Best memory usage for real-world signals

Understanding Sequence Run and Sequence Advance Modes

Application Note

Agilent Technologies

Table of Contents

Sequences and Run Mode Options

Electronic devices and system become more and more complex and force more complex measurements and so long and complex waveforms, which require in case of an AWG long play times. One approach is to increase the amount of memory. The Agilent 81180A is available with 16 or 64 Msamples memory. The other possibility is to make best usage of the memory with a sequencer. Many waveforms are repetitive depending of certain events.

This application note focuses on sequence parameters and how they can be used in different run mode options to generate and refine the output of the 81180A. Chapter 3 of the *81180A Arbitrary Waveform Generator (AWG)* manual describes the sequence function and front panel programming aspects of various sequence parameters. What is also important is being able to create a sequence that matches the requirements of real life scenarios.

This is normally difficult with simple sequence routines however, the 81180A AWG has capability that, if fully understood, can be used to duplicate and simulate extremely complex scenarios. Examples of such waveforms include communication signals from a ground to air station, battle field stealth techniques, complex I&Q modulation signals, and multi-tone phase-shifted communication signal.

Table 1 shows how advance modes affect the sequence in various run mode options. Just as a brief reminder, sequences are created in sequence tables. To create a sequence one must do the following:

1. Define waveform segments.
2. Download waveforms to the waveform segments.
3. Write a sequence table that defines steps, segments, repeats, and jump bits. (Figure 1 shows an example of a sequence table created on an 81180A front panel display.)
4. Select an advance mode. Advance mode defines how the waveforms in the sequence table step through the sequence table entries. There are three advance mode options: auto, once, and stepped.
5. Select run mode. Run mode defines if the sequence will be generated continuously without intervention from the outside world, or if the sequence requires an enable, trigger, or gating event to start a sequence.

If the above steps are properly defined, your sequence will generate at the output connector. Note that downloading waveforms and creating the sequence table are prerequisites to generating any type of sequence. Advance and run modes can be modified later with no effect on the sequence table. Also note that the 81180A can store up to 1,000 different sequence tables which can be recalled one at a time to play different scenarios, for different applications, at the output connector.

Table 1: Sequence advance in various run mode options

Run mode	Waveform	Advance mode	Arm options	Idle waveform	Enable signal	Abort signal	Initiate signal	Wave loops	Seq loops	Jump flag	Jump signal	Smart trig
Continuous	Sequenced	Auto	Self-armed	Sequence	–	–	–(*)	1-1M	–	Bit (0|1)	Event	–
			Armed	Wave	BUS | Event	BUS	–	1-1M	–	Bit (0|1)	Event	–
		Once	Armed	Wave	BUS | Event	BUS	–	1-1M	1-1M	Bit (0|1)	–	–
		Stepped	Armed	Wave	BUS | Event	BUS	–	–	–	–	Event	–
	Adv'd Seq.	Auto	Self-armed	Adv'd seq		–	–	1-1M	–	Bit (0|1)	Event	–
			Armed	Sequence	BUS | Event	BUS	–	1-1M	–	Bit (0|1)	Event	–
		Once	Armed	Sequence	BUS | Event	BUS	–	1-1M	1-1M	Bit (0|1)	–	–
		Stepped	Armed	Sequence	BUS | Event	BUS	–	–	–	–	Event	–
Triggered	Sequenced	Auto	Self-armed	–	–	–	–	–	–	–	–	–
			Armed	DC	–	BUS	F.P.|BUS|Trigger	1-1M	–	Bit (0|1)	Event	Yes
		Once	Armed	DC	–	BUS	F.P.|BUS|Trigger	1-1M	1-1M	Bit (0|1)	Event	–
		Stepped	Armed	DC	–	BUS	F.P.|BUS|Trigger	–	–	–	Event	Yes
	Adv'd Seq.	Auto	Self-armed	–	–	–	–	–	–	–	–	–
			Armed	DC	–	BUS	F.P.|BUS|Trigger	1-1M	–	Bit (0|1)	Event	Yes
		Once	Armed	DC	–	BUS	F.P.|BUS|Trigger	1-1M	1-1M	Bit (0|1)	Event	–
		Stepped	Armed	DC	–	BUS	F.P.|BUS|Trigger	–	–	–	Event	Yes
Gated	Sequenced	Auto	Self-armed	–	–	–	–	–	–	–	–	–
			Armed	DC	–	BUS	F.P.|Trigger	1-1M	–	Bit (0|1)	Event	–
		Once	Armed	–	–	–	–	–	–	–	–	–
		Stepped	Armed	–	–	–	–	–	–	–	–	–
	Adv'd Seq.	Auto	Self-armed	–	–	–	–	–	–	–	–	–
			Armed	DC	–	BUS	F.P.|Trigger	1-1M	1-1M	Bit (0|1)	Event	–
		Once	–	–	–	–	–	–	–	–	–	–
		Stepped	–	–	–	–	–	–	–	–	Event	–

1. *Note: "–" denotes not relevant*

Table's glossary of terms

Run mode – Defines basic instrument run mode options

Continuous – Waveform is generated continuously

Triggered – Output waits for trigger; waveform is generated once

Gated – Output waits for gating signal; waveform is generated as long as the gating signal is true

Waveform – Sequenced or advanced sequencing waveforms; waveform coordinates must be downloaded to the instrument and sequence tables updated to generate these waveforms

Advance mode – Basic sequence advance modes; relevant for sequenced waveforms and sequenced sequences only

Auto – A sequence is generated continuously

Once – A sequence is generated once

Stepped – The sequence waits for an event to jump to the next step

Arm options – Defines if a waveform is self-enabled or requires a control signal to initiate

Self-armed – The output does not require a control signal to generate waveforms

Armed – A control signal is required to enable waveform generation

Idle waveform – Defines the state of the output waveform before an event (enable, trigger, or gate) initiates a waveform

DC – First waveform point

Wave – First waveform in a sequence

Enable signal – Defines the source from where an enable signal is expected. Affects the output only when the arm option is set to armed

BUS – A remote command enables the output

Event – A transition at the event input enables the output

Abort signal – Defines the source from where an abort signal is expected

BUS – A remote command aborts all output activities. This signal does not affect the generator when set to operate in self-armed mode

Initiate signal – Defines the wait for trigger entry port to initiate a waveform. Relevant for trigger and gated run modes only

F.P. – A front panel push button is used to initiate a waveform or a sequence

BUS – A remote command initiates a waveform or a sequence. Selecting this option automatically disables front panel operation

Trigger – A valid signal at the trigger input initiates a waveform or a sequence

Wave loops – Defines how many times the waveform repeats after a valid initiate signal

Seq loops – Defines how many times the sequence and sequenced sequences repeat after a valid initiate signal

Jump flag – Marks a step in a sequence as either wait for an event to jump or unconditional jump

Bit = 0 – Unconditional jump to the next step upon completion of the waveform loops

Bit = 1 – Dwell on current step and wait for an event to jump to the next step

Jump signal – Defines the entry port for a signal to cause a jump

Event – Assigns the event input as the jump signal

Smart trigger – Defines if smart trigger features are available for a specific function.

Using sequences in continuous run mode

This section describes how the sequence generator behaves in continuous run mode and in conjunction with the three advance mode options: auto, once, and stepped. The examples are driven from the sequence table shown in Figure 1. The *Continuous, auto sequence* section below relates to Table 2 and the section's flow charts depict the sequence flows for the various advance modes.

