Agile RF Synthesizer & AOM driver

XRF421

Revision 1.8.3
Serials A09xxx and newer
Firmware v1.12.2

8. Simple table mode

Table mode performs sequential execution of up to 8191 instructions with precise timing. This enables generation of complicated pulse sequences, custom envelope shapes, and automated control of experiment sequences through digital I/O.

There are two versions of table mode: *simple* table mode (TSB mode) which utilises the DDS serial interface, and *advanced* table mode (TPA mode) that utilises the parallel interface. Advanced table mode has increased functionality and improved timing resolution, as described in chapter 9, and is only available in XRF devices.

8.1 Operational principle

A table is defined as a number of entries that describe the frequency, amplitude and phase of the rf output at each step, as well as any desired I/O. These are preloaded by the FPGA into a DDS*profile*, so that when the sequence is executed the parameters are updated instantaneously. However, this makes table mode incompatible with both external modulation and PID control.

The speed of the serial interface limits the rate at which new instructions can be loaded into the DDS, so the duration of each table entry is discretised at 1 µs.[1]

Once the sequence has been defined using the `TABLE,ENTRY` commands, it is readied for execution using the `TABLE,ARM` command. The table is checked for errors, and will fail if an incompatibility is detected. For example, to use digital output, the associated pin must be configured for AUTO control (§7.4).

Once the table is armed, execution is started by either a hardware TTL trigger on the SEQ input or using the `TABLE,START` command. The

[1] Advanced table mode (XRF) is capable of 16 ns steps using the parallel bus.

phase-accumulator of the DDS is then reset and the table executes autonomously under FPGA control. This provides a very high degree of reproducibility in terms of both timing of instructions and output of the DDS generators, as the DDS phase accumulator is reset for every execution.

The table can be automatically restarted after completion by enabling the TABLE,RESTART option, and execution can be stopped mid-sequence using the TABLE,STOP command.

Each channel has its own independent table, and there are *slots* for four distinct tables in non-volatile memory. Commonly-used tables can then be stored on the device for later use. However, it should be noted that stored tables may become inoperable after a firmware upgrade and tables should be archived in human-readable form.

When a table is armed, the RF is switched on (including the amplifiers for 421-models), and upon completion of the table the final RF state remains ongoing. If it is required that the output be disabled when the table is complete, the final entry should set the power to 0x0 (zero amplitude, not 0 dBm).

8.2 Defining table entries

Table entries can be defined or queried using the TABLE,ENTRY command. Once the entries have been set, the TABLE,ENTRIES command should be used to set the length of the table. Alternately, the TABLE,APPEND command can be used to add an entry to the end of the table and update the count automatically.

It is recommended to always explicitly empty the table with the TABLE,CLEAR command before defining the entries.

TABLE,ENTRY `TABLE,ENTRY,ch,num,[freq,pow,phas,dur,flags]`

Configure the specified table entry. If only `ch` and `num` are given, the current entry of the table is returned.

ch The channel to edit (1 or 2)

num The entry to edit (1 to 8191)

freq Frequency to output during this step

pow Output power during this step

phas Phase of the RF for this step

dur Duration of this step (discretised at 1 us)

flags A comma-separated list of flags, comprised of the following:

OFF Switch off the RF signal for this step, disabling the output. Must be repeated in subsequent steps for the signal to remain off.

TRIG Repeat the current instruction until a hardware trigger is received, optionally specifying trigger source and edge (§8.4.1). More advanced behaviour is available with the `TABLE,LOOP` command instead (§8.4).

IOxy Perform a digital ouput action on pin *x* as specified by *y* (§8.3.1). Only one I/O operation can be performed in a given table entry.

IOSETx Write multiple HSB digital outputs simultaneously (§8.3.2) in conjunction with an IOMASK flag.

IOMASKy Specifies which outputs should be controlled by IOSET (§8.3.2).

The maximum duration for simple table mode is normally $2^{20} - 1\,\mu$s, 1.048 s. For multiple output mode (§8.3.2), the limit is $2^{16} - 1\,\mu$s, 65.535 ms. For advanced table mode, the maximum duration is $2^{32}-1$ times 16 ns, about 68 s.

The following commands are also provided to simplify editing tables in memory. They take the same parameters as the `TABLE,ENTRY` command but automatically update the table entry count as relevant.

TABLE,APPEND `TABLE,APPEND,ch,freq,pow,phas,dur,flags`
Add the specified table entry to the end of the table and increment the entry counter.

TABLE,INSERT `TABLE,INSERT,ch,num,freq,pow,phas,dur,flags`
Insert the specified entry into the table at the specified index, and shift all other entries down.

TABLE,DELETE `TABLE,DELETE,ch,num`
Remove only the specified entry from the table.

TABLE,CLEAR `TABLE,CLEAR,ch`
Deletes the entire table from memory and resets any state variables.

Listing 8.1 shows basic operation of table mode using the `TABLE,ENTRY` command to define the instructions. Entries are loaded in individually then the total number is set with `TABLE,ENTRIES`. The table is armed and executed using `TABLE,START`, resulting in typical output shown in Figure 8.1.

The last instruction is held after the table completes, so it is good practice to include a final instruction that sets the output power to 0x0 if it is desired for the RF to be off after execution finishes.

```
# Example of table output
MODE,1,TSB
# Begin table
TABLE,ENTRY,1,1,100MHz,-10dBm,0,100us
TABLE,ENTRY,1,2,100MHz,0dBm,0,100us
TABLE,ENTRY,1,3,80MHz,-5dBm,0,100us
TABLE,ENTRY,1,4,80MHz,-15.0dBm,0,100us
TABLE,ENTRY,1,5,100MHz,-2.0dBm,0,100us
TABLE,ENTRY,1,6,100MHz,0x0C00,0,100us
TABLE,ENTRY,1,7,100MHz,0x0200,0,100us
TABLE,ENTRY,1,8,100MHz,0x0,0,100us
TABLE,ENTRIES,1,8
TABLE,START,1
```

Listing 8.1: Simple table mode demonstration.

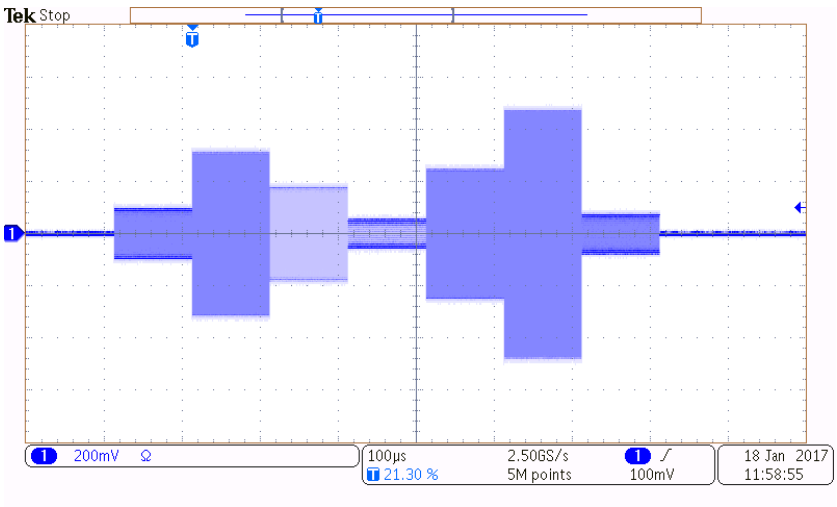

Figure 8.1: Demonstration of RF output generated by Listing 8.1.

8.3 Digital I/O

Each entry in the table can control a single digital I/O pin (§8.3.1), or write multiple pins simultaneously (§8.3.2). However, pins must be correctly configured using the `EXTIO` to be used in table mode: inputs must be set to READ mode, and outputs must be set to WRITE mode with AUTO control.

