Subsystem Design

Function Generator Using DAC8

1 Description

This *DAC8 function generator subsystem example* shows how to implement a basic function generator to output a sine wave using the integrated DAC8, a timer triggered DMA, and an OPAx module. A 256-point sine wave data table written in the code is loaded to the DAC8 about every 40µs using a timer to trigger the DMA transfers. This generates an approximate 98Hz, 256-point sine wave. The OPA is used to buffer the DAC output and route the signal to an external pin, PA22 (OPA0_Out).

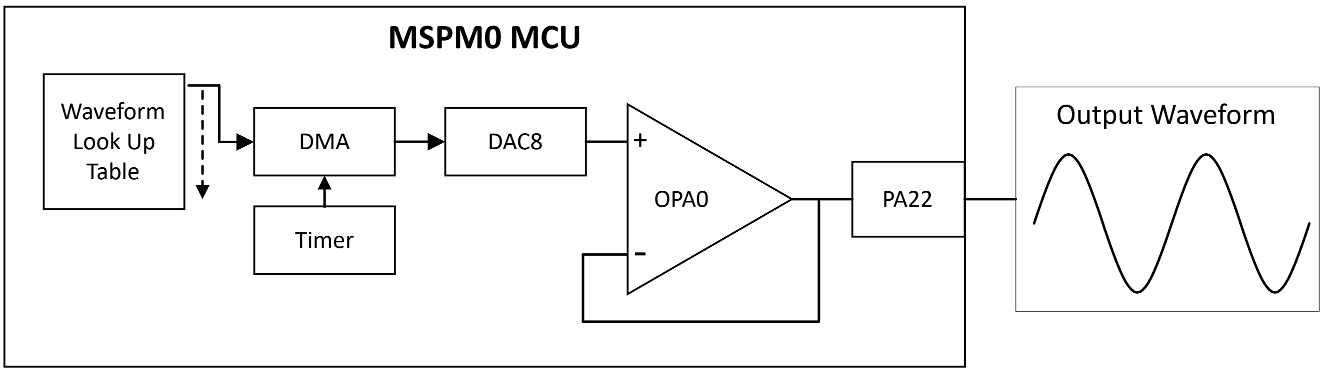

Figure 1-1. DAC8 Function Generator Block Diagram

2 Required Peripherals

Table 2-1 describes the required integrated peripherals.

Table 2-1. Required Peripherals

Subblock Functionality	Peripheral Use	Notes
DAC8	1 × COMP	Shown as `COMP_0_INST` in code
DMA	1 × DMA Channel	Shown as `DMA_CH0` in code
Buffer	1 × OPA	Shown as `OPA_0_INST` in code
Timer	1 × TIMG	Shown as `TIMER_0_INST` in code

3 Compatible Devices

Based on the requirements shown in Table 2-1, the example code is compatible with the devices shown in Table 3-1.

Table 3-1. Compatible Devices

Compatible Devices	EVM
MSPM0L13x	LP-MSPM0L1306
MSPM0Gx	LP-MSPM0G3507
MSPM0Lx22x	LP-MSPM0L2228

4 Design Steps

Complete the following to implement or update the simple function generator output:

1. Build and load a 256-point table of 8-bit DAC values for the desired waveform output.
2. Configure the DMA in SysConfig to transfer a block of data to a fixed address of the DAC8. Subscribe the DMA transfer trigger to an event so the exact timing can be controlled by a timer.
3. Configure a timer in SysConfig to publish a periodic event and trigger the DMA transfers.
4. Enable the COMP in SysConfig to power the DAC8.
5. Use SysConfig to configure and OPAx in buffer mode with the DAC8 as the input source and enable the output.

5 Design Considerations

When generating an output waveform, here are a few things to consider:

1. Waveform period is equal to the DMA update timer period times the number of points in the waveform. More points enable a smoother waveform but a longer period.
   a. At maximum speed, consider the DAC8 settling time of about 1.5µs
2. In this example, there are 256-points and the DMA updates the points every 40µs.
   a. 256 point × 40µs = 10.24ms waveform period = approximately 98Hz output
3. Some MSPM0Gxx devices have standalone 12-bit DAC modules that do not have to be buffered by the integrated OPAx.