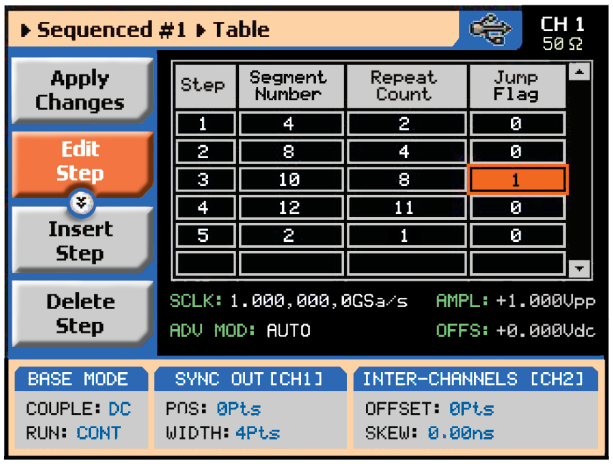

Figure 1: Sequence table example

Table 2: Sequence advance in continuous run mode option

Advance mode	Arm options	Idle waveform	Enable signal	Abort signal	Wave loops	Seq loops	Jump flag	Jump signal
Auto	Self-armed	Sequence	–	–	1-1M	–	Bit (0|1)	Event
	Armed	Wave	BUS|Event	BUS	1-1M	–	Bit (0|1)	Event
Once	Armed	Wave	BUS|Event	BUS	1-1M	1-1M	Bit (0|1)	–
Stepped	Armed	Wave	BUS|Event	BUS	–	–	–	Event

Continuous, auto sequence

Let us take the first line in Table 2 and analyze how a sequence behaves if you placed the run mode in continuous, advance mode in auto and the arm option in self-armed. Figure 1 shows the actual sequence table entries and Figure 2 depicts the sequence flow as listed in the sequence table.

This is the simplest form of sequence. Notice that as soon as the sequence is programmed, the output starts generating waveforms in an ascending order per the advanced sequence table. Also notice that the jump flag for Step 3 is set to "1". The jump flag controls the flow of the sequence. When encountered, the step halts until a jump event is asserted, in this case, Step 3 of the sequence. The sequence continues until it reaches Step 5 and then sequence generation resumes without delay and without waiting for external signals to trigger the sequence.

If there was no jump flag set for Step 3, the same sequence as shown in Figure 2 repeats itself continuously and indefinitely until turned off by the user.

Now, let us add a bit of complexity to the continuous, self-armed auto mode by adding an arming option. The arm option idles on the first waveform and waits until a valid event starts the sequence. After this event, the sequence behaves as expected from a self-armed sequence scheme. The user can select when to abort the sequence.

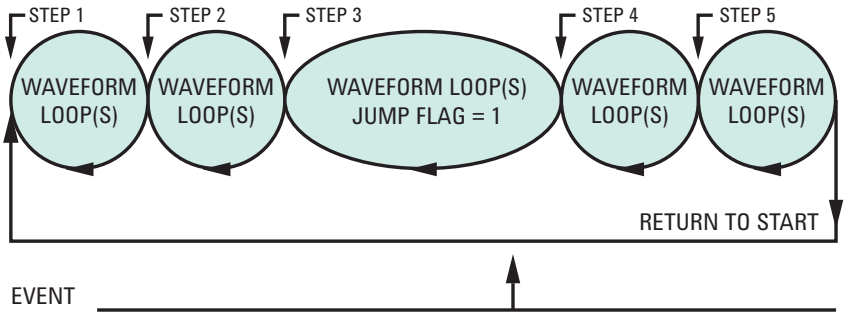

Figure 2: Continuous, self armed auto sequence mode

Figure 3 depicts the continuous auto sequence in arm mode, using the same sequence table that was used for the self-armed mode.

Immediately after the sequence table and modes have been programmed, the output starts idling on the waveform, which is defined in Step 1. An enable event is required to start the sequence and when asserted, the sequence pointer shifts to the start of the first waveform and begins generating the sequence.

Notice that the jump flag for Step 3 is set to "1". The jump flag controls the flow of the sequence. When encountered, the step halts until a jump event is asserted, in this case, Step 3 of the sequence. The sequence continues until it reaches Step 5 and resumes generating the sequence without delay and without waiting for external signals to trigger the sequence; the same sequence as shown in Figure 2. If there was no jump flag set for Step 3, the sequence repeats itself continuously and indefinitely until turned off by the user.

The sequence can be stopped when one of two conditions occurs: 1) an abort signal is asserted or 2) another output function is programmed by the user. An abort signal returns the sequence pointer to Step 1 where the output idles on this step until the sequence is re-enabled.

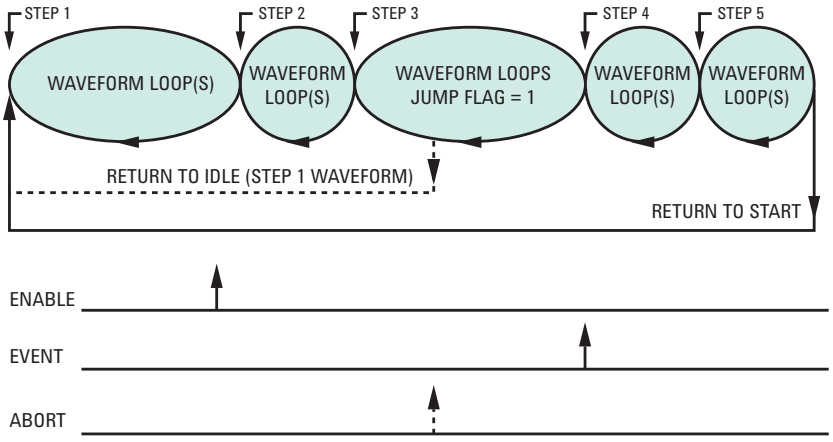

Figure 3: Continuous, armed auto mode

Continuous, once sequence

Let us take the third line in Table 2 and analyze how a sequence behaves if you place the run mode in continuous and set the advance mode to once. Figure 1 shows the actual sequence table entries and Figure 4 depicts the sequence flow as listed in the sequence table.

The once mode is similar to the continuous, armed auto sequence except that the sequence loops only the number of times that have been defined by the sequence loop's counter.

Immediately after the sequence table and modes have been programmed, the output starts idling on the waveform which is defined in Step 1. An enable event is required to start the sequence and when asserted, the sequence pointer shifts to the start of the first waveform and initiates generation of the sequence.

Notice that the jump flag for Step 3 is set to "1". The jump flag controls the flow of the sequence. When encountered, the step halts until a jump event is asserted, in this case, Step 3 of the sequence. The sequence continues until it reaches Step 5. If the sequence loop's counter is set to "1", the output will cease generating the sequence and will idle on the waveform as programmed in Step 1. If the sequence loop counter has been programmed to a certain value, the sequence will repeat itself until it completes the number of loops specified and then will idle on the waveform as specified in Step 1.

The sequence can be stopped when one of three conditions occurs: 1) an abort signal is asserted and the signal returns the sequence pointer to Step 1 where the output idles until the sequence is re-enabled, 2) another output function is programmed by the user, or 3) the sequence loop counter reaches the final number of loops.

