Users Manual
for
# tsadclib1624

16 & 24 Channel
Analog to Digital Acquisition Peripherals

Version	Date	Description	Engineer
0.1	2010-01-05	Initial Release	J Lydic

Abstract

The analog to digital acquisition library (tsadclib1624) is a general purpose library used to control and access samples from both the ADC16 and ADC24 analog to digital converters. Up to four ADC boards (when properly configured) may be connected to the TS-7200, TS-7350, TS-7370, and TS-7800 Single Board Computers. Additionally, the Digital IO and Digital to Analog conversion hardware may be accessed on the ADC16. Aggregate sample rates of up to 250,000 samples per second are possible, with certain restrictions described later in this document.

Caveats and restrictions

As with any real time system, there are a finite number of CPU cycles available for managing, moving, and processing data. The following guidelines will assist you in planning for the best use of your hardware. Additionally, the library contains certain restrictions which allow better performance with less memory usage. Since full source code is provided, the user may make modifications as required, but the initial library is written to balance the memory and cpu resource requirements of the library, with best acquisition performance.

1. NOTE: If you are re-compiling the library, be sure to use the following compiler directive: **-mcpu=arm9**. Failing to do so may generate code for the ARM processor that will not work correctly when performing IO access to the ADC hardware.
2. Channel pairs & channel quads: On the ADC16 all data acquisition is performed on pairs of channels (i.e., 2, 4, 6, …, 16). On the ADC24 all data acquisition is performed on quads of channels (i.e., 4, 8, 12, …, 24). These channel selections must be used when calculating per channel throughput.
3. Even though a system may contain up to 4 ADC boards, the library may be used to access the channels on only one board at a time. Attempting to read channel data from more than one board in polling mode, will result in undefined behavior.
4. One thing the library will attempt to do is to modify the priority of the process, raising it while waiting for data, and lowering it back to nominal when the wait is complete. The library can do this if the process running as root. If you are seeing intermittent acquisition errors, this may help in the real time responsiveness of the library.
5. Call the acquisition function to start the acquisition. Then call a finish function to await completion. The call may be:

   - A blocking call which will poll until data acquisition completion.
   - OR, A non-blocking call. This call must be called repeatedly, with the user able to perform additional work between calls. It returns a negative value for error, a zero for continued polling, and a positive number when the transfer completes

6.

The general processing steps are as follows:

User Initiated Function	**Library action**	**Caveats / Description**
Initialize Library	Configure SBC hardware; identify & enumerate attached ADC hardware; Allocate buffers based on requested channels & size	This allocates Highest channel number x 3 x requested sample size in bytes; e.g., 24 channels @ 100K samples require 720000 bytes of memory in the library PLUS user supplied buffers per channels actually sampled.
Initiate data acquisition request	Using the Acquire or Sample functions a data request it initialized.	This does no acquisition but sets up the next step to acquire the proper channels for the requested size and sample rate
Poll for data – Blocking mode	Data is acquired from the hardware and placed into the internal buffer defined and created during initialization. Polling for the data including process pause is performed within the function. This function does not return to the user until all requested data has been retrieved.	
Poll for data – Non-blocking mode	Data is acquired from the hardware and placed into the internal buffer defined and created during initialization. Poling for the data is the responsibility of the calling program. This function returns to the user when no data is available or when all requested data has been retrieved.	The user may perform other housekeeping or computing functions between the poling calls, but under polling can and will generate **FIFO overrun errors**. This is meant to be used during low speed sample rates.
At the termination of data acquisition in either polling mode (not a user initiated action) – Phase 1	All data (12 & 16 bit) is moved from the internal master collection buffer and sorted into user supplied sample buffers, based on channel data collected and requested.	
At the termination of data acquisition in either polling mode (not a user initiated action) – Phase 2	12-bit (ADC-24) only, modified in place in the user supplied sample buffers, based on channel data collected and requested. All data is sign extended to 16 bits and in certain instances twos-complemented to correct for hardware ADC acquisition modifications)	

NOTE: Because of the way the data is presented from the hardware, it is not currently possible to provide continuous data into a ring buffer at high speeds.

Library Functions

Name
tsadclib1624_init – Initialize the library

Synopsis
int tsadclib1_init(sbctype sbc, uint16_t maxChannel, uint32_t maxBufferSize);

Description
tsadclib1624_init locates and initializes all of the installed ADC peripheral boards and allocates required memory.

sbc indicates the single board computer to which the hardware is connected as follows: SBC_TS7200, SBC_TS7350, or SBC_TS7800. (SBC_TS7350 is used to specify the SBC_TS7370)

maxChannel specifies the maximum channel that will be accessed from within the library, range 1-24.

maxBufferSize specifies the maximum buffer size that will be requested in subsequent library calls, range 1-n*.