6 Software Flow Chart

This flow chart provides a high-level overview of the *software for the DAC8 function generator*. Most of the effort for this subsystem is properly configuring the peripherals through SysConfig. The application code only needs to point the DMA at the Waveform look up table (LUT) and the DAC8 control register and then start the periodic timer. The timer publishes an event every 40µs which triggers the DMA to transfer the next byte of the waveform.

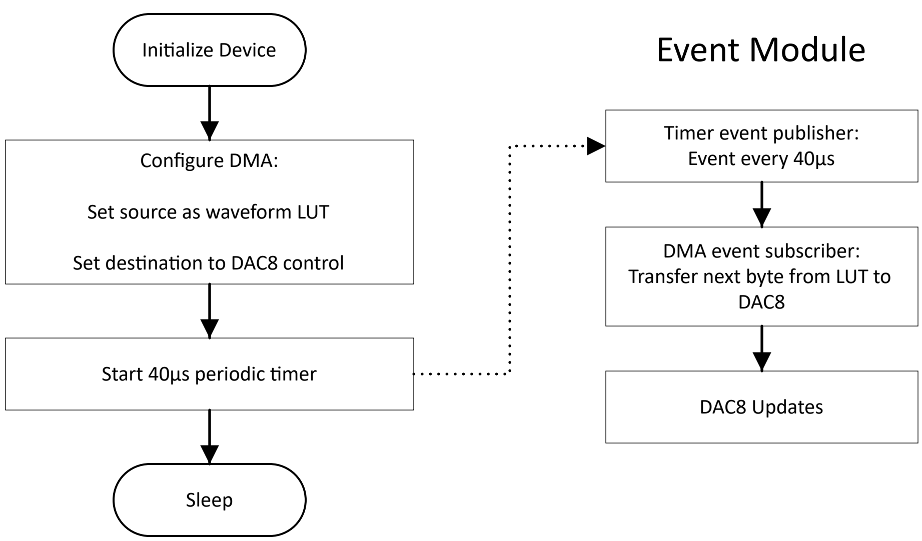

Figure 6-1. Application Software Flow Chart

7 Application Code

There are two main sections of the application code. The first is the waveform look up table (LUT). The following code has a table for a 256 point sine wave for an 8-bit peak. This DAC look up table can be easily generated with the use of online calculators.

```
/* Number of samples in the function array */
#define SINE_ARRAY_SAMPLES 256

/* Array of samples for function stored in flash */
const uint8_t gSineArray[SINE_ARRAY_SAMPLES] =
            {
            0x80,0x83,0x86,0x89,0x8c,0x8f,0x92,0x95,
            0x98,0x9c,0x9f,0xa2,0xa5,0xa8,0xab,0xae,
            0xb0,0xb3,0xb6,0xb9,0xbc,0xbf,0xc1,0xc4,
            0xc7,0xc9,0xcc,0xce,0xd1,0xd3,0xd5,0xd8,
            0xda,0xdc,0xde,0xe0,0xe2,0xe4,0xe6,0xe8,
            0xea,0xeb,0xed,0xef,0xf0,0xf2,0xf3,0xf4,
            0xf6,0xf7,0xf8,0xf9,0xfa,0xfb,0xfb,0xfc,
            0xfd,0xfd,0xfe,0xfe,0xfe,0xff,0xff,0xff,
            0xff,0xff,0xff,0xff,0xfe,0xfe,0xfd,0xfd,
            0xfc,0xfc,0xfb,0xfa,0xf9,0xf8,0xf7,0xf6,
            0xf5,0xf4,0xf2,0xf1,0xef,0xee,0xec,0xeb,
            0xe9,0xe7,0xe5,0xe3,0xe1,0xdf,0xdd,0xdb,
            0xd9,0xd7,0xd4,0xd2,0xcf,0xcd,0xca,0xc8,
            0xc5,0xc3,0xc0,0xbd,0xba,0xb8,0xb5,0xb2,
            0xaf,0xac,0xa9,0xa6,0xa3,0xa0,0x9d,0x9a,
            0x97,0x94,0x91,0x8e,0x8a,0x87,0x84,0x81,
            0x7e,0x7b,0x78,0x75,0x71,0x6e,0x6b,0x68,
            0x65,0x62,0x5f,0x5c,0x59,0x56,0x53,0x50,
            0x4d,0x4a,0x47,0x45,0x42,0x3f,0x3c,0x3a,
            0x37,0x35,0x32,0x30,0x2d,0x2b,0x28,0x26,
            0x24,0x22,0x20,0x1e,0x1c,0x1a,0x18,0x16,
            0x14,0x13,0x11,0x10,0xe,0xd,0xb,0xa,
            0x9,0x8,0x7,0x6,0x5,0x4,0x3,0x3,
            0x2,0x2,0x1,0x1,0x0,0x0,0x0,0x0,
            0x0,0x0,0x0,0x1,0x1,0x1,0x2,0x2,
            0x3,0x4,0x4,0x5,0x6,0x7,0x8,0x9,
            0xb,0xc,0xd,0xf,0x10,0x12,0x14,0x15,
            0x17,0x19,0x1b,0x1d,0x1f,0x21,0x23,0x25,
            0x27,0x2a,0x2c,0x2e,0x31,0x33,0x36,0x38,
            0x3b,0x3e,0x40,0x43,0x46,0x49,0x4c,0x4f,
            0x51,0x54,0x57,0x5a,0x5d,0x60,0x63,0x67,
            0x6a,0x6d,0x70,0x73,0x76,0x79,0x7c,0x80
            };
```

