



# **CompuGen PCI SDK User's Guide**

**For CompuGen 4300,  
CompuGen 8150, CompuGen 8152,  
and CompuGen 11G**

**Supporting:**

- **CompuGen PCI SDK for C/C++**
- **CompuGen PCI SDKs for LabVIEW**
- **CompuGen SDKs for MATLAB**

Reorder #: MKT-SWM-CGPCI02  
0511

© Copyright Gage Applied Technologies 2005, 2006

## Second Edition (March 2006)

CompuGen is a registered trademark of Gage Applied Technologies. Windows is a registered trademark of Microsoft Incorporated. LabVIEW is a registered trademark of National Instruments Inc. MATLAB is a registered trademark of The MathWorks Inc. All other trademarks are registered trademarks of their respective companies.

Changes are periodically made to the information herein; these changes will be incorporated into new editions of the publication. Gage Applied Technologies may make improvements and/or changes in the products and/or programs described in this publication at any time.

Copyright © 2005, 2006 Gage Applied Technologies. All Rights Reserved, including those to reproduce this publication or parts thereof in any form without permission in writing from Gage Applied Technologies.

The installation program used to install the CompuGen PCI SDKs for Windows, **InstallShield**, is licensed software provided by InstallShield Software Corp., 900 National Parkway, Ste. 125, Schaumburg, IL. **InstallShield** is Copyright © 2000 by InstallShield Software Corp., which reserves all copyright protection worldwide. **InstallShield** is provided to you for the exclusive purpose of installing the CompuGen SDKs for Windows. Gage Applied Technologies is exclusively responsible for the support of the CompuGen SDKs for Windows, including support during the installation phase. In no event will **InstallShield** Software Corp. be able to provide any technical support for the CompuGen SDKs for Windows.

Please complete the following section and keep it handy when calling for GaGe **technical support**:

Owned by: \_\_\_\_\_  
Serial Number: \_\_\_\_\_  
Purchase Date: \_\_\_\_\_  
Purchased From: \_\_\_\_\_

You must also have the following information when you call:

- Brand name and type of computer
- Processor and bus speed
- Total memory size
- Information on all other hardware in the computer

### ***How to reach GaGe Product Support***

Toll-free phone: (800) 567-GAGE

Toll-free fax: (800) 780-8411

### ***To reach GaGe from outside North America***

Tel: +1-514-633-7447

Fax: +1-514-633-0770

**E-mail:** [prodinfo@gage-applied.com](mailto:prodinfo@gage-applied.com)

**Website:** <http://www.gage-applied.com>

# Table of Contents

---

<b>Preface .....</b>	<b>5</b>
<b>CompuGen PCI C/C++ SDK Overview.....</b>	<b>6</b>
CompuGen PCI C/C++ API.....	7
Constants returned by CompuGen API functions .....	7
Structures used by CompuGen API functions.....	9
CompuGen API Functions .....	10
CgInitialize .....	10
CgGetBoardName .....	11
CgGetBoardType.....	12
CgGetCaps .....	13
CgSet.....	15
CgDo.....	17
CgRelease.....	18
<b>CompuGen PCI LabVIEW SDK Overview .....</b>	<b>19</b>
CgLv Sub-VIs.....	21
CgLv_ConfigureBoard.vi.....	21
CgLv_ConfigureChannel.vi .....	22
CgLv_ConfigureChannelAtten.vi.....	23
CgLv_ConfigureSegment.vi .....	24
CgLv_ErrorHandler.vi .....	25
CgLv_GetBoardName.vi.....	25
CgLv_GetChannelCount.vi .....	26
CgLv_Intialize.vi.....	26
CgLv_LoadSegBuffer.vi .....	27
CgLv_ResetSegments.vi .....	28
CgLv_Start.vi .....	28
CgLv_Stop.vi .....	29
CgLv_Trigger.vi .....	29
<b>CompuGen PCI MATLAB SDK Overview.....</b>	<b>30</b>
CompuGen PCI MATLAB CgCall Functionality.....	31
CompuGen MATLAB CgCall Modes.....	32
CgCall(INITIALIZE).....	32
CgCall(QUERY, board_number, command_item) .....	33
QUERY_CHANNEL_COUNT .....	33
QUERY_RATE_COUNT.....	33
QUERY_ATTENUATION.....	33
QUERY_EXTERNAL_CLOCK.....	33
QUERY_RATES .....	33
CgCall(CONFIGURE, board_number, command_item, structure, signal_buffer) .....	34
CONFIGURE_SEGMENT.....	34
CONFIGURE_CHANNEL.....	35
CONFIGURE_BOARD .....	35
CgCall(DO, board_number, command_item).....	36
ACTION_START_TRIGGERED .....	36
ACTION_START_FREE .....	36
ACTION_TRIGGER.....	36
ACTION_STOP .....	36

<b>Advanced CompuGen PCI SDK Functionality .....</b>	<b>37</b>
Link'n'Loop.....	37
Introduction .....	37
The seg Structure .....	38
Link'N'Loop Operation .....	39
Digital Output Marker Control .....	41

This manual is designed to describe CompuGen PCI Software Development Kits (SDKs) for the Windows 2000, and Windows XP environments. Separate SDKs are provided for C/C++, LabVIEW, and MATLAB.

All Software Development Kits are installed by the CompuGen driver installation in the **O/S system drive: \Program Files\Gage\CompuGen\sample** directory. This manual assumes that the user is familiar with operation of the programming language (C, LabVIEW, or MATLAB) in use. No description is given for these topics. The manual also refers to CompuGen functionality only in so far as it is necessary to describe CompuGen control from the SDK in use. For detailed description to CompuGen functionality, please refer to the CompuGen PCI Hardware Manual and Installation Guide.

# CompuGen PCI C/C++ SDK

---

## CompuGen PCI C/C++ SDK Overview

In order to operate the CompuGen PCI C/C++ SDK, the user requires Microsoft Visual Studio 6.0 or higher (both VS 6.0 and VS 7.1 projects are included). The CompuGen PCI C/C++ SDK consists of a single Microsoft Visual C project called CgSimple.dsw. The project includes the main C source code in CgSimple.c as well as all required supporting files. CgSimple.c may be compiled using different Windows compilers such as Borland C Builder, however, no project files are supplied and the user must construct the project file themselves.

The structure of CgSimple.c is very straightforward, but may be easily extended to more complex operation by reading about and understanding the basic C API CompuGen subroutine calls and making appropriate modifications.

An overview of the operation of CgSimple.c is given below.

1. Initialize the CompuGen hardware and determine the number of CompuGen boards installed in the host PC.
2. Obtain the name of the first CompuGen board in the host PC. All subsequent coding will operate only on this first CompuGen board.
3. Calculate the data points for a single cycle sine wave waveform that is 1024 points long. The values to be uploaded to the CompuGen board must take on the values from 0 to 4095. A 0 value corresponds to negative full scale output. A value of 4095 corresponds to positive full scale. A value of 2048 corresponds to 0 Volts.
4. Return the number of channels on the first CompuGen board in the host PC.
5. Set the segment configuration. This entails setting the Generation Mode to Free Run and setting the segment length on the board, which is 1024 by default. The segment length is the number of data points in the waveform segment that will be generated. This is distinct from the pattern buffer length, which may be different.
6. Upload the sinusoidal cycle pattern created in step 3 to all channels on the first CompuGen board. In this step, the length of the pattern, `dwBuffLength`, is passed to the CompuGen board. While `dwBuffLength` is set equal to `dwLength` within CgSimple.c, this need not be the case. If the uploaded pattern is longer (`dwBuffLength > dwLength`), then only the first `dwLength` points from the pattern are generated. If the uploaded pattern is shorter (`dwBuffLength < dwLength`), then pattern is extended so that it is `dwLength` points long. This is done by repeating the last point in the uploaded pattern until it is padded up to `dwLength`.
7. Set the CompuGen generation parameters. These parameters include the conversion rate and the flag to activate External Clocking. The default conversion rate is set to the maximum available for the CompuGen board in use.
8. Start generation on the CompuGen board in Free Run Mode.
9. Generation continues until a key is pressed at which point generation is stopped, all allocated buffers are freed, and the driver is released.

