



Application Note:

Arbitrary Waveform Generator on SDR14TX

Table of Contents

1	Introduction	2
2	Specification	2
3	General Description	2
4	Application	2
4.1	Segments	2
4.2	Laps	3
4.3	AWG operational modes	4
4.4	Enable AWG trigger	6
4.5	AWG trigger output	6
4.6	Uploading data	6
4.7	Playlist mode	6
4.8	Special features	9
4.9	Embed GPIO in data	7
4.10	Operation in the second Nyquist band	10
5	API commands	10
6	Supported revisions	10
7	Example code	10



1 INTRODUCTION

The SDR14TX signal generator, produced by Teledyne SP Devices, has two output channels capable of 14-bit output at 2000 MSPS each. Each output channel is fitted with an Arbitrary Waveform Generator (AWG) controller, which this document will provide usage instructions for. An AWG is a signal generator which can produce user defined sequences of output data.

2 SPECIFICATION

- **DAC data rate** 2000 MSps
- **Available memory** 0.5 GB / channel
- **Maximum waveform length** 256 Msamples / channel
- **Minimum waveform length** 1024 samples (512 ns)
- **Segment upload transfer** maximum 160 Mbyte/sec (depends on host connection)
- **Segments** 1 to 16384 segments
- **Number of laps** 1 to $2^{31}-1$ or infinite
- **Maximum playlist elements** 1024 playlist elements

3 GENERAL DESCRIPTION

The AWG data is stored in the DRAM memory of the digitizer. Waveform data can easily be uploaded to the DRAM using API commands. The AWG data can be subdivided into between 1 and 16384 data segments, where each segment is of a user defined length.

During AWG operation, the segments are read out to the DACs sequentially. The user can set how many laps each segment should be run before the AWG moves on to the next segment. This reduces the amount of data that is needed to run longer repetitive data patterns.

There are several modes of operation for the AWG with regards to triggering, which are detailed below.

4 APPLICATION

4.1 Segments

The AWG has the possibility to make use of up to 16384 different waveform segments. The segments are stored in the onboard DRAM memory on SDR14TX. Each segment may be set to any length, independently of other segments, with the only limitation being that the segment length must be set as a multiple of 16 samples, and that there is a minimum segment length of 1024 samples. This translates to a segment granularity of 8 ns and a minimum segment length of 512 ns. The maximum segment length is limited by the total available memory per channel, as seen in the specification

above.

Note: If there exists a need for shorter segment combinations than what can be achieved with 1024 samples, one path would be to un-wrap all combinations to be used into a number of different 1024 sample segments and use those accordingly.

The segment DRAM data may also be reprogrammed during operation of the AWG. As an example, this could be used to switch between two segments and replace the data of one segment while the other is looping. This may be used to remove the limit of 16384 unique segments set by the AWG logic, since dynamic segment reprogramming during operation allows an infinite number of different segments. Care has to be taken to avoid reprogramming the data sections currently being read out, in order to avoid glitches in the output.

4.2 Laps

The user defines the number of laps each segment should be run. The segments can be programmed individually to lap values between 1 and $2^{31}-1$. If the number of laps for one segment is set to 2^{31} or higher, it will run infinitely. This can be combined in order to run a series of segments first, and then loop the last segment infinitely, see Figure 1. The loop can only be broken via the trigger mode “seamless trigger” (see below), or by disarming the AWG.

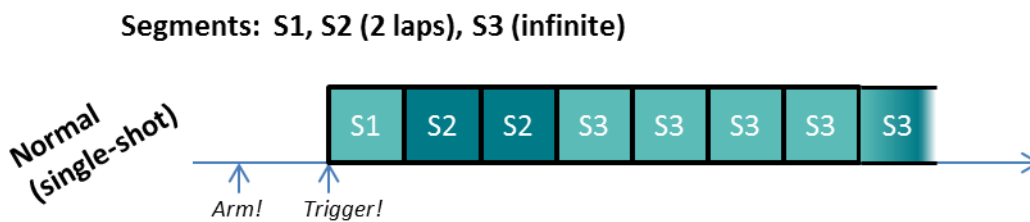


Figure 1. Infinite laps mode, with segment S3 having a lap value of 2^{31} or higher, thereby enabling infinite laps.



4.3 AWG operational modes

There are several different modes of operation for the AWG. The AWG is always started by a trigger. The operational modes react differently on the triggers. The modes are illustrated in Figure 2 and summarized here:

➤ **Normal mode**

If no special mode options are selected, the AWG will wait for a trigger event before outputting data, and will require a manual rearm before every trigger.

➤ **Triggered-lap mode**

In this mode, every single lap for every segment will require a trigger event before starting. This is useful if an output consisting of a large number of different pulses is desired, with long spaces of silence in between. Instead of programming the empty spaces as actual waveform data (which is not very memory-efficient), this can be done by letting each pulse be programmed as an AWG segment and then spacing out the trigger signals.