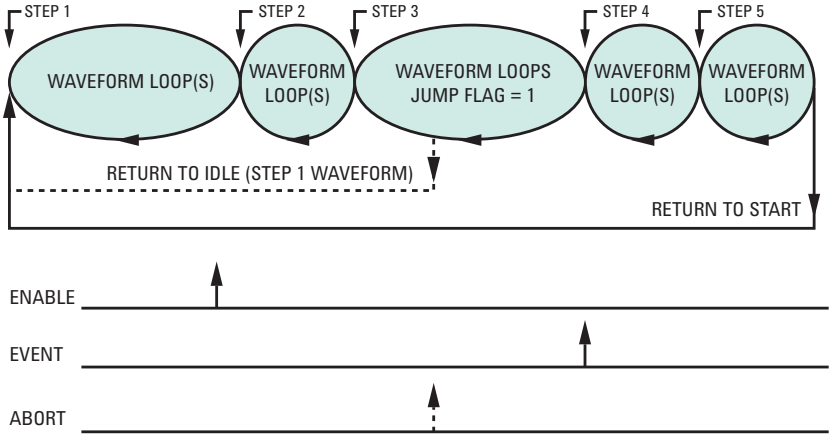

Figure 4: Continuous, once sequence mode

Continuous, stepped sequence

The forth line in Table 2 contains parameters that affect the continuous, stepped sequence. In general, stepped sequence requires events to advance from one step to the next. In fact, this is like programming all jump bits in the sequence table to "1".

Immediately after the sequence table and modes are programmed, the output starts idling on the waveform defined in Step 1. An enable event is required to start the sequence. When asserted, the sequence pointer shifts to the start of the first waveform and initiates generation of the first waveform in the sequence. From this point onwards, only events advance waveforms from step to step and when the sequence ends, it starts all over again without the need for another enable signal.

The sequence can be stopped when one of three conditions occurs: 1) an abort signal is asserted and returns the sequence pointer to Step 1, where the output idles until the sequence is re-enabled, 2) another output function is programmed by the user, or 3) events are no longer available to advance the steps.

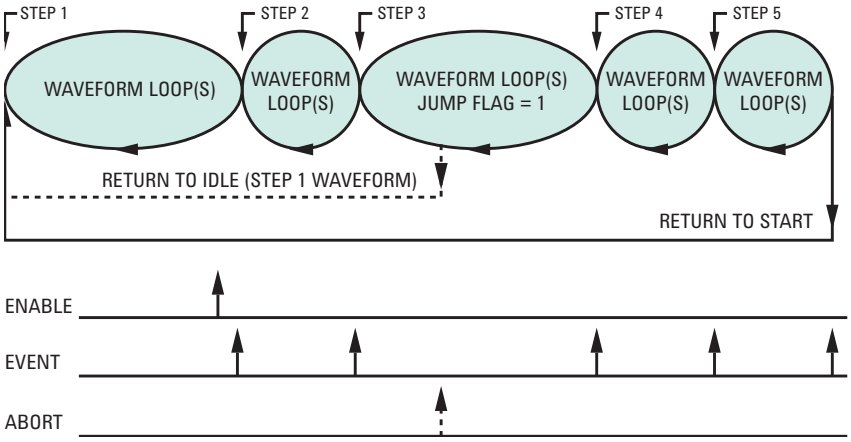

Figure 5: Continuous, stepped sequence mode

Using sequences in triggered run mode

The following section describes how the sequence generator behaves in triggered run mode and in conjunction with the three advance mode options: auto, once, and stepped. The examples are all driven from the sequence table example shown in Figure 6. The explanations below pertain to Table 3 and the flow charts that depict sequence flows for the various advance modes.

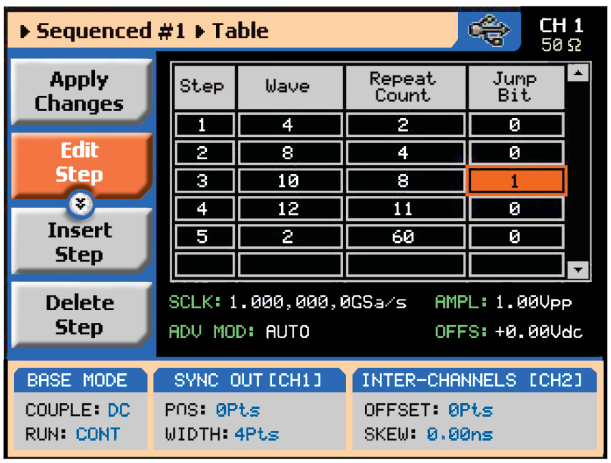

Figure 6: Sequence table example

Table 3: Sequence advance in triggered run mode option

Advance mode	Arm options	Idle waveform	Abort signal	Initiate signal	Wave loops	Seq loops	Jump flag	Jump signal	Smart trig
Auto	Self-armed	–	–	–	–	–	–	–	–
	Armed	DC	BUS	F.P. | BUS | Trigger	1-1M	–	Bit (0 | 1)	Event	Yes
Once	Armed	DC	BUS	F.P. | BUS | Trigger	1-1M	1-1M	Bit (0 | 1)	Event	–
Stepped	Armed	DC	BUS	F.P. | BUS | Trigger	–	–	–	Event	Yes

Triggered, auto sequence

Let us analyze how a sequence behaves if you place the run mode in triggered and the advance mode in auto. Figure 6 shows the actual sequence table entries and Figure 7 depicts the sequence flow as listed in the sequence table.

Notice that as soon as the sequence is programmed, the output starts idling on a DC level and waits there for a valid trigger signal. The output starts generating the waveforms in the sequence table in an ascending order, as listed in the sequence table. Also notice that the jump flag for Step 3 is set to "1". The jump flag is used for controlling the flow of the sequence. When encountered, the output dwells on the waveform until a jump event is asserted, in this case, Step 3 of the sequence. Then, the sequence continues until it reaches Step 5 and resumes generating the sequence again; without delay and without waiting for external signals to trigger the sequence.

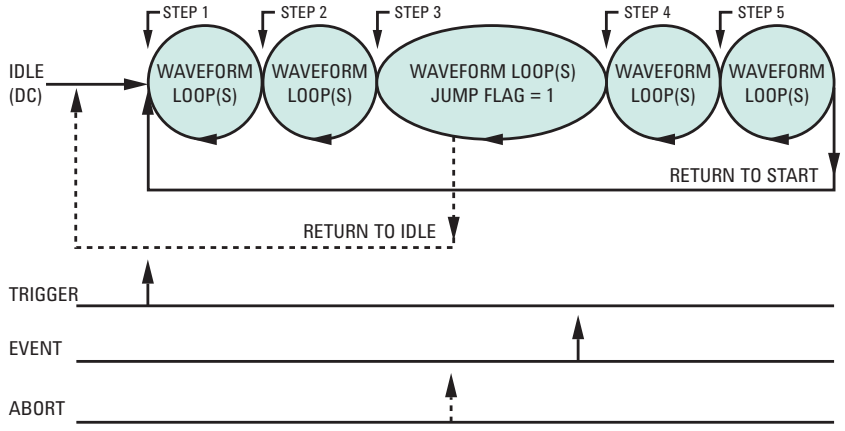

Figure 7: Triggered, auto sequence mode

If there was no jump flag set for Step 3, the same sequence shown in Figure 7 would repeat itself continuously and indefinitely until turned off by the user.

