

ATS-GPU SDK

Version 0.0.2

December 14, 2011

[Introduction](#)

[System requirements](#)

[File overview](#)

[Connecting input signals](#)

[Using the LabVIEW sample applications](#)

[Using the C++ sample applications](#)

[Using the MATLAB sample applications](#)

[Building the sample DLL](#)

[Programming Guide](#)

[FFT Function Reference](#)

[For more information](#)

[Release history](#)

Introduction

The ATS-GPU SDK provides a framework to allow real-time processing of data from AlazarTech PCIE digitizers, or from a file. It includes a DLL that calculates the FFT of sample data using an OpenCL compatible GPU, and the FFTW open-source library that calculates the FFT of sample data in software.

The sample code includes:

- C++ console applications that demonstrate how to call ATS_GPU.dll to make an ADMA acquisition, and perform FFT calculations in hardware in a C++ programming environment.
- LabVIEW applications and a subVI library that demonstrate how to call ATS_GPU.dll to make an ADMA acquisition, and perform FFT calculations in a LabVIEW programming environment.
- MATLAB applications and a library that demonstrate how to call ATS_GPU.dll to make an ADMA acquisition, and perform FFT calculations in a MATLAB programming environment.
- C/C++ DLL source code that demonstrates how to configure an AlazarTech digitizer to make an AMDA acquisition, and use OpenCL process the sample data in each DMA buffer.

System requirements

This software requires a PC with an OpenCL compatible GPU, and sufficient CPU resources to supply data to the GPU at the desired data acquisition rate. It was tested with an ASUS GTX560 and NVIDIA Tesla C2070.

The LabVIEW code was created with LabVIEW 8.6.1, and requires 8.6.1 or later to be run. It was tested with LabVIEW 8.6.1. ATS-VI 6.0.7 or later must be installed to run the LabVIEW sample code.

The C++ code was created with Microsoft Visual C++ 2008, and requires Microsoft Visual C++ 2008 or later. If you do not have a suitable C++ compiler, use Microsoft Visual C++ 2010 Express.

<http://www.microsoft.com/express/Downloads/#2010-Visual-CPP>

The MATLAB sample code was created with MATLAB 2006a, and requires 2006a or later. It was tested under MATLAB 2006a 32-bit and MATLAB 2009a 64-bit.

File overview

The ATS-GPU library code includes the following directories:

Directory	Description
Samples_LabVIEW	LabVIEW applications and subVIs that demonstrate how to call functions exported by ATS-GPU DLL in a LabVIEW programming environment.
Samples_MATLAB	MATLAB applications that demonstrate how to call functions exported by ATS-GPU DLL in a MATLAB programming environment.
Samples_C/<boardType>	C source files required to build sample applications that demonstrates how to call functions exported by the ATS_GPU.dll in a C++ application environment.
Samples_C/OpenCL	OpenCL headers and library files required to compile ATS_GPU.DLL. For more information, see http://www.khronos.org/opencl/
Samples_C/ATS_GPU_DLL	C++ source files required to build a sample DLL that uses OpenCL to invert the samples in each buffer.
Samples_C/fftw	The FFTW open-source library source code. For information, please see http://www.fftw.org .

Connecting input signals

Configure a function generator to output a 100 KHz TTL signal, and connect it to the TRIG IN connector of the digitizer. Configure another second function generator to supply +/-200 mV sine of the desired frequency, and connect it to CH A of the digitizer. Run a sample application, and adjust the input signals as required.

Using the LabVIEW sample applications

LabVIEW sample applications demonstrate how to make ADMA acquisitions, and perform FFT calculations in an OpenCL compatible GPU. The `_NPT` sample makes an NPT mode ADMA acquisition, while the `_CS` sample makes a continuous streaming mode acquisition.