➤ **Continuous mode**

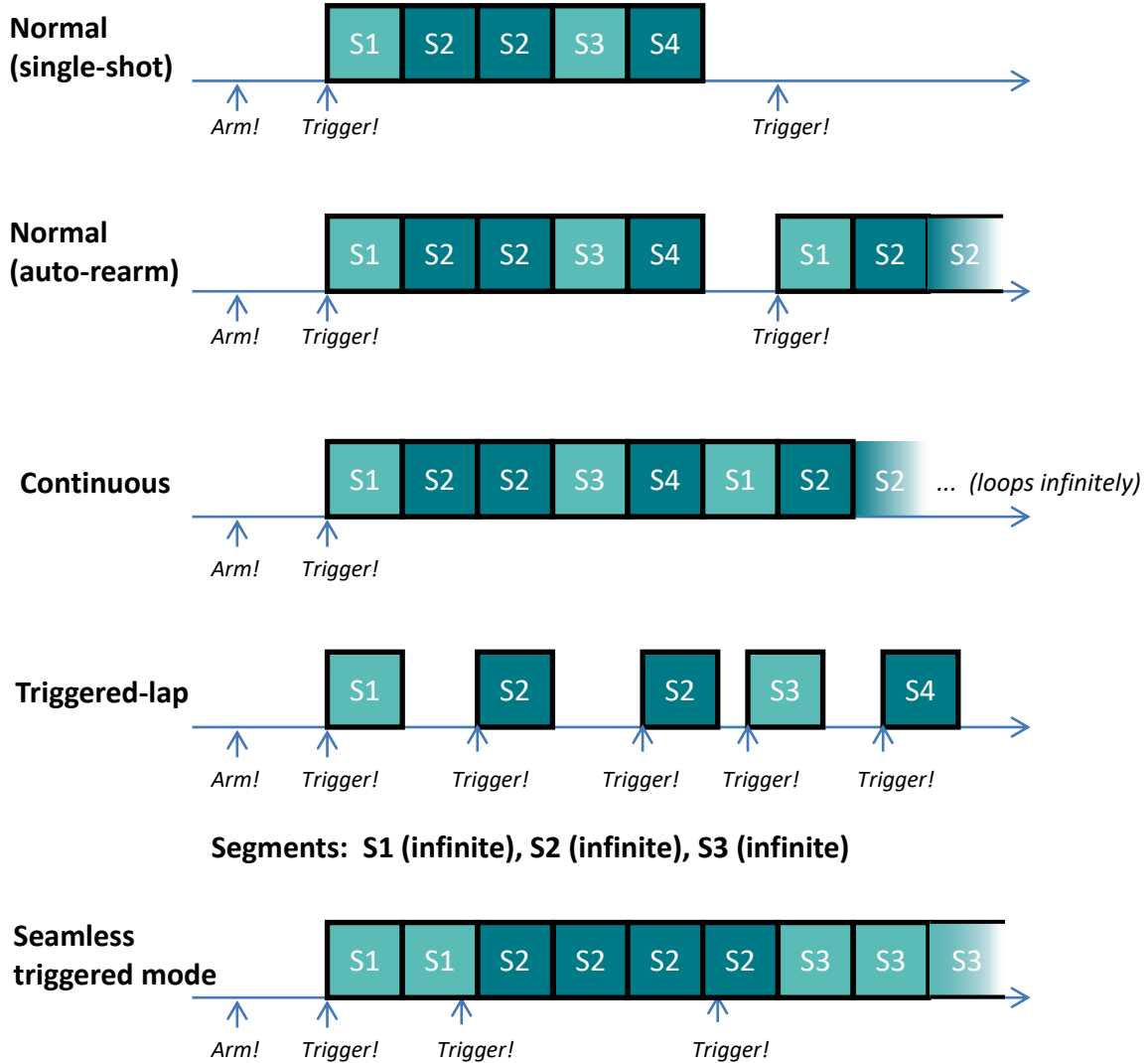
This mode starts the AWG on first accepted trigger after rearming, without waiting for any more triggers, and will then run the defined segment data until its end. If an autorearm is used instead of a normal arm function call, the AWG will start looping data infinitely without any pauses. For instance, to start immediately, the user can configure for a software trigger (*AWGSetTriggerEnable*) and issue it directly after arming (*AWGTrig*).

➤ **Seamless triggered mode**

Upon receiving a trigger signal in the seamless triggered mode, the currently running segment will skip any remaining laps (regardless of how many), and seamlessly cross to the next segment. By using the seamless mode in conjunction with segments programmed with infinite laps, the AWG will loop indefinitely on each segment and will switch over to the next segment seamlessly upon receiving a trigger. Note that when synchronizing several SDR14TX in a system with the seamless trigger mode, the minimum segment length has to be increased to at least 16384 samples to guarantee reliable synchronization.



