phil Daly (pnd).

**LICENSE:** (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

12.8. **comHdwRead**

**USE:** `void comHdwRead (long *istat, char *resp, int fd, uchar *addr, ulong *nBytes, va_list nArgs );`

**DESCRIPTION:** reads a number of bytes from a buffer. The inherited status and number of bytes read are updated.

**ARGUMENT(S):**

- `long *istat` inherited status.
- `char *resp` response message.
int fd  file descriptor.
uchar *addr  supplied address.
ulong *nBytes  updates number of bytes to read.
va_list nArgs  variable argument list.

RETURN(S):  void.

LAST MODIFIED:  Friday, 15 November 2002.

AUTHOR(S):  Phil Daly (pnd).

LICENSE:  (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

12.9.  comHdwStatus

USE:  void comHdwStatus(long *isstat, char *resp, int fd, hStatus_p status);

DESCRIPTION:  this function returns the hardware status in the generic structure. The inherited
status is updated. This function is deprecated: use hdwrIOctl instead.

ARGUMENT(S):
    long *isstat  the inherited status value.
    char *resp  a user supplied buffer for message strings.
    int fd  the file descriptor to read from.
    hStatus_p status  the updated status structure.

RETURN(S):  void.

LAST MODIFIED:  Monday, 4 November 2002.

AUTHOR(S):  Phil Daly (pnd).

LICENSE:  (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

12.10.  comHdwUninit

USE:  void comHdwUninit(long *isstat, char *resp);

DESCRIPTION:  this function un-initializes the hardware. The inherited status is updated.

ARGUMENT(S):
    long *isstat  the inherited status value.
    char *resp  a user supplied buffer for message strings.

RETURN(S):  void.

LAST MODIFIED:  Monday, 4 November 2002.

AUTHOR(S):  Phil Daly (pnd).

LICENSE:  (c) 2002 AURA Inc. All rights reserved, Released under the GPL.
12.11. **comHdwWrite**

**USE:** 
```c
void comHdwWrite (long *istat, char *resp, int fd, uchar *addr, ulong *nBytes, va_list nArgs );
```

**DESCRIPTION:** write a number of bytes to a buffer. The inherited status and number of bytes written are updated.

**ARGUMENT(S):**
- `long *istat` inherited status.
- `char *resp` response message.
- `int fd` file descriptor.
- `uchar *addr` supplied address.
- `ulong *nBytes` updates number of bytes to read.
- `va_list nArgs` variable argument list.

**RETURN(S):** void.

**LAST MODIFIED:** Friday, 15 November 2002.

**AUTHOR(S):** Phil Daly (pnd).

**LICENSE:** (c) 2002 AURA Inc. All rights reserved, Released under the GPL.
13. The simHdw 1.0.0 Library

13.1. simHdw.h

USE: #include “simHdw.h”

DESCRIPTION: this file contains all common code required by the functions needed to build the static and dynamic simHdw libraries. These libraries abstract the hardware communications simulator interface to the system.

ARGUMENT(S): not applicable.

RETURN(S): not applicable.

LAST MODIFIED: Monday, 4 November 2002.

13.2. hdwrClose

USE: void hdwrClose(long *istat, char *resp, int fd);

DESCRIPTION: this function closes the given file descriptor. The inherited status is updated.

ARGUMENT(S):

    long *istat  the inherited status value.
    char *resp   a user supplied buffer for message strings.
    int fd      the file descriptor to close.

RETURN(S): void.

LAST MODIFIED: Monday, 4 November 2002.

AUTHOR(S): Phil Daly (pnd).

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

13.3. hdwrConfig

USE: void hdwrConfig(long *istat, char *resp, int fd, u_char flag, hConfig *p config);

DESCRIPTION: this function get/sets the hardware configuration in the generic structure. The inherited status is updated. This function is deprecated: use hdwrIOctl instead.

ARGUMENT(S):

    long *istat  the inherited status value.
    char *resp   a user supplied buffer for message strings.
    int fd      the file descriptor to read from.
    u_char flag FALSE=get, TRUE=set.
    hConfig *p config  the updated configuration structure.

RETURN(S): void.
13.4. **hdwrDump**

**USE:** `void hdwrDump(long *istat, char *resp, int fd );`

**DESCRIPTION:** This function dumps hardware information onto stdout. The inherited status is updated.

**ARGUMENT(S):**
- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `int fd` the file descriptor to read from.

**RETURN(S):** `void`.

LAST MODIFIED: Monday, 4 November 2002.