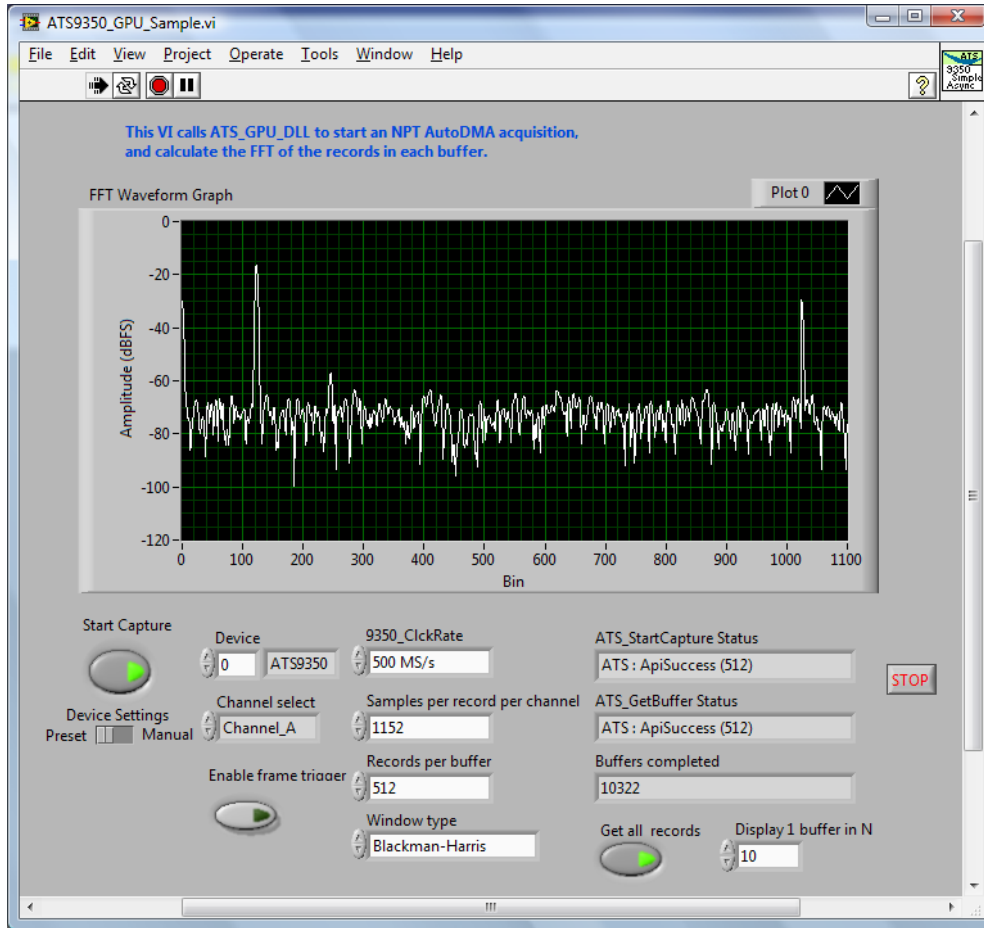


Figure 1 ATSS9350_FFT_NPT.vi Front Panel

To run sample vi, press the “Start Capture” button. The above screenshot shows the vi running with a 100 mV, 30 MHz sine connected to CH A, and a 100 KHz trigger signal connected to TRIG IN.

Start Capture

Press this button to start or stop the acquisition.

Device Settings

When this toggle switch is set to the “Manual” position, the vi displays a front panel where you can configure digitizer settings, such as the sample rate, clock source, input range, trigger source, and trigger level when the “Start Capture” button is pressed.

When this toggle switch is set to the “Preset” position, the vi uses the configuration settings coded in the block diagram to configure the digitizer.

Device

ATS-VI identifies each digitizer it detects by an index, where 0 is the index of the first board. If you have one board in your computer, set the index to 0. If you have more than one board in your computer, then use the device index to select the board to control.

Channel select

Select which channels that the board will capture. You must press the “Start Capture” button to apply this setting.

Enable frame trigger

Press this button to enable a frame trigger. If enabled, you must connect a TTL signal to the AUX connector on the digitizer. Each time that the board receives an edge on the AUX input, it will be armed to wait for “Records per buffer” triggers.

Samples per record per channel

Select the number of samples to capture per trigger per channel.

Note that the GPU FFT code requires that the number of samples per record be a power of two. As a result, if sample per record is not a power of two, ATS-GPU pads samples per record to the next larger power of two, zero filling unused samples. For example, if you request 1152 samples per record, ATS-GPU will calculate a 2048 sample FFT.

Records per buffer

Select the number of records per DMA buffer.

Window type

Select the FFT window type.

Get all records

If this button is pressed, the vi copies the FFT results for all records in a DMA buffer to a LabVIEW array. If this button is not pressed, the vi only copies the FFT results for the first record in the DMA buffer.

Display 1 in N records

The digitizer can acquire and process records faster than they can be displayed by LabVIEW. Use this edit control to reduce the screen update rate in LabVIEW.

Examine the block diagram for more information.

Using the C++ sample applications

C++ sample applications demonstrate how to configure a digitizer to make an ADMA acquisition, or read sample data from a file, and process the data in an OpenCL compatible GPU or using the FFTW open-source software library.

The FFT_ samples process sample data using an OpenCL compatible GPU device, while the FFTW_ samples process sample data using the FFTW software library. The _NPT samples make NPT mode ADMA acquisitions, while the _CS samples make continuous streaming ADMA acquisitions. The _File samples read sample data from a file rather than from a digitizer.

You can use the FFT_ and FFTW_ samples to compare the performance of hardware vs. software solutions. If the FFTW software provides sufficient throughput for your application, then it may not be necessary to use an OpenCL compatible GPU for real-time data processing.

The sample code contains parameters that should be changed as required. See the lines in “AcqToDisk.cpp” with “TODO” comments.

To build AcqToDisk.exe:

Microsoft Visual C++ 2008 Professional

- Open the “AcqToDisk_x64.sln” solution in the ATS_Average_APP directory.
- Select the “Build | Configuration Manager” to display the “Configuration Manager” dialog and select the desired configuration and platform.
- Select the “Build | Build Solution” command to build the sample program.