*Note that the maximum 'n' is limited to the amount of available memory on the SBC. The required memory size (in bytes) allocated may be calculated as 3n*maxChannel.

Return Value
tsadclib1624_init returns zero for correct initialization

Errors
Errors are returned as a negative value for a failure. The error values are defined in the header files under the TSADCERR_ section, and may be converted to strings by using the tsadclib1624_errors function.

Name
tsadclib1624_boardstatus – Return the status of each board installed in the system

Synopsis
adctype tsadclib1624_boardstatus(BoardIndex theBoard)

Description
After a successful completion from **tsadclib1624_init** the boards located in the system may be enumerated.

theBoard (range 0-3) specifies the board slot from which status is being requested. The definitions for the board selection in the header file in the **BoardIndex** enumeration.

Return Value
TS_ADCINVALID if no board installed in the slot
TS_ADC16 if a TS-ADC16 board is installed in the slot
TS_ADC24 if a TS-ADC24 board is installed in the slot

Errors
TSADCERR_NOINIT if the library has not been initialized
TSADCERR_BADBOARD if the board id is incorrect (i.e., > 3)

Name
tsadclib16_digitalOut - set or clear the digital output bit: */

Synopsis
int tsadclib16_digitalOut(BoardIndex theBoard, bool state)

Description
theBoard (range 0-3) specifies the board slot to which the DIO bit is being modified. The definitions for the board selection in the header file in the **BoardIndex** enumeration.

State indicate whether to set or clear the bit

Return Value
Zero if successful

Errors
TSADCERR_NOINIT if the library has not been initialized
TSADCERR_BADBOARD if the board id is incorrect (i.e., > 3)
TSADCERR_BADPARAMETER If the board is NOT a TS-ADC24

Name
tsadclib16_digitalIn – Return the 4 digital input bits

Synopsis
uint8_t tsadclib16_digitalIn(BoardIndex theBoard)

Description
theBoard (range 0-3) specifies the board slot from which input is being requested. The definitions for the board selection in the header file in the **BoardIndex** enumeration.

Return Value
The state of the digital input bits

Errors
TSADCERR_NOINIT if the library has not been initialized
TSADCERR_BADBOARD if the board id is incorrect (i.e., > 3)
TSADCERR_BADPARAMETER If the board is NOT a TS-ADC16

Name
tsadclib16_counterIn – Read a value from one of the free running counters

Synopsis
uint16_t tsadclib16_counterIn(BoardIndex theBoard, CounterIndex theCounter)

Description
theBoard (range 0-3) specifies the board slot from which a counter is being requested. The definitions for the board selection in the header file in the **BoardIndex** enumeration.

theCounter (range 0-3) specifies the counter from which the value is being requested. The definitions for the board selection in the header file in the **CounterIndex** enumeration.

Return Value
Returns the 16 bit unsigned value of the counter. The counters are incremented on the falling edge of the counter input bit at a rate up to 4 MHz. They may not be reset, but reset to zero oin overflow.

Errors
TSADCERR_NOINIT if the library has not been initialized
TSADCERR_BADBOARD if the board id is incorrect (i.e., > 3)
TSADCERR_BADPARAMETER If the board is NOT a TS-ADC16

Name
tsadclib16_dacset – Set an output value to one of the 16-bit DACs

Synopsis
int16_t tsadclib16_dacset(BoardIndex theBoard, uint8_t theDac, dacrefrange referenceRange, uint16_t value)

Description
theBoard (range 0-3) specifies the board slot to which the DAC is to be written. The definitions for the board selection is in the header file in the **BoardIndex** enumeration.

theDac (range 0-3) specifies the DAC to which the data will be written. The definitions for the DAC selection in the header file is in the **DacIndex** enumeration

referenceRange specifies the range to which the DAC will be set. The definitions for the DAC range selection in the header file is in the **dacrefrange** enumeration

value specifies the 16-bit value to which the DAC will be set.

Return Value
Zero if successful

Errors
TSADCERR_NOINIT if the library has not been initialized
TSADCERR_BADBOARD if the board id is incorrect (i.e., > 3)
TSADCERR_BADPARAMETER If the board is NOT a TS-ADC16, or an invalid range DAC or range selection is specified.

Name
tsadclib1624_startAcquisution – Acquire data from 1 or 2 adjacent channels.

Synopsis
int16_t tsadclib1624_startAcquisution(BoardIndex theBoard, int8_t channel, uint32_t sampleRate, int16_t sampleCnt, adcrange range, uint16_t *data1, uint16_t *data2, bool contineOperat)

Description
theBoard (range 0-3) specifies the board slot to which the DAC is to be written. The definitions for the board selection in the header file in the **BoardIndex** enumeration.