Finally, if you compare the triggered auto run mode sequence to the continuous and armed auto run mode, the difference is obvious; one idles on a waveform and the other on a DC level. Once enabled or triggered, the two types of sequences behave exactly the same.

Triggered, once sequence

The once advance mode differs from the auto advance mode only in the way the sequence stepper behaves at the end of the sequence. In auto advance mode the sequence repeats itself continuously until stopped by the user. In once advance mode, the user defines a loop counter that tells the sequence generator how many times it should be repeated. At the end of the repeat counter, the sequence resumes idle position on a DC level.

Figure 6 shows the actual sequence table entries and Figure 8 shows the sequence flow as listed in the sequence table. Notice that as soon as the sequence is programmed, the output starts idling on a DC level and will wait there for a valid trigger signal. Then, the output starts generating the waveforms in the sequence table in an ascending order as listed in the advanced sequence table. Also notice that the jump flag for Step 3 is set to "1". The jump flag controls the flow of the sequence. When encountered, the output dwells on the waveform until a jump event is asserted, in this case, Step 3 of the sequence. The sequence continues until it reaches Step 5 and then resumes to idle on a DC level, except if the loop counter has been programmed to generate "N" multiples of sequence loops.

If there was no jump flag set for Step 3, the same sequence as shown in Figure 2 would repeat itself for "N" times and then resume to idle on a DC level.

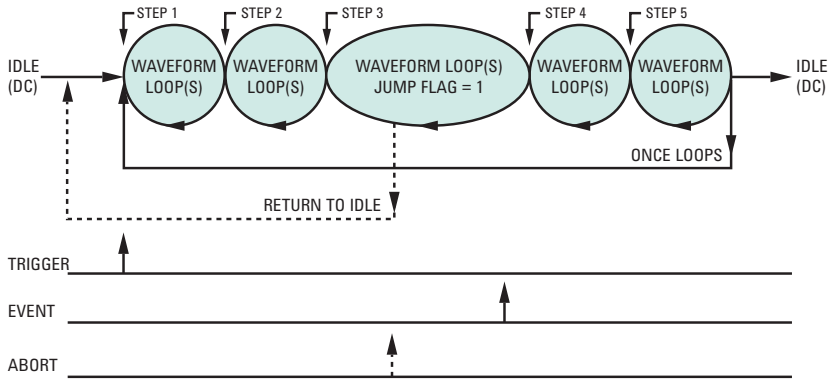

Figure 8: Triggered, once sequence mode

Triggered, stepped sequence

As described in the continuous stepped sequence mode, stepped sequence requires events to advance from one step to the next. The difference between the continuous and triggered modes is that the continuous stepped advance mode starts from a waveform (as programmed for Step 1, and the triggered step advance mode starts from a DC level.

Immediately after the sequence table and modes have been programmed, the output starts idling on a DC level. A trigger event is required to start the sequence. When asserted, the sequence pointer shifts to the start of the first waveform and initiates generation of the first waveform in the sequence. From this point onwards, only events advance waveforms from step to step, and when the sequence ends it starts all over again without the need for another trigger signal.

The sequence can be stopped when one of three conditions occurs: 1) an abort signal is asserted and returns the sequence pointer to Step 1 where the output idles until the sequence is re-enabled, 2) another output function is programmed by the user, or 3) events no longer are available to advance the steps.

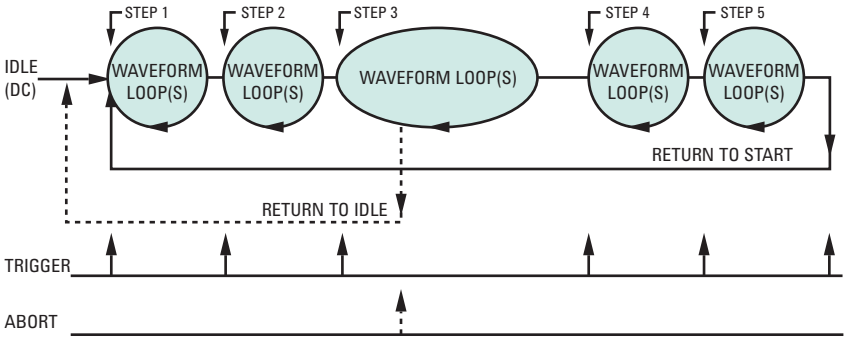

Figure 9: Triggered, stepped sequence mode

Using sequences in gated run mode

The following section describes how the sequence generator behaves in gated run mode. In fact, there is only one advance mode that can be used in conjunction with the gated run mode and that is auto advance. Once and stepped advance modes can not be used in gated run mode. The sequence example is driven from the sequence table shown in Figure 10. The explanations below pertain to Table 4 and the flow chart that depict sequence flow for the gated auto advance mode.

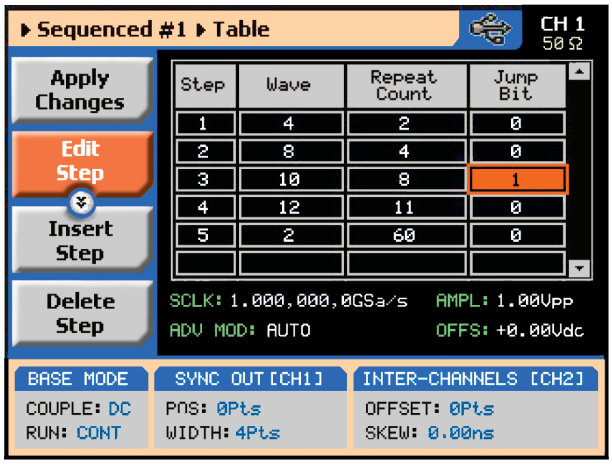

Figure 10: Sequence table example

Table 4: Sequence advance in gated run mode

Advance mode	Arm options	Idle waveform	Abort signal	Initiate signal	Wave loops	Jump flag	Jump signal
Auto	Armed	DC	BUS	F.P. | Trigger	1-1M	Bit (0 | 1)	Event

Gated, auto sequence

Now let us analyze how a sequence behaves if you place the run mode in gated and the advance mode in auto. Figure 10 shows the actual sequence table entries and Figure 11 depicts the sequence flow as listed in the sequence table.

Notice that as soon as the sequence is programmed, the output starts idling on a DC level and waits there for a valid gating signal. Then, the output starts generating the waveforms in the sequence table in an ascending order. Also notice that the jump flag for Step 3 is set to "1". The jump flag controls the flow of the sequence. When encountered, the output dwells on the waveform until a jump event is asserted, in this case, Step 3 of the sequence. Then, the sequence continues until it reaches Step 5. It then resumes generating the sequence without delay and without waiting for external signals to trigger the sequence.

If there was no jump flag set for Step 3, the same sequence as shown in Figure 7 would repeat itself continuously and indefinitely until turned off by the user.

The sequence can be stopped when one of three conditions occurs: 1) an abort signal is asserted and returns the sequence pointer to a DC level where the output idles until the sequence is re-enabled, 2) another output function is programmed by the user, or 3) the gating signal transitions to low and hence terminates the gated operation.