If a build succeeds, it creates one of the following executables:

32-bit Debug	Win32\Debug\AcqToDisk.exe
32-bit Release	Win32\Release\AcqToDisk.exe
64-bit Debug	x64\Debug\AcqToDisk.exe
64-bit Release	x64\Release\AcqToDisk.exe

Microsoft Visual C++ 2008 Express

- Open the “AcqToDisk.sln” solution in the ATS_Average_APP directory.
- Select the “Build | Configuration Manager” to display the “Configuration Manager” dialog and select the desired configuration and platform.
- Select the “Build | Build Solution” command to build the sample program.

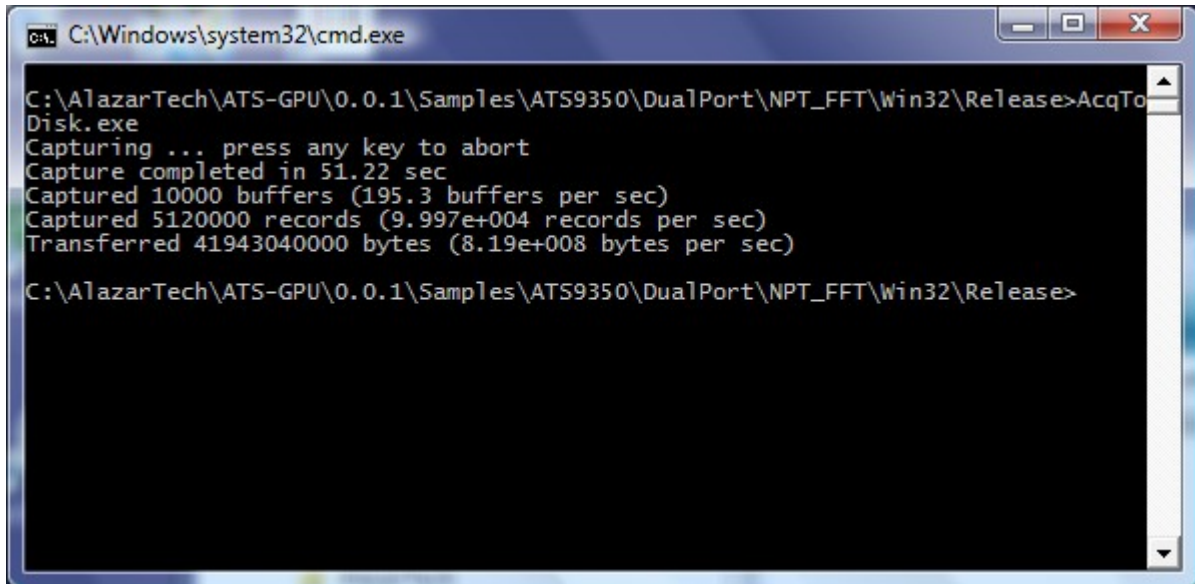
If a build succeeds, it creates one of the following executables:

32-bit Debug	Win32\Debug\AcqToDisk.exe
32-bit Release	Win32\Release\AcqToDisk.exe

To run the AcqToDisk.exe:

1. Open a command prompt and change the current directory to an “ATS_Average_APP” output directory (“Win32\Release”, “Win32\Debug”, “x64\Release”, or “x64\Debug”).
2. Type “AcqToDisk.exe” to start averaging buffers.
3. Press any key to stop collecting data.

The following screenshot shows the sample program output.



```
C:\Windows\system32\cmd.exe
C:\AlazarTech\ATS-GPU\0.0.1\Samples\ATS9350\DualPort\NPT_FFT\Win32\Release>AcqToDisk.exe
Capturing ... press any key to abort
Capture completed in 51.22 sec
Captured 10000 buffers (195.3 buffers per sec)
Captured 5120000 records (9.997e+004 records per sec)
Transferred 41943040000 bytes (8.19e+008 bytes per sec)
C:\AlazarTech\ATS-GPU\0.0.1\Samples\ATS9350\DualPort\NPT_FFT\Win32\Release>
```

Figure 2 C++ sample application

Using the MATLAB sample applications

MATLAB sample applications demonstrate how to configure a digitizer to make an ADMA acquisition or read sample data from a file, and process the data in an OpenCL compatible GPU or using the FFTW open-source software library.

The FFT_ samples process sample data using an OpenCL compatible GPU device, while the FFTW_ samples process sample data using the FFTW software library. The _NPT samples make NPT mode ADMA acquisitions, while the _CS samples make continuous streaming ADMA acquisitions. The _File samples read sample data from a file rather than from a digitizer.

You can use the FFT_ and FFTW_ samples to compare the performance of hardware vs. software solutions. If the FFTW software provides sufficient throughput for your application, then it may not be necessary to use an OpenCL compatible GPU for real-time data processing.

To run a MATLAB sample:

1. Open MATLAB
2. Change in the “C:\AlazarTech\ATS-GPU\- 3. Modify “configureBoard.m” and “acquireData.m” as required. Search for lines with “ToDo” comments.
- 4. Type “AcqToDisk” to run the sample.

The following screenshot shows sample program output.