Please note that in developing a C application from CgSimple.c, the user should maintain the above order in making C API subroutine calls. Making the calls in an arbitrary order may result in faulty operation.

# CompuGen PCI C/C++ SDK

---

## CompuGen PCI C/C++ API

CgSimple.c may be easily modified to implement different functionality by modifying the code accordingly. For instance, the user might easily calculate a pattern other than a sinusoidal cycle and upload it to one or more channels. The following sections completely document the elements of the CompuGen API (Application Program Interface). These elements are the values of the defined constants returned by the API functions, the data structures used by the API functions, and a list of the API functions themselves and their functionalities.

### Constants returned by CompuGen API functions

The following definitions are used to identify which CompuGen model was detected.

```
#define CG_11G 0x00000001
#define CG_4300 0x00000002
#define CG_8150 0x00000003
#define CG_8152 0x00000004
```

The following definitions are used to specify which capability to query during a call to CgGetCaps().

```
#define CG_CHAN_NUM 1
#define CG_RATE_NUM 2
#define CG_RATE 3
#define CG_ATTEN 4
#define CG_EXT_CLK 5
#define CG_MAX_MEM 6
```

The following definitions are used to specify which configuration parameter group to set during a call to CgSet().

```
#define CG_BOARD_CONF 1
#define CG_SEGMENT_CONF 2
#define CG_CHANNEL_CONF 3
```

The following definitions are used to specify which action should be executed during a call to CgDo().

```
#define CG_DO_START_TRIGGERED 1
#define CG_DO_STOP 2
#define CG_DO_TRIGGER 3
#define CG_DO_START_FREE 4
```

# CompuGen PCI C/C++ SDK

---

These macros are defined to help simplify validation of the return values from the CompuGen API functions.

```
#define CG_FAILED(x) ((x)<0?TRUE:FALSE)
#define CG_SUCCEEDED(x) ((x)<0?FALSE:TRUE)
```

The following definitions are used to identify which error occurred during a function call, if any.

```
#define CG_FALSE 0
#define CG_SUCCESS 1
#define CG_INVALID_CHANNEL -1
#define CG_NO_ATTENUATION -2
#define CG_SEGMENT_TOO_BIG -3
#define CG_TOO_MANY_SEGMENTS -4
#define CG_INVALID_SEGMENT -5
#define CG_ZERO_LENGTH -6
#define CG_INVALID_BOARD -7
#define CG_INVALID_CAPS_ID -8
#define CG_BAD_POINTER -9
#define CG_INVALID_ID -10
#define CG_FAIL 0x80000000
```

The following definitions are required by the CompuGen API.

```
#define CG_API __stdcall
typedef long CG_STATUS;
```



# CompuGen PCI C/C++ SDK

---

## Structures used by CompuGen API functions

```
typedef struct _CG_GEN_CONFIG
{
    Float    fConversionRate;    // Conversion rate in Hz for the signal to
                                // be generated.
    bool     bExtClock;         // Flag that activates external clocking
                                // when TRUE.
    bool     bReserved;        // This parameter is currently ignored and
                                // is reserved for future use.
}CG_GEN_CONFIG;

typedef struct _CG_SEG_CONFIG
{
    DWORD    dwLength;         // Length in points of the segment to be
                                // generated by the CompuGen board. It
                                // must be a multiple of 16 with a 64 point
                                // minimum.
    DWORD    dwLoopCount;     // Currently only 0 is supported. This
                                // parameter has no effect.
    bool     bTriggered;      // This flag sets the CompuGen
                                // conversion mode.
                                // For Triggered Mode, set to TRUE.
                                // For Free Run Mode, set to FALSE.
}CG_SEG_CONFIG, *PCG_SEG_CONFIG;

typedef struct _CG_CHAN_CONFIG
{
    DWORD    dwChanNum;       // Number of the channel to be configured.
                                // Use values 1 to n to configure a specific
                                // channel. A value of 0 will configure all
                                // channels identically.
    DWORD    dwBufLength;     // Size of the buffer that contains the
                                // waveform pattern to be uploaded to the
                                // CompuGen memory.
    Float    fAtten;         // Output attenuation factor for the
                                // CG4300, in dB. Ignored for other
                                // CompuGen models.
}CG_CHAN_CONFIG, *PCG_CHAN_CONFIG;
```

All structures are packed on 8 byte boundaries. (This information is only important if you are trying to call CompuGen C API functions from another programming language, such as Visual Basic.)

# CompuGen PCI C/C++ SDK

---

## CompuGen API Functions

### CgInitialize

#### Function Prototype:

```
CG_STATUS CG_API CgInitialize(void);
```

#### Purpose:

This function initializes the CompuGen hardware and driver. It also identifies the number of CompuGen boards installed in the host PC.

#### Parameters:

None

#### Returns:

A positive return value indicates the number of CompuGen boards found. A negative return value indicates an error. The value is the error code.

#### Example Code:

```
//Initialize the driver and query number of board available  
long lBoards = CgInitialize();
```

# CompuGen PCI C/C++ SDK

---

## CgGetBoardName

### Function Prototype:

```
CG_STATUS CG_API CgGetBoardName(DWORD dwBoardNumber, size_t stLen,  
char* pBoardName);
```

### Purpose:

This function obtains a text string containing the name of the CompuGen board.

### Parameters:

dwBoardNumber	Number of the CompuGen board to be addressed, starting from 0.
stLen	The size of the string variable to which pBoardName points.
pBoardName	A pointer to the string variable supplied by the user. This string will be returned containing the CompuGen board name.

### Returns:

CG\_SUCCESS upon success or error code upon failure.

### Example Code:

```
//Get name of the first board  
char strBoardName[MAX_PATH];  
CgGetBoardName(0, MAX_PATH, strBoardName);
```

# CompuGen PCI C/C++ SDK

---

## CgGetBoardType

### Function Prototype:

```
DWORD CG_API CgGetBoardType(DWORD dwBoardNumber);
```

### Purpose:

This function gets the CompuGen board type (model) for the selected board number.

### Parameters:

dwBoardNumber	Number of the CompuGen board to be addressed, starting from 0.
---------------	--

### Returns:

Returns the CompuGen board type or model, which is a positive number equal to CG\_11G, CG\_4300, CG\_8152, or CG\_8150.  
A negative return value indicates an error. The value is the error code.

### Example Code:

```
DWORD dwBoardType = CgGetBoardType(0);
```

# CompuGen PCI C/C++ SDK

---

## CgGetCaps

### Function Prototype:

```
CG_STATUS CG_API CgGetCaps(DWORD dwBoardNumber, DWORD dwCapsId, void* pBuffer, DWORD dwBufferSize);
```

### Purpose:

This function gets the capabilities of the selected CompuGen board.

