Texas Instruments
C2000 MCU

DCL Reference Generator

User's Guide

Document date: 7 February 2020

Contents

1. Overview

The reference generator is a software module which generates static and dynamic signals intended as the reference input to a digital control loop. The generator is capable of producing various dynamic signals including sine waves, square waves, pulses, and triangular waves, all of which are superimposed on an adjustable static offset. The design includes amplitude and frequency modulation, and output clamping. The user interface is a set of eleven C functions which may be easily integrated into any C2000 software project.

The reference generator is part of the C2000 Digital Control Library (DCL), which is deployed in the C2000Ware. C2000Ware is a collection of libraries and other supporting software collateral for the C2000 device and may be freely downloaded at: www.ti.com/tool/c2000ware.

Although packaged with the Digital Control Library, users are not obliged to integrate the DCL into their code in order to use the reference generator. It is straightforward to add the reference generator to existing C code even if no other DCL features are needed. Like the DCL, the reference generator is intended for use in any system in which a C2000 device is used. The DCL may not be used with any other devices. Refer to the C2000Ware license agreement for further information.

This User's Guide covers the reference generator module supplied with version 3.4 of the C2000 Digital Control Library (DCL). It contains technical descriptions of the reference generator functions and how to use them. The document is supplemental to the DCL User's Guide which can be found in the same sub-directory of your DCL installation path.

This User's Guide is divided into five chapters. This opening chapter presents a general introduction to the reference generator module and its use. Chapter 2 presents a technical description of the module and should be read carefully by all users. Chapter 3 provides information on how to integrate the reference generator with existing code. Chapter 4 tabulates the function benchmarks, and finally chapter 5 provides detailed information of the reference generator functions and examples of their use.

The DCL contains both single and double precision versions of the reference generator. These are similar in all respects except that of data type. Double precision functions and filenames are distinguished by a '64' begin appended to both. For this reason, this document describes only the single precision module; once this is understood users will find it straightforward to work with the double precision version.

2. Technical Description

The DCL reference generator module is capable of producing up to three channels each made up of a dynamic waveform superimposed on a common static offset. Dynamic waveforms include a single phase sine, square, saw-tooth, triangular, or pulse output; and two or three phase sine waves.

The reference generator consists of five sub-modules: a static offset generator, a three phase waveform generator, a frequency modulator, an amplitude modulator, and an output clamp. The relationship between these sub-modules is set out below.

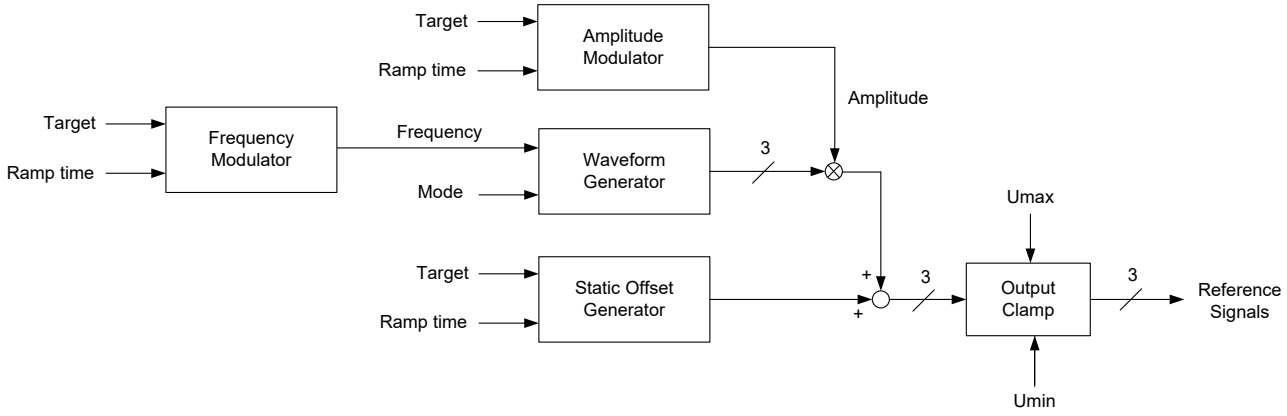

Figure 1. Reference generator architecture

2.1 Static Offset Generator

The static offset generator produces an adjustable offset onto which the dynamic component of the outputs is superimposed. The transition between two offsets is achieved using a linear ramp, the duration of which can be determined by the user.

The ramp is implemented by adding an increment to the offset each time the reference generator is run, the increment being computed when the ramp is loaded. In the floating point numerical format, resolution is lost as the size of the number increases, so adding a small number to a much larger number can result in the larger number not changing. For this reason there is an upper limit to the ramp transition time and a lower limit to the offset change which can be achieved. It is recommended that the reference generator module be used to generate outputs in the range ±1. This allows the computed ramp increment is checked against a lower limit (defined near the top of DCL_refgen.h) which is known to maintain accumulation up to the maximum accumulator value.

