Appendix A. Audio IRIG Receiver Programming Information

The following sections describe programming interfaces and calling sequences for the modified BSD audio driver for the Sun SPARCstation originally written by Van Jacobson and collaborators at Lawrence Berkeley National Laboratory. The modifications, written by Roy LeCates of the University of Delaware, provide for the connection of a standard Inter-Range Instrumentation Group (IRIG-B) timecode signal and the decoding of the signal to synchronize the local clock to the IRIG signal. The NTP Version 3 distribution supports the modified BSD driver when installed in the SunOS 4.1.x kernel.

The IRIG receiver modifications are integrated in the BSD audio driver bsd_audio.c without affecting its usual functions in transmitting and receiving ordinary speech, except when enabled by specific ioctl() system calls. However, the driver cannot be used for both speech and IRIG signals at the same time. Once activated by a designated ioctl() call, the driver remains active until it is explicitly deactivated by another ioctl() call. This allows applications to configure the audio device and pass the pre-configured driver to other applications. Since the driver is currently only a receiver, it does not affect the operation of the BSD audio output driver.

Data are read using the standard read() system call. Since the output formats have constant lengths, the application receives the data into a fixed-length buffer or structure. The read() call never blocks; it simply returns the most recent IRIG data received during the last second. It may happen that, due to unavoidable race conditions, data for other than the most recent second are returned. However, all data in the structure are updated as an atomic unit; thus, the entire structure is valid, even if it contains old data. The software daemon or driver can determine the validity from a status byte returned with the structure.

A.1. Application Program Interface

The header file bsd_audioirig.h defines the irig_time structure and ioctl() codes used by the receiver. Following are those codes specific to the IRIG functions of the BSD audio driver. Unless indicated otherwise, the (third) argument of the ioctl() system call points to an integer or string.

AUDIO_IRIG_OPEN

> This command activates the IRIG receiver. The BSD audio driver must be opened with this command before other commands can be issued. The argument is ignored. When the IRIG receiver is initialized, all internal data are purged and any buffered data are lost.

AUDIO_IRIG_CLOSE

> This command deactivates the IRIG receiver. The argument is ignored. The buffers are purged and any buffered time data are lost. The original BSD audio driver functions are enabled and it resumes operating normally.

AUDIO_IRIG_SETFORMAT

> The argument is a pointer to an integer designating the output format for the IRIG data. There are currently two formats defined, 0 (default) and 1. If an invalid format is selected, the default format is used.

The data returned by a read() system call in format 0 is a character string in the format “ddd hh:mm:ss*\n”, which consists of 13 ASCII characters followed by a newline terminator for a total of 14 characters. The “*” status character is an ASCII space “ ” if the status byte determined by the receiver is zero and “?” if not. This format is intended to be used with simple user programs that care only about the time to the nearest second.

The data returned by a read() system call in format 1 is a structure defined in the bsd_audioirig.h header file:

```
struct irig_time {
        struct timeval stamp;           /* timestamp */
        u_char bits[13];                /* 100 irig data bits */
        u_char status;                  /* status byte */
        char time[14];                  /* time string */
};
```

The irig_time.stamp is a pair of 32-bit longwords in Unix timeval format, as defined in the sys/time.h header file. The first word is the number of seconds since 1 January 1970, while the second is the number of microseconds in the current second. The timestamp is captured at the most recent on-time instant of the IRIG timecode and applies to all other values returned in the irig_time structure.

The irig_time.bits[13] is a vector of 13 bytes to hold the 100-bit, zero-padded raw binary timecode, packed 8 symbols per byte. The symbol encoding maps the IRIG code element one to 1 and both the IRIG code element zero and IRIG position identifier to 0. The order of encoding is illustrated by the following diagram (the padding bits are represented by xxxx, which are set to zero):

```
                            1         2            8         9         10
IRIG code element  12345678901234567890123  ...  0123456789012345678 9xxxx
byte number        0       1       2             10      11      12
bit in byte        01234567012345670123456  ...  012345670123456701234567
```

The irig_time.status is a single byte with bits defined in the bsd_audioirig.h header file. In ordinary operation all bits of the status byte are zero and the “ ” status character is set in the ASCII timecode. If any of these bits are nonzero, the “?” status character is set in the ASCII timecode.

AUDIO_IRIG_BADSIGNAL

The signal amplitude is outside tolerance limits, either in amplitude or modulation depth. The indicated time may or may not be in error. If the signal is too high, it may be clipped by the codec, so that the pulse width cannot be reliably determined. If too low, it may be obscured by noise. The nominal expectation is that the peak amplitude of the signal be maintained by the codec AGC at about 10 dB below the clipping level and that the modulation index be at least 0.5 (6 dB). Since the prescribed modulation index of the standard IRIG-B signal is 10 dB, this leaves an adequate margin of 4 dB.

