Data Acquisition Software Library

Daqlib Extensions

For daqdrv and

General Standards Daq Cards

16AISS8AO4

16AISS16AO2

Linux Version

Dr. M. C. Nelson

Sensor Realtime, LLC

daqlib

Rev. 070221

Notice and Disclaimer:

This document and the software described herein are provided with no claim or guarantee of safety, reliability or suitability for any purpose whatsoever.

Always test your software and systems exhaustively under safe, controlled circumstances before deployment.

Bug reports and feature requests are welcome.

Dr. M. C. Nelson
Sensor Realtime, LLC
12/26/2006

Contents

1 Introduction

These are the additional modules that make up the internal version of daqlib. Refer to the daqlib manual for information on the base library.

2 Files

./daqlib

libdaq.a	The full data acquisition software library.
daqfile.h daqiom.h daqiodblbuf.h daqparse.h daqthreshm.h daqwaveforms.h stringlib.h	C-language include files

3 Daqiom

Note that the control settings have to be written to the cards before calling any of the daq routines.

3.1 Data Structures

```
extern DaqBoard daqboardlist[];
extern int daqboardcount;

extern int daqboardreading;
extern int daqboardwriting;
```

3.2 Functions

3.2.1 Multi Card Access and Controls

3.2.1.1 int daqOpenM(DAQDRV_Ctl *ctl);

3.2.1.2 void daqCloseM( );

3.2.1.3 int daqReadControlsM(DAQDRV_Ctl *ctl);

3.2.1.4 int daqWriteControlsM(DAQDRV_Ctl *ctl);

3.2.1.5 int daqInitM(DAQDRV_Ctl *ctl);

3.2.1.6 void daqDumpM( );

3.2.1.7 void daqDumpFormattedM( );

3.2.2 Multi Card Read

3.2.2.1 int daqReadRawM(unsigned int *udata, int ndata, int (*readyfunc)(void *), void *readyarg);

3.2.2.1.1 int daqReadRawStartM( );

3.2.2.1.2 int daqReadRawStopM( );

3.2.2.1.3 int daqReadWaitM(unsigned int *buffer, int ndata);

3.2.3 Multi Card Write

3.2.3.1 int daqWriteRawM(unsigned int *udata, int ndata, int (*readyfunc)(void *), void *readyarg);

3.2.3.1.1 int daqWriteRawStartM(unsigned int *buffer, int ndata);

3.2.3.1.2 int daqWriteRawStopM( );

3.2.3.1.3 int daqWriteWaitM( );

3.2.4 Multi Card Write and Read

3.2.4.1 int daqWriteReadRawM(unsigned int *uwrite, int nwrite, unsigned int *uread, int nread, int (*readyfunc)(void *), void *readyarg);

3.2.5 Internals

3.2.5.1 void demuxdata(unsigned int *out, int outcols, unsigned int *in, int incols, int nrows);

3.2.5.2 void remuxdata(unsigned int *out, int outcols, unsigned int *in, int incols, int nrows);

3.2.5.3 int daqCountInputChannelsM(int nboards);

3.2.5.4 void daqFreeInputBuffersM(int nboards);

3.2.5.5 int daqMallocInputBuffersM(int nboards);

3.2.5.6 int daqCancelInputDmaM(int nboards);

3.2.5.7 int daqStartInputDmaM(int nboards);

3.2.5.8 int daqWaitInputM(int nboards);

3.2.5.9 int daqCountOutputChannelsM(int nboards);

3.2.5.10 void daqFreeOutputBuffersM(int nboards);

3.2.5.11 int daqMallocOutputBuffersM(int nboards, int nrows);

3.2.5.12 int daqCancelOutputDmaM(int nboards);

3.2.5.13 int daqStartOutputDmaM(int nboards);

3.2.5.14 int daqWaitOutputM(int nboards);

4 Daqthreshm

4.1 Data Structures

```
typedef struct daqtresholdlevelargm {
 int   channel;
```

```
  unsigned int level;
  int   direction;
} DaqThresholdLevelArgM;

typedef struct daqtresholdtoneargm {
  int   channel;
  float level;
  float f;
  FFT   fft;
  float vfs;
} DaqThresholdToneArgM;
```