Segments: S1, S2 (2 laps), S3, S4



Segments: S1 (infinite), S2 (infinite), S3 (infinite)

Figure 2. Segment sequences for various operational modes, showing some of the things that can be performed using the AWG.



4.4 AWG trigger sources

Several trigger sources such as external trigger, backplane trigger, internal trigger or software trigger are available for use with the AWG. The command *AWGSetTriggerEnable* controls the configuration of the AWG triggering.

4.5 AWG trigger output

The AWG can generate a trigger output to the PXIe backplane or on the front panel trigger connector, synchronous to the waveform generation. This trigger output has several configuration parameters, including:

- Output a trigger signal at the start, or the end of a segment.
- Re-arm automatically after triggering, or require manual rearm.
- Variable trigger output pulse length.

Use the command *AWGSetupTrigout*, *AWGTrigoutArm*, *AWGTrigoutDisarm* to control the trigger output. This trigger signal may be routed to the PXIe backplane trigger outputs DSTARC, PXI TRIG 0, or PXI TRIG 1 by the *EnablePXIeTrigout* command.

4.6 Uploading data

Data upload is done in two stages; segment memory allocation and segment writing. A segment memory allocation allocates a slice of memory space for a specific segment. This is done using the API function *AWGSegmentMalloc*. The segment space can then be set to any length from the minimum of 16 samples up to the maximum as determined by the available memory (see specification above). Once the segment memory is allocated, the API function *AWGWriteSegment* can be used to upload the actual segment waveform. The waveform data length can be anywhere between the minimum of 16 samples and up to the entire allocated memory size. During operation, the AWG will only read out as much data as was programmed using this function, not the entire allocated memory. There is also a faster command called *AWGWriteSegments* where you upload all segments at once.

If the AWG has been allocated and programmed with a set of segments, and a segment is then reallocated with a larger memory space, a write to this segment will overwrite the data of the following segment. In order to avoid errors, a reallocation should therefore be followed by a reprogramming all the segments. A good idea is to allocate as much memory as possible for each segment, to avoid having to reallocate during operation just because a single segment needs to be written to with a particularly long set of data.

4.7 Playlist mode

There is also an advanced segment playlist mode, which can be enabled by setting *AWGPlaylistMode* to 1. The number of laps setting associated with each segment is ignored in playlist mode, as each playlist in itself sets the number of laps.

Each playlist element defines which segment ID to play, which playlist index to play next (jump to), how many segment laps to play and also a trigger output generation definition. It is possible to generate a trigger which is played once per segment lap or just once for the first lap, with



configurable time offset, polarity and length. It is also possible to define how the playlist element generates switching signals to the user logic inside the FPGA (only valid for FPGA DevKit users).

The playlist is uploaded by the API command *AWGWritePlaylist* or *AWGWritePlaylistItem*. It is possible to change the playlist during playback, affecting the sequence played. Any changes to a currently playing segment will be disregarded. When you upload or change elements you may choose to only update the segment ID, number of laps, next to play info or trigger information by using individual masking. Note that you must enable all segment IDs up to the maximum playlist element index used, otherwise if the playlist reaches a disabled segment, it will stop playing.

Example:

Playlist			
Index	1	2	3
Segment	11	34	24
Next index	2	3	1
Number of laps	2	5	4
User logic signals	0	1	0
Trigger type	2	1	3
Trigger length	200	0	200
Trigger sample	16	0	0
Trigger polarity	0	0	0

When AWG is triggered, segment 11 will be played twice – issuing a single trigger with a time offset of 16 samples and a length of 200 ns. The next index in the playlist is 2. Segment 34 is played five times, issuing no triggers, but issuing a user logic signal for both segment and lap switch. Finally, a jump to index 3 will be made. Segment 24 is played 4 times, issuing a trigger only on the first lap at a sample offset of 0 for 200 ns. After that, the sequence will jump back to index 1 and go for segment 11 again.

4.8 Embed GPIO in data

It is possible to embed GPIO output waveforms in the data. This is performed by using the 2 extra MSB bits of each sample in the 8 sample-parallel stream. In total there are 8×16 bits = 128 bits in each such 8-sample word. These are handled with the 200MHz clock, meaning that GPIO output will be aligned to this 200MHz clock.