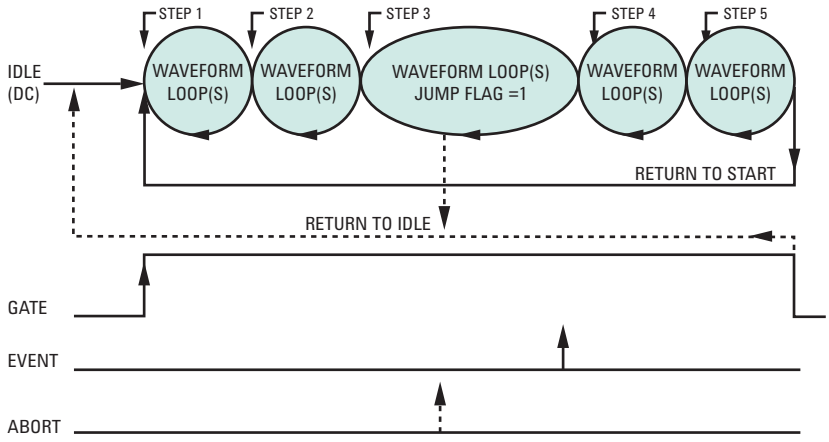

Figure 11: Gated, auto sequence mode

Advanced Sequence and Run Mode Options

Table 1 shows how advance modes affect the advanced sequence in various run mode options. Just as a brief reminder, an advanced sequence is created in a special advanced sequence table. To create an advanced sequence one must do the following:

1. Define waveform segments.
2. Download waveforms to the waveform segments.
3. Write sequence tables that defines steps, segments, repeats, and jump flags.
4. Write an advanced sequence table that defines steps, sequences, repeats and jump flags, just as is done for normal sequences except the table entries require sequence numbers, not segment numbers. Figure 12 shows an example of an advanced sequence table created on an 81180A front panel display.
5. Select an advance mode. Advance mode defines how the waveforms in the sequence table step through the sequence table entries. There are three advance mode options: auto, once and stepped.
6. Select run mode. Run mode defines if the sequence will be generated continuously without intervention from the outside world, or if it requires an enable, trigger, or gating events to start a sequence.

If the above steps are defined properly, your advanced sequence will be generated at the output connector. Note that downloading waveforms, creating sequence tables, and then an advanced sequence table, are prerequisites to generating any type of sequence. Advance and run modes can be modified later with no effect on the sequence table. The 81180A AWG can store only one advanced sequence table, but can store up to 1,000 different normal sequence tables that can be used for the advanced sequencing technique.

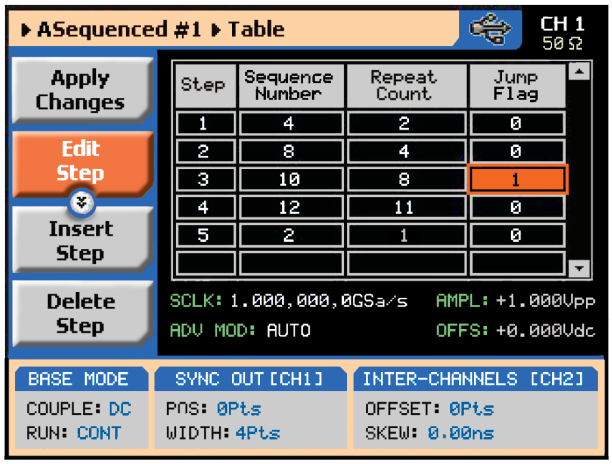

Figure 12: Advanced sequence table example

Using an advanced sequence in continuous run mode

The following section describes how the advanced sequence generator behaves in continuous run mode and in conjunction with the three advance mode options: auto, once, and stepped. The examples are all driven from the advanced sequence table example as shown in Figure 1. The explanations below pertain to Table 5 and the flow charts that depict sequence flows for the various advance modes.

Table 5: Advanced sequence in continuous Run Mode

Advance mode	Arm options	Idle waveform	Enable signal	Abort signal	Wave loops	Seq loops	Jump flag	Jump signal
Auto	Self-armed	Adv'd seq	–	–	1-1M	–	Bit (0|1)	Event
	Armed	Sequence	BUS|Event	BUS	1-1M	–	Bit (0|1)	Event
Once	Armed	Sequence	BUS|Event	BUS	1-1M	1-1M	Bit (0|1)	–
Stepped	Armed	Sequence	BUS|Event	BUS	–	–	–	Event

Continuous, auto advanced sequence

Let us take the first line in Table 5 and analyze how a sequence behaves if you place the run mode in continuous, advance mode in auto and the arm option in self-armed. Figure 12 shows the actual sequence table entries and Figure 13 depicts the advanced sequence flow as listed in the advanced sequence table.

This is the simplest form of advanced sequence. Notice that as soon as the sequence is programmed; the output generates sequences in an ascending order, as listed in the advanced sequence table. Also notice that the jump flag for Step 3 is set to "1". The jump flag controls the flow of the sequence. When encountered, the step halts until a jump event is asserted, in this case, Step 3 of the sequence. Then, the sequence

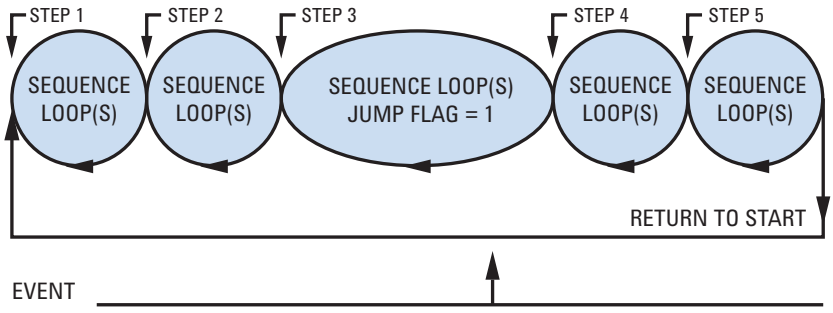

Figure 13: Continuous, self-armed auto advanced sequence mode

continues until it reaches Step 5 and resumes generating the sequence again without delay and without waiting for external signals to trigger the sequence.

If there was no jump flag set for Step 3, the same sequence as shown in Figure 13 would repeat itself continuously and indefinitely until turned off by the user.

Now, let us add complexity to the continuous, self-armed auto advanced sequence mode by adding an arming option. The arm option idles on the first sequence and waits until a valid event starts the advanced sequence as listed in the advanced sequence table. After this event, the advanced sequence behaves just as a self-armed advanced sequence scheme, except the user can select when to abort the sequence when it is no longer required.

Figure 14 depicts the continuous auto advanced sequence in armed mode, using the same sequence table as was used for the self armed mode.

Immediately after the advanced sequence table and modes have been programmed, the output starts idling on the sequence which is defined in Step 1. An enable event is required to start the sequence and when asserted, the advanced sequence pointer shifts to the start of the first sequence and initiates generation of the sequence.

Notice that the jump flag for Step 3 is set to "1". The jump flag is controls the flow of the sequence. When encountered, the step halts until a jump event is asserted, in this case, Step 3 of the sequence. The sequence continues until it reaches step 5 and then resumes generating the sequence again without delay and without waiting for external signals to trigger the sequence. The same sequence as shown in Figure 14. If there was no jump flag set for Step 3, it would have repeated itself continuously and indefinitely until turned off by the user.