### Parameters:

dwBoardNumber	Number of the CompuGen board to be addressed, starting from 0.
dwCapsId	Specifies which CompuGen capabilities are to be obtained by CgGetCaps(). Must be one of the following:
CG_CHAN_NUM	Causes CgGetCaps() to return the number of CompuGen channels.
CG_RATE_NUM	Causes CgGetCaps() to return the number of different internal CompuGen conversion rates.
CG_MAX_MEM	Causes CgGetCaps() to return the size of the maximum memory available per channel on the selected CompuGen card.
CG_ATTEN	Causes CgGetCaps() to return a 1 if an output attenuator is available on the selected CompuGen card. Otherwise, a zero value is returned.
CG_EXT_CLK	Causes CgGetCaps() to return a 1 if the selected CompuGen card supports external clock. Otherwise, a zero value is returned.
CG_RATE	Causes CgGetCaps() to fill up the buffer variable to which pBuffer points with the valid internal conversion rates for the selected CompuGen board. pBuffer should point to an array of float variables with enough elements to hold the retrieved information (CG_RATE_NUM or more).
pBuffer	Ignored unless dwCapsId = CG_RATE. If dwCapsId = CG_RATE, pBuffer is a pointer to a buffer variable that will be returned containing requested information.
dwBufferSize	Ignored unless dwCapsId = CG_RATE. If dwCapsId = CG_RATE, dwBufferSize is the size in bytes of the buffer to which pBuffer points.

# CompuGen PCI C/C++ SDK

---

## Returns:

Return value depends on the value of dwCapsId.  
An error code is returned upon failure.

## Example Code:

```
//Query number of channels on the first board  
    long lChanNum = CgGetCaps(0, CG_CHAN_NUM, NULL, 0);
```

# CompuGen PCI C/C++ SDK

---

## CgSet

### Function Prototype:

```
CG_STATUS CG_API CgSet(DWORD dwBoardNumber, DWORD dwSetId, void* pStruct, void* pBuffer);
```

### Purpose:

This function configures the selected CompuGen board

### Parameters:

dwBoardNumber	Number of the CompuGen board to be addressed, starting from 0.
dwSetId	Specifies which board settings to configure. Must be one of the following:
CG_BOARD_CONF	To configure overall conversion parameters
CG_SEG_MARKER_CONF	To configure the segment with one marker
CG_SEG_MARKER_MOD	To modify configuration of the segment with one marker
CG_CHANNEL_CONF	To configure the channel attenuation and load channel data. This Id used only in single segment mode
CG_CHANNEL_ATTEN	To configure the channel attenuation in Link'N'Loop mode
CG_SEG_BUFFER	To load data for specified segment of the specified channel in Link'N'Loop mode
CG_SEGMENT_RESET	To reset or clear previous segment configuration
pStruct	Points to a variable whose type must be appropriate for the dwSetId being used.
	For dwSetId = CG_BOARD_CONF, use type CG_GEN_CONFIG
	For dwSetId = CG_SEG_MARKER_CONF or CG_SEG_MARKER_MOD, use type CG_SEG_MARK_CONFIG
	For dwSetId = CG_SEG_BUFFER, use type CG_SEG_BUF_CONFIG
	For dwSetId = CG_CHANNEL_CONF or CG_CHANNEL_ATTEN, use type CG_CHAN_CONFIG
	For deSetId = CG_SEGMENT_RESET, use null pointer

# CompuGen PCI C/C++ SDK

---

pBuffer

Use when dwSetId = CG\_CHANNEL\_CONF or CG\_SEG\_BUFFER, otherwise ignored. Pointer to the buffer that contains the waveform pattern to be uploaded to the CompuGen memory. pBuffer must point to an array of short variables with dwBuffLength elements. A short variable value of 4095 corresponds to negative full scale. A value of 2048 corresponds to 0 Volts.

## Returns:

CG\_SUCCESS upon success or error code upon failure.

## Example Code:

```
//Configure board generation parameters
CG_GEN_CONFIG gen;
gen.fConversionRate = 300e6F;
gen.bExtClock = false;
gen.bReserved = false;
st = CgSet(0, CG_BOARD_CONF, &gen, NULL);
```



# CompuGen PCI C/C++ SDK

---

## CgDo

### Function Prototype:

```
CG_STATUS CG_API CgDo(DWORD dwBoardNumber, DWORD dwActionId);
```

### Purpose:

This function performs an action on the selected CompuGen board.

### Parameters:

dwBoardNumber	Number of the CompuGen board to be addressed, starting from 0.
dwActionId	Specifies which action is to be executed by CgDo(). Must be one of the following: CG_DO_START_TRIGGERED Start signal generation in Triggered Mode. CG_DO_STOP Stop signal generation. CG_DO_TRIGGER Force generation of a single waveform in Triggered Mode. CG_DO_START_FREE Start signal generation in Free Run Mode.

### Returns:

CG\_SUCCESS upon success or error code upon failure.

### Example Code:

```
//Start generation in Free Run Mode  
st = CgDo(0, CG_DO_START_FREE);
```

# CompuGen PCI C/C++ SDK

---

## CgRelease

### Function Prototype:

```
CG_STATUS CG_API CgRelease(void);
```

### Purpose:

This function causes the C application to release the CompuGen driver and frees all allocated resources.

### Parameters:

None

### Returns:

A positive return value indicates that driver successfully released.  
A negative return value indicates an error. The value is the error code.

### Example Code:

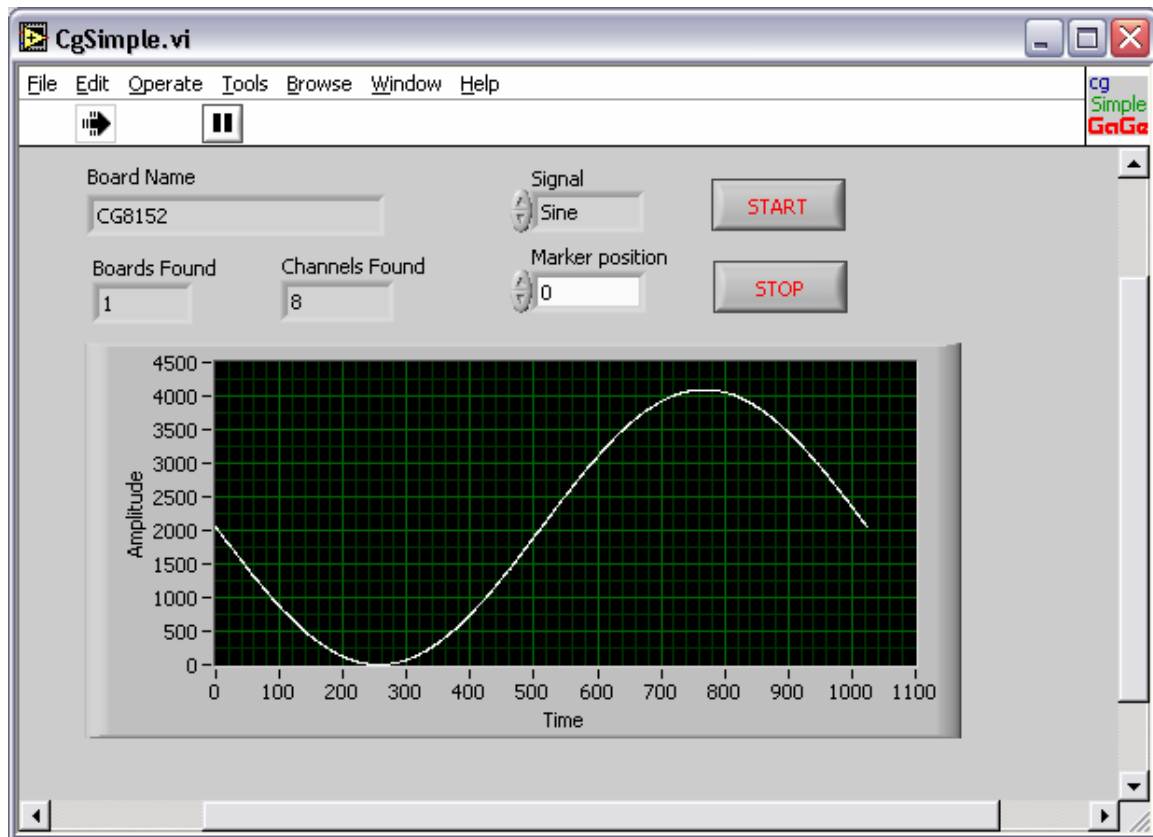
```
//Release the driver and free allocated resources  
st = CgRelease();
```

# CompuGen PCI LabVIEW SDK

## CompuGen PCI LabVIEW SDK Overview

In order to operate the CompuGen PCI LabVIEW SDK, the user requires National Instruments LabVIEW version 6.1 or higher. The CompuGen PCI LabVIEW SDK consists of two LabVIEW VI called CgSimple.vi and CgLnL.vi. The CgSimple VI allows generation of a sine, triangle, square, or ground waveforms on the CompuGen hardware.