Channel is the channel (0-15, or 0-23) to acquire. Since channel pairs are involved, the channel and it's adjacent channel pair will be acquired.

sampleRate is the sample rate in Hz, up to 100,000

sampleCnt is the number of samples to acquire. This value cannot exceed the value specificed in the initialization call.

range is the range setting from the adcrange enumeration in the header file.

data1 pointer to the buffer to contain the data samples from the lower numbered channel.

data2 pointer to the buffer to contain the data samples from the higher numbered channel.

contineOperation indicates if this transaction is a continuation of the previous transaction. Normally this is not the case, and the parameter is set to false, which clears the FIFO before initiating the next transaction.

Return Value
Zero if successful

Errors
TSADCERR_NOINIT if the library has not been initialized
TSADCERR_BADBOARD if the board id is incorrect (i.e., > 3)
TSADCERR_BADPARAMETER If an invalid channel for the selected board has been specified.

Name
int32_t tsadclib1624_seperateData - Separate data from the main buffer into the supplied buffer(s)

Synopsis
int32_t tsadclib1624_seperateData(boarddata *theAdc)

Description
Internal function used by tsadclib1624_pollResults to demultiplex the FIFO data into the appropriate channel buffers.

Return Value
Current data count

Errors
None

Name
static int32_t tsadclib1624_moveData - Move data from the FIFO into the supplied buffer(s)

Synopsis
static int32_t tsadclib1624_moveData(boarddata *theAdc)

Description
This is an internal helper function

Return Value
Requested/Acquired sample count

Errors
TSADCERR_FIFOOVERRUN if the data has not been removed from the FIFO quickly enough.
TSADCSTATUS_NODATA if the FIFO is prematurely empty

Name
tsadclib1624_pollResults – Acquire the data requested by the tsadclib1624_startAcquisution or tsadclib1624_sample functions.

Synopsis
int32_t tsadclib1624_pollResults(BoardIndex theBoard, bool blocking)

Description
After the acquisition setup calling this function will start the actual sampling on the voard selected by **theBoard** and will then perform one of two actions dependent on the **blocking** parameter.

If blocking is true, the function will not return until either all requested data has been acquired, or a FIFO overrun is detected.

If blocking is false, the function must be called repeatedly by the user. On each call, any available data will be moved, and the function will return zero. The final call, during which the data request is completely satisfied, will return a count of the acquired data. The rate at which the call is repeated is dependent on the number of channels being sampled, and the sample rate.

Return Value
A positive count of the data that has been acquired
A zero if using non-blocking polling and the transaction has not yet completed.

Errors
TSADCERR_NOINIT if the library has not been initialized
TSADCERR_BADBOARD if the board id is incorrect (i.e., > 3)

Name
tsadclib1624_sample – Setup multiple channel sample acquisition

Synopsis
int16_t tsadclib1624_sample(BoardIndex theBoard, int16_t maxChannel, int32_t sampleRate, int16_t sampleCnt, adcrange range, int16_t *channelBuffers[], bool contineOperation)

Description
Acquire **numberOfSamples** samples from all channels of the ADC on **theBoard** from zero through **maxChannel** in adcrange **range** at **sampleRate** in hertz into the buffer array **channelBuffer[0...maxChannel]**

The buffer array **channelBuffer[…]** consists of an array (1-24 entries) of pointers to properly sized buffers of type uint16_t. The channelBuffer is indexed by the channel from which the data is acquired. If the channelBuffer[channel] is non-NULL, then the data is placed into the next location for the entry. If the channelBuffer[channel] is NULL, the data sample is discarded. In this way, by properly initializing the entries of the channelBuffer array, data may be selectively acquired from a matrix of the channels of interest.

Return Value
Zero if successful

Errors
TSADCERR_NOINIT if the library has not been initialized
TSADCERR_BADBOARD if the board id is incorrect (i.e., > 3)
TSADCERR_BADPARAMETER If an invalid channel for the selected board has been specified.

Name

uint8_t *tsadclib1624_errors(int errorCode)

Synopsis
Description
Return Value
Errors

<u>Name</u>
tsadclib1624_verbose – This function sets or clears the verbose output flag.