8.3.1 Digital output

Digital output can be controlled from the table on any pin which has been configured for AUTO control using the `EXTIO,CTRL` command (§7.4). The output is configured using a flag of the form `IOxy`, where `x` is the pin to use and `y` the function to perform.

The pin can specified as:

D The digital output (DOUT) associated with this channel on the DE15 connector

0-7 The specified pin of the high-speed bank with the same number

as this channel (A for CH1, B for CH2)

A0-A7 The specified pin of high-speed bank A (irrespective of channel)

B0-B7 The specified pin of high-speed bank B (irrespective of channel)

Both channels are capable of accessing the same I/O pins using the second notation. It is up to the user to ensure that this does not cause conflicts, particularly when loading a table from memory.

Available functions are listed below. Only the first letter is significant when specifying the command.

L[OW] Set the pin to output digital LOW

H[IGH] Set the pin to output digital HIGH

T[OGGLE] Switch the output level of specified pin

P[ULSE] Output a short pulse (~ 500 ns) on specified pin

Examples of I/O flags:

IO3H Set pin 3 of associated high-speed bank to output HIGH

IODT Toggle the output of the associated DOUT pin on the DE15 connector

IOA2P Output a short pulse on pin 2 of high-speed bank A

IOB1L Set pin 1 of high-speed bank B to output digital LOW

The example below shows how to toggle an output in the high-speed bank, resulting in the output shown in Figure 8.2.

```
MODE,1,TSB
# set HSB—A to output
EXTIO,MODE,1,HSB,WRITE
# set HSB pin A1 to AUTO mode
EXTIO,CONTROL,1,HS1,AUTO
# define table —— same frequency, five different amplitudes
TABLE,ENTRY,1,1,100MHz,0x200,0,2
# next entry sets pin 1 high
```

```
TABLE,ENTRY,1,2,100MHz,0x600,0,2,IO1H
# 2us later, set pin 1 low
TABLE,ENTRY,1,3,100MHz,0x1000,0,2,IO1L
TABLE,ENTRY,1,4,100MHz,0x1400,0,2
# create 500ns pulse
TABLE,ENTRY,1,5,100MHz,0x2000,0,2,IO1P
TABLE,ENTRY,1,6,100MHz,0x200,0,2
# toggle pin 1 (from low to high)
TABLE,ENTRY,1,7,100MHz,0x600,0,2,IO1T
TABLE,ENTRY,1,8,100MHz,0x1000,0,2
# toggle pin 1 again (i.e. from high to low)
TABLE,ENTRY,1,9,100MHz,0x1400,0,2,IO1T
TABLE,ENTRY,1,10,100MHz,0x2000,0,2
TABLE,ENTRIES,1,10
```

Listing 8.2: Simple example showing digital output in table mode.

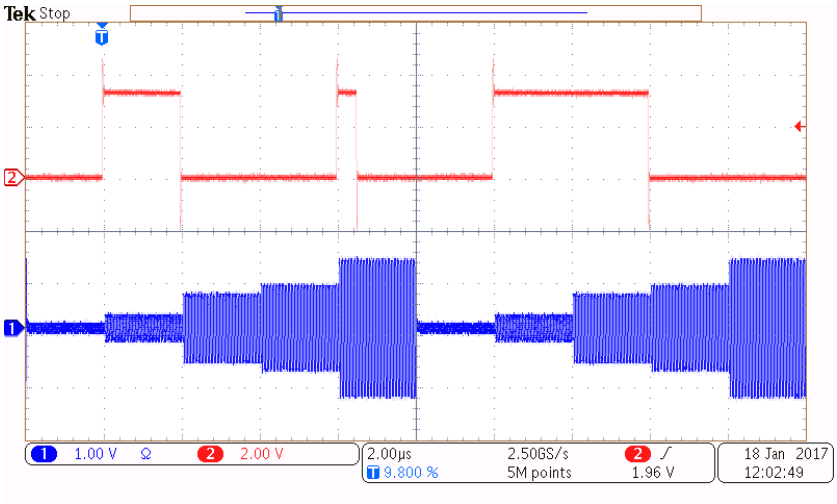

Figure 8.2: Example of table mode, showing changing RF output (blue, lower trace) and synchronised TTL output (red, upper trace) generated by the example in Listing 8.2.

8.3.2 Simultaneous digital output

As of firmware v1.3.0, multiple digital outputs can be set simultaneously in a single table instruction using the IOSET and IOMASK flags. Both instructions take a 16-bit number, where each bit corresponds to a pin of the high-speed output banks. If a bit is set in the IOMASK value, then the corresponding value of IOSET is written to that pin, overwriting any previous value.

This allows all pins of both high-speed banks to be changed in each table entry, or only a subset of the pins. Table 8.1 demonstrates an example that simultaneously writes to 5 pins of bank A and 4 pins of bank B.

	Bank B								Bank A							
Bit	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
IOSET	0	0	1	0	1	1	1	1	1	0	0	1	0	0	1	1
IOMASK	0	1	0	0	1	1	0	1	1	1	1	0	1	0	1	0
Outcome	-	0	-	-	1	1	-	1	1	0	0	-	0	-	1	-

Table 8.1: Example of simultaneous output using IOSET0x2F93 and IOMASK0x4DEA. Only the 9 pins corresponding to bits set in IOMASK are affected by the command, with "-" denoting no change.

If IOMASK is not specified, the value 0xFFFF is assumed and all HSB outputs are written at once. Care must be taken when running two tables simultaneously to ensure that the same pin isn't being written to simultaneously by both tables, which can result in undefined behaviour.

Where only a few outputs need to be combined, it is possible to chain together multiple IOxH and IOxL flags on the same table entry to avoid needing to construct the mask word. For example, the following set of output flags could be used,

```
IOA3H, IOA4L, IOB1H,
```

which is equivalent to

```
IOSET0x0208, IOMASK0x0218,
```

but has improved clarity.

For reference, the flag IOMASK0x00FF will only write to bank A, and IOMASK0xFF00 will only write to bank B. It is recommended to encode values in hexadecimal in this way to improve readability.

Note: Multiple digital output mode cannot be used in combination with LOOP or TRIG.

8.4 Loops and triggers

Table mode supports the use of loops, to simplify the generation of pulse sequences or to wait for a hardware trigger to synchronise execution with external devices. The general syntax for defining a loop is described below. A simplified option for external triggering (the TRIG flag) is described in §8.4.1.

TABLE,LOOP TABLE,LOOP,ch,source,dest,condition

Set the table to jump from the source entry to the dest entry until the condition is satisfied. If source and dest are the same, or if dest is 0, then the table effectively holds the instruction.

source and dest can be negative numbers, which are then taken as offsets. If source is negative, it is taken as an offset from the end of the table (requires TABLE,ENTRIES to be set). If dest is negative, it is taken as the offset from the source. This is particularly convenient when using the TABLE,APPEND command which automatically updates TABLE,ENTRIES.

The condition can be an integer in the range [1, 4095], corresponding to the number of times to execute the loop, or a hardware descriptor flag of the form IOxy, indicating to repeat until the digital input pin x exhibits behaviour y, as described below.

The pin syntax is described in §8.3.1, with the addition that "D" refers to the associated trigger input (CHx-SEQ) of the DE15 connector. The behaviour y is one of the following options:

H[IGH] Terminate loop on logic level HIGH *at* the loop instruction

L[OW] Terminate loop on logic level LOW *at* the loop instruction

F[ALLING] Terminate loop *after* falling edge occurs

R[ISING] Terminate loop *after* rising edge occurs

Note: When using the TTL inputs as a trigger in table mode, it is strongly recommended to ensure that the input debouncer is disabled using the DEBOUNCE command.