Bits	Description
0-13	Sample 0
14-15	Command (see special features)
16-29	Sample 1
30	New value for GPIO bit 0 (see also mask below)
31	New value for GPIO bit 1 (see also mask below)
32-45	Sample 2
46	New value for GPIO bit 2 (see also mask below)
47	New value for GPIO bit 3 (see also mask below)
48-61	Sample 3
62	New value for GPIO bit 4 (see also mask below)
63	Don't care
64-77	Sample 4
78	Mask for GPIO bit 0 (if 1 update GPIO, if 0 do not change)
79	Mask for GPIO bit 1 (if 1 update GPIO, if 0 do not change)
80-93	Sample 5
94	Mask for GPIO bit 2 (if 1 update GPIO, if 0 do not change)
95	Mask for GPIO bit 3 (if 1 update GPIO, if 0 do not change)
96-109	Sample 6
110	Mask for GPIO bit 4 (if 1 update GPIO, if 0 do not change)
111	Don't care
112-125	Sample 7
126	Don't care
127	Don't care

The API command *SetDirectionGPIO* must be used to select the embedded GPIO from the desired channel as output on the GPIO ports.



4.9 Special features

The DAC data is 14 bits per sample, but the AWG data is stored in DRAM as 16 bit samples. It is possible to use the two extra bits to encode special features into the data stream. At the moment two different features are available; zero encoding (by command) and trigger encoding in data. The 2 MSB bits are used to issue specific commands.

Data (14 bits LSB aligned) (D = Data[13:0])	MSB Code (Data[15:14])	Instruction
Data D	0 0	Send data D to DAC (default)
Encode trigger	0 1	Send data D to DAC. Set trigout_o to high if TRIGOUTMODE is also set to TRIGOUTMODE_DATATRIG
Command	1 0	A command to the state-machine that controls the data flow to the DACs.
Reserved	1 1	Reserved

Since eight samples are processed in parallel in the FPGA, a command will affect the following eight samples. The command will use all the bits in the 16 bytes where it is placed. Thereby, commands may only be issued on 8-sample increments (sample $N=0$, sample $N=8$, etc). When a command is detected on sample N , samples $N+1$ to $N+7$ are viewed as parameters for this command. When the data field of sample N is a command (Sample N MSB = 1 0) there is a command detected.

Command name: Zero Output

Command indicator: Sample 1 = 16'd0

Command parameter: Sample 3-4 = 32 bits = number of 8-sample cycles to output zero only

The number of samples with zero output can be set in steps of 8. The minimum amount of sample cycles to zero for each command is 3, corresponding to 24 samples. (Value of sample 3-4, 32 bits must be at least 3) otherwise corrupt behavior may occur.

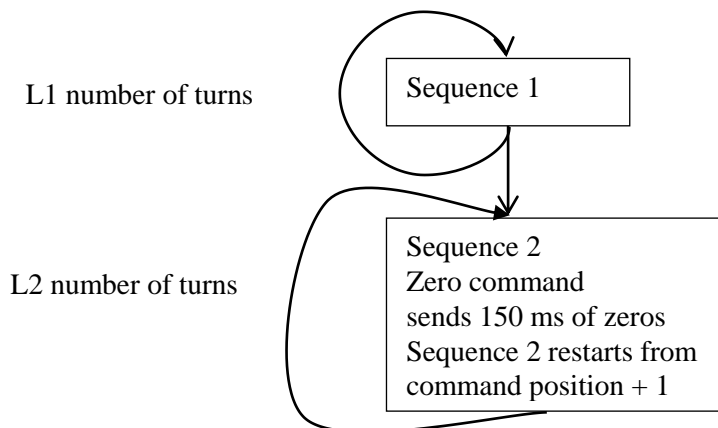


FIGURE 3: Sequence example with zero encoding.



Limitations

- A command can only be placed on every 8th sample.
- A command word will consume 8 samples (16 bytes).
- A zero output command must output at least 24 samples of zeros. If you have a lower amount of zeros, place as ordinary data in the stream.

4.10 Operation in the second Nyquist band

The SDR14TX hardware maintains a good frequency characteristic and distortion performance throughout the second Nyquist band (between 1 GHz and 2 GHz). For optimal performance, the API command *SetDACNyquistBand* should be issued to manually select operation in the second Nyquist band.

The AWG output will be aliased across 1 GHz and can then be filtered out using a bandpass filter. To achieve, for example, a sinusoid output at 1500 MHz, a segment of 500 MHz sinusoid data should be uploaded to the unit.

5 API COMMANDS

This application note does not intend to contain all available API commands. The ADQAPI user's guide, which contains descriptions of all commands used for controlling the AWG, can be found in the ADQ SDK at:

"SP Devices\Documentation\ADQAPI Reference Guide"

6 SUPPORTED REVISIONS

Required firmware revision: 35411 or higher
Required API revision: 35411 or higher

7 EXAMPLE CODE

The ADQ SDK installer includes several examples of AWG usage.

Python

SP Devices\Python_examples\ADQAPI_python.zip\SDR14TX_toneoutput.py

This example sets up and triggers output of a single segment containing coherent single-tone or two-tone data in the AWG.

