```
// PARCIAL 4 - DISEÑO CON uP y uC. 2024-1.
// NOMBRE: Diego Andrés García Díaz.
// CÓDIGO: 2195533.
// ------------------------------------------------------------------------

// PRIMERA FORMA USANDO SIN de la libreria C_Math

//// Se definen las variables
//float deg;
//int out;
//
//void InitMain(){
//    ANSELA = 0;   // Se declaran como digitales los puertos A
//    // Se declaran como salida los pines que estaran conectados al DAC.
//    TRISA.F0 = 0;  // Salida pin SDO.
//    TRISA.F1 = 0;  // Salida pin SCK.
//    TRISA.F4 = 0;  // Salida pin LDAC.
//    TRISA.F5 = 0;  // Salida pin CS.
//    // Se define el estado de los pines que transmiten los datos.
//    LATA.F4 = 1;  // LDAC en 1 --> Hace que el DAC no transfiera datos a la salida.
//    LATA.F5 = 1;  // CS en 1 --> Desactiva la transerencia de datos.
//    // Se define la conexion SPI
//    SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV4, _SPI_DATA_SAMPLE_MIDDLE,
_SPI_CLK_IDLE_LOW, _SPI_LOW_2_HIGH);
//}
//void signal_sen(int values){
//    char temporal; // Porque tiene 8 bits = 1 byte
//    // Envia el byte en alto
//    temporal = (values >> 8) & 0x0F; // Desplazamiento para los bits más
significativos.
//    temporal |= 0x30; // OR lógica para agregar palabras de configuración del DAC.
//    LATA.F5 = 0;  // Se desactiva el pin CS del DAC, para poder transmitir los datos.
//    SPI1_Write(temporal); // Envia los correspondientes datos de los 8 bits más
significativos.
//
//    // Envia el byte en bajo
//    temporal = values;   // temporal solo guarda los 8 primeros bits
//    SPI1_Write(temporal); // Envia el byte restante
//    LATA.F5 = 1; // Se desactiva el CS para dejar de transmitir datos.
//    LATA.F4 = 0; // Se activa el LDAC para que transfiera los datos de los bits
restantes.
//    LATA.F4 = 1; // Enseguida se desactiva el LDAC para no transmitir datos.
//}
//
//void main(void){
//    OSCCON = 0b11110011;  //Configuración del oscilador a 32 MHz
//    deg = 0;  // Inicializar variable de grados en 0.
//    InitMain();
//    while(1){
//        // Se calcula el valor del seno y se transforma el valor para la lectura del
DAC.
//        out = (int)(4095/5)*4*(fabs(sin(deg)));
//        signal_sen(out);  // Se envian los datos al DAC
//        deg = deg + 0.017453; // Se aumenta la resolución grado a grado.
//    }
```

```
//}

// ------------------------------------------------------------------------------------------
-----------------------------------------------

// SEGUNDA FORMA USANDO TABLAS:

// Se declaran variables
int i=0;
int temporal;

const unsigned int sen[] = {
0, 57, 114, 171, 229, 286, 342, 399, 456, 512, 569, 625, 681, 737, 793, 848, 903, 958,
1012,
1067, 1120, 1174, 1227, 1280, 1332, 1384, 1436, 1487, 1538, 1588, 1638, 1687, 1736, 1784,
1832,
1879, 1926, 1972, 2017, 2062, 2106, 2149, 2192, 2234, 2276, 2316, 2357, 2396, 2435, 2472,
2510,
2546, 2582, 2616, 2650, 2684, 2716, 2747, 2778, 2808, 2837, 2865, 2893, 2919, 2944, 2969,
2993,
3016, 3037, 3058, 3078, 3098, 3116, 3133, 3149, 3164, 3179, 3192, 3204, 3216, 3226, 3236,
3244,