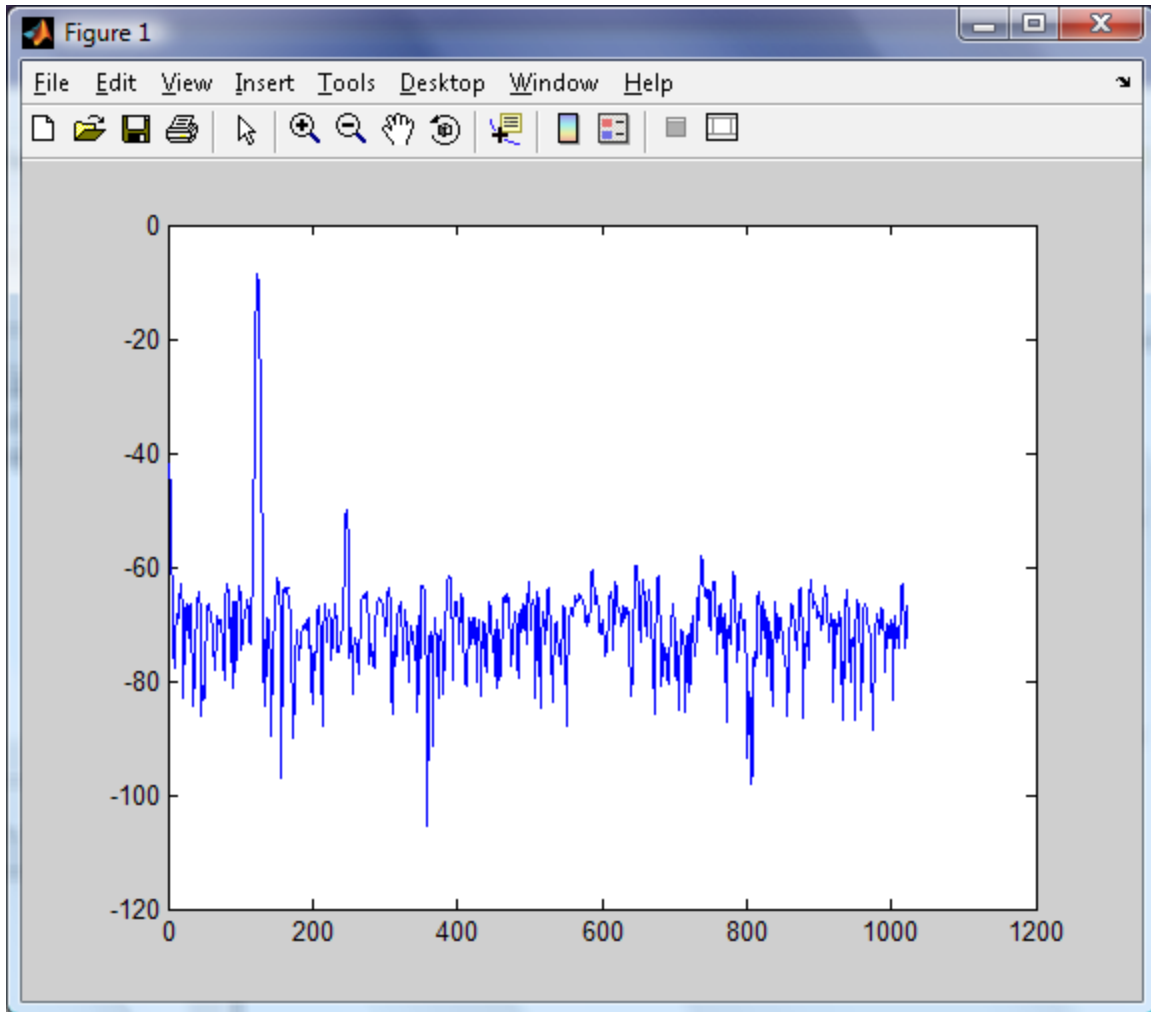


Figure 3 MATLAB sample program output

Building the sample DLL

The ATS_GPU SDK includes the source code for a C++ DLL that calls ATSApi.dll functions to configure an AlazarTech digitizer to make an ADMA acquisition, and transfer sample data to an OpenCL compatible GPU for processing. You should modify the OpenCL kernel code as required; the sample DLL inverts sample values in each DMA buffer.

To build the sample DLL:

Microsoft Visual C++ 2008 Professional

- Open the “ATS_GPU_DLL_x64.sln” solution in the ATS_GPU_DLL directory.
- Select the “Build | Configuration Manager” to display the “Configuration Manager” dialog and select the desired configuration and platform.
- Select the “Build | Build Solution” command to build the sample program.

If a build succeeds, it creates one of the following executables:

32-bit Debug	Win32\Debug\ATS_GPU_DLL.dll
--------------	-----------------------------

32-bit Release	Win32\Release\ATS_GPU_DLL.dll
64-bit Debug	x64\Debug\ATS_GPU_DLL.dll
64-bit Release	x64\Release\ATS_GPU_DLL.dll

Microsoft Visual C++ 2008 Express

- Open the “ATS_GPU_DLL.sln” solution in the sample directory.
- Select the “Build | Configuration Manager” to display the “Configuration Manager” dialog and select the desired configuration and platform.
- Select the “Build | Build Solution” command to build the sample program.