The function DCL_setRefgenRamp() allows the user to change both the static level and the transition time between level changes. When the user specifies a new target level, the generator outputs a linear ramp between the new and old levels in the specified time interval. Figure 2 shows an example of a static level change taking place over a specific time interval (tr).

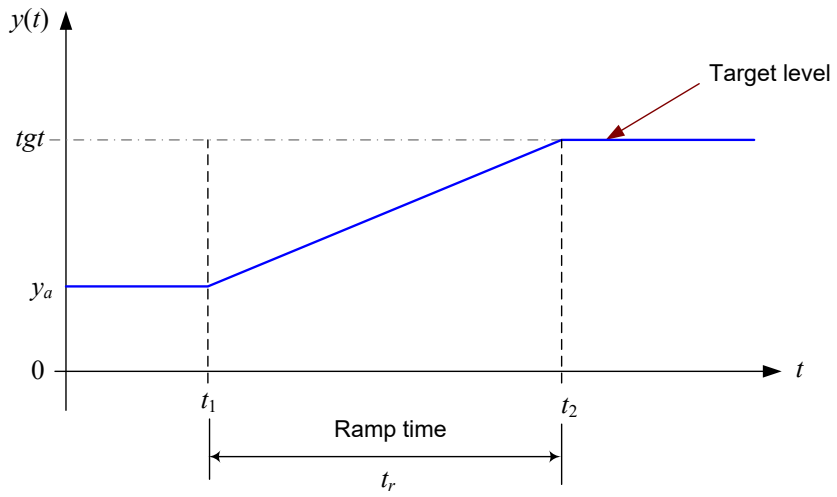

Figure 2. Static level ramp

The ramp interval is programmable in units of seconds, and the function uses the generator update rate held in the CSS sub-structure to compute the incremental change in level on each update during the adjustment interval. For this reason, the update rate must be loaded into the CSS sub-structure before the level change function is called. To force an immediate (step) change to a new offset level, the ramp interval should be set to zero.

2.2 Waveform Generator

The waveform generator produces the dynamic component of the reference (sine waves, square waves, and so on). The frequency of the dynamic component is determined by two factors: the frequency of the desired waveform, and the update rate of the generator. Each time the DCL_runRefgen() function is called, the REFGEN generator determines a normalized angle from which the next waveform point is computed. This is done by adding a fixed angular increment to the angle variable. The increment is computed in the DCL_setRefgenFreq() function. At very low frequencies or very high update rates, the increment may violate the smallest allowable value defined in DCL_REFGEN_MIN_INC. In this case, if error handling is enabled an error will be generated, or if error handling is disabled the angle increment will be clamped to DCL_REFGEN_MIN_INC. The maximum recommended frequency of the generated waveform is one quarter of the update rate.

Prior to using the waveform generator, the user must load the correct update period into the CSS sub-structure before calling any of the REFGEN functions. This can be done using the DCL_SET_SAMPLE_RATE macro (see the example code in the function descriptions section).

The waveform generator produces three outputs denoted phase A, phase B, and phase C. The generator can produce the following waveforms.

- sinusoidal
- square
- saw-tooth
- triangular
- pulse

- 2 phase sine
- 3 phase sine

The waveform type is selected by the DCL_setRefgenMode() function which takes as an argument an enumerated 16-bit integer denoting the mode. In all cases, the frequency and amplitude of the waveforms are freely adjustable by the user.

On TMU equipped devices, sinusoidal waveform generation modes make use of TMU C intrinsics to accelerate the run function. Devices which do not have a TMU must use RTS library support functions so will execute more slowly. None of the waveforms use look-up tables so the data memory footprint of the REFGEN module is very small.

The table below shows the operating modes available in the reference generator. The mode number is in the leftmost column and the enumerated mode name in the second column from the left. Double precision modes have a '64' appended to the 'REFGEN', for example 'REFGEN64_STATIC'.

Table 1. Reference Generator Operating Modes

No	Mode Name	Type	Phase A	Phase B	Phase C
0	REFGEN_OFF	No output	0	0	0
1	REFGEN_STATIC	Static offset only	Static	Static	Static
2	REFGEN_SINE	Sine wave	Sine	Static	Static
3	REFGEN_SQUARE	Square wave	Square	Static	Static
4	REFGEN_SAW	Saw tooth wave	Saw tooth	Static	Static
5	REFGEN_TRIANGLE	Triangle wave	Triangle	Static	Static
6	REFGEN_PULSE	Pulsed wave	Pulse	Static	Static
7	REFGEN_SINE2	2 phase sine wave	Sine	Cosine	Static
8	REFGEN_SINE3	3 phase sine wave	Sine	120 deg. sine	240 deg. sine

The same amplitude adjustment is applied to all phases prior to their addition to the static offset. Amplitude adjustment is made using the DCL_setRefgenAmpl() function.

2.2.1 REFGEN_OFF Mode

In this mode all three outputs are forced to zero. This is the default mode.