AUTHOR(S): Phil Daly (pnd).

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

13.5. **hdwrIOctl**

**USE:** `void hdwrConfig(long *istat, char *resp, int fd, ulong iofunc, va_list nArgs );`

**DESCRIPTION:** This function performs IOCTL style i/o on the device. The inherited status is updated.

**ARGUMENT(S):**
- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `int fd` the file descriptor to read from.
- `ulong iofunc` the IOCTL code to use.
- `va_list nArgs` the variable argument list.

**RETURN(S):** `void`.

LAST MODIFIED: Monday, 4 November 2002.

AUTHOR(S): Phil Daly (pnd).

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.
13.6. **hdwrInit**

**USE:** `void hdwrInit(long *istat, char *resp);`

**DESCRIPTION:** this function initializes the hardware. The inherited status is updated.

**ARGUMENT(S):**

- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.

**RETURN(S):** `void`.

**LAST MODIFIED:** Monday, 4 November 2002.

**AUTHOR(S):** Phil Daly (pnd).

**LICENSE:** (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

13.7. **hdwrOpen**

**USE:** `void hdwrOpen(long *istat, char *resp, ulong unit, int *fd);`

**DESCRIPTION:** this function opens the given unit and returns a file descriptor. The inherited status is updated.

**ARGUMENT(S):**

- `long *istat` the inherited status value.
- `char *resp` a user supplied buffer for message strings.
- `ulong unit` the unit number.
- `int *fd` the returned file descriptor.

**RETURN(S):** `void`.

**LAST MODIFIED:** Monday, 4 November 2002.

**AUTHOR(S):** Phil Daly (pnd).

**LICENSE:** (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

13.8. **hdwrRead**

**USE:** `void hdwrRead (long *istat, char *resp, int fd, uchar *addr, ulong *nBytes, va_list nArgs );`

**DESCRIPTION:** reads a number of bytes from a buffer. The inherited status and number of bytes read are updated.

**ARGUMENT(S):**

- `long *istat` inherited status.
- `char *resp` response message.
- `int fd` file descriptor.
uchar *addr supplied address.
ulong *nBytes updates number of bytes to read.
va_list nArgs variable argument list.

RETURN(S): void.

LAST MODIFIED: Friday, 15 November 2002.

AUTHOR(S): Phil Daly (pnd).

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

13.9. hdwrStatus

USE: void hdwrStatus(long *istat, char *resp, int fd, hStatus p status);

DESCRIPTION: this function returns the hardware status in the generic structure. The inherited status is updated. This function is deprecated: use hdwrIOctl instead.

ARGUMENT(S):

long *istat the inherited status value.
char *resp a user supplied buffer for message strings.
int fd the file descriptor to read from.
hStatus p status the updated status structure.

RETURN(S): void.

LAST MODIFIED: Monday, 4 November 2002.

AUTHOR(S): Phil Daly (pnd).

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

13.10. hdwrUninit

USE: void hdwrUninit(long *istat, char *resp);

DESCRIPTION: this function un-initializes the hardware. The inherited status is updated.

ARGUMENT(S):

long *istat the inherited status value.
char *resp a user supplied buffer for message strings.

RETURN(S): void.

LAST MODIFIED: Monday, 4 November 2002.

AUTHOR(S): Phil Daly (pnd).

LICENSE: (c) 2002 AURA Inc. All rights reserved, Released under the GPL.

DRAFT
13.11. **hdwrWrite**

**USE:** 
```c
void hdwrWrite (long *istat, char *resp, int fd, uchar *addr, ulong *nBytes, va_list nArgs );
```

**DESCRIPTION:** write a number of bytes to a buffer. The inherited status and number of bytes written are updated.

**ARGUMENT(S):**
- `long *istat` inherited status.
- `char *resp` response message.
- `int fd` file descriptor.
- `uchar *addr` supplied address.
- `ulong *nBytes` updates number of bytes to read.
- `va_list nArgs` variable argument list.

**RETURN(S):** void.

**LAST MODIFIED:** Friday, 15 November 2002.

**AUTHOR(S):** Phil Daly (pnd).

**LICENSE:** (c) 2002 AURA Inc. All rights reserved, Released under the GPL.
14. Document Revision History


Acknowledgments. Linux is a registered trade mark of Linus Torvalds. FibreXtreme is a registered trade mark of Systran Corporation.

References