Loops are subject to the following restrictions:

- Nested loops are not supported.
- The TABLE,LOOP command can only be used after the source entry has been defined.
- The *first* entry and *last three* entries of the table cannot contain a LOOP or TRIG.
- Loop conditions are checked at the *start* of the instruction, so if the condition is met *during* the instruction, it will take effect on the next repetition.
- All loops will execute *at least once*, even if the condition is met when the loop is first entered.
- In simple table mode there must be a minimum of 4 table entries between consecutive LOOP instructions.

The example below shows how to define a loop with a fixed number of iterations. The loop counter is 4, so instructions 1–3 are executed a total of 5 times (Figure 8.3). Since the final three instructions cannot contain a loop, several dummy instruction are added to the end which also turn off the RF.

```
TABLE,CLEAR,1
TABLE,APPEND,1,100MHz,0dBm,0,1us
TABLE,APPEND,1,100MHz,-5dBm,0,4us
TABLE,APPEND,1,100MHz,-10dBm,0,2us
TABLE,LOOP,1,3,1,4
TABLE,APPEND,1,100MHz,-30dBm,0,1us,OFF
```

```
TABLE,APPEND,1,100MHz,-30dBm,0,1us,OFF
TABLE,APPEND,1,100MHz,-30dBm,0,1us,OFF
```

Listing 8.3: Demonstration of a simple loop

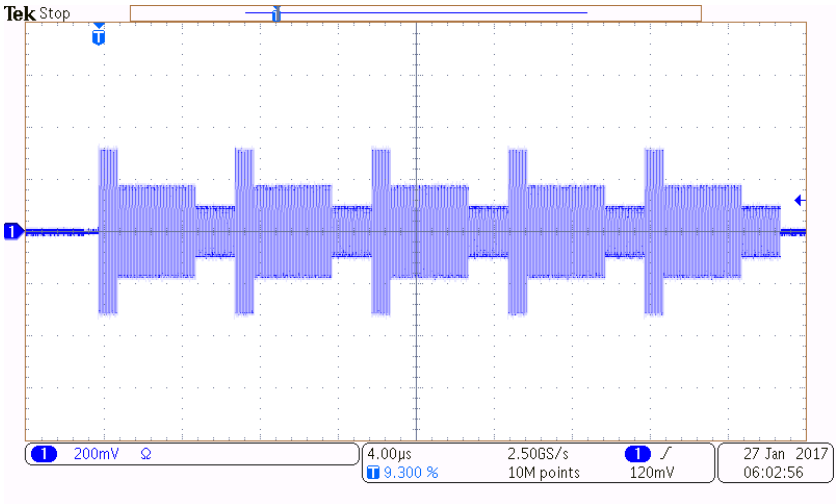

Figure 8.3: Demonstration of a simple loop.

An alternate way of specifying the `LOOP` instruction in the above example is `TABLE,LOOP,1,-1,-2,4`, since the loop should activate after the most recently added entry (source -1), and the destination (entry 1) is 2 instructions earlier.

8.4.1 External trigger flag

An alternate way to wait for an external trigger is to use the `TRIG` flag on a table entry. This alleviates the need for a separate call to the `TABLE,LOOP` command. The flag is of the form `TRIGxy` where `x` is the pin (e.g. `D` or `A0`) and `y` the condition. If neither `x` nor `y` are specified, the default is `TRIGDF`.

Similarly to the general loop, the trigger behaviour is to **repeat the current instruction until a trigger is received**. If the trigger

condition is met *during* the instruction period, the duration of the current instruction is completed first. Hence the duration of a TRIG instruction should be *as small as possible* to ensure rapid response.

The example below shows how to use the TRIG flag, with typical result shown in Figure 8.4.

```
MODE,1,TSB # set table mode
EXTIO,CTRL,1,HSB,AUTO # ready HSB1 for output
TABLE,CLEAR,1
TABLE,APPEND,1,100,5,0,10
TABLE,APPEND,1,100,10,0,1,IO1T,TRIG # wait for trigger
TABLE,APPEND,1,100,-5,0,10
TABLE,APPEND,1,100,-30,0,3,IO1L
TABLE,ARM,1 # ready table for execution
```

Listing 8.4: Demonstration of a table with a trigger command

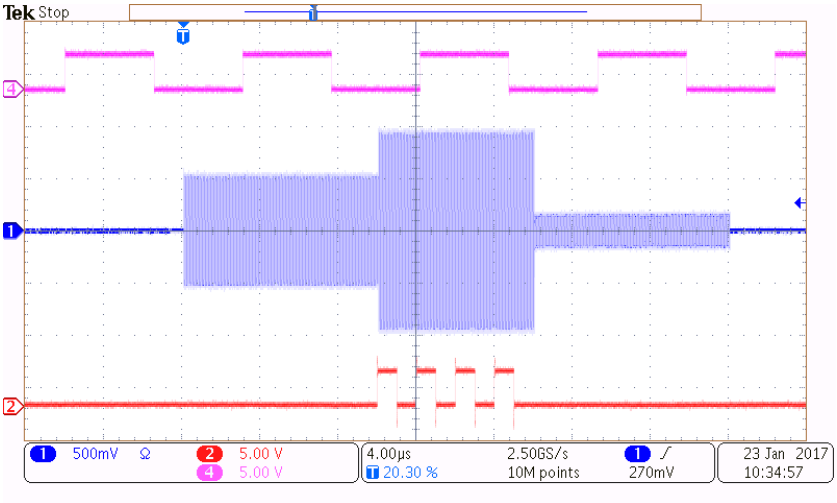

Figure 8.4: Waiting for a trigger causes the entry to be repeated until a falling edge in the trigger (magenta) is detected. The digital output (red) toggles several times, showing the repetition.

Once a table is armed with the TABLE,ARM command it will automatically begin executing upon a falling external trigger on the DE15

input. Starting the table execution from an external trigger therefore does not require the TRIG flag on the first table entry. If a trigger is not received but an instruction is waiting for a trigger, the TABLE,STOP or OFF command should be used to abort operation.

8.5 Upload and download

The host software mogrf provides convenient functionality that enables upload and download of tables in both binary and CSV format. This simplifies handling of tables, and allows simple generation of sequences from scripting languages like matlab or python, and the human-readable structure simplifies trouble-shooting.

Binary format enables direct upload of table data into non-volatile memory and is provided for fast back-up purposes. **The internal binary representation of tables is likely to change between minor firmware revisions** due to the implementation of new functionality that makes old binary tables incompatible with new firmware. It is strongly recommended that all tables be stored in ASCII (human-readable) format for long-term storage. An example of this format is shown below.

```
100 MHz, -5 dBm,  0 deg, 10us
150 MHz,  0 dBm, 90 deg, 2us, IODH
 80 MHz,  5 dBm,  0 deg, 1us, TRIG
100 MHz,  0 dBm,  0 deg, 5us
```

Listing 8.5: Example of table mode CSV file for use with mogrf.

A distinction should be made between *CSV tables* and *scripts*. In this context, CSV tables contain only the table instructions directly, whereas scripts may contain arbitrary sequences of command for the device. Listing 8.5 demonstrates the format of a CSV table. The intention is that such files are more easily generated by scripting languages (such as matlab or python).

```
MODE,1,TSB
TABLE,CLEAR,1
TABLE,ENTRIES,1,4
```

```
TABLE,ENTRY,1,1,100MHz,-5dBm,0deg,10us
TABLE,ENTRY,1,2,150MHz,0dBm,90deg,2us,IODH
TABLE,ENTRY,1,3,80MHz,5dBm,0deg,1us,TRIG
TABLE,ENTRY,1,4,100MHz,0dBm,0deg,5us
```

Listing 8.6: Script equivalent to Listing 8.5.

8.6 Re-arm and restart

The FPGA can be instructed to automatically re-arm the table after a successful execution using the `TABLE,REARM` command. This automatically prepares the table for execution from either hardware or software trigger once execution has finished.