If a build succeeds, it creates one of the following executables:

32-bit Debug	Win32\Debug\ATS_GPU_DLL.dll
32-bit Release	Win32\Release\ATS_GPU_DLL.dll

Programming Guide

The ATS-GPU SDK includes a DLL that exports functions to calculate the FFT of sample buffers in an OpenCL compatible GPU, and return floating point power spectrum data to an application. The DLL can operate in data acquisition mode, or data verification mode.

Data acquisition mode allows an application to configure an AlazarTech digitizer to make an ADMA acquisition, process the data in an OpenCL compatible GPU, and return power spectrum data as an array of floating point values to an application. To use acquisition mode:

1. Call ATS-SDK functions as required to configure the sample rate, inputs, and trigger parameters.
2. Call `ATS_GPU_SetComputeDevice` to select a GPU device, if required.
3. Call `ATS_GPU_BeforeCapture` to set acquisition and GPU parameters.
4. Call `AlazarStartCapture` to arm the digitizer to begin an acquisition.
5. Call `ATS_GPU_GetBuffer` to wait for a buffer to complete, process the buffer, and return the processed data to the application.
6. Repeat from 5 to acquire and process more data.
7. Call `ATS_GPU_AbortCapture` to abort the acquisition and free resources allocated by `ATS_GPU_BeforeCapture`.

Data validation mode allows the GPU to process sample data supplied by an application, rather than sample data acquired by a digitizer. It can be used to pass known values to ATS-GPU SDK in order to validate its results. To use data validation mode:

1. Call `ATS_GPU_SetBoardType` to set the simulated board type enable data validation mode.
2. Call `ATS_GPU_SetComputeDevice` to select a GPU device, if required.
3. Call `ATS_GPU_BeforeCapture` to set acquisition parameters.
4. Call `ATS_GPU_SetBuffer` to set sample data buffer to be processed.
5. Call `ATS_GPU_GetBuffer` to process the buffer and get processed data.
6. Repeat from step 4 to supply more data for processing, or from 5 to benchmark `ATS_GPU` performance.
7. Call `ATS_GPU_AbortCapture` to free resources allocated by `ATS_GPU_BeforeCapture`.

The arrangement of data in a buffer depends on the board type and acquisition mode. The board type is set by a call to `ATS_GPU_SetBoardType`, and the acquisition mode specified in the call to `ATS_GPU_BeforeCapture`. The arrangement of data must match that of the digitizer hardware in the specified acquisition mode. See the ATS-SDK manual for more information about data organization in DMA buffers.

FFT Function Reference

The following section describes FFT functions exported by `ATS_GPU` DLL.

ATS_GPU_AbortCapture

The `ATS_GPU_AbortCapture` function aborts an acquisition, stops data processing, and releases resources allocated by `ATS_GPU_BeforeCapture`.

Syntax

```
C/C++
RETURN_CODE
ATS_GPU_AbortCapture (
    HANDLE boardHandle
);
```

Parameters

boardHandle

[in] Handle to a digitizer board.

Return values

This function returns `ApiSuccess` (512).

Remarks

`ATS_GPU_AbortCapture` must be called to abort an acquisition and resources allocated in the call to `ATS_GPU_BeforeCapture`.

ATS_GPU_BeforeCapture

The `ATS_GPU_BeforeCapture` function configures a digitizer to make an AutoDMA acquisition, and configures a GPU calculate the FFT of sample data in DMA buffers from the digitizer.

Syntax

```
C/C++
RETURN_CODE
ATS_GPU_BeforeCapture (
    HANDLE boardHandle,
    U32     channelMask,
    U32     samplesPerRecordBeforeTrigger,
    U32     samplesPerRecordAfterTrigger,
    U32     recordsPerBuffer,
    U32     windowType,
    U32     options,
    U32*    pBinsPerRecord
);
```

Parameters

boardHandle

[in] Handle to a digitizer board.

channelMask

[in] Select which channels to control.

This parameter may have one of the following identifiers or values.

Identifier	Value	Meaning
CHANNEL_A	1	Acquire from CH A only
CHANNEL_B	2	Acquire from CH B only
CHANNEL_A CHANNEL_B	3	Acquire from both CH A and B.

On digitizer with 4-input channels (ATS9440), this parameter may be one of the following identifiers or values:

Identifier	Value	Meaning
CHANNEL_C	4	CH C only
CHANNEL_A CHANNEL_C	5	CH A and CH C
CHANNEL_B CHANNEL_C	6	CH B and CH C
CHANNEL_D	8	CH D only
CHANNEL_A CHANNEL_D	9	CH A and CH D
CHANNEL_B CHANNEL_D	10	CHB and CHD
CHANNEL_C CHANNEL_D	12	CHC and CHD
CHANNEL_A CHANNEL_B CHANNEL_C CHANNEL_D	15	CHA, CHB, CHC, and CHD

samplesPerRecordBeforeTrigger

[in] Specify the number of samples before the trigger event in each record. See the remarks section below.

samplesPerRecordAfterTrigger

[in] Specify the number of samples after the trigger event in each record. See the remarks section below.

recordsPerBuffer

[in] Specify the number records per DMA buffer. See the remarks section below.

windowType

[in] Specify the FFT window function.