Synopsis
void tsadclib1624_verbose(bool Verbose)

Description
If Verbose is set to true, additional diagnostic messages will be output to standard error.
Return Value
None

Errors
None

<u>Name</u>
tsadclib1624_buffers - Return true if buffers assigned or false if not

Synopsis
bool tsadclib1624_buffers(void)

Description
Return true if buffers assigned or false if not

Return Value
Return true if buffers assigned or false if not

Errors
None

Appendix A

ADC Estimated Channel capacity

ADC Board	Channel Range	Channels Sampled	Estimated max rate per channel (73x0)	Estimated max rate per channel (7800)
16	0-1	2	110000	108000
16	0-3	4	55000	53000
16	0-5	6	32000	32000
16	0-7	8	25000	25000
16	0-9	10	20000	20000
16	0-11	12	16000	3000
16	0-13	14	14500	14500
16	0-15	16	12500	12500
24	0-3	4	54000	50000
24	0-7	8	26000	25000
24	0-11	12	16000	16000
24	0-15	16	12000	12000
24	0-19	20	10000	10000
24	0-23	24	8000	8000

Appendix B

tsadc1624ctl – Test and measurement program

The tsadc1624ctl program is a general purpose utility used to setup the ADC hardware, and acquire samples. The samples can be displayed to the standard output device in several useful formats, and the output may be redirected to save the acquired data for later use. Additionally, the source code for the utility is available for use as an example of how the different library functions are used, and is therefore usable as a starting point for customer applications.

Display the utility functionality

```
# tsadc1624ctl -?
Usage: tsadc1624ctl -m model [OPTION] ...
Interact with the ADC-16 and ADC-24 in the 7200, 7350, 7370, & 7800 SBCs
General options:
  -a  channels*        Acquire ADC channels (1 or 2), e.g. "1-2", "3,4"
                       Note: you must always specify an odd channel and
                       the subsequent even channel, e.g., '1-2' or '7,8'
  -b  board*           ADC board number (0-3) on whici to sample
                       Required prior to 'r', 's', 'd' and 'D' options
  -B                   Maximum buffer size
  -D                   Set DAC output - Range
  -d                   Set DAC otuput - Channel:Value
  -e                   Enumerate the list of installed ADC boards and quit
  -h                   This help screen
  -m  model (Required) Model of the processor (m modeal [from list above])
                       NOTE: This parameter must be first on the line
  -M                   Maximum channels (Range 1-16 OR 1-24)
  -n  number*          Number of samples to acquire (Default: 10000)
  -o  format           printf output to stdout in format: decimal, hex, csv
  -p                   Utilize non-blocking Polling
  -r  rate*            Sample rate in Hz (Default: 10000)
  -R                   display details on range selectionz
  -s  channels         Sample ADC channels CHANS, e.g. "1-3,5", "2,3,5"
                       output string parseable data to standard out
  -v                   Verbose output
  -x  range*           Range and measurement mode (Refer to enum 'adcrange'
                       in the library header file
                       ADC16: 55S, 05S (default), 1010S, 0010S, 55D, 05D,
                       1010D, 0010D
                       ADC24: 0VS, 02VS (default), 02VAS, 02VBS, 0VD, 02VD,
                       02VAD, 02VBD
  -?                   This help screen
   *                   Option required for 'a' and 's'
```

Display the utility sampling hardware ranges

```
# tsadc1624ctl -m 7800 -R
SBC Model assigned as: 7800
ADC and DAC range definitions
ADC16
55S     -5 to +5V  single ended
05S      0 to +5V  single ended (default)
1010S  -10 to +10V single ended
0010S    0 to +10V single ended
55D     -5 to +5V  differential
05D      0 to +5V  differential
1010D  -10 to +10V differential
0010D    0 to +10V differential

ADC24
0VS     0 to +VRef   single ended
02VS    0 to +2xVRef single ended (default)
02VAS   0 to +2xVRef single ended
02VBS   0 to +2xVRef single ended
0VD     0 to +VRef   differential
02VD    0 to +2xVRef differential
02VAD   0 to +2xVRef differential
02VBD   0 to +2xVRef differential

DAC
0VU     0 to VRef  - Unbuffered
02VU    0 to 2VRef - Unbuffered
0VB     0 to VRef  - Buffered
02VB    0 to 2VRef – Buffered
```

Initialize the hardware and enumerate the attached boards

```
# tsadc1624ctl -m 7800 -e
SBC Model assigned as: 7800

Board 0: TSADC24
Board 1: TSADC16
Board 2: Not installed
Board 3: Not installed
```

Use the utility to sample channels 0-3 for 10000 samples each at a sample rate of 51.5 KHz

```
# tsadc1624ctl -m 7800 -b 1 -s 0-3 -n 100000 -r 51500 -x 55S -B 100000
main: Await data completion
Complete transfer: ADC Machine PROGRAM STOP: 0x0120
seperateData: ADC_SAMPLE
```

Use the utility to sample channels 0, 1, 4 & 5 for 100 samples each at a sample rate of 100 KHz and display the output of each channel in csv format, suitable for inclusion in a spreadsheet.

```
# tsadc1624ctl -m 7800 -b 1 -s 0,1,4-5 -n 100 -r 100000 -x 55S -B 10000 –o csv
SBC Model assigned as: 7800
main: Await data completion
Complete transfer: ADC Machine PROGRAM STOP: 0x0120
```

```
0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
1,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
4,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
5,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,
root@ts7800:adc#
```