The front panel of CgSimple.vi is shown below.



CgSimple.vi is constructed using lower level sub-VIs, each of which is a simple wrapper VI that calls the CompuGen driver Dynamically Linked Library (DLL). CgSimple.vi may be easily extended to more complex operation by reading and understanding the descriptions of the CgLv VIs and making appropriate modifications.

An overview of the operation of CgSimple.vi is given below.

1. Initialize the CompuGen hardware and determine the number of CompuGen boards installed in the host PC.
2. Obtain and display the name of the first CompuGen board in the host PC. All subsequent coding will operate only on this first CompuGen board.
3. Obtain and display the number of channels on the first CompuGen board in the host PC.
4. Wait until the Start button is pressed.

# CompuGen PCI LabVIEW SDK

---

5. Reset previous segment configuration.
6. Set the segment configuration. This entails setting the Generation Mode to Free Run and setting the segment Length on the board, which is 1024 by default. The segment Length is the number of data points in the waveform segment that will be generated. This is distinct from the pattern Buffer Length, which may be different.
7. Calculate the pattern data values for the selected waveform type. The values to be uploaded to the CompuGen board must take on the values from 0 to 4095. A 0 value corresponds to negative full scale output. A value of 4095 corresponds to positive full scale. A value of 2048 corresponds to 0 Volts.
8. Upload the selected pattern created in step 6 to all channels on the first CompuGen board. In this step, the length of the pattern, Buffer Length, is passed to the CompuGen board. While Buffer Length is set equal to Length within CgSimple.vi, this need not be the case. If the uploaded pattern is longer (Buffer Length > Length), then only the first Length points from the pattern are generated. If the uploaded pattern is shorter (Buffer Length < Length), then pattern is extended so that it is Length points long. This is done by repeating the last point in the uploaded pattern until it is padded up to Length.
9. Set the CompuGen generation parameters. These parameters include the conversion rate and the flag to activate External Clocking. The default conversion rate is set to the maximum available for the CompuGen board in use.
10. Start generation on the CompuGen board in Free Run Mode.
11. Generation continues until the Stop button is pressed at which point generation is stopped. Please note that this is the end of the sequence. In order to start generation again, you will need to restart the VI.

Please note that in developing a VI from CgSimple.vi, the user should maintain the above order in making calls to CsLv VIs. Making the calls in an arbitrary order may result in faulty operation. The following section describes the operation of each CgLv VI.

# CompuGen PCI LabVIEW SDK

---

## CgLv Sub-VIs

### CgLv\_ConfigureBoard.vi

This VI configures the overall CompuGen generation parameters.

Board Number – number of the CompuGen board to be addressed, starting from 0.

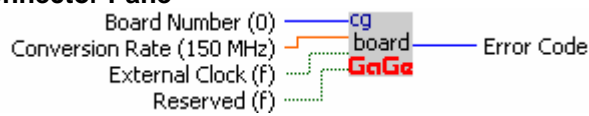
Conversion Rate – conversion rate in Hz.

External Clock – flag to activate external clocking.

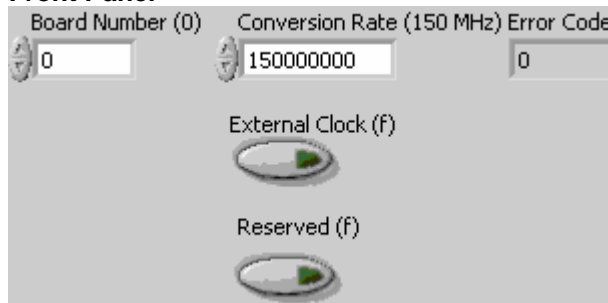
Reserved – this control has no effect.

The VI returns an Error Code.






#### Connector Pane



#### Front Panel



#### Controls and Indicators

-  Conversion Rate (150 MHz)
-  External Clock (f)
-  Reserved (f)
-  Board Number (0)
-  Error Code

# CompuGen PCI LabVIEW SDK

## CgLv\_ConfigureChannel.vi

This VI configures a CompuGen channel specified by Channel Number on the board specified by Board Number.

Board Number – number of the CompuGen board to be addressed, starting from 0.

Buffer – array of U16 variables containing the segment data. A value of 0 corresponds to positive full scale. A value of 4095 corresponds to negative full scale. A value of 2048 corresponds to 0 Volts.

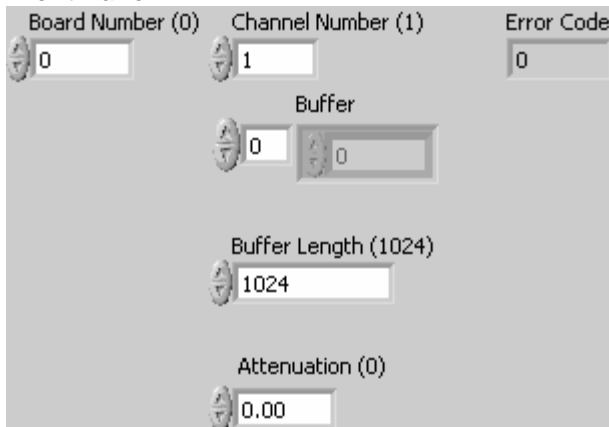
Buffer Length – number of elements in the Buffer array. If Buffer Length is greater than segment Length, then only the first Length samples in the pattern will be generated. If Buffer Length is less than segment Length, then the last sample in the pattern buffer will be repeated to fill the rest of the segment Length.

Attenuation sets the attenuation factor, in dB, on the CompuGen output attenuator, if available. The VI returns an Error Code.








### Connector Pane



### Front Panel



### Controls and Indicators

-  Channel Number (1)
-  Buffer Length (1024)
-  Attenuation (0)
-  Board Number (0)
-  Buffer
-  Numeric
-  Error Code

# CompuGen PCI LabVIEW SDK

---

## CgLv\_ConfigureChannelAtten.vi

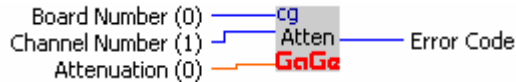
This VI configures a channel, specified by Channel Number, of the board specified by Board Number.

Buffer - array of U16 variables containing the segment data. A value of 0 corresponds to positive full scale. A value of 4095 corresponds to negative full scale. A value of 2048 corresponds to 0 Volts.