4.2 Functions

4.2.1 Core Threshold routine

4.2.1.1 int daqReadThresholdM(unsigned int *udata, int ndata, int (*readyfunc)(void *), void *readyarg, int (*thresholdfuncM)(void *, int, int, int, void *), void *thresholdargM, int nlatent, int npretrig);

Performs multi card analog input , with dynamic triggering performed in the user supplied threshold function. Latency is the scanning length, pretrig is the pretrigger record length.

4.2.1.1.1 int daqReadThresholdStartM(int nlatent, int npretrig);

*4.2.1.1.2 int daqReadThresholdWaitM(unsigned int *udata, int ndata, int nchans, int (*thresholdfuncM)(void *, int, int, int, void *), void *thresholdargM);*

4.2.1.1.3 int daqReadThresholdStopM( );

4.2.1.1.3.1 int daqReadThresholdStopM_(int nboards);

4.2.2 Level threshold

4.2.2.1 int daqReadThresholdLevelM(unsigned int *udata, int ndata, int (*readyfunc)(void *), void *readyarg, int channel, float level, int direction, int nlatency, int npretrig);

Performs multi card analog input, with dynamic triggering by level and direction (above or below threshold).

*4.2.2.1.1 int daqThresholdLevelMf(unsigned int *data, int ndata, int nchan0, int nchannels, DaqThresholdLevelArgM *a);*

4.2.3 Edge level threshold

4.2.3.1 int daqReadThresholdEdgeM(unsigned int *udata, int ndata, int (*readyfunc)(void *), void *readyarg, int channel, float level, int direction, int nlatency, int npretrig);

Performs multi card analog input, with dynamic triggering by level transition and direction (going above or below threshold).

*4.2.3.1.1 int daqThresholdEdgeMf(unsigned int *data, int ndata, int nchan0, int nchannels, DaqThresholdLevelArgM *a);*

4.2.4 Tone level Threshold

4.2.4.1 int daqReadThresholdToneM(unsigned int *udata, int ndata, int (*readyfunc)(void *), void *readyarg, int channel, float level, float f, int nlatency, int npretrig);

Performs multi card analog input, with dynamic triggering by level of a specific tone frequency.

*4.2.4.1.1 int daqThresholdToneMf(unsigned int *data, int ndata, int nchan0, int nchannels, DaqThresholdToneArgM *a);*

4.2.5 Internals

4.2.5.1 inline int daqMuxBuffer_(unsigned int *out, int nout, int nchans, unsigned int *in, int nin, int inchans);

4.2.5.2 int daqMuxRing_(unsigned int *udata, int ndata, int nchans, DaqBuff *b0, DaqBuff *b1, int inchans);

4.2.5.3 int daqCopyInputRingBufferM(unsigned int *udata, int ndata, int nchans);

4.2.5.4 int daqCopyInputRingM(unsigned int *udata, int ndata, int nchans);

4.2.5.5 int daqWaitInputRingBufferM( );

4.2.5.6 int daqThresholdM(int (*thresholdfuncM)(void *, int, int, int, void *), void *thresholdarg);

4.3 Daq I/O Double Buffered

4.3.1 Functions

4.3.1.1 Double Buffered Input Functions

*4.3.1.1.1 int daqReadDblBuf(DaqBoard *b, int (*readyfunc)(void *), void *readyarg, int (*datafunc) (void *, int, void *), void *dataarg, int ndata_per_buffer);*

Peforms double bufferd input, for large data transfers or transfers directly to disk. The function readyfunc() is called to start the input. The function datafunc() is called for each buffer. The input continues while datafunc() returns zero.