2.2.2 REFGEN_STATIC Mode

In this mode all outputs are controlled by the static reference generator only. The waveform generator is not used.

2.2.3 REFGEN_SINE Mode

In this mode phase A is a sinewave of configurable frequency and amplitude, while phases B & C are static outputs.

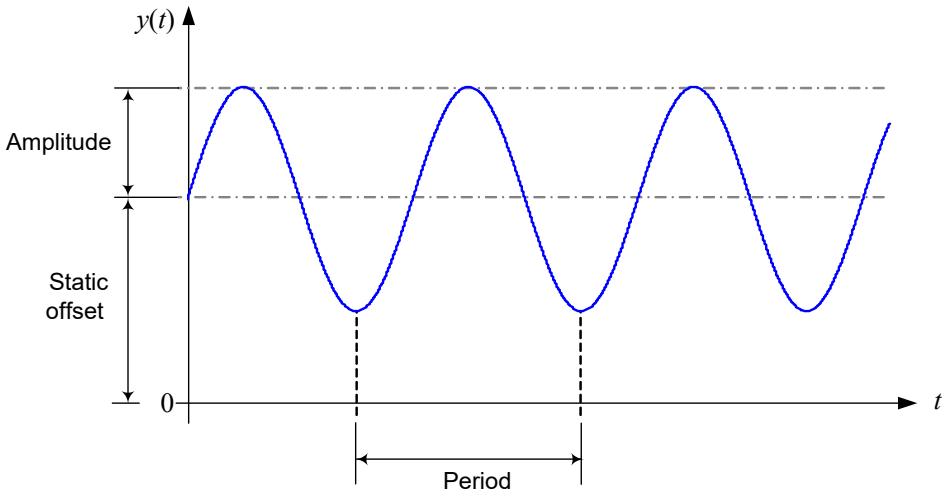

Figure 3. Sine wave mode

2.2.4 REFGEN_SQUARE Mode

In this mode phase A is a square wave of configurable frequency and amplitude, superimposed on the static generator output. Phases B & C are static outputs.

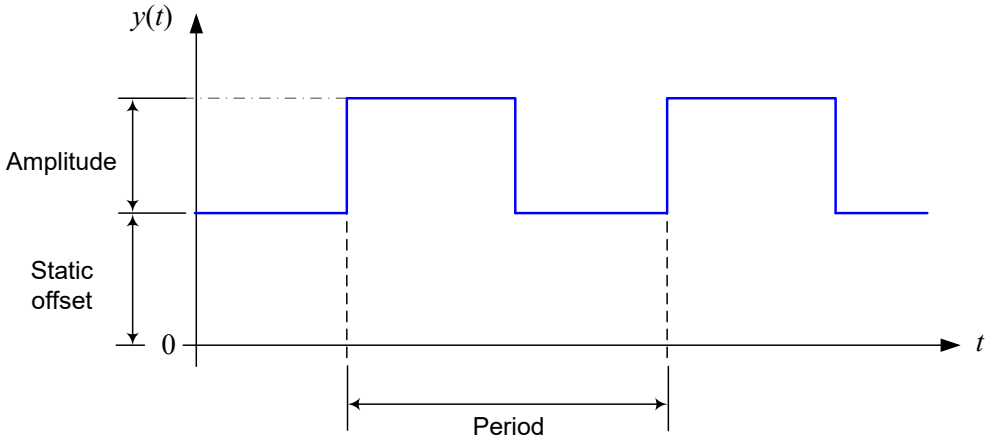

Figure 4. Square wave mode

2.2.5 REFGEN_SAW Mode

In this mode phase A is a saw-tooth wave of configurable frequency and amplitude superimposed on the output of the static generator. Phases B & C are static outputs. The saw-tooth output is zero when the angle is zero, and rises to the amplitude when the angle is 1.

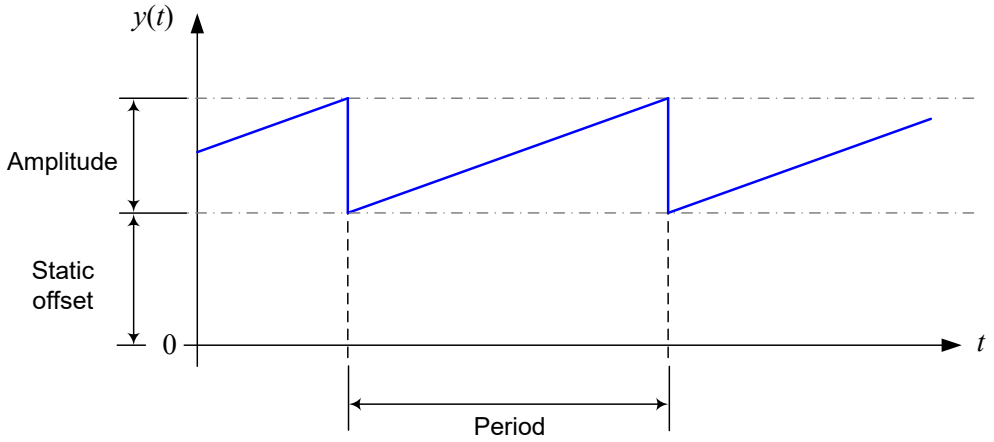

Figure 5. Saw tooth mode

2.2.6 REFGEN_TRIANGLE Mode

In this mode phase A is a triangle wave of configurable frequency and amplitude superimposed on the output of the static generator. Phases B & C are static outputs. The saw-tooth output is zero when the angle is zero, and increases linearly to the commanded amplitude when the angle is 0.5.