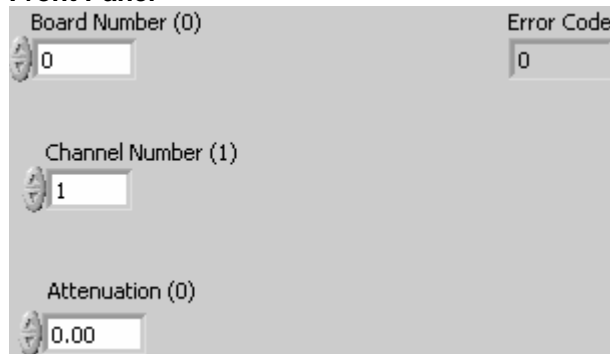
Buffer Length - number of elements in the Buffer array. If Buffer Length is greater than segment Length, then only the first Length samples in the pattern will be generated. If Buffer Length is less than segment Length, then the last sample in the pattern buffer will be repeated to fill the rest of the segment Length.

Attenuation sets the attenuation factor, in dB, on the CompuGen output attenuator, if available. The VI returns an Error Code.





### Connector Pane



### Front Panel



### Controls and Indicators

-  Channel Number (1)
-  Attenuation (0)
-  Board Number (0)
-  Error Code

# CompuGen PCI LabVIEW SDK

## CgLv\_ConfigureSegment.vi

This VI configures the output waveform segment for the CompuGen board specified by Board Number.

Board Number – number of the CompuGen board to be addressed, starting from 0.

The Length must conform to the limitations specified in the CompuGen PCI hardware manual for the CompuGen model in use.

Loop count specifies how many times to repeat a segment before switching to the next one. It should be set to 0 for continuous operation. This control is only necessary in Link'N'Loop mode and should be left disconnected in normal generation mode.

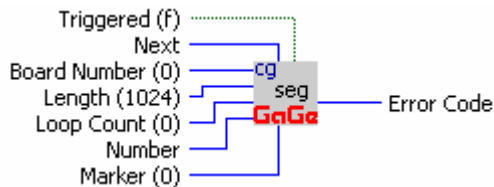
The Triggered input sets the CompuGen generation mode. For Triggered Mode, set to TRUE. For Free Run Mode, set to FALSE.

Marker specifies the position of the marker output for this segment. If the marker value is greater than the segment Length, no marker pulse is generated.

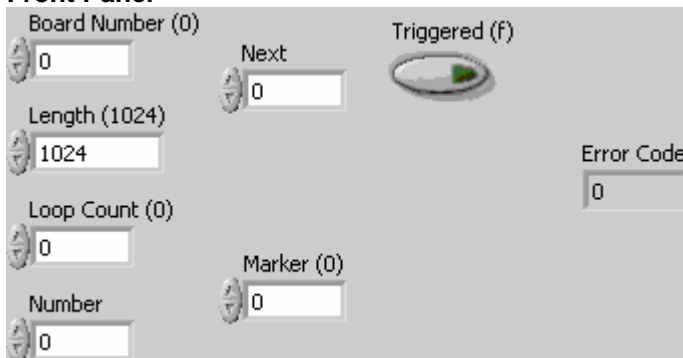
Next specifies the subsequent segment number to be generated in Link'N'Loop mode. This control is only necessary in Link'N'Loop mode and should be left disconnected in normal generation mode.

The VI returns an Error Code.









### Connector Pane



### Front Panel



### Controls and Indicators

-  Length (1024)
-  Loop Count (0)
-  Triggered (f)
-  Board Number (0)
-  Marker (0)
-  Number
-  Next
-  Error Code



# CompuGen PCI LabVIEW SDK

---

## CgLv\_ErrorHandler.vi

This VI returns a descriptive error string that corresponds to the input Error Code. If StopOnError is TRUE, execution of the main VI is stopped if the Error Code indicates that an error has occurred.




### Connector Pane



### Front Panel



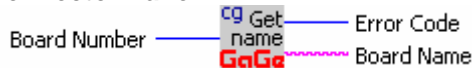
### Controls and Indicators

-  Error Code
-  StopOnError (true)
-  Error String

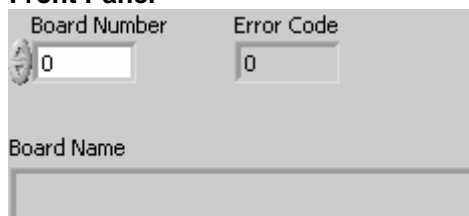
## CgLv\_GetBoardName.vi

This VI retrieves the model name of the CompuGen board specified by Board Number. Returns the Board Name as a string variable and an Error Code. Board Number – number of the CompuGen board to be addressed, starting from 0. The VI returns an Error Code.




### Connector Pane



### Front Panel



### Controls and Indicators

-  Board Number
-  Error Code
-  Board Name

# CompuGen PCI LabVIEW SDK

---

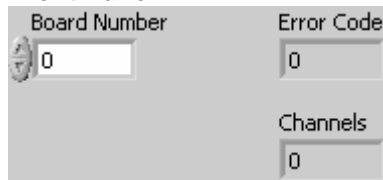
## CgLv\_GetChannelCount.vi

This VI retrieves number of channels of the specified CompuGen board.  
Board Number – number of the CompuGen board to be addressed, starting from 0.  
The VI returns an Error Code.

### Connector Pane



### Front Panel



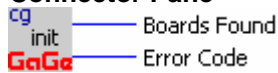
### Controls and Indicators

- Board Number
- Channels
- Error Code

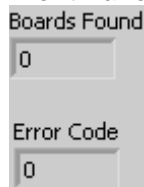
## CgLv\_Intialize.vi

This VI initializes the CompuGen driver and returns the number of CompuGen boards found. The VI returns an Error Code.

### Connector Pane



### Front Panel



### Controls and Indicators

- Boards Found
- Error Code

# CompuGen PCI LabVIEW SDK

## CgLv\_LoadSegBuffer.vi

This VI loads data for the specified segment.

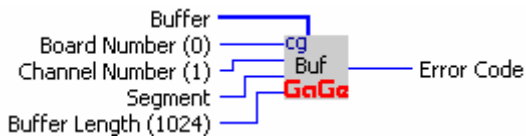
Board Number, Channel Number and Segment identify the target for upload.

Buffer - array of U16 variables containing the segment data. A value of 0 corresponds to positive full scale. A value of 4095 corresponds to negative full scale. A value of 2048 corresponds to 0 Volts.

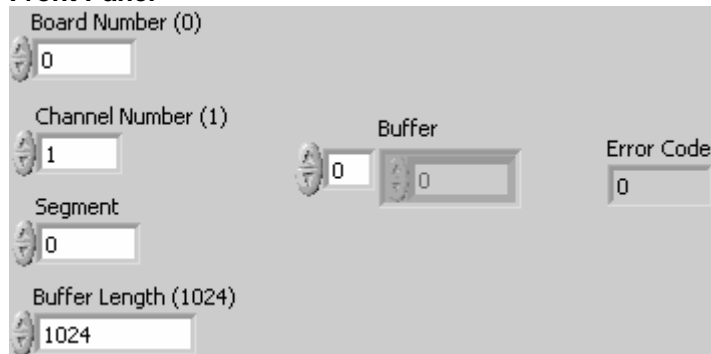
Buffer Length - number elements in the Buffer array. If Buffer Length is greater than segment Length, then only the first Length samples in the pattern will be generated. If Buffer Length is less than Segment length, then the last sample in the pattern buffer will be repeated to fill the rest of the segment Length.

The VI returns an Error Code.








### Connector Pane



### Front Panel



### Controls and Indicators

-  Channel Number (1)
-  Buffer Length (1024)
-  Board Number (0)
-  Buffer
-  Numeric
-  Segment
-  Error Code

# CompuGen PCI LabVIEW SDK

---

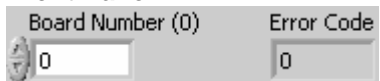
## CgLv\_ResetSegments.vi

This VI removes any previous segment data on the CompuGen hardware.  
This VI should be called prior to the first call to CgLv\_ConfigureSegment.vi.  
The VI returns an Error Code.