(int) (*datafunc)((unsigned int *)buffer, (int) ndata, void *) data_arg)

*4.3.1.1.2 int daqReadtoFile(DaqBoard *b, FILE *fd, char *comments, int ndatatotal, int (*readyfunc)(void *), void *readyarg);*

*4.3.1.1.3 int daqReadtoDoubleTransposed(DaqBoard *b, double *d, int ndatatotal, int (*readyfunc)(void *), void *readyarg);*

4.3.1.2 Double Buffered Echo Input Functions

4.3.1.2.1 int daqWriteReadDblBuf(DaqBoard *b, unsigned int *uwrite, int nwrite, int (*readyfunc)(void *), void *readyarg, int (*datafunc) (void *, int, void *), void *dataarg, int ndata_per_buffer);

4.3.1.2.2 int daqWriteReadtoFile(DaqBoard *b, unsigned int *uwrite, int nwrite, FILE *fd, char *comments, int ndatatotal, int (*readyfunc)(void *), void *readyarg);

4.3.1.2.3 int daqWriteReadtoDoubleTransposed(DaqBoard *b, unsigned int *uwrite, int nwrite, double *d, int ndatatotal, int (*readyfunc)(void *), void *readyarg);

4.3.2 Internals

4.3.2.1 Structures

```
typedef struct daqdblbuf {
  unsigned int *a;
  unsigned int *b;
} DaqDblBuf;
```

4.3.2.2 Functions

4.3.2.2.1 void daqFreeDblBuf_(DaqDblBuf *p);

4.3.2.2.2 int daqMallocDblBuf_(DaqDblBuf *p, int ndataeach);

4.3.2.2.3 void daqReleaseInputDblBuf_(DaqBoard *b, DaqDblBuf *p);

4.3.2.2.4 int daqAddInputDblBuf_(DaqBoard *b, DaqDblBuf *p, int ndataeach);

4.3.2.2.5 int daqReadDblBufStart(DaqBoard *b, DaqDblBuf *pdb, int ndata_per_buffer);

4.3.2.2.6 int daqReadDblBufStop(DaqBoard *b, DaqDblBuf *pdb);

4.4 Daqfile

The daqfile component provides functions for saving data to disk files and reading data from disk files. The data sets are written with a header that includes all of the control settings and waveform settings if present.

4.4.1 Functions

4.4.1.1 Open Close Files

4.4.1.2 int daqCloseFile(FILE *fd);

4.4.1.3 FILE *daqOpenInputFile(char *filename);

Open input file read only. If the file does not exist, NULL is returned. If specified as “-“, stdin is returned.

4.4.1.4 FILE *daqOpenOutputFile(char *filename);

Open output file for write. If the file already exists, NULL is returned. If specified as “-“, stdout is returned.

4.4.2 Write data files

4.4.2.1 int daqWriteFrame(FILE *fd, void *data, DAQDRV_Ctl *ctl, char * comments);

Write an entire data frame with control settings to an ascii file.

*4.4.2.1.1 int daqWriteHeader(FILE *fd, DAQDRV_Ctl *ctl, char * comments);*

*4.4.2.1.2 int daqWriteDataFloat(FILE *fd, void *p, DAQDRV_Ctl *ctl);*

4.4.2.2 int daqWriteFrameRaw(FILE *fd, void *data, DAQDRV_Ctl *ctl, char * comments);

Write a data frame with control headers and data formatted as unsigned integers.

*4.4.2.2.1 int daqWriteDataRaw(FILE *fd, void *p, DAQDRV_Ctl *ctl);*

4.4.2.3 int daqWriteFrameBinary(FILE *fd, void *data, DAQDRV_Ctl *ctl, char * comments);

Write a data frame with formatted control headers and binary data.

*4.4.2.3.1 int daqWriteDataBinary(FILE *fd, void *p, DAQDRV_Ctl *ctl);*

4.4.2.3.1.1 int daqWriteDataBinary_(FILE *fd, unsigned int *udata, int ndata);

4.4.3 Read data files

4.4.3.1 int daqReadFrame(FILE *fd, float *data, int maxdata, DAQDRV_Ctl *ctl, char * comments, int maxc);

Read a completed data frame, automatically detects float, raw, and binary.