Furthermore, the table can be repeated continuously by enabling the `TABLE,RESTART` option. This will cause the table to immediately re-execute after it has been rearmed, although a hardware or software trigger must be provided to begin the first execution.

Typically, the time between a table completing and subsequently restarting under FPGA control is $\sim 5\mu s$, depending on the length of the table and any synchronisation features being used. Enabling the `NORESET` option will prevent resetting the DDS phase accumulator between executions and reduces the delay. Note that unless the `NORESET` option is enabled, the output will be switched off during the phase accumulator reset ($\sim 2\mu s$).

If this delay is unacceptable, an alternative is to specify a loop back to the beginning of the sequence using the `TABLE,LOOP` command which will repeat the table with no delay. As the last table entry cannot be a loop, a dummy instruction needs to be appended, as in the example below. This table will terminate after 4096 executions; an alternative is to specify to repeat until an external trigger that is never provided, which will loop forever.

```
# ... Instructions that define the CH1 table ...
# add a LOOP from the most recent instruction (−1) back to the start (1)
TABLE,LOOP,1,-1,1,4095
# last three instructions cannot be a loop −− add dummy instructions
TABLE,APPEND,1,100MHz,-30dBm,0,1us
TABLE,APPEND,1,100MHz,-30dBm,0,1us
TABLE,APPEND,1,100MHz,-30dBm,0,1us
```

Listing 8.7: Demonstration of restarting CH1 table using a loop.

8.7 Linear ramps

It is often desirable to linearly ramp a parameter without having to specify the individual table entries manually. The `TABLE,RAMP` command provides a convenient way to generate such ramps.

TABLE,RAMP `TABLE,RAMP,ch,param,start,stop,duration,count`

Appends table entries to the channel `ch` that linearly ramp `param` from `start` to `stop` in `count` steps, with each step lasting for the given `duration`. `param` is one of `FREQ`, `AMPL` or `PHAS`. The `start` and `stop` values can be specified with real units as appropriate for `param`. The values of the other parameters are taken from the last entry in the table. In simple table mode, `count` entries are generated, whereas in advanced table mode only 3 entries are required.

For example, the command `TABLE,RAMP,1,FREQ,80,100,100us,2000` in TSB mode will generate 2000 table entries corresponding to a frequency ramp that sweeps the RF frequency from 80 MHz to 100 MHz in a total time of 200 ms (each step being 10 kHz and taking 100 us).

The command can also be used to specify piecewise-linear envelopes using the `AMPL` parameter, such as in Listing 8.8 resulting in the RF output in Figure 8.5.

```
MODE,1,TSB
# clear the table
TABLE,CLEAR,1
# define initial conditions (i.e. freq and phase)
TABLE,APPEND,1,80MHz,-30dBm,0deg,1us
# ramp power from -30dBm to 0dBm in 100us, then down again
TABLE,RAMP,1,POW,-30,0,1us,100
TABLE,RAMP,1,POW,0,-30,1us,100
# should now have 201 entries
TABLE,ENTRIES,1
# arm the table
TABLE,ARM,1
```

Listing 8.8: Example using `TABLE,RAMP` to create an envelope.

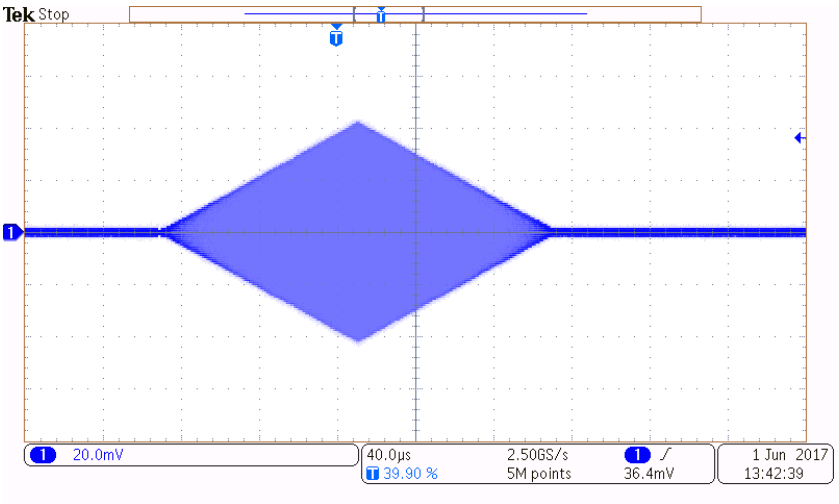

Figure 8.5: Output generated by Listing 8.8 showing linear ramps in amplitude.

The ramp functionality can also be applied to frequency or phase. Listing 8.9 shows how multiple `TABLE,RAMP` commands can be used to execute a sequence of slow frequency ramps that can be watched on a spectrum analyser. A graph of the corresponding frequency of the RF over time is shown in Figure 8.6.

```
MODE,1,TSB
TABLE,CLEAR,1
# set initial conditions
TABLE,APPEND,1,80MHz,0dBm,0,1us
# define first ramp
TABLE,RAMP,1,FREQ,70,80,1m,1000
# append a pause
TABLE,APPEND,1,80,-5dbm,0,1s
# append second and third ramps
TABLE,RAMP,1,FREQ,80,75,5m,200
TABLE,RAMP,1,FREQ,75,85,2m,500
```

Listing 8.9: Example demonstrating use of `TABLE,RAMP` to create multiple frequency ramps.

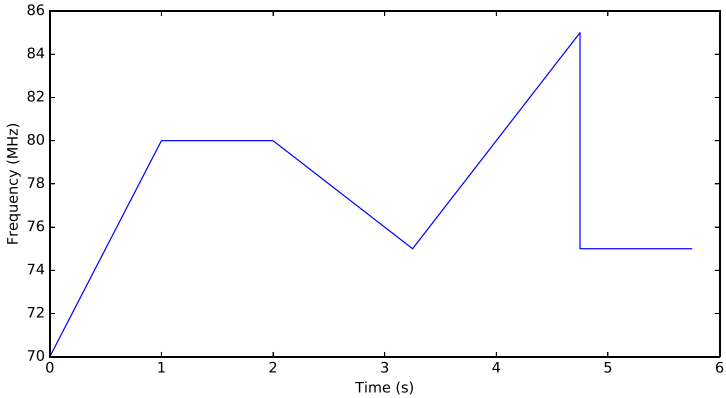

Figure 8.6: Frequency ramps achieved by chaining together `RAMPFREQ` commands in TSB mode.

8.8 Synchronous table execution

When operating both channels in table mode, it is possible to synchronise the two channels. This synchronisation occurs both at FPGA level (so that the table entries are stepped through simultaneously), and at the DDS level (so that the RF generated by the two channels remain in phase).

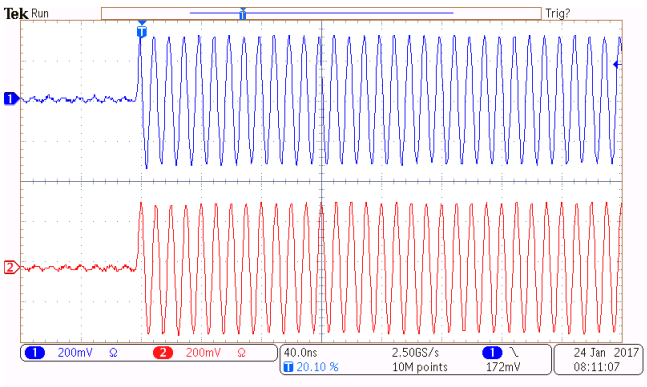

Figure 8.7: Example showing the two RF outputs when executing the same table synchronously on both channels.