This parameter may have one of the following identifiers or values.

Identifier	Value	Window type
FFT_WINDOW_NONE	0	Square
FFT_WINDOW_HANNING	1	Hanning
FFT_WINDOW_HAMMING	2	Hamming
FFT_WINDOW_BLACKMAN	3	Blackman
FFT_WINDOW_BLACKMAN_HARRIS	4	Blackman-Harris
FFT_WINDOW_BARTLETT	5	Bartlett

options

[in] Specify acquisition options.

This parameter may have one of the following identifiers or values.

Identifier	Value	Description
FFT_MODE_NPT	0	Acquire multiple records (one per trigger) without pre-trigger samples.
FFT_MODE_TRADITIONAL	1	Acquire multiple records (one per trigger) optionally with pre-trigger samples.
FFT_MODE_CONTINUOUS_STREAMING	2	Acquire a single continuous record spanning multiple buffers. Do not wait for a trigger event to start the acquisition.
FFT_MODE_TRIGGERED_STREAMING	3	Acquire a single continuous record spanning multiple buffers. Wait for a trigger event to start the acquisition.

pBinsPerRecord

[out] Specify the address of a variable to receive the number of FFT bins per record that will be returned by `ATS_GPU_GetBuffer`. The number of bins per

record is equal to the number of samples per record per channel rounded up to the next power of two, multiplied by the number of enabled channels.

Return values

This function returns `ApiSuccess` (512) if it is able to start an acquisition. It may return one of the following error codes if it fails.

Return code	Description
<code>ApiDmaInProgress</code> (518)	An acquisition is already in progress. Call <code>ATS_GPU_AbortCapture</code> to stop the current acquisition.
<code>ApiInvalidHandle</code> (572)	The <code>boardHandle</code> parameter is not valid.
<code>ApiInvalidData</code> (574)	The <code>channelMask</code> , <code>samplesPerRecordPerChannel</code> , <code>recordsPerBuffer</code> , or <code>bufferCount</code> parameters are invalid.
<code>ApiInsufficientResources</code> (553)	The operating system does not have sufficient resources to allocate DMA buffers or create a worker threads.
<code>ApiUnsupportedFunction</code> (533)	ATS-GPU is not supported with this board model. Only PCIe digitizers with on-board memory are currently supported.
<code>ApiFailed</code> (513)	The OpenCL driver may have returned an error. See <code>%TempDir%\ATS_GPU_DLL.log</code> file for more information.

Remarks

In data acquisition mode, an application must call `AlazarStartCapture` to arm the board to begin the acquisition after calling `ATS_GPU_BeforeCapture`.

An application must call `ATS_GPU_AbortCapture` after calling `ATS_GPU_BeforeCapture`. `ATS_GPU_AbortCapture` aborts an acquisition in data acquisition mode, and release resources allocated by `ATS_GPU_BeforeCapture` in both data acquisition and data validation modes.

To make a traditional mod AutoDMA acquisition, set the *options* parameter to `FFT_MODE_TRADITIONAL` and the *samplesPerRecordBeforeTrigger* to the desired number of pre-trigger samples per record per channel. Otherwise, set *samplesPerRecordBeforeTrigger* to zero.

Set the board handle to `NULL` in data validation mode.

ATS_GPU_GetBuffer

The `ATS_GPU_GetBuffer` function waits for a DMA buffer to complete, transfers the buffer to a GPU to calculate an FFT of the data in the buffer, and returns a float array of FFT bin values in to the caller.

Syntax

```
C/C++
ATS_GPU_DLL_API RETURN_CODE
ATS_GetBuffer(
    HANDLE    boardHandle,
    void*     bufferOut,
    U32       samplesPerBuffer,
    U32       timeout_ms
)
```

Parameters

boardHandle

[in] Handle to a digitizer board, or NULL in data validation mode.

bufferOut

[out] Specify the address of a buffer to receive the FFT records.

samplesPerBuffer

[in] Specify the size of the FFT buffer in floats.

Timeout_ms

[in] Specify the maximum amount of time in milliseconds to wait for a DMA buffer to complete.

Return values

This function returns `ApiSuccess` (512) if the board received sufficient triggers to fill a DMA buffer, and the records in the buffer were processed to a single co-averaged record. The function may return one of the following error codes if it fails.