### Connector Pane



### Front Panel



### Controls and Indicators

- Board Number (0)
- Error Code

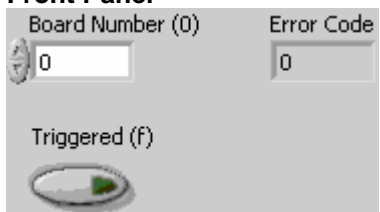
## CgLv\_Start.vi

This VI starts CompuGen signal generation.  
The Triggered input sets the CompuGen generation mode. For Triggered Mode, set to TRUE.  
For Free Run Mode, set to FALSE.  
Board Number – number of the CompuGen board to be addressed, starting from 0.  
The VI returns an Error Code.

### Connector Pane



### Front Panel



### Controls and Indicators

- Board Number (0)
- Triggered (f)
- Error Code

# CompuGen PCI LabVIEW SDK

---

## CgLv\_Stop.vi

This VI stops CompuGen signal generation.

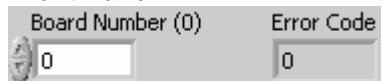
Board Number – number of the CompuGen board to be addressed, starting from 0.

The VI returns an Error Code.

### Connector Pane




### Front Panel



### Controls and Indicators

 Board Number (0)

 Error Code

## CgLv\_Trigger.vi

This VI forces generation of a single waveform in Triggered Mode.

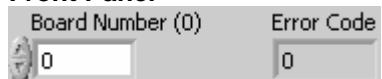
Board Number – number of the CompuGen board to be addressed, starting from 0.

The VI returns an Error Code.

### Connector Pane




### Front Panel



### Controls and Indicators

 Board Number (0)

 Error Code

# CompuGen PCI MATLAB SDK

---

## CompuGen PCI MATLAB SDK Overview

In order to operate the CompuGen PCI MATLAB SDK, the user requires Mathwork's MATLAB 6.5 or higher. The CompuGen PCI MATLAB SDK consists of a single MATLAB program called CgSimple.m. The MATLAB SDK folder includes CgSimple.m as well as all required supporting files.

CgSimple.m works by making function calls to a Dynamically Linked Library (DLL) called CgCall.dll. The structure of CgSimple.m is very straightforward, but may be easily extended to more complex operation by reading about and understanding the basic CgCall function calls and making appropriate modifications.

An overview of the operation of CgSimple.m is given below.

1. Initialize the CompuGen hardware and determine the number of CompuGen boards installed in the host PC.
2. Query the CompuGen hardware in order to determine its capabilities.
3. Set the CompuGen board parameters. These parameters include the conversion rate and the flag to activate External Clocking. The default conversion rate is set to 150000000.
4. Calculate the data points for a single cycle sine wave waveform. The values to be uploaded to the CompuGen board must take on the values from 0 to 4095. A 0 value corresponds to negative full scale output. A value of 4095 corresponds to positive full scale. A value of 2048 corresponds to 0 Volts.
5. Set the segment configuration. This entails setting the Generation Mode to Free Run and setting the segment length on the board. The segment length is the number of data points in the waveform segment that will be generated. This is distinct from the pattern buffer length, which may be different.
6. Calculate a new segment pattern.
7. Upload the sinusoidal cycle pattern created in step 4 to the first half of the channels on the first CompuGen board. In this step, the length of the pattern, `channel.buffer_length`, is passed to the CompuGen board. While `channel.buffer_length` is set equal to `segment.length` within CgSimple.c, this need not be the case.  
If the uploaded pattern is longer (`channel.buffer_length > segment.length`), then only the first `segment.length` points from the pattern are generated. If the uploaded pattern is shorter (`channel.buffer_length < segment.length`), then pattern is extended so that it is `segment.length` points long. This is done by repeating the last point in the uploaded pattern until it is padded up to `segment.length`.
8. Upload the pattern created in step 6 to the second half of the channels on the first CompuGen board.
9. Start generation on the CompuGen board in Free Run Mode.

Please note that in developing a MATLAB application from CgSimple.m, the user should maintain the above order in making CgCall() subroutine calls. Making the calls in an arbitrary order may result in faulty operation.

# CompuGen PCI MATLAB SDK

---

## CompuGen PCI MATLAB CgCall Functionality

CgSimple.m controls the CompuGen hardware by making function calls to an intermediate DLL written in C called CgCall.dll.

The function calling convention to CgCall.dll is as follows:

```
Return_value =  
CgCall(system_command, board_number, item, structure, signal_buffer)
```

The most important parameter is `system_command` which determines the type of operation that `CgCall()` will perform on the CompuGen hardware. This parameter may be set to 1, 2, 3, and 4. For convenience, `CgSimple.m` makes the following assignments:

```
INITIALIZE = 1;  
QUERY = 2;  
CONFIGURE = 3;  
DO = 4.
```

Remaining parameters for `CgCall()` behave differently depending on the value of `system_command`. It is not necessary to include all `CgCall()` parameters for every call. Some parameters are unnecessary, depending on the value of `system_command`.

The remainder of this manual describes the operation of `CgCall()` for the four different possible values of `system_command`.

In the event of an error, `CgCall()` returns an error code. Please refer to page 8 of the C/C++ SDK section for the meaning of each error code.

# CompuGen PCI MATLAB SDK

---

## CompuGen MATLAB CgCall Modes

### CgCall(INITIALIZE)

**Purpose:**

This CgCall() command initializes the CompuGen hardware and driver. It also identifies the number of CompuGen boards installed in the host PC.

**Additional parameters:**

None

**Returns:**

A positive return value indicates the number of CompuGen boards found. If an error occurs, then the return value is an error code.



# CompuGen PCI MATLAB SDK

---

## **CgCall(QUERY, board\_number, command\_item)**

### **Purpose:**

This CgCall() command queries the driver and returns a value that is the response to the query.

### **Additional parameters:**

board_number	Number of the CompuGen board to be addressed, starting from 0.
command_item	Can be assigned to QUERY_CHANNEL_COUNT, QUERY_RATE_COUNT, QUERY_ATTENUATION, QUERY_EXTERNAL_CLOCK, or QUERY_RATES, which are defined within CgSimple.m.
QUERY_CHANNEL_COUNT	Queries the CompuGen hardware and returns the number of available output channels.
QUERY_RATE_COUNT	Queries the CompuGen hardware and returns the number of different available conversion rates.
QUERY_ATTENUATION	Queries the CompuGen hardware and returns a 1 if a CompuGen output attenuator is available.
QUERY_EXTERNAL_CLOCK	Queries the CompuGen hardware and returns a 1 if external clocking functionality is available.
QUERY_RATES	Queries the CompuGen hardware in order to determine the available internal conversion rates. If the call is successful, the return value is an array of doubles containing the available conversion rates. The size of the array can be determined by a QUERY_RATE_COUNT query.

### **Returns:**

The return value depends on the query type, as described above. If an error occurs, then the return value is an error code.

# CompuGen PCI MATLAB SDK

---

## **CgCall(CONFIGURE, board\_number, command\_item, structure, signal\_buffer)**

### **Purpose:**

This CgCall() command configures the CompuGen hardware. The command can be used to configure the pattern segment, the channel configuration, or the board configuration.