This feature is activated by issuing the command `TABLE,SYNC,1`. This configures CH1 as the master and CH2 as the slave, such that the CH2 DDS derives its clock from the CH1 DDS for phase stability. Once enabled, CH2 cannot be started independently of CH1, and CH1 cannot be started unless CH2 has been armed. For this reason, it is recommended to activate the auto-rearm feature on CH2 using the command `TABLE,REARM,2,1`.

The feature must be deactivated using `TABLE,SYNC,0` to run the tables independently, or to take one channel out of table mode. The FPGA is also slightly slower ($\sim$ 1µs) to respond to a start instruction (software or hardware trigger) in synchronous mode, as it must prepare the each channel's DDS for execution.

It is also possible to synchronise the DE15 triggers received by the two tables. The two trigger inputs on the DE15 connector can be hardwired together, but the FPGA can also be instructed to trigger both channels from the same input. This functionality can be controlled by the `TABLE,TRIGSYNC` command.

This does not apply to the high-speed bus, as both tables can be instructed to use the same input as a trigger regardless.

9. Advanced table mode (XRF)

The XRF has access to an *advanced table mode* with increased functionality, flexibility and significantly faster execution than normal (SIF) table mode. Due to the added complexity, it is strongly recommended that users be familiar with simple table mode before reading this chapter.

9.1 Operational principle

Advanced table (TPA) mode performs table execution via the parallel DDS interface. This allows one parameter to be updated at the maximum supported rate, currently 16 ns per instruction. Due to limitations of the AD9910 DDS devices, only one parameter can be modified via PIF, and the other parameters must be updated at the slower rate supported by the serial interface (SIF). Furthermore neither external modulation nor PID can be used in TPA mode.

There are therefore two kinds of instructions in advanced mode: parallel and serial. Parallel instructions allow a single parameter to be updated rapidly, whereas serial instructions are preloaded into the serial interface and then activated in a subsequent command. This load/activate cycle allows changes to be made to the parallel parameter without having to wait for the serial load to complete.

Advanced mode also supports a more extensive selection of loop options, including the ability to increment the parallel parameter for piecewise linear interpolation. This is advantageous for generating smooth envelopes with only a few table instructions.

The FPGA supports I/O at the full update rate, but the rise-time (particularly on the DE15 connector) must be taken into account. The MOGLabs B3120 break-out board is recommended when using the high-speed bus as it has matched track lengths to minimise relative delay between the signals.

It should also be noted that the while the DDS outputs can be synchronised, the rf components following the DDS introduce a small frequency-dependent propagation delay. However, this delay is fixed for a given frequency, and can be calibrated in applications where it is important. Such applications likely need calibration anyway, due to propagation delay introduced by cables and other external components.

9.2 Defining table entries

There are two forms of syntax for defining table entries in advanced mode, corresponding to parallel or serial instructions. Before populating the table, the parallel parameter needs to be set using the `TABLE,XPARAM` command, after which entries can be set for that parameter (syntax 1). A second format is provided for loading serial commands, which is identical to simple table mode (syntax 2).

TABLE,XPARAM `TABLE,XPARAM,ch,param,[fmgain]`
Sets which parameter is to be controlled on the parallel interface. Must be called before the table is populated with entries. The parameter is one of `FREQ`, `PHAS`, `POW` or `AMPL`. The parameters `POW` and `AMPL` are synonyms in this mode. Subsequent PIF table entries can then modify this parameter. When entering `FREQ` mode, the `fmgain` must also be specified (see §9.7).

TABLE,ENTRY(1) `TABLE,ENTRY,ch,num,param,value,duration,flags`
Set the specified table entry to change `param` to the specified `value`, while other parameters remain unchanged. During normal use, `param` should match the parameter set by `TABLE,XPARAM`, but some special instructions are available (§9.8). The `duration` of the entry is rounded to the nearest multiple of 16 ns, and the duration 0x1 can be used to set the minimum possible duration. The same `flags` are supported as in simple table mode (including `TRIG` and `IOxy`), with some extra options as explained in this chapter.

The listing below demonstrates how to set up advanced table mode to control the envelope (amplitude) of the RF signal,

```
# enter fast table mode
MODE,1,TPA
# clear any table entries or settings
TABLE,CLEAR,1
# set amplitude (power) as the parallel parameter
TABLE,XPARAM,1,POW
# define a table
TABLE,APPEND,1,POW,5dbm,16ns
TABLE,APPEND,1,POW,10dbm,50ns
TABLE,APPEND,1,POW,0dbm,16ns
TABLE,APPEND,1,POW,-40dbm,16ns
```

Listing 9.1: Parallel instruction example in advanced table mode.

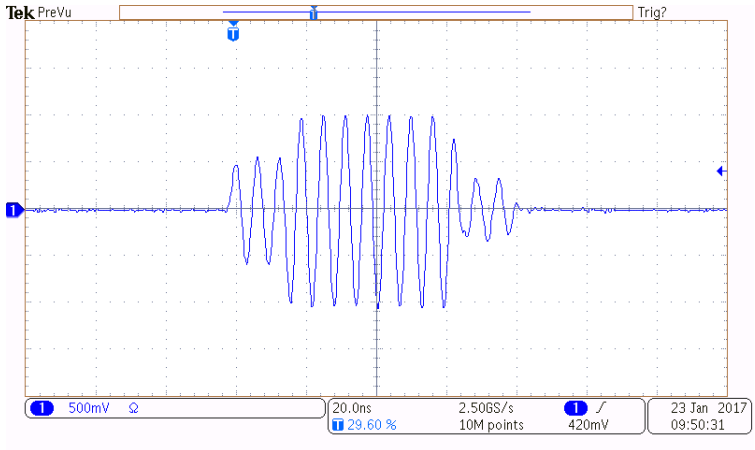

Figure 9.1: Output generated by Listing 9.1.

Parallel-mode instructions also provide functionality for piecewise linear-interpolations, as described in §9.6. This enables the creation of smooth ramps without requiring a large number of instructions. The functions `TABLE,APPEND` and `TABLE,INSERT` can also be used, supporting either syntax.

It is anticipated that many applications will only seek to control a single parameter, in which case only the parallel interface need be used. However, a second command format is provided to simultaneously set all three parameters using the serial interface.

TABLE,ENTRY(2) `TABLE,ENTRY,ch,num,freq,pow,phase,duration,flags`

Defines a serial (SIF) instruction that updates all parameters using the same syntax as simple table mode (§8.2), but the instruction does not take effect immediately. The instructions are queued for load over the serial interface and must be activated by a subsequent table entry using the `UPD` flag. This ensures that the output only changes in accordance with a table instruction.

The time between issuing the serial instruction and the update flag must be at least 960ns otherwise the DDS does not have time to load the instructions. However, any flags specified in the instruction (such as digital output) will occur immediately.

The example below demonstrates how to queue and then activate a serial instruction in advanced table mode.

```
# define a SIF entry using simple mode syntax
TABLE,APPEND,1,100MHz,5dbm,0,1us
# trigger the update to take effect
TABLE,APPEND,1,POW,0dbm,0x1,UPD
```

Listing 9.2: Serial instruction example in advanced table mode.