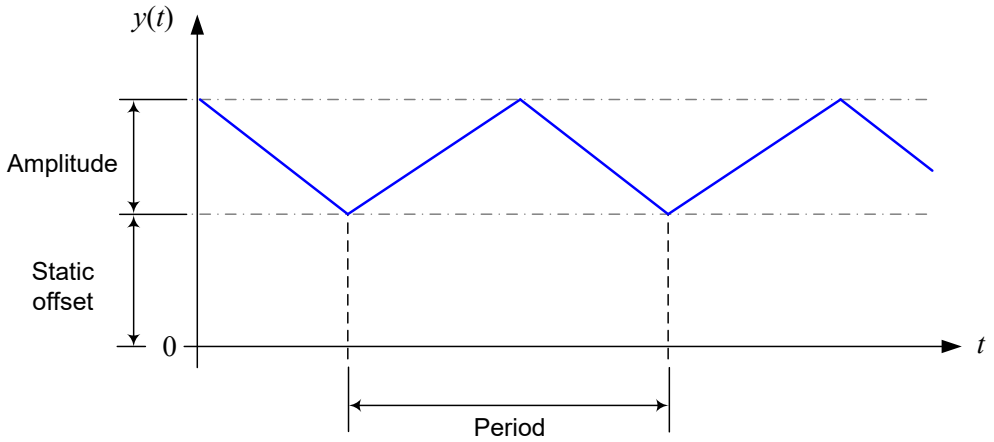

Figure 6. Triangle mode

2.2.7 REFGEN_PULSE Mode

In this mode phase A is a pulsed wave of configurable frequency and amplitude superimposed on the output of the static generator. Phases B & C are static outputs. The pulse width is controlled by the duty cycle setting in the DCL_REFGEN structure, which may be set using the DCL_setRefgenDuty() function. The pulse

output is one when the angle is below the normalized duty cycle, and zero when the angle is greater than the normalized duty cycle value.

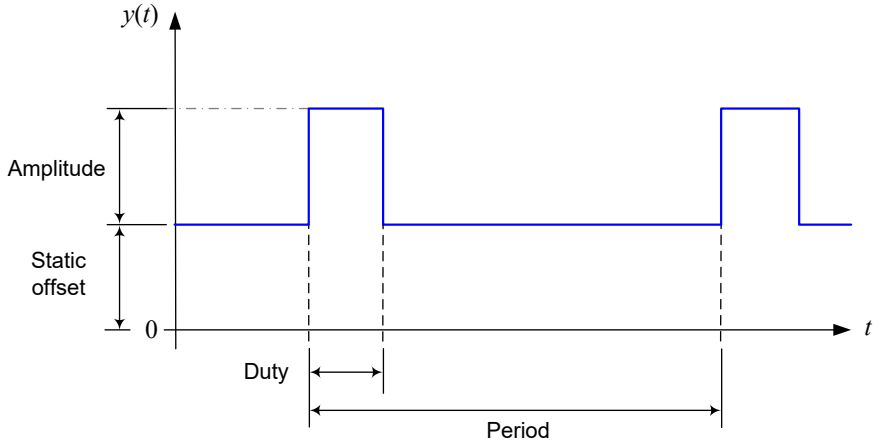

Figure 7. Pulse mode

2.2.8 REFGEN_2PHASE Mode

In this mode, phases A & B are two sine waves of similar frequency and amplitude superimposed on the static reference generator output. The phases are separated by 90 degrees, so that the outputs form a sine/cosine pair. Phase C is the static generator output.

