```
#include <device.h>
#include <cytypes.h>
#define True 1
#define False 0
#define TABLE_LENGTH 128
#define DMA_BYTES_PER_BURST 1
#define DMA_REQUEST_PER_BURST 1
int32 Shift=0;
int32 degree;
char PotChanged = False;
int32 oldValue;
int32 ADC_POT;

CY_ISR(Interrupt)
{
    Shift++;
    if(oldValue - ADC_POT > 0x400||ADC_POT - oldValue > 0x100){
        PotChanged = True;
        CyPins_SetPin(LED_1_0);
    }
    else{
        PotChanged = False;
        CyPins_ClearPin(LED_1_0);
    }
    /*if(ADC_POT - oldValue > 0x100){
        PotChanged = True;
        CyPins_SetPin(LED_1_0);
    }
    else{
        PotChanged = False;
        CyPins_ClearPin(LED_1_0);
    }*/
    oldValue = ADC_POT;
    Timer_ReadStatusRegister();
}

/* This table stores the 128 points in Flash for smoother sine wave ↵
generation */
CYCODE const uint8 sineTable[TABLE_LENGTH*2] =
{

    128, 134, 140, 147, 153, 159, 165, 171,
    177, 183, 188, 194, 199, 204, 209, 214,
    218, 223, 227, 231, 234, 238, 241, 244,
    246, 248, 250, 252, 253, 254, 255, 255,
    255, 255, 255, 254, 253, 252, 250, 248,
    246, 244, 241, 238, 234, 231, 227, 223,
    218, 214, 209, 204, 199, 194, 188, 183,
    177, 171, 165, 159, 153, 147, 140, 134,
    128, 122, 115, 109, 103,  97,  91,  85,
     79,  73,  68,  62,  57,  52,  47,  42,
```

```
     37,  33,  29,  25,  22,  18,  15,  12,
     10,   7,   6,   4,   2,   1,   1,   0,
      0,   0,   1,   1,   2,   4,   6,   7,
     10,  12,  15,  18,  22,  25,  29,  33,
     37,  42,  47,  52,  57,  62,  68,  73,
     79,  85,  91,  97, 103, 109, 115, 122,
    128, 134, 140, 147, 153, 159, 165, 171,
    177, 183, 188, 194, 199, 204, 209, 214,
    218, 223, 227, 231, 234, 238, 241, 244,
    246, 248, 250, 252, 253, 254, 255, 255,
    255, 255, 255, 254, 253, 252, 250, 248,
    246, 244, 241, 238, 234, 231, 227, 223,
    218, 214, 209, 204, 199, 194, 188, 183,
    177, 171, 165, 159, 153, 147, 140, 134,
    128, 122, 115, 109, 103,  97,  91,  85,
     79,  73,  68,  62,  57,  52,  47,  42,
     37,  33,  29,  25,  22,  18,  15,  12,
     10,   7,   6,   4,   2,   1,   1,   0,
      0,   0,   1,   1,   2,   4,   6,   7,
     10,  12,  15,  18,  22,  25,  29,  33,
     37,  42,  47,  52,  57,  62,  68,  73,
     79,  85,  91,  97, 103, 109, 115, 122,
};
CYCODE const uint8 Constant[TABLE_LENGTH] =
{
    128,128,128,128,128,128,128,128,
    128,128,128,128,128,128,128,128,
    128,128,128,128,128,128,128,128,
    128,128,128,128,128,128,128,128,
    128,128,128,128,128,128,128,128,
    128,128,128,128,128,128,128,128,
    128,128,128,128,128,128,128,128,
    128,128,128,128,128,128,128,128,
    128,128,128,128,128,128,128,128,
    128,128,128,128,128,128,128,128,
    128,128,128,128,128,128,128,128,
    128,128,128,128,128,128,128,128,
    128,128,128,128,128,128,128,128,
    128,128,128,128,128,128,128,128,
    128,128,128,128,128,128,128,128,
    128,128,128,128,128,128,128,128
};
/* Variable declarations for DMA .