This may appear complicated, however it makes it possible to execute instructions *during* a serial load. For example, when making a chirped pulse, the amplitude can still be changing on the parallel bus while the next frequency is being loaded on the serial bus.

```
# set up the table
TABLE,CLEAR,1
TABLE,XPARAM,1,POW
# set the initial conditions
TABLE,APPEND,1,120MHz,-5dbm,0,1us
TABLE,APPEND,1,POW,-5dbm,0x1,UPD
# start loading next serial instruction
TABLE,APPEND,1,40MHz,0dbm,0,0x1
```

```
# some other instructions while the SIF loads
TABLE,APPEND,1,POW,-5dbm,320ns
TABLE,APPEND,1,POW,-10dbm,320ns
TABLE,APPEND,1,POW,-5dbm,320ns
# trigger the serial instruction
TABLE,APPEND,1,POW,5dbm,200ns,UPD
# another parallel instruction at new frequency
TABLE,APPEND,1,POW,-5dbm,100ns
# final instruction to power down
TABLE,APPEND,1,POW,0x0,0x1
```

Listing 9.3: Demonstration of parallel instructions during a SIF load

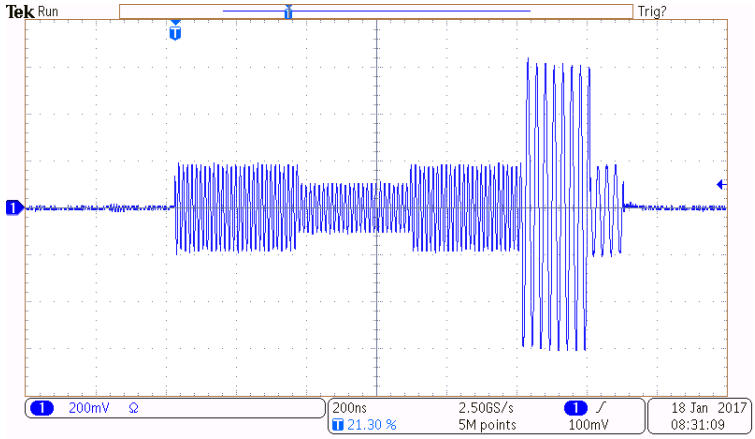

Figure 9.2: Output generated by Listing 9.3

Note: The value corresponding to the parallel parameter is ignored when loading a serial update and must be set separately. For example, in Listing 9.2, the output power will be 0 dBm (not 5 dBm).

9.3 Initial and final states

As in simple table mode, when the TABLE,ARM command is used to ready the table for execution, the output is enabled and amplifiers switched on (if present). The state of the RF after arming is therefore the same as the last table instruction that was run.

In simple table mode, the first instruction specified all three parameters, so the state at each step is well-defined and repeatable. However, in parallel mode only one parameter is specified and the other two may be undefined. It is strongly recommended that the start and end of every table reset all parameters to known values, as demonstrated in the listing below.

```
TABLE,CLEAR,1
TABLE,XPARAM,1,POW
# create table entry specifying a known start condition
TABLE,APPEND,1,80MHz,5dbm,45deg,976ns,OFF
# activate the update
TABLE,APPEND,1,POW,5dbm,16ns,UPD
# —— 1us has elapsed up to here ——
#
# ... other table mode entries ...
#
# add a final entry to reset rf state
TABLE,APPEND,80MHz,5dbm,45deg,1us
# activate the update, and turn off the rf
TABLE,APPEND,1,POW,-30dbm,UPD,OFF
```

Listing 9.4: Setting initial and final states in parallel mode

The RF state at the last instruction is held after the table has finished executing, which may involve the RF being on or off depending on the desired application. If it is desired for the RF to be switched off, then the amplitude should be set to zero on the final instruction instead of using the OFF flag, as the RF output will be switched on immediately following the next table rearm.

It is also possible to manually use the commands FREQ, POW and PHASE in advanced table mode to set the initial conditions *when the table is not running*. Once the table has finished executing, the last

instruction will remain unless subsequently overwritten by one of these commands. It is therefore strongly recommended to specify the initial and final states.

9.4 Counters

XRF devices are capable of controlling the digital input counters in advanced table mode, and using the counters as a loop condition. This assists in experiment automation, whereby the execution can be paused until a critical count is received. For example, the experiment can be paused until sufficient pulses are recorded from an avalanche photodetector, indicating the experiment is ready to proceed.

Counters can be controlled through the following advanced table mode flags, appended to any table entry. The pin must be enabled as a counter (§7.7) before arming the table, and each command takes effect at the *start* of each table instruction.

CxS[TART] Reset counter `x` to zero and then begin counting

CxP[AUSE] Disable counter `x` and stop accumulating counts

CxR[ESUME] Re-enable counter `x` and resume accumulating counts

CA[LL] Stop and reset all counters

The parameter `x` in each of these flags is the counter pin, such as `A3` for pin 3 of bank A, or `D` for OFFn on the DE15 connector.

Unlike basic mode, where the counter begins accumulating counts as soon as it is enabled, the counter in table mode only accumulates counts after a `CxSTART` or `CxRESUME` flag. This allows precise control of the counting interval.

The syntax for looping until a threshold counter value is reached is

```
LOOP,ch,source,dest,COUNT,IOx,N
```

where `x` is the counter pin and `N` is the count threshold, which is an integer in the range [1,65535].

The example below shows how to configure a counter, control it with table flags, and use it as a loop condition.

```
# configure the counter
EXTIO,MODE,1,HSB,READ # set bank A into read mode
EXTIO,COUNTER,1,HS1,FALLING # set pin A1 to count falling edges
# create the table
MODE,1,TPA
TABLE,CLEAR,1
TABLE,XPARAM,1,POW
TABLE,APPEND,1,POW,-5dBm,100ns,CA1S # start the counter
TABLE,APPEND,1,POW,0dBm,16ns # accumulate counts
TABLE,LOOP,1,-1,0,COUNT,IOA1,1000 # loop until 1000 counts
TABLE,APPEND,1,POW,-30dBm,16ns,CA1P # pause the counter
# run the table
TABLE,START,1
# wait for completion then read back the count
SLEEP,10
TABLE,STATUS,1
EXTIO,COUNTER,1,HS1,READ
```

Listing 9.5: Demonstration of configuring HSA1 as a counter and controlling it in advanced table mode.

9.5 Loops and triggers

Loops and triggers can be specified in advanced table mode using the TABLE,LOOP command similarly to simple table mode (§8.4). However, some different restrictions apply to loops in advanced table mode:

- Nested loops are not supported
- Neither the first nor last table instruction can be a loop
- The TABLE,LOOP command cannot be applied to a table entry that uses the EXTRAPOLATE feature (§9.6)
- A loop cannot jump by more than 1024 instructions
- The loop condition can specify up to 65535 repeats
- Sequential instructions may contain loops, unlike TSB mode where loops need to be separated by four instructions.

9.6 Linear ramps using extrapolation

One of the powerful features provided in advanced table mode is the ability to specify linear ramps in parallel mode, which reduces the number of instructions necessary to produce smooth piecewise-linear ramps. The FPGA performs the extrapolation and updates the DDS as required. This means that a 1000-point ramp can be implemented as a single table instruction instead of requiring 1000 different individual instructions.

Note: This section describes the low-level implementation of parameter extrapolation. It is strongly recommended to use the high-level helper functionality provided by the TABLE,RAMP command, which makes use of the extrapolate feature in TPA mode and provides a simpler user interface.

The extrapolation feature is activated by specifying the REPn flag in a table entry, using the syntax shown below.

TABLE,ENTRY(3) TABLE,ENTRY,ch,num,param,delta,duration,REPn,flags

Sets the associated table entry to EXTRAPOLATE, adding delta to the parameter param on each execution. The instruction is repeated *n* times as specified by REPn.

The delta argument should be specified in hexadecimal when extrapolating power/amplitude, as in this example. Hexadecimal values can be obtained from the output of the POW command for each end-point. Subtract the two and divide by the number of steps to obtain the delta. ***Do not specify delta in dBm or W.*** Conversely, the TABLE,RAMP function does accept values in real-world units.

Warning: Bounds checking is not performed in extrapolation mode; it is up to the user to ensure that parameters do not go out of bounds. In particular, the power LIMIT is not obeyed, and amplitude must not go negative. It is ***strongly recommended*** to check the output on an oscilloscope through an RF attenuator before connecting to a device that could be damaged by maximum output power.