The advanced sequence can be stopped when one of two conditions occurs: 1) an abort signal is asserted or 2) another output function is programmed by the user. An abort signal returns the advanced sequence pointer to Step 1. The output idles on this step until the sequence is re-enabled.

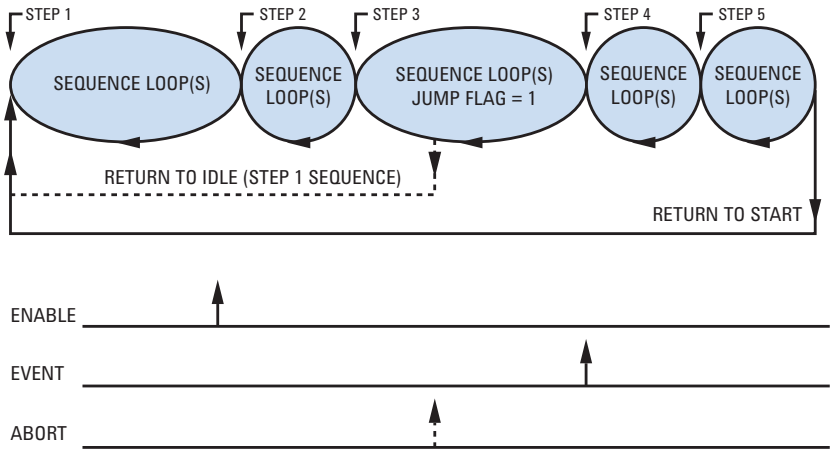

Figure 14: Continuous, armed auto advanced sequence mode

Continuous, once advanced sequence

Let us take the third line in Table 5 and analyze how an advanced sequence behaves if you place the run mode in continuous and the advance mode in once. Figure 12 shows the actual sequence table entries and Figure 15 depicts the sequence flow as listed in the sequence table.

The once mode is similar to the continuous, armed auto sequence except that the advanced sequence loops only the number of times that are defined by the sequence loop's counter.

Immediately after the advanced sequence table and modes have been programmed, the output starts idling on the sequence defined in Step 1. An enable event is required to start the sequence and when asserted, the sequence pointer shifts to the start of the first sequence and initiates generation of the advanced sequence.

Notice that the jump flag for Step 3 is set to "1". The jump flag controls the flow of the sequence. When encountered, the step halts until a jump event is asserted, in this case, Step 3 of the sequence. The sequence continues until it reaches Step 5. If the sequence loop's counter is set to "1", the output will cease generating the sequence and will idle on the sequence as programmed in Step 1. If the sequence loop's counter has been programmed to a certain value, the sequence will repeat itself until it completes the number of loops and then resume idle on the sequence as specified in Step 1.

The sequence can be stopped when one of three conditions occurs: 1) an abort signal is asserted and the signal returns the sequence pointer to Step 1 where the output idles on this step until the sequence is re-enabled, 2) another output function is programmed by the use, or 3) the sequence loop counter has reached the final number of loops.

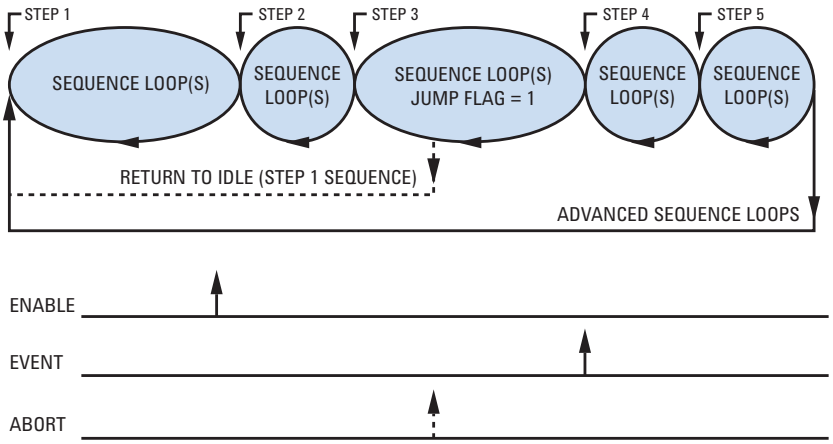

Figure 15: Continuous, once advanced sequence mode

Continuous, stepped sequence

The forth line in Table 5 contains parameters that affect the continuous, stepped advanced sequence. In general, stepped advanced sequence requires events to advance from one step to the next. In fact, this is like programming all jump bits in the sequence table to "1".

Immediately, after the advanced sequence table and modes have been programmed, the output starts idling on the sequence defined in Step 1.

An enable event is required to start the sequence and when asserted, the sequence pointer shifts to the start of the first sequence and initiates generation of the first sequence in the advanced sequence loop. From this point onwards, only events advance sequences from step to step and when the advanced sequence ends, it starts all over again without the need for another enable signal.

The advanced sequence can be stopped when one of three conditions occurs: 1) an abort signal is asserted and returns the sequence pointer to Step 1 where the output idles on this step until the sequence is re-enabled, 2) another output function is programmed by the user, or 3) events no longer are available to advance the steps.

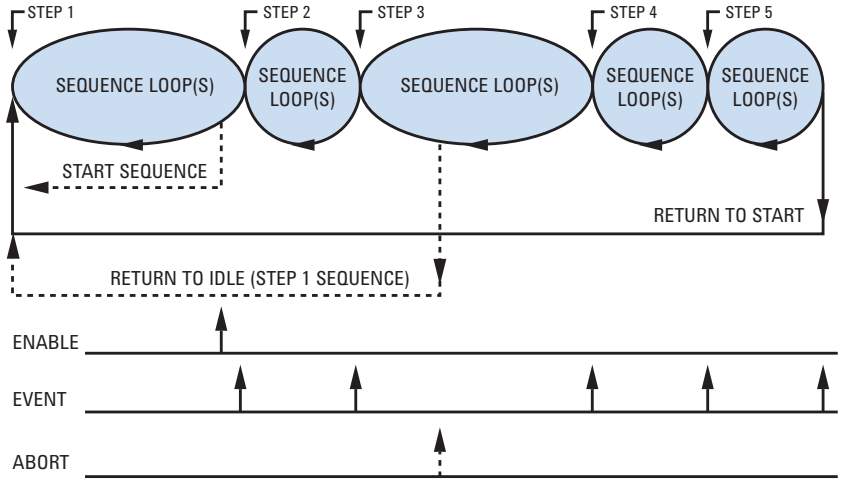

Figure 16: Continuous, stepped advanced sequence mode

Using an Advanced Sequence in Triggered Run Mode

The following paragraphs describe how the advanced sequence generator behaves in triggered run mode and in conjunction with the three advance mode options: auto, once, and stepped. The examples are all driven from the advanced sequence table shown in Figure 17 and the explanations below pertain to Table 6. The flow charts depict sequence flows for the various advance modes.