*4.4.3.1.1 int daqReadHeader(FILE *fd, DAQDRV_Ctl *ctl, char * comments, int maxc);*

*4.4.3.1.2 int daqReadData(FILE *fd, float *data, int maxdata, float vfs);*

4.4.3.1.2.1 Int daqReadDataBinary_(FILE *fd, float *data, int ndata, float vfs);

4.4.3.2 float *daqReadFrameMalloc(FILE *fd, DAQDRV_Ctl *ctl, char * comments, int maxc);

Allocates data buffer and reads a completed data frame, automatically detects float, raw, and binary.

4.4.3.3 float *daqReadFrameRealloc(FILE *fd, DAQDRV_Ctl *ctl, char * comments, int maxc, void *old);

Reallocates data buffer to new size as needed and reads a completed data frame, automatically detects float, raw, and binary. If the reallocate fails, NULL is returned and the old space is freed.

4.4.4 Save/Load configuration files

4.4.4.1 int daqLoadCfg(DAQDRV_Ctl *ctl, char *comments, int maxc);

4.4.4.2 int daqSaveCfg(char *filename, DAQDRV_Ctl *ctl, char *commments);

5 Daqparse

5.1 Functions

5.1.1 int daqParse(char *s, DAQDRV_Ctl *ctl, WAVESPEC *wspec, int maxwspec);

Parses everything including control settings and wave specs.

5.1.1.1 int daqUnparse(char *s, DAQDRV_Ctl *ctl, WAVESPEC *wspec, int maxwspec);

5.1.2 int daqCtlParse(DAQDRV_Ctl *ctl, char *s);

Parse a string looking for control settings, by keywords “input” and “output” and “trigger”, followed by a list of par=val statements.

5.1.2.1 int daqCtlParse_(DaqBoardConfig *cfg, DaqAioCtl *ctl, char *s);

Parse control parameters for an input or output, as a list of par=val statements.

5.1.3 int daqCtlHelpPrint( );

Lists descriptions for control keywords and values.

5.1.3.1 int daqCtlHelpSprint(char *s);

5.1.4 int daqCtlPrint(FILE *fd, DAQDRV_Ctl *ctl);

Writes formatted current control settings.

5.1.4.1 int daqCtlSprint(char *s, DAQDRV_Ctl *ctl);

*5.1.4.1.1 int daqAioCtlSprint(char *s, DaqAioCtl *a);*

5.1.5 int daqStatusPrint(FILE *fout, DAQDRV_Status *p);

6 Daqwaveforms

6.1 Data Structures

```
typedef float (*ValueFunc)(float, float, float, float, int);

typedef int  (*OperatorFunc)(float *, int, int, float, float, float,
                    ValueFunc);

typedef struct wavepars {
  ValueFunc func;
  float a;
  float f;
  float p0;
  float offset;
  float duty;
  float cycles;
  char  name[16];
} WAVEPARS;

typedef struct wavespec {
  struct wavepars carrier;
  struct wavepars modulation;
} WAVESPEC;

typedef struct wavedatastruct {
  float *data;
  int ndata;
  int ncols;
  int nrows;
  float clockrate;
  float vfs;
} WAVEDATA;
```

6.2 Functions

6.2.1 Parse functions

6.2.1.1 int parsewavespec(WAVESPEC *spec, int maxchans, char *s);

Parse for waveform specifiers following the key word “wave”, optionally followed by “clear” to reset the specified list, and channels in the form

wave clear
wave channel{n}=type,f={Hz}a={Volts},cycles={n},p={phase(0-2pi)},offset={Volts},duty={fraction}

type is decay, pulse, sine, square, null, or for modulation prefix with an asterisk, c.f. *decay .