Listing 9.6 demonstrates how to linearly ramp the RF power in 100 steps using a single instruction instead of 100 individual instructions by using the REPn notation. The result is shown in Figure 9.3. Listing 9.7 demonstrates an equivalent formulation based on the TABLE,RAMP command, which provides the convenience of specifying the power in real units (dBm) instead of hexadecimal increments.

```
MODE,1,TPA
TABLE,CLEAR,1
# fast parameter is power
TABLE,XPARAM,1,POW
# set power to OFF
TABLE,APPEND,1,POW,0x0,0x1
# linear ramp up (100 steps)
TABLE,APPEND,1,POW,0x10,0x1,REP100
# linear ramp down (100 steps)
TABLE,APPEND,1,POW,-0x10,0x1,REP100
# reset power to OFF
TABLE,APPEND,1,POW,0x0,0x1
# loop the ramp 2 more times
TABLE,LOOP,1,-1,1,2
# loop the ramp 2 more times
TABLE,APPEND,1,POW,0x0,0x1
```

Listing 9.6: Creation of a triangle wave envelope in advanced table mode, using only five table entries.

```
TABLE,CLEAR,1
TABLE,XPARAM,1,POW
# define a ramp from off (0x0) to 0dBm
TABLE,RAMP,1,POW,0x0,0dBm,16ns,100
# append the reverse of the ramp
TABLE,RAMP,1,POW,0dBm,0x0,16ns,100
# loop back to the beginning twice more
TABLE,LOOP,1,-1,1,2
# last entry cannot be a loop
TABLE,APPEND,1,POW,0x0,16ns
```

Listing 9.7: Alternate specification of triangle wave using TABLE,RAMP.

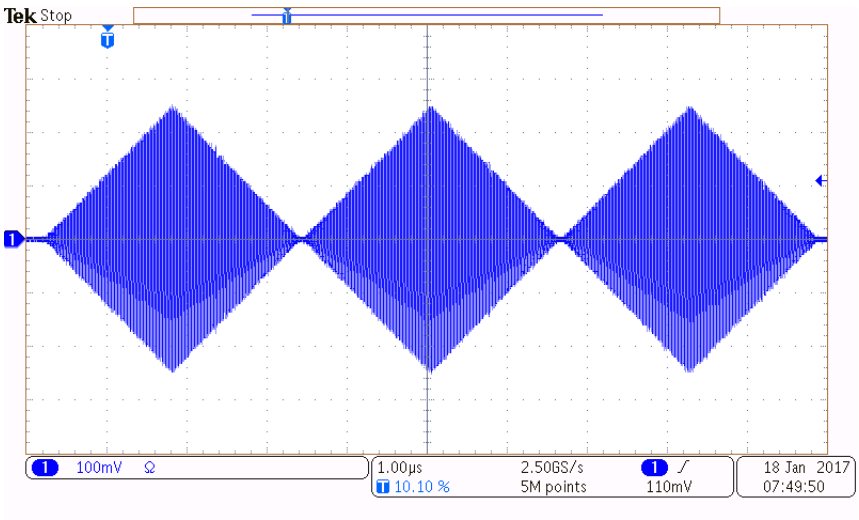

Figure 9.3: Output generated by Listing 9.6.

9.7 Frequency gain

The fast (parallel) interface to the DDS is 16-bit, which is sufficient to fully define the amplitude (14-bits) and phase (16-bits) but not frequency (32-bits). This is an inherent restriction of the DDS, so in order to specify frequency on the parallel interface, a frequency gain is applied by the DDS when receiving values, which amounts to a *bit-shift* of the incoming value (Figure 9.4).

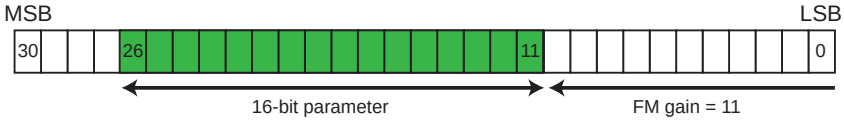

Figure 9.4: Visualisation of how frequency gain allows the 32-bit frequency word to be modified by the 16-bit parallel bus. In this example, the gain is 11 and only the indicated bits can be modified on the parallel bus.

This has the effect of restricting the smallest and largest changes that can be made in parallel frequency mode through the associated frequency discretisation (step size) and value range. If large changes to frequency are required, high gain should be used. If high resolution is required, small gain should be used. The outcome of different gain settings are shown in Table 9.1.

Gain	Step	Max	Gain	Step	Max
0	0.23 Hz	7.54 kHz	8	59.6 Hz	1.95 MHz
1	0.47 Hz	15.3 kHz	9	119 Hz	3.91 MHz
2	0.93 Hz	30.5 kHz	10	238 Hz	7.81 MHz
3	1.86 Hz	61.0 kHz	11	477 Hz	15.6 MHz
4	3.73 Hz	122 kHz	12	953 Hz	31.2 MHz
5	7.45 Hz	244 kHz	13	1.91 kHz	62.5 MHz
6	14.9 Hz	488 kHz	14	3.81 kHz	125 MHz
7	29.8 Hz	977 kHz	15	7.63 kHz	250 MHz

Table 9.1: Effect of frequency gain with default clock configuration.

The gain is specified initially using the `TABLE,XPARAM` command,

```
TABLE,XPARAM,ch,FREQ,gain
```

where `ch` is the channel and `gain` is the desired gain (0–15).

The range of frequencies that can be achieved in advanced table mode is $f_0 \pm df_{\text{max}}$ where f_0 is the center frequency set with the `FREQ` command, and df_{max} is the value in the above table corresponding to the gain. To assist with understanding these ranges, the `TABLE,XPARAM,ch,FREQ` command will output information about what combinations are possible for the *current* set of parameters.

Note that the frequency gain can presently only be set before the table is populated, and cannot be changed mid-sequence. It is therefore important to ensure that the desired frequency range fits entirely within the accessible range.

For example, to vary the frequency between 70 MHz and 80 MHz with maximum dynamic range, the frequency should be set to 75 MHz using the FREQ command, and the frequency gain set to 10. However, to vary between 65 MHz and 85 MHz, the gain must be increased to 11.

9.8 Other instruction parameters

The following parameters can also be specified in parallel table entries, for special behaviours, as listed below. Instructions that use the serial interface must be followed with a subsequent instruction containing the UPD flag to activate them.

HOLD Do not change the output (also known as a "nop" or "no operation"). Intended to perform I/O operations or trigger a serial update (using the UPD flag) without changing the RF.

REGx Write a 32-bit value directly to register *x* of the DDS using the serial interface. Provided for advanced functionality in consultation with the AD9910 datasheet. Requires subsequent entry with UPD parameter to take effect.