Return code	Description
<code>ApiNotInitialized</code> (556)	<code>ATS_StartCapture</code> was not called before calling this function, or it was called and failed.
<code>ApiInvalidHandle</code> (572)	The <code>boardHandle</code> parameter is not valid.
<code>ApiBufferOverflow</code> (582)	The board filled all of the available DMA buffers and on-board memory. This may happen if the data acquisition rate exceeds the bus bandwidth (1.6GB/s).
<code>ApiWaitTimeout</code> (579)	The timeout interval expired before the board received sufficient triggers to complete a DMA buffer. Either the board is not triggering, or the timeout value is too short.
<code>ApiFailed</code> (513)	A system or internal error occurred. Call <code>ATS_GetLastErrorText()</code> for more information.

Remarks

This function must be called at average rate that is equal to or greater than the rate at which DMA buffers complete.

This function returns the first half of the bins from each FFT. For example, if the digitizer captures 1152 samples per record, the GPU applies the specified window function to 1152 samples, calculates an FFT on 2048 samples, and returns the first 1024 bins per record per channel to the application.

The FFT bin values units are dBFS.

ATS_GPU_GetLastErrorText

The `ATS_GPU_GetLastErrorText` function returns information about the most recent error from ATS-GPU DLL for a board.

Syntax

```
C/C++
RETURN_CODE
ATS_GPU_GetLastErrorText (
    HANDLE boardHandle,
    char * buffer,
    U32    maxChars
);
```

Parameters

boardHandle

[in] Handle to a digitizer board.

buffer

[out] Specify the address of a buffer to receive the string.

maxChars

[in] Specify the size of the buffer in bytes.

Return values

This function returns a NULL terminated string with information about the most recent error from the DLL.

Remarks

If an ATS-GPU function returns an error code other than `ApiSuccess` or `ApiWaitTimeout`, call `ATS_GPU_GetLastErrorText` to get more information about the error.

ATS_GPU_QueryDeviceCount

The `ATS_GPU_QueryDeviceCount` function returns the number of compute devices in a specified OpenCL platform.

Syntax

```
C/C++
RETURN_CODE
ATS_GPU_QueryDeviceCount (
    U32  platformIndex,
    U32  *deviceCount
);
```

Parameters

platformIndex

[in] Specify the index of an OpenCL platform.

deviceCount

[out] Specify a pointer to a variable to receive the number of compute devices in the selected OpenCL platform.

Return values

This function returns `ApiSuccess` (512) and sets the value pointed to by *deviceCount* if it succeeds. It returns `ApiFailed` (513) if it fails. See the `%TempDir%\ATS_GPU_DLL.log` file for more information.

Remarks

OpenCL organizes compute devices into platforms and devices. If your computer includes more than one GPU device in an OpenCL platform, then it can be addressed using the 0-based device index.

ATS_GPU_QueryDeviceName

The `ATS_GPU_QueryDeviceName` function returns the name of a compute device in an OpenCL platform.

Syntax

```
C/C++
RETURN_CODE
ATS_GPU_QueryDeviceName (
    U32    platformIndex,
    U32    deviceIndex,
    char *buffer,
    int    maxChars
);
```

Parameters

platformIndex

[in] Specify the index of the platform.

deviceIndex

[in] Specify the index of a device within a compute platform.

buffer

[out] Specify the address of a buffer to receive the platform name.

maxChars

[in] Specify the size of the buffer in bytes.

Return values

This function returns `ApiSuccess` (512) and copies the platform name to user supplied buffer if it is succeeds. It returns `ApiFailed` (513) if it fails. See the `%TempDir %\ATS_GPU_DLL.log` file for more information.

Remarks

OpenCL organizes compute devices into platforms and devices. If your computer includes more than one OpenCL platform, then you can address a platform using the 0-based platform index.

ATS_GPU_QueryPlatformCount

The `ATS_GPU_QueryPlatformCount` function returns the number of OpenCL platforms detected.

Syntax

```
C/C++
RETURN_CODE
ATS_GPU_QueryPlatformCount (
    U32 *platformCount
);
```

Parameters

platformCount

[out] Specify a pointer to a variable to receive the number of OpenCL platforms detected.

Return values

This function returns `ApiSuccess` (512) and sets value pointed to by *platformCount* if it succeeds. It may return one of the following error codes if it fails.

Return code	Description
ApiFailed (513)	The OpenCL driver returned an error. See %TempDir %\ATS_GPU_DLL.log file for more information.

Remarks

If your computer includes more than one OpenCL platform, then select a platform using the 0-based platform index.

ATS_GPU_QueryPlatformName

The `ATS_GPU_QueryPlatformName` function returns the name of an OpenCL platform.

Syntax

```
C/C++
RETURN_CODE
ATS_GPU_QueryPlatformName (
    U32    platformIndex,
    char *buffer,
    int    maxChars
);
```

Parameters

platformIndex

[in] Specify the index of the platform.

buffer

[in|out] Specify the address of a buffer to receive the platform name.

maxChars

[in] Specify the size of the buffer in bytes.

Return values

This function returns `ApiSuccess` (512) and copies the platform name to user supplied buffer if it is succeeds. It may return one of the following error codes if it fails.