AUDIO_IRIG_BADDATA

An invalid hex code (A through F) has been found where BCD data is expected. The ASCII representation of the invalid code is set to “?”. Errors of this type are most likely due to noise on the IRIG signal due to ground loops, coupling to other noise sources, etc.

AUDIO_IRIG_BADSYNC

A code element has been found where a position identifier should be or a position identifier has been found where a code element should be. Errors of this type can be due to severe noise on the IRIG signal due to ground loops, coupling to other noise sources, etc., or during initial acquisition of the signal.

AUDIO_IRIG_BADCLOCK

Some IRIG timecode generators can indicate whether or not the generator is operating correctly or synchronized to its source of standard time using a designated field in the raw binary timecode. Where such information is available and the IRIG decoder can detect it, this bit is set when the generator reports anything except normal operating conditions.

AUDIO_IRIG_OLDDATA

The IRIG time has not changed since the last time it was returned in a read() call. This is not normally considered an error, unless it persists for longer than a few seconds, in which case it probably indicates a hardware problem.

The irig_time.time[14] vector is a character string in the format “ddd hh:mm:ss*\0”, which consists of 13 ASCII characters followed by a zero terminator. The “*” status character is an ASCII space “ ” if the status byte is zero and “?” if not. This format is identical to format 0, except that in format 1 the time string is null-terminated.

A.2. Programming Example

The following pseudo-code demonstrates how the IRIG receiver may be used by a simple user program. Of course, real code should include error checking after each call to ensure the receiver is communicating properly. It should also verify that the correct fields in the structure are being filled by the read() call.

```
#include "bsd_audioirig.h"

main()

{
        int format = 1;
        struct irig_time it;

        Audio_fd = open("/dev/audio", O_RDONLY);
        ioctl(Audio_fd, AUDIO_IRIG_OPEN, NULL);
        ioctl(Audio_fd, AUDIO_IRIG_SETFORMAT, &format);
        while (condition)
                read(Audio_fd, &it, sizeof(it);
                printf("%s\n", it.time);
        ioctl(Audio_fd, AUDIO_IRIG_CLOSE, NULL);
        close(Audio_fd);

}
```

A.3. Implementation and Configuration Notes

The signal level produced by most IRIG signal generators is on the order of a few volts peak-peak, which is far larger than the audio codec can accept; therefore, an attenuator in the form of a voltage divider is needed. The codec can handle IRIG signals at the microphone input from 4.2 mV to 230 mV peak-peak. A suitable attenuator consists of a series-connected 100K-Ohm resistor at the input and a parallel-connected 1K-Ohm resistor at the output, both contained along with suitable connectors in a small aluminum box. The exact values of these resistors are not critical, since the IRIG receiver includes an automatic level-adjustment capability.

For the most accurate time using the IRIG signal and a particular IRIG-equipped radio, it may be necessary to adjust the time1 parameter of the fudge command to compensate for the codec delay and any additional delay due to IRIG signal processing in the radio itself. Since the codec samples at an 8-kHz rate, the average delay is about 62 μs; however, the internal radio delay can vary. For instance, in the Austron receivers the radio delay is essentially zero, while in the Spectracom receivers the delay is about 240 μs relative to the pulse-per-second (PPS) signal. In addition, the poll interval can be reduced from the usual 64 s to 16 s to reduce wander of the local clock. Finally, the "prefer" parameter can be used to bias the clock-selection algorithm to favor the IRIG time, which is ordinarily the best time available. For example, the following two lines in the NTP configuration file ntp.conf are appropriate for the Spectracom Netclock/1 WWVB Synchronized Clock with IRIG Option:

```
server 127.127.6.0 prefer minpoll 4 maxpoll 4 # irig audio decoder
fudge 127.127.6.0 time1 0.0005
```

The time1 value of .0005 s (500 μs) was determined by actual measurement. Since the IRIG delay in Austron receivers is essentially zero, the fudge command is not necessary with these receivers. The correct value in case of other radios may have to be determined by actual measurement. A convenient way of doing this is to configure the PPSPPS feature in the NTP Version 3 distribution and adjust time1 until the PPS signal and IRIG signal both show the same offset.

The BSD_audio distribution includes the modified BSD driver bsd_audio.c and the IRIG header file bsd_audioirig.h, as well as modified header files bsd_audiovar.h and bsd_audioio.h. The driver is installed in the same way as described in the BSD driver documentation, with the addition of the following define in the kernel configuration file:

```
options         AUDIO_IRIG                  # IRIG driver
```

This causes the IRIG code to be included in the BSD audio driver, as well as a C-coded codec interrupt routine which replaces the assembly-coded routine and provides the IRIG functionality. While the C-coded routine is somewhat slower than the assembly-coded routine, the extra overhead is not expected to be significant. Note that the IRIG receiver calls the kernel routine microtime() as included in the ppsclock directory of the NTP Version 3 distribution. It is highly recommended that this routine be installed in the kernel configuration as well. The instructions for doing this are contained in the ppsclock directory of the distribution.