### **Additional parameters:**

board_number	Number of the CompuGen board to be addressed, starting from 0.
command_item	Can be assigned to CONFIGURE_SEGMENT, CONFIGURE_CHANNEL, or CONFIGURE_BOARD, which are defined within CgSimple.m.
CONFIGURE_SEGMENT	Assigning the command_item to CONFIGURE_SEGMENT sets the segment configuration. The segment configuration parameters are contained in a structure called segment, which must be passed as the structure parameter of CgCall(). The elements of the structure called segment are: <ul style="list-style-type: none"><li>segment.trigger This flag sets the CompuGen conversion mode. For Triggered Mode, set to 1. For Free Run Mode, set to 0.</li><li>segment.loop_count Currently only 0 is supported. This parameter has no effect.</li><li>segment.length Length in points of the segment to be generated by the CompuGen board. It must be a multiple of 16 with a 64 point minimum.</li><li>signal_buffer This is not used for CONFIGURE_SEGMENT.</li></ul>

# CompuGen PCI MATLAB SDK

---

**CONFIGURE\_CHANNEL** Assigning the `command_item` to **CONFIGURE\_CHANNEL** configures the channel settings. The most important channel setting is the pattern to be uploaded. The channel configuration parameters are contained in a structure called `channel`, which must be passed as the structure parameter of `CgCall()`. The elements of the structure called `channel` are:

<code>channel.channel_number</code>	Number of the channel to be configured. Use values 1 to n to configure a specific channel. A value of 0 will configure all channels identically.
<code>channel.attenuation</code>	Output attenuation factor for the CG4300, in dB. Ignored for other CompuGen models.
<code>channel.buffer_length</code>	Size of the buffer that contains the waveform pattern to be uploaded to the CompuGen memory.
<code>signal_buffer</code>	The buffer that contains the waveform pattern to be uploaded to the CompuGen memory. <code>signal_buffer</code> must be an array of variables with <code>channel.buffer_length</code> elements. A variable value of 4095 corresponds to negative full scale. A value of 2048 corresponds to 0 Volts.

**CONFIGURE\_BOARD** Assigning the `command_item` to **CONFIGURE\_BOARD** sets the board configuration. The board configuration parameters are contained in a structure called `board`, which must be passed as the structure parameter of `CgCall()`. The elements of the structure called `board` are:

<code>board.conversion_rate</code>	Sets the conversion rate in Hz.
<code>board.external_clock</code>	Flag that activates external clocking when set to 1, set to 0 for internal clocking.
<code>board.external_trigger</code>	This flag sets the CompuGen conversion mode. For Triggered Mode, set to 1. For Free Run Mode, set to 0.
<code>signal_buffer</code>	This is not used for <b>CONFIGURE_BOARD</b> .

## Returns:

If an error occurs, then the return value is an error code.

# CompuGen PCI MATLAB SDK

---

## **CgCall(DO, board\_number, command\_item)**

### **Purpose:**

This CgCall() command performs an action on the selected CompuGen board.

### **Additional parameters:**

board_number	Number of the CompuGen board to be addressed, starting from 0.
command_item	Can be assigned to ACTION_START_TRIGGERED, ACTION_START_FREE, ACTION_TRIGGER, or ACTION_STOP, which are defined within CgSimple.m.
ACTION_START_TRIGGERED	Start signal generation in Triggered Mode.
ACTION_START_FREE	Start signal generation in Free Run Mode.
ACTION_TRIGGER	Force generation of a single waveform in Triggered Mode.
ACTION_STOP	Stop signal generation.

### **Returns:**

If an error occurs, then the return value is an error code.

# Advanced CompuGen PCI SDK Functionality

---

## Advanced CompuGen PCI SDK Functionality

CompuGen PCI Software Development Kits include advanced sample programs that are for use by customers who are already comfortable with normal CompuGen operation

This section describes the advanced sample programs in a generic fashion that is not specific to any of the three CompuGen SDKs.

## Link'n'Loop

### Introduction

All PCI CompuGen cards support *CompuGen Link'N'Loop Mode*, which allows multiple pattern segments to be uploaded to the CompuGen card for later selective generation. Link'N'Loop Mode allows CompuGen memory limitations to be overcome for a wide variety of signals. For instance, a user may generate a succession of different cyclical waveforms simply by uploading the pattern for a single cycle of each waveform and then seamlessly looping each single cycle in order to obtain the required number of cycles. As an example, a Link'N'Loop generation sequence might generate 100 cycles of a 1 MHz sine wave, followed by 2000 cycles of a 2.5 MHz triangle wave, followed by 200 cycles of a 500 kHz square wave. Alternatively, a user who wants to generate short distinct waveforms that are separated by a long duration where the signal is at 0 Volts need not waste CompuGen memory by filling it up with 0 Volt values. Instead, only the short distinct waveform are uploaded and, in Link'N'Loop mode, each waveform is generated upon receipt of a software or external trigger. Using Link'N'Loop mode, a user may even initially upload a portfolio of waveforms and then may select to generate them in different orders simply by uploading new segment order lists.

Link'N'Loop Mode is somewhat more complex than normal CompuGen mode and so should be used only by users who are already comfortable with Normal CompuGen operation. Link'N'Loop Mode is accessible only using the CompuGen Software Development Kits.

A big advantage of Link'N'Loop Mode is that if the user needs to rapidly switch between different output waveforms, then different waveform data need not be uploaded in order to switch the output waveform. Instead, waveform segments may be preloaded onto the CompuGen hardware and then the user may rapidly switch between different segments during generation. For instance, for a CompuGen 11G, which has 4 MegaSamples of pattern memory per channel, over 4000 waveform segments of 1024 points each may be pre-loaded into the CompuGen memory for later generation. Different triggering and looping conditions may be pre-selected for each Link'N'Loop segment. Segment configurations are pre-loaded to the CompuGen hardware so that, during generation, the user may

# Advanced CompuGen PCI SDK Functionality

---

switch between different waveform segments very quickly with no software interaction required.

From any of the three CompuGen SDKs, Link'N'Loop mode is implemented as a simple extension of Normal CompuGen operation mode. Each SDK includes a sample program called *CgLnL*, which illustrates use of Link'N'Loop mode. *CgLnL* calculates the distinct waveform data for a few segments and assigns different Link'N'Loop settings for each segment. These segments and settings are then uploaded to the CompuGen hardware and generated in the prescribed fashion.

Link'N'Loop mode is controlled using the same sub-routine set that is used for the control of Normal CompuGen operating mode. The only difference is that new Link'N'Loop mode variables and constants must be used for Link'N'Loop control. In what follows, a description of the software elements required for Link'N'Loop control is given. The description is given only for the C programming language. From MATLAB or LabVIEW, Link'N'Loop is controlled in the same fashion but following the syntax required for these environments. The user may refer to the Link'N'Loop sample programs (*CgLnL.M* and *CgLnL.VI*) as guides on correct syntax.

## The seg Structure

Settings for Link'N'Loop operation are contained within the *seg* structure in *CgLnL.c*. Link'N'Loop mode is actually enabled simply by using a loop to upload multiple copies of the *seg* structure to the CompuGen drivers, rather than only one. The *seg* structure contains a variable (*seg.dwSegNumber*) that specifies to which segment the entries refer. The variables contained in the *seg* structure and their operation in Link'N'Loop mode are described below.

*seg.bTriggered* - As in Normal operating mode, this Boolean variable determines how the CompuGen will initiate segment generation. If *seg[i].bTriggered* is set to FALSE, then Free Run generation mode is selected. In this mode, the segment loops seamlessly and successive segment generations occur immediately after the previous generation. This mode is useful for generating sine waves, for instance, where the segment data for only a single cycle of the sine wave are uploaded and the segment is looped seamlessly to generate multiple sine wave cycles. If *seg[i].bTriggered* is set to TRUE, then Triggered Mode generation mode is selected. In this mode, the CompuGen awaits the occurrence of an external or software trigger event before generating its segment. *bTriggered* may be set independently for each segment in the Link'N'Loop sequence.