3252, 3258, 3264, 3268, 3272, 3274, 3276, 3276, 3276, 3274, 3272, 3268, 3264, 3258, 3252,
3244,
3236, 3226, 3216, 3204, 3192, 3179, 3164, 3149, 3133, 3116, 3098, 3078, 3058, 3037, 3016,
2993,
2969, 2944, 2919, 2893, 2865, 2837, 2808, 2778, 2747, 2716, 2684, 2650, 2616, 2582, 2546,
2510,
2472, 2435, 2396, 2357, 2316, 2276, 2234, 2192, 2149, 2106, 2062, 2017, 1972, 1926, 1879,
1832,
1784, 1736, 1687, 1638, 1588, 1538, 1487, 1436, 1384, 1332, 1280, 1227, 1174, 1120, 1067,
1012,
958, 903, 848, 793, 737, 681, 625, 569, 512, 456, 399, 342, 286, 229, 171, 114, 57, 0,
57, 114,
171, 229, 286, 342, 399, 456, 512, 569, 625, 681, 737, 793, 848, 903, 958, 1012, 1067,
1120, 1174,
1227, 1280, 1332, 1384, 1436, 1487, 1538, 1588, 1638, 1687, 1736, 1784, 1832, 1879, 1926,
1972,
2017, 2062, 2106, 2149, 2192, 2234, 2276, 2316, 2357, 2396, 2435, 2472, 2510, 2546, 2582,
2616,
2650, 2684, 2716, 2747, 2778, 2808, 2837, 2865, 2893, 2919, 2944, 2969, 2993, 3016, 3037,
3058,
3078, 3098, 3116, 3133, 3149, 3164, 3179, 3192, 3204, 3216, 3226, 3236, 3244, 3252, 3258,
3264,
3268, 3272, 3274, 3276, 3276, 3276, 3274, 3272, 3268, 3264, 3258, 3252, 3244, 3236, 3226,
3216,
3204, 3192, 3179, 3164, 3149, 3133, 3116, 3098, 3078, 3058, 3037, 3016, 2993, 2969, 2944,
2919,
2893, 2865, 2837, 2808, 2778, 2747, 2716, 2684, 2650, 2616, 2582, 2546, 2510, 2472, 2435,
2396,
2357, 2316, 2276, 2234, 2192, 2149, 2106, 2062, 2017, 1972, 1926, 1879, 1832, 1784, 1736,
1687,
1638, 1588, 1538, 1487, 1436, 1384, 1332, 1280, 1227, 1174, 1120, 1067, 1012, 958, 903,
848, 793,
737, 681, 625, 569, 512, 456, 399, 342, 286, 229, 171, 114, 57, 0};
```

```
void InitMain(){
      TRISA.F0 = 0;  // Salida pin SDO.
      TRISA.F1 = 0;  // Salida pin SCK.
      TRISA.F4 = 0;  // Salida pin LDAC.
      TRISA.F5 = 0;  // Salida pin CS.
      LATA.F4 = 1;   // LDAC en 1 --> Hace que el DAC no transfiera datos a la salida.
      LATA.F5 = 1;   // CS en 1 --> Desactiva la transerencia de datos.
      SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV4, _SPI_DATA_SAMPLE_MIDDLE,
_SPI_CLK_IDLE_LOW, _SPI_LOW_2_HIGH);
}
void signal_sen(int values){
      char temporal;
      //Envio byte en alto
      temporal = (values >> 8) & 0x0F; // Desplazamiento para los bits más
significativos.
      temporal |= 0x30; // OR lógica para agregar palabras de configuración del DAC.
      LATA.F5 = 0;  // Se desactiva el pin CS del DAC, para poder transmitir los datos.
      SPI1_Write(temporal); // Envia los correspondientes datos de los 8 bits más
significativos.

      //Envio byte en bajo
      temporal = values;   // temporal solo guarda los 8 primeros bits
      SPI1_Write(temporal); // Envia el byte restante
      LATA.F5 = 1; // Se desactiva el CS para dejar de transmitir datos.
      LATA.F4 = 0; // Se activa el LDAC para que transfiera los datos de los bits
restantes.
      LATA.F4 = 1; // Enseguida se desactiva el LDAC para no transmitir datos.
}

void main(void){
    OSCCON = 0b11110011;  //Configuración del oscilador a 32 MHz
    InitMain();
    while(1){
        signal_sen(sen[i]);  // Señal seno por tablas.
        i++;
        if(i==360){i=0;}
        Delay_us(1);  // Retardo
    }
}
```