9.9 Additional examples

9.9.1 Gaussian envelope

The following example demonstrates creation of a short (300 ns) Gaussian pulse by specifying the output power at the maximum update rate (instruction duration $0x1 = 16$ ns). The resulting output waveform is shown in Figure 9.5.

```
# set up table mode
MODE,1,TPA
TABLE,CLEAR,1
TABLE,XPARAM,1,POW
# define points along Gaussian
TABLE,APPEND,1,POW,-24.59dBm,0x1
TABLE,APPEND,1,POW,-22.08dBm,0x1
TABLE,APPEND,1,POW,-18.88dBm,0x1
TABLE,APPEND,1,POW,-14.99dBm,0x1
TABLE,APPEND,1,POW,-10.53dBm,0x1
TABLE,APPEND,1,POW,-5.74dBm,0x1
TABLE,APPEND,1,POW,-0.95dBm,0x1
TABLE,APPEND,1,POW, 3.41dBm,0x1
TABLE,APPEND,1,POW, 6.92dBm,0x1
TABLE,APPEND,1,POW, 9.21dBm,0x1
TABLE,APPEND,1,POW,10.00dBm,0x1
TABLE,APPEND,1,POW, 9.21dBm,0x1
TABLE,APPEND,1,POW, 6.92dBm,0x1
TABLE,APPEND,1,POW, 3.41dBm,0x1
TABLE,APPEND,1,POW,-0.95dBm,0x1
TABLE,APPEND,1,POW,-5.74dBm,0x1
TABLE,APPEND,1,POW,-10.53dBm,0x1
TABLE,APPEND,1,POW,-14.99dBm,0x1
TABLE,APPEND,1,POW,-18.88dBm,0x1
TABLE,APPEND,1,POW,-22.08dBm,0x1
TABLE,APPEND,1,POW,-24.59dBm,0x1
TABLE,ARM,1
```

Listing 9.8: Rapid Gaussian pulse (340ns) in fast table mode

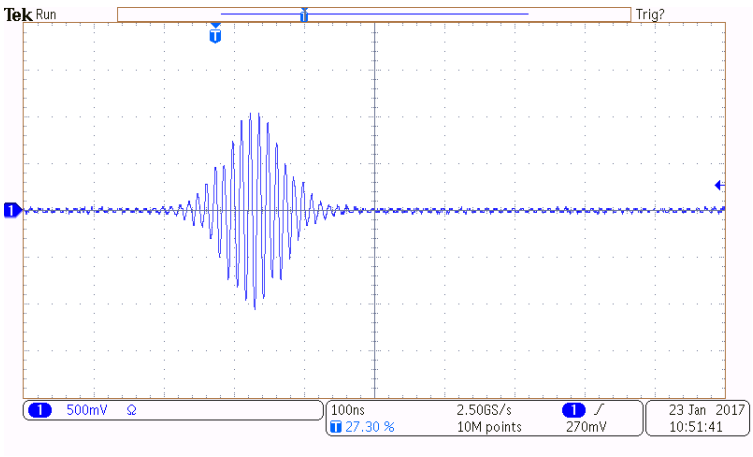

Figure 9.5: Example of a short Gaussian pulse.

9.9.2 Back-to-back pulses with different frequency

This example demonstrates how to generate two back-to-back 1µs pulses with different frequencies by loading the second frequency during the first pulse. The EXTRAPOLATE feature is used to smooth the envelope, and the LOOP feature is used to generate the second pulse using the instructions for the first. The amplitude steps are specified in hexadecimal so that the EXTRAPOLATE feature can be used.

```
MODE,1,TPA
TABLE,CLEAR,1
# serial load and trigger first frequency
TABLE,APPEND,1,40MHz,0dbm,0deg,1us
TABLE,APPEND,1,HOLD,0x1,UPD
# begin serial load of second frequency
TABLE,APPEND,1,120MHz,0dbm,0deg,0x1
# define smooth pulse envelope using EXTRAPOLATE
TABLE,APPEND,1,POW,0x001f,0x1,REP3
TABLE,APPEND,1,POW,0x006a,0x1,REP3
TABLE,APPEND,1,POW,0x00d1,0x1,REP3
TABLE,APPEND,1,POW,0x0153,0x1,REP3
TABLE,APPEND,1,POW,0x01db,0x1,REP3
TABLE,APPEND,1,POW,0x0244,0x1,REP3
```

```
TABLE,APPEND,1,POW,0x0264,0x1,REP3
TABLE,APPEND,1,POW,0x0222,0x1,REP3
TABLE,APPEND,1,POW,0x017b,0x1,REP3
TABLE,APPEND,1,POW,0x0088,0x1,REP3
TABLE,APPEND,1,POW,-0x088,0x1,REP3
TABLE,APPEND,1,POW,-0x17b,0x1,REP3
TABLE,APPEND,1,POW,-0x222,0x1,REP3
TABLE,APPEND,1,POW,-0x264,0x1,REP3
TABLE,APPEND,1,POW,-0x244,0x1,REP3
TABLE,APPEND,1,POW,-0x1db,0x1,REP3
TABLE,APPEND,1,POW,-0x153,0x1,REP3
TABLE,APPEND,1,POW,-0x0d1,0x1,REP3
TABLE,APPEND,1,POW,-0x06a,0x1,REP3
TABLE,APPEND,1,POW,-0x01f,0x1,REP3
# trigger the frequency change
TABLE,APPEND,1,HOLD,0x1,UPD
# loop back to create second pulse
TABLE,LOOP,1,-1,4,1
TABLE,APPEND,1,POW,0x0,0x1
```

Listing 9.9: Back-to-back shaped pulses with different frequencies

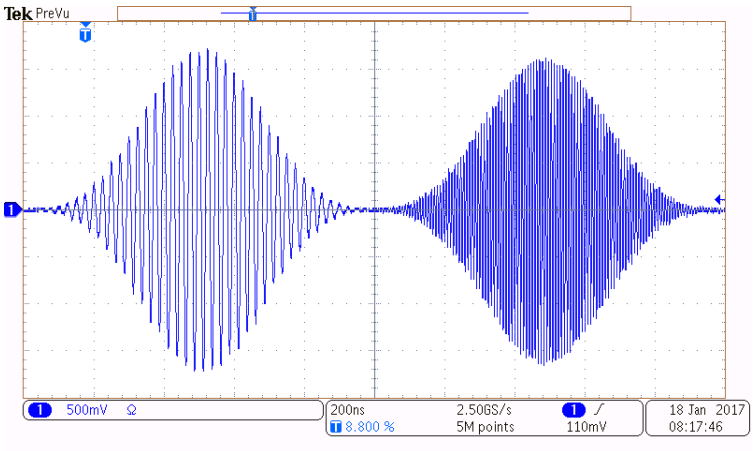

Figure 9.6: Two 1µs Gaussian pulses generated in advanced table mode with amplitude on the parallel bus. A serial instruction changes the frequency from 40 MHz to 120 MHz between the two pulses, allowing control of the frequency and phase of the second pulse.

9.9.3 Parallel frequency mode

The following example demonstrates control of the RF frequency with the parallel interface.

```
# set up table mode
MODE,1,TPA
TABLE,CLEAR,1
# use HSB for triggering
EXTIO,CTRL,1,HSB,AUTO
# change to FREQ mode and set the FM gain
TABLE,XPARAM,1,FREQ,15
# step through a number of frequencies
TABLE,APPEND,1,FREQ,20,0x5,IOA1H
TABLE,APPEND,1,FREQ,40,0x5,IOA1L
TABLE,APPEND,1,FREQ,80,0x5,IOA1H
TABLE,APPEND,1,FREQ,160,0x5,IOA1L
```

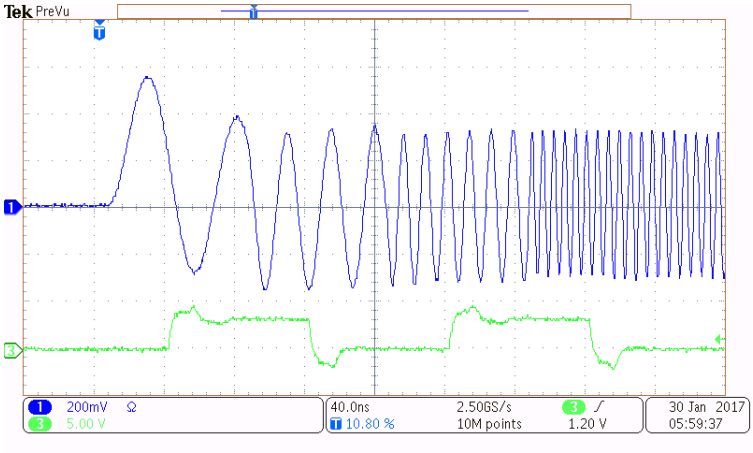

Figure 9.7: Example showing phase continuity with stepwise changes in frequency between 20 MHz and 400 MHz in advanced table mode.