*seg.dwLoopCount* - In Normal Mode, segment looping in Free Run Mode is endless while segments are only generated once in Triggered Mode. In Link'N'Loop mode, however, each segment may be looped a selectable finite number of times in both Free Run and Triggered modes. The value of the *dwLoopCount* variable sets the number of times that the pattern will be looped. The maximum value is 16384.

# Advanced CompuGen PCI SDK Functionality

---

*seg.dwLength* – This variable must be set equal to the number of points in the segment data pattern that is to be uploaded to the CompuGen hardware. The variable value must conform to the specified minimum length and length increment for the CompuGen model in use.

*seg.dwSegNumber* – This variable must be equal to the index of the current segment, starting from 0. For instance, for the first segment the value must be 0, for the second segment the value must be 1, etc.

*seg.dwNextSegment* – In Link’N’Loop mode, all but the first of the uploaded segments may be generated in any arbitrary order. The segments may be generated, for instance, in the order #1, #4, #5, #3, #2. By uploading only a new *segment* structure, the segment order may be changed without uploading new segment pattern data. The *dwNextSegment* variable selects which segment will be generated after the current one.

*seg.dwMarkerAddress* – This variable specifies the marker pulse position, as it does in Normal mode. In Link’N’Loop mode, a marker may be independently specified for each Link’N’Loop segment.

## Link’N’Loop Operation

Once the values have been set within the *seg* structure, these values are set on the CompuGen hardware using *CgSet()* with the *CG\_SEGMENT\_CONF* modifier, as in Normal generation mode. Board configuration is done using *CgSet()* with the *CG\_BOARD\_CONF* modifier, as in Normal generation mode.

Before loading new segment data, the user should normally first clear previous segment data on the CompuGen hardware using *CgSet()* with the *RESET\_SEGMENT* modifier. The user may opt not to make this call, however, and only change *seg* structure variables and upload them to CompuGen hardware. In this way, using the same segment data, changes may be made only to the Link’N’Loop settings without having to re-upload the segment data. In this case, changes to *seg* structure variables must be consistent with the already loaded segment data. For instance, while the *dwLoopCount* and *bTriggered* values may be arbitrarily changed, *dwLength* must remain equal to the already uploaded segment pattern length.

Data for each Link’N’Loop segment are uploaded to the CompuGen hardware using *CgSet()* with the *CG\_CHANNEL\_CONF* modifier, as in Normal generation mode. Segments are uploaded separately and in order. New uploaded segment data is automatically appended as the next segment in the list. As discussed, *CgSet()* with the *RESET\_SEGMENT* modifier is required to clear all segments and restart the list.

Once the CompuGen hardware has been configured and the segment data have been uploaded, the generation of the Link’N’Loop sequence may be initiated. In

# Advanced CompuGen PCI SDK Functionality

---

order to understand the details of Link'N'Loop generation, further explanation is necessary.

First of all, the overall Link'N'Loop sequence is always looped so that it repeats forever. For instance, if the user uploads three Link'N'Loop segments, the generation sequence will be:

Segment #1 sequence, segment #2 sequence, segment #3 sequence, segment #1 sequence, segment #2 sequence, segment #3 sequence, segment #1 sequence  
.....

By “segment #N sequence”, we mean the generation of segment #N for the pre-set number of loops and using the triggering conditions specified in the *seg* structure. By “over-all Link'N'Loop sequence”, we mean a complete single generation of all segment sequences.

A user may wish to generate the overall Link'N'Loop sequence only once, however. This may be done by simply adding a bogus final segment containing all 0 Volt values. Next, set the *seg.dwNextSegment* value for the final segment so that it points to itself. This way, at the end of the first generation of the overall Link'N'Loop sequence, the CompuGen hardware will become stuck in a loop generating a 0 Volt pattern forever. The user may then abort the CompuGen generation by calling *CsDo()* with the *CG\_DO\_STOP* modifier.

Since Link'N'Loop settings are all pre-loaded onto the CompuGen hardware before generation, software interaction is not required for the hardware to switch between sequential segments. Consequently, switching between segments is extremely fast but is not instantaneous. The CompuGen boards require a 64 point delay in order to switch between segments. During this delay time, the output voltage is held at a value that corresponds to the last value of the previous segment. Since no software interaction is required, the 64 point delay is always fixed and does not depend on the operating system task load in any way. The user may, therefore, compensate for this fixed delay if phase coherence from one segment to the next is required.

The CompuGen drivers logically order the *dwLoopCount* and *bTriggered* variables so that they are associated with the current segment. Internally, however, on the CompuGen hardware, these variables and the *dwNextSegment* value are held in memory associated with the previously generated segment. As a result, when the Link'N'Loop sequence begins, the first segment cannot be associated with any *dwLoopCount* and *bTriggered* variables, since no previous segment has yet been generated. Consequently, for the first generation of the first segment in a Link'N'Loop, The *dwLoopCount* value is always 1 and the *bTriggered* value is always FALSE. On the second time through the Link'N'Loop sequence, the first segment gets its *dwLoopCount* and *bTriggered* values from the last segment (which was the previous segment generated) so that the specified *dwLoopCount* and *bTriggered* values for the first segment are used. The irregularity of generation of first segment is no real limitation since it may be rendered irrelevant. For instance, if the user



# Advanced CompuGen PCI SDK Functionality

---

requires the first segment to loop a finite number of times, then they may create a short bogus first segment that contains all 0 V output values and then load the real first segment as the second segment. This way, when Link'N'Loop generation begins, a short, innocuous 0V generation occurs before the real first segment is looped as required.

Once Link'N'Loop configuration and segment data have been loaded to the CompuGen hardware, generation is initiated in the same way as Normal Generation. In order to begin generation immediately, simply call `CsDo()` with the `CG_DO_START_FREE` modifier. If `CsDo()` is called with the `CG_DO_START_TRIGGERED` modifier, then the CompuGen hardware will await an external trigger pulse or software trigger before generating.

## Digital Output Marker Control

Each model of PCI CompuGen card comes equipped with a Digital Marker Output, which produces signals with a TRUE level of 1.3 V and a FALSE level of 0 Volts, when the output is terminated by a 50 Ohm load. The main usage of the Marker Output signal is for creating a digital pulse signal that may be used as an external trigger source for other devices, such as a CompuScope digitizer card. Software control allows placement of a Marker pulse that is N samples wide and occurs during pattern generation. The marker position may be specified with a resolution of N samples anywhere within the CompuGen pattern. For the CompuGen 815x and 4300, N=4 and for the CompuGen 11G, N=16. The Marker position value specifies the position of the falling edge of the Marker Pulse. For instance, consider a pattern of length 8192 points. A marker position of 0 will cause the Marker Output to generate a digital pulse at the beginning of the pattern. Similarly, a Marker position of 4096 will create a Marker output pulse in the middle of the pattern, while a marker position of 8180 will create a marker pulse near the end of the pattern.

From `CgTest`, the position in time of the digital marker position is adjusted using the Marker control, which selects the sample number on which the falling edge of the marker pulse will occur. From the CompuGen C Software Development Kit, the marker position is selected by assigning a value to the variable `dwmarker`, within the `seg` structure (i.e. `seg.dwmarker`). The name of the marker variable is slightly different in the MATLAB and LabVIEW SDKs and its usage is illustrated in the `CgSimple.M` and `CgSimple.VI` sample programs. As in `CgTest`, the value of the marker variable in `CgSimple` determines the sample number on which the falling edge of the marker pulse occurs. Each SDK sample program contains code that illustrates the assignment of the marker pulse position. Disabling of the output marker is done by simply assigning a value to the marker position that this outside limits of the current segment. For instance, if the pattern segment is 4096 points long, then a marker position of 16384 will disable the marker output.