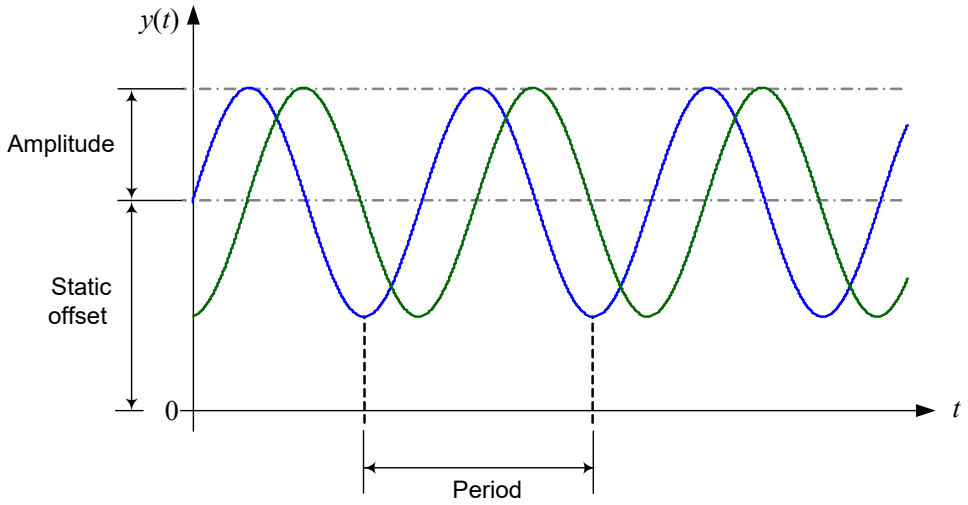

Figure 8. 2-phase sine mode

2.2.9 REFGEN_3PHASE Mode

In this mode, the three output phases are sine waves of similar frequency and amplitude superimposed on the static reference generator output. The angular separation of the phases is 120 degrees.

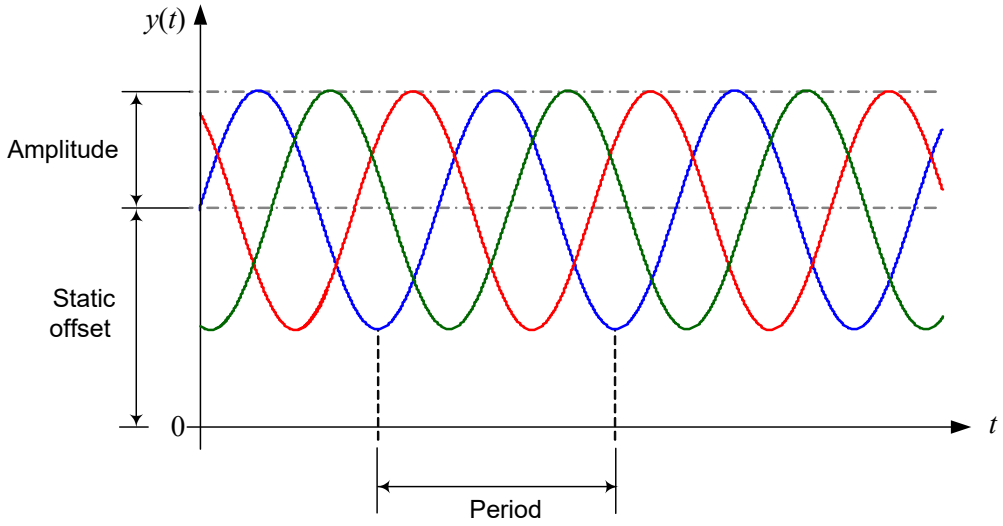

Figure 9. 3-phase sine mode

2.3 Frequency Modulation

Changes to the frequency of the dynamic waveforms can be made over a specified time interval. The interval is the last argument in the function DCL_setRefgenFreq(). The transition from the frequency at the time this function is called to the target frequency occurs linearly over the transition interval.

The linear change is implemented by increasing or decreasing the angle increment in small steps. The maximum transition time is limited by the numerical format to the value defined by DCL_REFGEN_MIN_INC in the file DCL_refgen.h. Attempts to force a large transition time together with a small frequency change may violate this limit, in which case one of two things will happen:

- If error handling is enabled an error will be generated.
- If error handling is not enabled, the smallest increment resulting in a change to the accumulated frequency variable will be adopted.

2.4 Amplitude Modulation

Amplitude modulation is implemented in a similar fashion to frequency modulation. The transition interval is the last argument in the function DCL_setRefgenAmpl(). Similarly, an attempt to force large transition time together with a small amplitude change will result in either an error or the minimum increment being adopted, according to whether error handling is enabled.

2.5 Output Clamp

All three reference outputs are clamped to the same upper and lower limits. Limits are loaded by the user using the DCL_setRefgenClamp() function with the only restriction that the upper limit be equal to or greater than the lower limit.

3. Using the Reference Generator

This chapter describes how to add the DCL reference generator module to existing user code.

3.1 How to Add the Reference Generator to User Code

The reference generator is integrated into the DCL, however it is straightforward to use the module independently of the library. In either case, follow these steps to use the reference generator module.

Step 1.

The user must include the header file DCL_refgen.h in their project (for double precision, the header file DCL_refgen64.h should be included in place of DCL_refgen.h). Note that this file includes the universal library header DCL.h which must be in the project include search path. The user must also add the source file DCL_error.c to the project. No other DCL files are required to use the reference generator.

Step 2.

Allocate a section in the project linker command file to hold the "dclfuncs" section in page 0. For example:

```
     dclfuncs          : > RAMGS11,    PAGE = 0
```

This section is where the "run" function and the three "get" functions are placed. These are most likely to be used in a time critical part of the user code, and their placement in this named section allows the user to control their location in memory. In some cases, it may be desirable to copy the "dclfuncs" section from flash to internal RAM to improve execution speed.