*6.2.1.1.1 int parsewavepars(WAVEPARS *spec, char *s);*

*6.2.1.1.2 int channelspec(char *s, int *chan);*

6.2.1.2 void clearwavespecs(WAVESPEC *spec, int maxchans);

6.2.1.3 int printwavespecs(FILE *fd, WAVESPEC *spec, int maxchans);

*6.2.1.3.1 int sprintwavespecs(char *s, WAVESPEC *spec, int maxchans);*

6.2.1.4 void printwavehelp( );

6.2.2 Wave Generator

6.2.2.1 int wavedataGenerate(WAVEDATA *w, WAVESPEC *spec, int maxchans, float clockrate);

*6.2.2.1.1 int wavedataPopulate_(WAVEDATA *w, WAVESPEC *spec);*

6.2.2.1.1.1 int wavedataCountChannels_(WAVESPEC *spec, int maxchans);

6.2.2.1.1.2 int wavedataMaxRows_(WAVESPEC *spec, int maxchans, float clockrate);

6.2.2.1.1.3 int wavedataFillChannels_(WAVESPEC *spec, int maxchans);

*6.2.2.1.2 int wavedataMalloc_(WAVEDATA *w, WAVESPEC *spec, int maxchans, float clockrate);*

*6.2.2.1.3 void wavedataFree(WAVEDATA *w);*

6.2.3 Waveform Generator Functions

6.2.3.1 float decayvalue(float a, float f, float p0, float fdummy, int n);

6.2.3.2 float pulsevalue(float a, float f, float p0, float fduty, int n);

6.2.3.3 float sinevalue(float a, float f, float p0, float fdummy, int n);

6.2.3.4 float squarevalue(float a, float f, float p0, float fduty, int n);

6.2.4 Write/Read wave files

6.2.4.1 int daqWriteWaveFrame(FILE *fd, WAVEDATA *w);

Write a complete wave form formatted as floating point.

6.2.4.2 float *daqReadWaveFrame(FILE *fd, WAVEDATA *w, DaqAioCtl *octl);

Read a complete wave form, allocates storage as w->data.

*6.2.4.2.1 int daqReadWaveHeader(FILE *fd, WAVEDATA *w);*

7 Stringlib

7.1 Functions

7.1.1 Basic string functions

7.1.1.1 char *strNext(char *s);

7.1.1.2 int strCopy(char *out, char *in);

7.1.1.3 int strTokenCopy(char *out, char *s);

7.1.1.4 int strMatch(char *s, char *key);

7.1.1.5 int strMatchPrefix(char *s, char *key);

7.1.1.6 int strMatchList(char *s, char *list[]);

7.1.1.7 int strTrim(char *s);

7.1.1.8 char *strAfter(char *s, char *key);

7.1.1.9 char *strArg(char *s, char *key);

7.1.2 Numerical value

7.1.2.1 int strFlt(char *s, float *f);

7.1.2.2 int strInt(char *s, int *i);

7.1.2.3 int strUInt(char *s, unsigned int *u);

7.1.2.4 int strOUInt(char *s, unsigned int *u);

7.1.3 Numerical value with range checking

7.1.3.1 int strFltRange(char *s, float *f, float fmax, float fmin);

7.1.3.2 int strIntRange(char *s, int *i, int imax, int imin);

7.1.3.3 int strUIntRange(char *s, unsigned int *u, unsigned int umax, unsigned int umin);

7.1.4 Numerical value as prefix

7.1.4.1 int strFltPrefix(char *s, float *f);

7.1.4.2 int strIntPrefix(char *s, int *i);

7.1.4.3 int strUIntPrefix(char *s, unsigned int *i);

7.1.5 Numerical value as argument

7.1.5.1 int strFltAfter(char *s, char *key, float *val);

7.1.5.2 int strIntAfter(char *s, char *key, int *val);

7.1.5.3 int strFltArg(char *s, char *key, float *val);

7.1.5.4 int strIntArg(char *s, char *key, int *val);

7.1.6 Numerical values as vector

7.1.6.1 int strFltVector(char *s, float *data, int ndata);

7.1.6.2 int strIntVector(char *s, int *data, int ndata);

7.1.6.3 int strUIntVector(char *s, unsigned int *data, int ndata);

7.1.7 Values from file

7.1.7.1 int readFlt(FILE *fin, float *data, int ndata);

7.1.7.2 int readUInt(FILE *fin, unsigned int *data, int ndata);

7.1.7.3 int readStr(FILE *fin, char *s, int nmax);