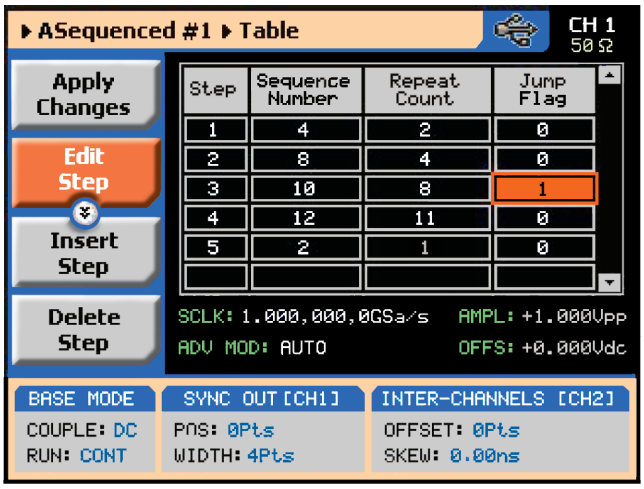

Figure 17: Sequence table example

Table 6: Advanced sequence advance in triggered run mode

Advance mode	Arm options	Idle waveform	Abort signal	Initiate signal	Wave loops	Seq loops	Jump flag	Jump signal	Smart trig
Auto	Self-armed	–	–	–	–	–	–	–	–
	Armed	DC	BUS	F.P. | BUS | Trigger	1-1M	–	Bit (0 | 1)	Event	Yes
Once	Armed	DC	BUS	F.P. | BUS | Trigger	1-1M	1-1M	Bit (0 | 1)	Event	–
Stepped	Armed	DC	BUS	F.P. | BUS | Trigger	–	–	–	Event	Yes

Triggered, auto advanced sequence

Now let us analyze how a sequence behaves if you place the run mode in triggered and advance mode in auto. Figure 17 shows the actual sequence table entries and Figure 18 depicts the advanced sequence flow as listed in the sequence table.

Notice that as soon as the sequence is programmed, the output starts idling on a DC level and waits there for a valid trigger signal. The output then starts generating the sequences in the sequence table in ascending order, as listed in the advanced sequence table. Also notice that the jump flag for Step 3 is set to "1". The jump flag controls the flow of the sequence. When encountered, the output dwells on the sequence until a jump event is asserted, in this case, Step 3 of the sequence. Then the sequence continues until it reaches Step 5 and resumes generating the sequence again without delay and without waiting for external signals to trigger the sequence.

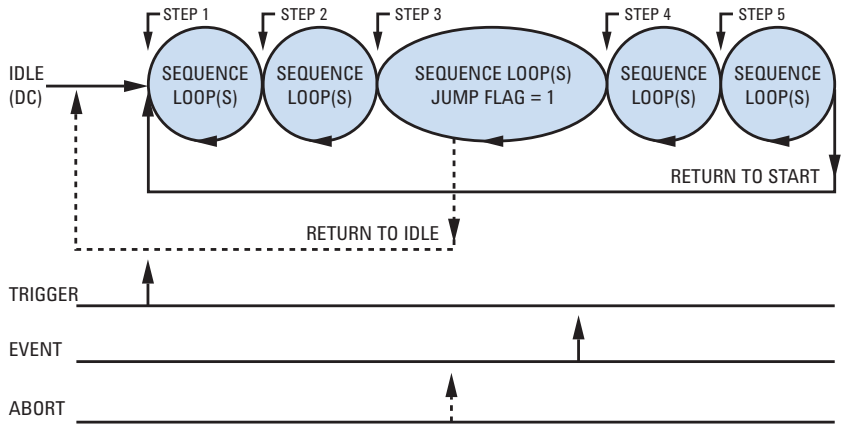

Figure 18: Triggered, auto advanced sequence mode

If there was no jump flag set for Step 3, the same sequence as shown in Figure 18 would have repeated itself continuously and indefinitely until turned off by the user.

Finally, if you compare the triggered auto run mode sequence to the continuous and armed auto run mode, the difference is obvious: one idles on a sequence and the other on a DC level. Once enabled or triggered, the two types of sequences behave exactly the same.

Triggered, once advanced sequence

The once advance mode differs from the auto advance mode only in the way the sequence stepper behaves at the end of the sequence. In auto advance mode the sequence repeats itself continuously until stopped by the user. In once advance mode, the user defines a loop counter that tells the sequence generator how many times to repeat. At the end of the repeat counter, the sequence resumes idle position on a DC level.

Figure 17 shows the actual sequence table entries and Figure 19 depicts the sequence flow as listed in the advanced sequence table. Notice that as soon as the sequence is programmed, the output starts idling on a DC-level and will wait there for a valid trigger signal. The output then starts generating the sequences in the sequence table in ascending order, as listed in the advanced sequence table. Also notice that the jump flag for Step 3 is set to "1". The jump flag controls the flow of the sequence. When encountered, the output dwells on the sequence until a jump event is asserted, in this case, Step 3 of the sequence. The sequence continues until it reaches Step 5 and then resumes to idle on a DC level except if the loop counter has been programmed to generate "N" multiples of sequence loops.

If there was no jump flag set for Step 3, the same sequence as shown in Figure 19 would repeat itself for "N" times and then resume to idle on a DC level.

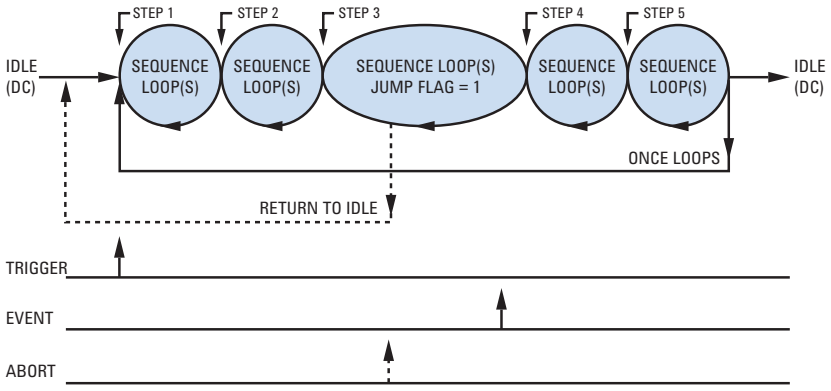

Figure 19: Triggered, once advanced sequence mode

Triggered, stepped advanced sequence

As described in the continuous stepped advanced sequence mode, stepped sequence requires events to advance from one step to the next. The difference between the two modes is that the continuous stepped advance mode starts from a sequence (as programmed for step 1) and the triggered step advance mode starts from a DC level.

Immediately, after the advanced sequence table and modes have been programmed, the output starts idling on a DC level. A trigger event is required to start the sequence and when asserted, the sequence pointer shifts to the start of the first sequence and initiates generation of the first sequence in the advanced sequence. From this point onwards, only events advance sequences from step to step and when the sequence ends, it starts all over again without the need for another trigger signal.

The advanced sequence can be stopped when one of three conditions occurs: 1) an abort signal is asserted and returns the sequence pointer to Step 1 where the output idles on this step until the sequence is re-enabled 2) another output function is programmed by the user, or 3) events no longer are available to advance the steps.