Step 3.

Create an instance of a reference generator and it's common support structure (CSS) in the main source and ensure it is visible to all source files which reference it. The CSS contains status and error information which is required by the reference generator module. Further information on the CSS can be found in the DCL User's Guide.

```
// variable declarations
DCL_REFGEN rgen = DCL_REFGEN_DEFAULTS;
DCL_CSS rgen_css = DCL_CSS_DEFAULTS;
```

The user might decide to declare the reference generator as 'extern' in a header file which is included in several different C files.

```
// extern variables
extern DCL_REFGEN rgen;
```

Initialize the reference generator as required using the functions listed in chapter 5. For example, the following code configures the generator to produce two sine waves of 150 Hz frequency and amplitude 0.25, superimposed on a static offset of 0.15.

```
    // initialize reference generator
    rgen.css = &rgen_css;
    DCL_SET_CONTROLLER_PERIOD(&rgen, 0.00001);
    DCL_resetRefgen(&rgen);
    DCL_setRefgenMode(&rgen, REFGEN_SINE2);
    DCL_setRefgenFreq(&rgen, 150.0f, 0.0f);
    DCL_setRefgenAmpl(&rgen, 0.25f, 0.0f);
    DCL_setRefgenRamp(&rgen, 0.15f, 0.01f);
```

Notice the first two lines which initialize the address of the CSS in the refgen structure, and set the sample rate of the reference generator.

Step 5.

Call the reference generator and read the output(s) as required. Typically these lines would be placed in an interrupt service routine to ensure they are executed at a deterministic rate. The CSS would be initialized with the ISR frequency using the DCL_SET_CONTROLLER_PERIOD macro as above.

```
DCL_runRefgen(&rgen);
x = DCL_getRefgenPhaseA(&rgen);
y = DCL_getRefgenPhaseB(&rgen);
```

3.2 Error Handling

Error handling is implemented in the same way as the DCL. Many of the reference generator functions perform range checks on parameters and input variables to ensure they fall within allowable ranges. These checks consume cycles, and in performance critical situations the user may elect to disable error checking by commenting out the following line in DCL.h.

```
#define DCL_ERROR_HANDLING_ENABLED
```

Users should inspect the source code for the relevant functions to determine which checks are performed.

If an error is detected, the code will set the “err” field of the CSS sub-structure of the controller with an error code, and load the “loc” field with the source line number where the error was detected. A list of enumerated error codes can be found in DCL.h.

4. Reference Generator Benchmarks

The list below shows the reference generator functions together with their execution cycles and code size in 16-bit words. Benchmarks were conducted with compiler optimization turned off, and with error checking disabled. Cycle counts include function call overhead. The compiler version used was v18.12.3.LTS. For more information refer to chapter 5.

Table 2. Single Precision Reference Generator Function Benchmarks

Function	Description	Cycles
DCL_resetRefgen	Resets the REFGEN module internal variables	90
DCL_setRefgenRamp	Loads the static offset generator	98
DCL_setRefgenAmpl	Sets the waveform amplitude	117
DCL_setRefgenFreq	Sets the waveform frequency	127
DCL_setRefgenDuty	Sets the pulse duty cycle	26
DCL_setRefgenClamp	Sets the output clamp limits	36
DCL_setRefgenMode	Sets the waveform generator operating mode	26
DCL_runRefgen	Runs the reference generator	(1)
DCL_getRefgenPhaseA	Gets the phase A reference output	24
DCL_getRefgenPhaseB	Gets the phase B reference output	24
DCL_getRefgenPhaseC	Gets the phase C reference output	24

(1) – see below

Table 3. DCL_runRefgen() Benchmarks

Mode	Cycles
0	420
1	430
2	383
3	397
4	367
5	387
6	381
7	389
8	478

Table 4. Double Precision Reference Generator Function Benchmarks

Function	Description	Cycles
DCL_resetRefgen64	Resets the REFGEN module internal variables	101
DCL_setRefgen64Ramp	Loads the static offset generator	122
DCL_setRefgen64Ampl	Sets the waveform amplitude	127
DCL_setRefgen64Freq	Sets the waveform frequency	126
DCL_setRefgen64Duty	Sets the pulse duty cycle	27
DCL_setRefgen64Clamp	Sets the output clamp limits	39
DCL_setRefgen64Mode	Sets the waveform generator operating mode	27
DCL_runRefgen64	Runs the reference generator	(1)
DCL_getRefgen64PhaseA	Gets the phase A reference output	26
DCL_getRefgen64PhaseB	Gets the phase B reference output	26
DCL_getRefgen64PhaseC	Gets the phase C reference output	26

(2) – see below

Table 5. DCL_runRefgen64() Benchmarks

Mode	Cycles
0	1340
1	1337
2	1541
3	1365
4	1334
5	1356
6	1501
7	1582
8	1806

5. Function Descriptions

This chapter provides details of the reference generator functions in the DCL. Detailed descriptions are given for single precision functions only. The table below shows equivalent single and double precision function names.