The second section is the main application code which configures the DMA and starts the timer as Figure 6-1 shows.

```
int main(void)
{
    SYSCFG_DL_init();

    /* Configure DMA to load samples from the gSineArray to the CTL3 register of COMP_0_INST */
    DL_DMA_setSrcAddr(DMA, DMA_CH0_CHAN_ID, (uint32_t) &gSineArray[0]);
    DL_DMA_setDestAddr(DMA, DMA_CH0_CHAN_ID, (uint32_t) &COMP_0_INST->CTL3);
    DL_DMA_setTransferSize(DMA, DMA_CH0_CHAN_ID, SINE_ARRAY_SAMPLES);
    DL_DMA_setSubscriberChanID(DMA, DL_DMA_SUBSCRIBER_INDEX_0, 1);
    DL_DMA_enableChannel(DMA, DMA_CH0_CHAN_ID);

    /* Start the timer counting, the zero event of this timer acts as the trigger for transfer */
    DL_TimerG_startCounter(TIMER_0_INST);

    while (1) {
        __WFI();
    }
}
```

8 Results

Figure 8-1 shows the output sine waveform with a measured period of 97.4Hz.

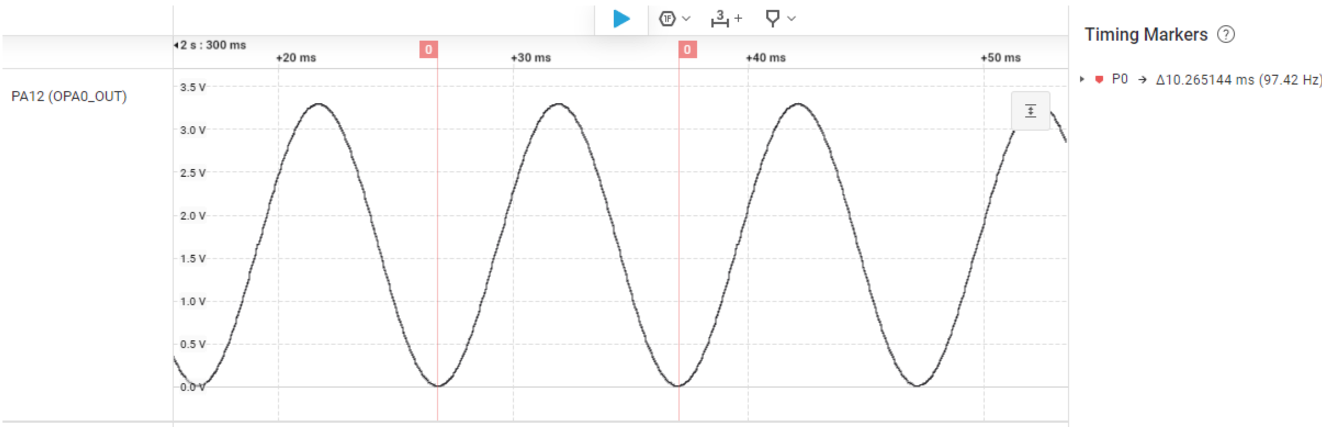

Figure 8-1. Waveform Output With Period

Zooming in on the *waveform* highlights the steps from DAC8 and the waveform LUT. The wave can be smoothed more by using a small amount of filtering with some external passive components and the OPA.

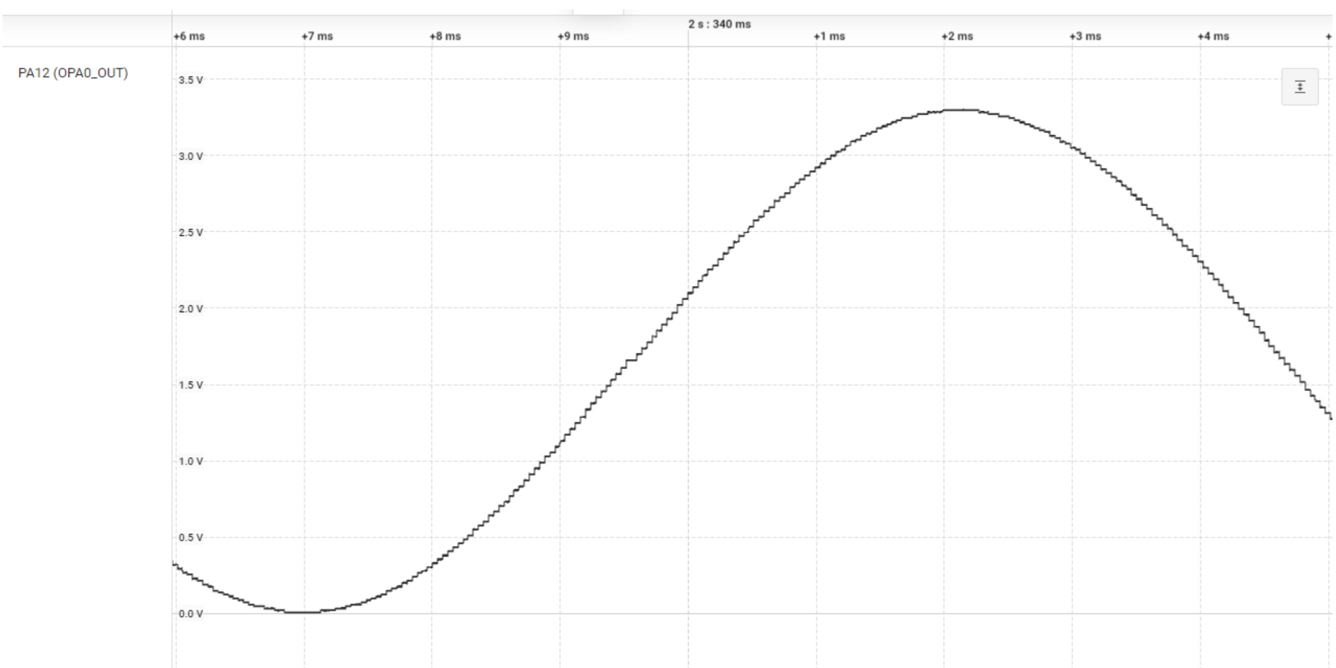

Figure 8-2. Waveform Output Close Up

9 Additional Resources

- Texas Instruments, Download the MSPM0 SDK
- Texas Instruments, Learn more about SysConfig
- Texas Instruments, MSPM0C LaunchPad™
- Texas Instruments, MSPM0L LaunchPad™
- Texas Instruments, MSPM0G LaunchPad™
- Texas Instruments, MSPM0 Academy

10 E2E

See TI's E2E™ support forums to view discussions and post new threads to get technical support for utilizing MSPM0 devices in designs.

IMPORTANT NOTICE AND DISCLAIMER