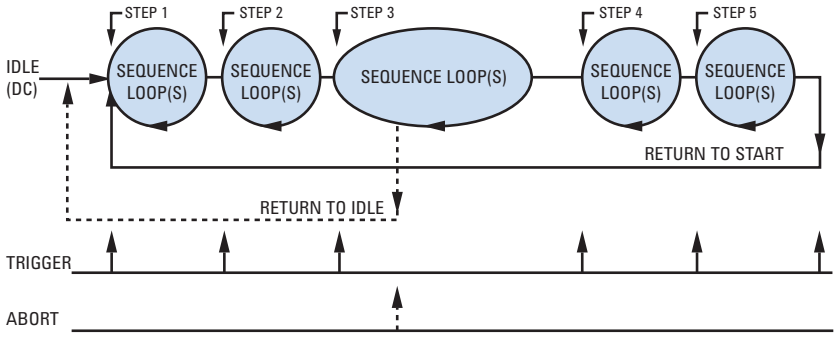

Figure 20: Triggered, stepped advanced sequence mode

Using an Advanced Sequence in Gated Run Mode

The following section describes how the advanced sequence generator behaves in gated run mode. In fact, there is only one advance mode that can be used in conjunction with the gated run mode is auto advance. Once and stepped advance modes can not be used in gated run mode. The sequence example is driven from the advanced sequence table shown in Figure 21. The explanations below pertain to Table 7 and the flow chart depicts sequence flow for the gated auto advance mode.

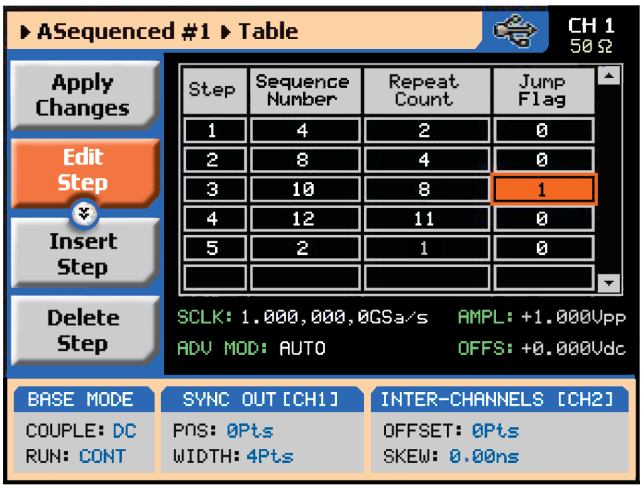

Figure 21: Advanced sequence table example

Table 7: Sequence advance in gated run mode option

Advance mode	Arm options	Idle waveform	Abort signal	Initiate signal	Wave loops	Jump flag	Jump signal
Auto	Armed	DC	BUS	F.P. | Trigger	1-1M	Bit (0 | 1)	Event

Gated, auto sequence

Now let us analyze how a sequence behaves if you place the run mode in gated and advance mode in auto. Figure 21 shows the actual sequence table entries and Figure 22 depicts the sequence flow as listed in the sequence table.

Notice that as soon as the sequence is programmed, the output starts idling on a DC level and waits there for a valid gating signal. The output then starts generating the sequences in an ascending order, as listed in the advanced sequence table. Also notice that the jump flag for Step 3 is set to "1". The jump flag controls the flow of the sequence. When encountered, the output dwells on the sequence until a jump event is asserted, in this case, Step 3 of the sequence. The sequence continues until it reaches Step 5 and then resumes generating the sequence again without delay and without waiting for external signals to trigger the sequence.

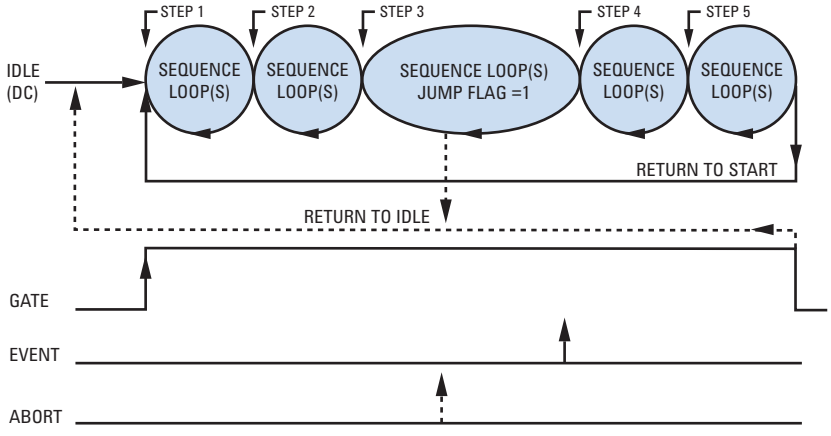

Figure 22: Gated, auto sequence mode

If there was no jump flag set for Step 3, The same sequence shown in Figure 22 would repeat itself continuously and indefinitely until turned off by the user.

The sequence can be stopped when one of three conditions occurs: 1) an abort signal is asserted and returns the sequence pointer to a DC level where the output idles until the sequence is re-enabled 2) another output function is programmed by the user, or 3) the gating signal transitions to low and hence terminates the gated operation.

Related Literature

81180A Arbitrary Waveform Generator, Data Sheet, literature number 5990-5697EN

81180A Arbitrary Waveform Generator, Quick Fact Sheet, literature number 5990-5711EN

Agilent Email Updates

www.agilent.com/find/emailupdates
Get the latest information on the products and applications you select.

LXI

www.lxistandard.org
LXI is the LAN-based successor to GPIB, providing faster, more efficient connectivity. Agilent is a founding member of the LXI consortium.

Agilent Channel Partners

www.agilent.com/find/channelpartners
Get the best of both worlds: Agilent's measurement expertise and product breadth, combined with channel partner convenience.

Agilent Advantage Services

Agilent Advantage Services is committed to your success throughout your equipment's lifetime. We share measurement and service expertise to help you create the products that change our world. To keep you competitive, we continually invest in tools and processes that speed up calibration and repair, reduce your cost of ownership, and move us ahead of your development curve.

www.agilent.com/find/advantageservices

Agilent Electronic Measurement Group
KEMA Certified
ISO 9001:2008
Quality Management System

www.agilent.com/quality

www.agilent.com
www.agilent.com/find/81180

For more information on Agilent Technologies' products, applications or services, please contact your local Agilent office. The complete list is available at:

www.agilent.com/find/contactus

Americas

Canada	(877) 894 4414
Brazil	(11) 4197 3500
Latin America	305 269 7500
Mexico	01800 5064 800
United States	(800) 829 4444

Asia Pacific

Australia	1 800 629 485
China	800 810 0189
Hong Kong	800 938 693
India	1 800 112 929
Japan	0120 (421) 345
Korea	080 769 0800
Malaysia	1 800 888 848
Singapore	1 800 375 8100
Taiwan	0800 047 866
Thailand	1 800 226 008

Europe & Middle East

Austria	43 (0) 1 360 277 1571
Belgium	32 (0) 2 404 93 40
Denmark	45 70 13 15 15
Finland	358 (0) 10 855 2100
France	0825 010 700*
	*0.125 €/minute
Germany	49 (0) 7031 464 6333
Ireland	1890 924 204
Israel	972-3-9288-504/544
Italy	39 02 92 60 8484
Netherlands	31 (0) 20 547 2111
Spain	34 (91) 631 3300
Sweden	0200-88 22 55
Switzerland	0800 80 53 53
United Kingdom	44 (0) 118 9276201

Other European Countries:
www.agilent.com/find/contactus

Revised: July 8, 2010

Product specifications and descriptions in this document subject to change without notice.

Printed in USA, August 11, 2010
5990-5965EN

Agilent Technologies