Return code	Description
<code>ApiInsufficientResources</code> (533)	ATS_GPU_DLL was unable to allocate host memory.
<code>ApiInvalidIndex</code> (539)	The <i>platformIndex</i> parameter is greater than or equal to the number of platforms detected.
<code>ApiFailed</code> (513)	The OpenCL driver returned an error. See %TempDir %\ATS_GPU_DLL.log file for more information.

Remarks

If your computer includes more than one OpenCL platform, then select a platform using the 0-based platform index.

ATS_GPU_SetBoardType

The `ATS_GPU_SetBoardType` function sets the digitizer type for data validation mode.

Syntax

```
C/C++
RETURN_CODE
ATS_GPU_SetBoardType (
    U32 boardType
)
```

Parameters

boardType

[in] Specify a digitizer model for data verification mode, or 0 to disable data verification mode.

This parameter may have one of the following identifiers or values.

Identifier	Value	Bit per sample	Channels	Sample Interleave
ATS9462	11	16	2	No
ATS9870	13	8	2	Yes
ATS9350	14	12	2	Yes
ATS9325	15	12	2	Yes
ATS9440	16	14	4	Yes
ATS9410	17	14	4	Yes
ATS9351	18	12	2	Yes
ATS9310	19	12	2	Yes
ATS9461	20	16	2	Yes
ATS9850	21	8	2	Yes

Return values

This function returns `ApiSuccess` (512) if it succeeds or `ApiFailed` (513) if the board type is not supported.

Remarks

This function must be called before `ATS_GPU_BeforeCapture` to enable data validation mode. In this mode, an application calls `ATS_GPU_SetBuffer` to write sample data to the GPU, and `ATS_GPU_GetBuffer` to get processed buffers from the GPU. This mode can be used to verify the data returned by a GPU, and benchmark the time required to perform calculations.

ATS_GPU_SetBuffer

The ATS_GPU_SetBuffer sets sample data in validation mode.

Syntax

```
C/C++
RETURN_CODE
ATS_GPU_SetBuffer (
    void *buffer,
    U32    samplesPerBuffer
)
```

Parameters

buffer

[in] Specify the address of a buffer containing sample data. See the remarks below.

samplesPerBuffer

[in] Specify the size of the buffer in samples.

Return values

This function returns ApiSuccess (512) if it succeeds. It may return one of the following error codes if it fails.

Return code	Description
ApiNotInitialized (556)	The board type was not set. Call ATS_GPU_SetBoardType before calling this function.
ApiBufferNotReady (573)	DMA buffer were not allocated. Call ATS_GPU_BeforeCapture before calling this function.

Remarks

Sample data must be arranged in a buffer the same way as sample data from a digitizer.

- The number of bits and bytes per sample value must match those of the board type specified in the call to ATS_GPU_SetBoardType.
- The organization of sample values into records in a buffer must match the organization of sample data in the AutoDMA mode specified in the call to ATS_GPU_BeforeCapture.

See the ATS-SDK Guide for more information about how a digitizer organizes sample data in AutoDMA buffers.

ATS_GPU_SetComputeDevice

The `ATS_GPU_SetComputeDevice` function allows you to specify which GPU should be used to process sample data from a digitizer, if more than one GPU is available.

Syntax

```
C/C++
RETURN_CODE
ATS_GPU_SetComputeDevice(
    HANDLE boardHandle,
    U32 platformIndex,
    U32 deviceIndex
);
```

Parameters

boardHandle

[in] Handle to a digitizer board.

platformIndex

[in] Specify the 0-based index of the OpenCL platform containing the compute unit that will process sample data. Set this value to 0 if only one compute unit is available.

deviceIndex

[in] Specify the 0-based index of the OpenCL device within the specified platform of the compute unit that will process sample data. Set this value to 0 if only one compute unit is available.

Return values

This function returns `ApiSuccess` (512) if it detects the specified compute device. It returns `ApiFailed` (513) if it fails. See the `%TempDir%\ATS_GPU_DLL.log` file for more information.

Remarks

OpenCL organizes compute devices into platforms and devices. If your computer includes more than one OpenCL compute device, then address a specific GPU by specifying a 0-based platform and device index.

For more information

Refer to the ATS-SDK Programmer's Guide for more information about the ATS-SDK functions.

<http://www.alazartech.com/support/Download%20Files/ATS-SDK-Guide-6.0.1.pdf>

Contact us if you have any other questions or comments, or need assistance.

Web	http://www.alazartech.com
Email	support@alazartech.com
Phone	514-426-4899
Fax	514-426-2723
Mail	Alazar Technologies Inc. 6600 Trans-Canada Highway, Suite 310 Pointe-Claire, QC Canada H9R 4S2

Release history

Version	Date	Description
0.0.2	2011/12/14	<ul style="list-style-type: none">• Add support for ATS9440, ATS9462, ATS9351 and ATS9870 digitizers.• Add support for continuous streaming, triggered streaming, and traditional mode ADMA acquisitions.• Add functions to display information about available compute devices, and allow an application to select a device to be used for data processing.• Add support for data verification mode to allow an application to supply sample data to a GPU for processing.
0.0.1	2011/11/07	First release