Single Precision	Double Precision
DCL_resetRefgen	DCL_resetRefgen64
DCL_setRefgenRamp	DCL_setRefgen64Ramp
DCL_setRefgenAmpl	DCL_setRefgen64Ampl
DCL_setRefgenFreq	DCL_setRefgen64Freq
DCL_setRefgenDuty	DCL_setRefgen64Duty
DCL_setRefgenClamp	DCL_setRefgen64Clamp
DCL_setRefgenMode	DCL_setRefgen64Mode
DCL_runRefgen	DCL_runRefgen64
DCL_getRefgenPhaseA	DCL_getRefgen64PhaseA
DCL_getRefgenPhaseB	DCL_getRefgen64PhaseB
DCL_getRefgenPhaseC	DCL_getRefgen64PhaseC

Table 6. Single and double precision function names

5.1 DCL_resetRefgen

This function resets the internal dynamic variables in the DCL_REFGEN structure to their default values.

HEADER FILE DCL_refgen.h

SOURCE FILE N/A

DECLARATION void DCL_resetRefgen(DCL_REFGEN *p)

PARAMETERS

p	The DCL_REFGEN structure
Return	void

EXAMPLE

```
// variable declarations
DCL_REFGEN rgen;
DCL_CSS rgen_css = DCL_CSS_DEFAULTS;

// reset the reference generator
DCL_resetRefgen(&rgen);
```

5.2 DCL_setRefgenRamp

This function loads a new desired offset level and the transition time from the current offset to the new one. The update rate parameter (T) in the CSS sub-module must be loaded prior to calling this function. The ramp will begin immediately the active parameters are changed.

HEADER FILE DCL_refgen.h

SOURCE FILE N/A

DECLARATION void DCL_setRefgenRamp(DCL_REFGEN *p, float32_t tgt, float32_t tr)

PARAMETERS

p	The DCL_REFGEN structure
tgt	The target offset level
tr	The transition time between levels in seconds
Return	void

EXAMPLE

```
// variable declarations
DCL_REFGEN rgen = DCL_REFGEN_DEFAULTS;
DCL_CSS rgen_css = DCL_CSS_DEFAULTS;

// set the sample period
rgen.css = &rgen_css;
DCL_SET_CONTROLLER_PERIOD(&rgen, 0.00001);

// configure the static offset to +0.5 in a 10 ms ramp
DCL_setRefgenRamp(&rgen, 0.5f, 0.01f);
```

5.3 DCL_setRefgenAmpl

This function loads the amplitude parameter of the waveform generator. All three outputs of the dynamic waveform generator are multiplied by the amplitude prior to addition of the static offset. The sign of the amplitude is not tested so it is permissible to apply negative amplitude.

HEADER FILE DCL_refgen.h

SOURCE FILE N/A

DECLARATION void DCL_setRefgenAmpl(DCL_REFGEN *p, float32_t ampl, float32_t tr)

PARAMETERS

p	The DCL_REFGEN structure
ampl	The target amplitude
tr	Transition time between amplitudes in seconds
Return	void

EXAMPLE

```
// variable declarations
DCL_REFGEN rgen = DCL_REFGEN_DEFAULTS;
DCL_CSS rgen_css = DCL_CSS_DEFAULTS;

// set the sample period
rgen.css = &rgen_css;
DCL_SET_CONTROLLER_PERIOD(&rgen, 0.00001);

// set the waveform amplitude to 1.95 in 10 ms
DCL_setRefgenAmpl(&rgen, 1.95f, 0.01f);
```

5.4 DCL_setRefgenFreq

This function loads the frequency and transition time of the waveform generator.

HEADER FILE DCL_refgen.h

SOURCE FILE N/A

DECLARATION void DCL_setRefgenFreq(DCL_REFGEN *p, float32_t freq, float32_t tr)

PARAMETERS

p	The DCL_REFGEN structure
freq	The target frequency
tr	Transition time between frequencies in seconds
Return	void

EXAMPLE

```
// variable declarations
DCL_REFGEN rgen = DCL_REFGEN_DEFAULTS;
DCL_CSS rgen_css = DCL_CSS_DEFAULTS;

// set the sample period
rgen.css = &rgen_css;
DCL_SET_CONTROLLER_PERIOD(&rgen, 0.00001);

// set the waveform frequency to 500 Hz in 100 ms ramp
DCL_setRefgenFreq(&rgen, 500.0f, 0.1f);
```

5.5 DCL_setRefgenDuty

This function loads the duty cycle parameter of the waveform generator when set to pulse generation mode. The duty cycle is used to control the duration of the high portion of the pulse waveform and is a normalized value between 0 and 1.

