

## Alarms Utility

---

Access to the Netra ft 1800 alarms is provided by the `u4ftalctl(1M)` utility, which enables you to toggle the alarms and UNIX-running watchdog, and report their status.

You can implement `u4ftalctl` utility directly from the command line using the following format. You can also incorporate these commands into a shell script.

- To set or clear alarms:

```
% u4ftalctl alarm=[off|on]
```

- To set or clear watchdog:

```
% u4ftalctl watchdog=[off|on]
```

- To reset alarms or UNIX-running watchdog:

```
% u4ftalctl reset
```

- To report status of alarms and UNIX-running watchdog (in fault-tolerant mode):

```
% u4ftalctl status  
watchdog=off  
alarm0=off  
alarm1=off  
alarm2=off  
alarm3=unavailable  
alarm4=unavailable  
alarm5=unavailable
```

- To report status of alarms and UNIX-running watchdog (in nonfault-tolerant mode):

```
% u4ftalctl status
watchdog=off
alarm0=off
alarm1=off
alarm2=off
alarm3=off
alarm4=off
alarm5=off
```

The following example shows how to assign the status of each alarm to a (Bourne or Korn) shell variable:

```
% eval u4ftalctl status
% echo $alarm1
off
```

The following code example shows how to set and clear an alarm.

#### CODE EXAMPLE D-1 Setting and Clearing an Alarm

```
/*
 * alarmctl.c
 *
 * program to set/clear alarms
 * synopsis - this program expects 2 args :- alarm# and setting
 *
 */

#include <unistd.h>
#include <sys/ioccom.h>
#include <fcntl.h>
#include <errno.h>
#include "u4ftalarm_io.h"

char *dev = "/dev/u4ftalarm:ctl";

int main(int argc, char **argv)
{
    int fd;
    int read_state = 0;
    u4ft_aldata_t *ad = NULL;
```

**CODE EXAMPLE D-1** Setting and Clearing an Alarm (*Continued*)

```
if (argc < 3) {
    printf("Usage: alarmctl [alarm#] [on/off]\n");
    perror(argv[0]);
    exit(1);
}

if ((fd = open(dev, O_RDONLY)) < 0) {
    printf("open failed\n");
    perror(argv[0]);
    exit(1);
}

ad = (u4ft_aldata_t *)calloc(1, sizeof(u4ft_aldata_t));
if (ad == NULL) {
    printf("%s: calloc failed\n", argv[0]);
    perror(argv[0]);
    exit(1);
}

ad->u4ftalarm_no = atoi(argv[1]);
if (ad->u4ftalarm_no < 0 || ad->u4ftalarm_no > 5) {
    printf("%s :- invalid alarm number\n", argv[0]);
    exit(1);
}

if (strcmp(argv[2], "on") == 0)
    ad->u4ftalarm_state = 1;
else if (strcmp(argv[2], "off") == 0)
    ad->u4ftalarm_state = 0;
else if (strcmp(argv[2], "state") == 0)
    read_state = 1;
else {
    printf("%s :- invalid alarm state\n", argv[0]);
    exit(1);
}

if (read_state) {
    if (ioctl(fd, U4FTIOCALSTATE, ad) < 0) {
        printf("U4FTIOCALSTATE failed for alarm %d\n", ad-
>u4ftalarm_no);
        perror(argv[0]);
    } else {
        printf("The state of alarm %d is %d\n", ad->u4ftalarm_no,
ad->u4ftalarm_state);
    }
} else if (ioctl(fd, U4FTIOCALCTL, ad) < 0) {
    printf("U4FTIOCALCTL failed for alarm %d\n", ad->u4ftalarm_no);
}
```

**CODE EXAMPLE D-1** Setting and Clearing an Alarm *(Continued)*

```
        perror(argv[0]);  
    }  
    close(fd);  
}
```