* These variables are defined as global variables to avoid "may be used ↵
before being set" warning
* issued by the PSoC 5 compilers  MDK/RVDS.In this case these variables are ↵
automatically initialized to zero */
uint8 DMA_1_Chan;               /* The DMA Channel */
uint8 DMA_2_Chan;

uint8 DMA_1_TD[1];            /* The DMA Transaction Descriptor (TD) */
uint8 DMA_2_TD[2];
```

```
void main()
{
    LCD_Start();
     ADC_Start();
    ADC_StartConvert();
    VDAC8_Start();
    VDAC8_Shift_Start();
    isr_StartEx(Interrupt);
    Timer_Start();
    CYGlobalIntEnable;

    #if (defined(__C51__))  /* Source base address when PSoC3 is used */
        #define DMA_SRC_BASE (CYDEV_FLS_BASE)
    #else                   /* Source base address when PSoC5 is used */
        #define DMA_SRC_BASE (&sineTable[0])
    #endif

    #define DMA_DST_BASE (CYDEV_PERIPH_BASE)  /* Destination base address */

    /* Step1 : DmaInitialize - Initialize the DMA channel
     * Bytes per burst = 1, (8 bit data transferred to VDAC one at a time)
     * Request per burst = 1 (this will cause transfer of the bytes only with ↵
every new request)
     * High byte of source address = Upper 16 bits of Flash Base address for ↵
PSoC 3,
                                    = HI16(&sineTable) for PSoC 5
     * High byte of destination address =  Upper 16 bits of peripheral base ↵
address */
    DMA_1_Chan = DMA_1_DmaInitialize(DMA_BYTES_PER_BURST, ↵
DMA_REQUEST_PER_BURST, HI16(DMA_SRC_BASE), HI16(DMA_DST_BASE) );
    DMA_2_Chan = DMA_2_DmaInitialize(DMA_BYTES_PER_BURST, ↵
DMA_REQUEST_PER_BURST, HI16(DMA_SRC_BASE), HI16(DMA_DST_BASE) );


    /* Step2 :CyDmaTdAllocate - Allocate TD */
    DMA_1_TD[0] = CyDmaTdAllocate();
    DMA_2_TD[0] = CyDmaTdAllocate();

    /* Step3 :CyDmaTdSetConfiguration - Configures the TD:
     * tdHandle = DMA_TD[0] - TD handle previously returned by CyDmaTdAlloc()
     * Transfer count = table_length (number of bytes to transfer for a sine ↵
wave)
     * Next Td = DMA_TD[0] ; loop back to the same TD to generate a continous ↵
sien wave
     * Configuration = The source address is incremented after every burst ↵
transfer
     */
    CyDmaTdSetConfiguration(DMA_1_TD[0], 128, DMA_1_TD[0], TD_INC_SRC_ADR);
    CyDmaTdSetConfiguration(DMA_2_TD[0], 128, DMA_2_TD[0], TD_INC_SRC_ADR);
```

```
    /* Step 4 : CyDmaTdSetAddress - Configure the lower 16 bit source and destination addresses
     * Source address = Lower 16 bits of sineTable array
     * Destination address = Lower 16 bits of VDAC8_Data_PTR register */
    CyDmaTdSetAddress(DMA_1_TD[0], LO16((uint32)sineTable), LO16((uint32)VDAC8_Data_PTR) );
    CyDmaTdSetAddress(DMA_2_TD[0], LO16((uint32)sineTable), LO16((uint32)VDAC8_Shift_Data_PTR) );

    /* Step 5: Map the TD to the DMA Channel */
    CyDmaChSetInitialTd(DMA_1_Chan, DMA_1_TD[0]);
    CyDmaChSetInitialTd(DMA_2_Chan, DMA_2_TD[0]);

    /* Step 6: Enable the DMA channel */
    CyDmaChEnable(DMA_1_Chan, 1);
    CyDmaChEnable(DMA_2_Chan, 1);

    for(;;)
    {
        /* Your code here. */
        if(ADC_IsEndConversion(ADC_RETURN_STATUS))
        {
        ADC_POT = ADC_GetResult32();
        if(ADC_POT<0x0000){
                ADC_POT = 0x0000;
        }

        if(PotChanged){
            Shift =0;
    CyDmaTdSetAddress(DMA_1_TD[0], LO16((uint32)Constant), LO16((uint32)VDAC8_Data_PTR) );
        }
        else{
            CyDmaTdSetAddress(DMA_1_TD[0], LO16((uint32)&sineTable[128-Shift]), LO16((uint32)VDAC8_Data_PTR) );
        }
        if(Shift == 125)
            Shift =0;
        LCD_Position(1u,0u);
        LCD_PrintInt16(ADC_POT);

    }
    }
}
```