HEADER FILE DCL_refgen.h

SOURCE FILE N/A

DECLARATION void DCL_setRefgenDuty(DCL_REFGEN *p, float32_t duty)

PARAMETERS

p	The DCL_REFGEN structure
duty	The new duty cycle
Return	void

EXAMPLE

```
// variable declarations
DCL_REFGEN rgen = DCL_REFGEN_DEFAULTS;

// set the pulse waveform duty cycle to 15%
DCL_setRefgenDuty(&rgen, 0.15f);
```

5.6 DCL_setRefgenClamp

This function loads the upper and lower limits of the output clamp of the reference generator.

HEADER FILE DCL_refgen.h

SOURCE FILE N/A

DECLARATION void DCL_setRefgenClamp(DCL_REFGEN *p, float32_t max, float32_t min)

PARAMETERS

p	The DCL_REFGEN structure
max	The upper clamp limit
min	The lower clamp limit
Return	void

EXAMPLE

```
// variable declarations
DCL_REFGEN rgen = DCL_REFGEN_DEFAULTS;

// set the output clamp limits
DCL_setRefgenClamp(&rgen, 0.75f, -0.35f);
```

5.7 DCL_setRefgenMode

This function configures the operating mode of the waveform generator. The input parameter determines the generated waveform type. Refer to table 1 for a list of operating modes.

HEADER FILE DCL_refgen.h

SOURCE FILE N/A

DECLARATION void DCL_setRefgenMode(DCL_REFGEN *p, uint16_t mode)

PARAMETERS

p	The DCL_REFGEN structure
mode	The new operating mode
Return	void

EXAMPLE

```
// variable declarations
DCL_REFGEN rgen = DCL_REFGEN_DEFAULTS;
DCL_CSS rgen_css = DCL_CSS_DEFAULTS;

// set the waveform type to square wave
DCL_setRefgenMode(&rgen, REFGEN_SQUARE);
```

5.8 DCL_runRefgen

This function runs the reference generator. The function should be called from an interrupt service routine so that it runs at deterministic rate, and the update rate should be loaded into the 'T' parameter of the CSS sub-structure. This function is mapped to the "dclfuncs" section in the linker command file.

HEADER FILE DCL_refgen.h

SOURCE FILE N/A

DECLARATION void DCL_runRefgen(DCL_REFGEN *p)

PARAMETERS

p	The DCL_REFGEN structure
Return	void

EXAMPLE

```
// variable declarations
DCL_REFGEN rgen = DCL_REFGEN_DEFAULTS;
DCL_CSS rgen_css = DCL_CSS_DEFAULTS;
float32_t v;

// configure the reference generator to produce a 150 Hz sine wave
// superimposed on a 0.5 offset
rgen.css = &rgen_css;
DCL_SET_CONTROLLER_PERIOD(&rgen, 0.00001);
DCL_resetRefgen(&rgen);
DCL_setRefgenMode(&rgen, REFGEN_SINE);
DCL_setRefgenFreq(&rgen, 150.0f, 0.0f);
DCL_setRefgenAmpl(&rgen, 0.25f, 0.0f);
DCL_setRefgenRamp(&rgen, 0.5f, 0.01f);

// run the reference generator
DCL_runRefgen(&rgen);
v = DCL_getRefgenPhaseA(&rgen);
```

5.9 DCL_getRefgenPhaseX (X = A, B, C)

This set of three functions returns the reference generator outputs. Each is implemented as a static inline function and returns one of the three generated outputs. These functions are mapped to the "dclfuncs" section in the linker command file.

HEADER FILE DCL_refgen.h

SOURCE FILE N/A

DECLARATION float32_t DCL_getRefgenPhaseA(DCL_REFGEN *p)

float32_t DCL_getRefgenPhaseB(DCL_REFGEN *p)

float32_t DCL_getRefgenPhaseC(DCL_REFGEN *p)

PARAMETERS

p	The DCL_REFGEN structure
Return	The phase A, B, or C output reference

EXAMPLE

```
// variable declarations
DCL_REFGEN rgen = DCL_REFGEN_DEFAULTS;
DCL_CSS rgen_css = DCL_CSS_DEFAULTS;
float32_t x, y, z;

// configure the reference generator to produce a 3 phase sine wave
// of 150 Hz superimposed on a 0.5 offset
rgen.css = &rgen_css;
DCL_SET_CONTROLLER_PERIOD(&rgen, 0.00001);
DCL_resetRefgen(&rgen);
DCL_setRefgenMode(&rgen, REFGEN_3PHASE);
DCL_setRefgenFreq(&rgen, 150.0f, 0.0f);
DCL_setRefgenAmpl(&rgen, 0.25f, 0.0f);
DCL_setRefgenRamp(&rgen, 0.5f, 0.01f);

// run the reference generator
DCL_runRefgen(&rgen);
x = DCL_getRefgenPhaseA(&rgen);
y = DCL_getRefgenPhaseB(&rgen);
z = DCL_getRefgenPhaseC(&rgen);
```

6. DOCUMENT HISTORY

Date	Changes
2/7/2020	Initial release