DESARROLLO DE UN SISTEMA DE MEDICIÓN DE TEMPERATURA BASADO EN UN TERMOPAR TIPO K Y UN PSoC®.

*Juan Antonio Jaramillo Gómez**

jantonioj@yahoo.com, jjaramillo@ipn.mx, Ext. 56850

*Luis Guillermo Venegas Pineda**

luisguillermovp@gmail.com

**UPIITA – IPN*

Resumen

En este documento se presenta el diseño, simulación y desarrollo de un sistema de medición de temperatura utilizando un termopar tipo K, un AD595 como acondicionador de señal, un PSoC® CY8C29466 como controlador del sistema y una LCD para el despliegue de la información medida.

Abstract

In this document is presented the design, simulation and development of a temperature measurement system using a thermocouple type K, an AD595 as signal conditioner, a PSoC® CY8C29466 as controller of the system and a LCD used to display the data measured.

Introducción

Día con día la tecnología crece de manera importante, permitiendo solucionar problemas más complejos. Un claro ejemplo de lo anterior es el desarrollo y la evolución de los microcontroladores, circuitos integrados que en la actualidad se encuentran en la mayoría de los dispositivos electrónicos.

Una variante de microcontroladores son los PSoC®, dispositivos desarrollados por la compañía Cypress Semiconductor® en 2002. Desafortunadamente dichos elementos no son tan conocidos y sus ventajas no son aún explotadas.

Un PSoC® (Programmable System on Chip o Sistema Programable en Chip, Figura 1) es un microcontrolador a grandes rasgos, ya que cuenta con CPU, entradas y salidas programables (I/O), así como con memoria interna. Su programación se realiza en lenguaje C.

Además, PSoC® cuenta con lógica programable y dispositivos lógicos con lo que es posible generar circuitos lógicos sin la necesidad de elementos externos. El sistema cuenta con un arreglo universal de bloques digitales (UDB, Universal Digital Block) programable en Verilog, con lo que se tiene una lógica altamente flexible basada solo en hardware.

Sin embargo, la principal diferencia que hace a un PSoC® más que un microcontrolador es que cuenta con una sección analógica programable, lo que le permite interactuar de manera directa con su ambiente y lo convierte en un elemento para la medición y procesamiento de señales en tiempo real (obteniendo una señal, amplificando, filtrando, etc.)

Otra ventaja que tiene el sistema es su alto nivel de flexibilidad, ya que todos sus pines pueden ser asignados como GPIO´s (General Purpose Input / Output) para datos digitales o señales analógicas.

Figura 1: PSoC® 4 (Cypress Semiconductor, 2016).

(Cypress Semiconductor, 2013)

(Tapia Farías, 2016)

Desarrollo

En el presente trabajo se desea implementar un sistema de medición de temperatura similar al presentado en publicaciones anteriores, con la diferencia de que el microcontrolador empleado será el PSoC® CY8C29466, ya que a futuro se planea expandir dicho sistema y se hará uso de las ventajas que presenta dicho tipo de controlador.

De igual manera, en pasadas publicaciones se han presentado los termopares, sus ventajas, tipos y aplicaciones. En este proyecto se desea implementar un termopar tipo K con su respectivo acondicionador de señal (AD595) y el despliegue de la información en una LCD, todo lo anterior controlado por el PSoC®.

El PSoC® es programado mediante el software PSoC Creator® (Christiano, 2015), el cual permite realizar una codificación usando bloques o una variante utilizando lenguaje C.

En primer lugar se generó un nuevo proyecto dentro de la interfaz del software, especificando el tipo de chip que se va a programar y donde se seleccionará la opción de codificar mediante lenguaje C o un lenguaje gráfico descrito como “Chip level”; para el desarrollo del proyecto se utilizó la programación gráfica.

La codificación consistió en colocar un bloque de termopar tipo k y un bloque de LCD, al interconectarlos se generó automáticamente la función de transferencia del sistema y después se modificaron los parámetros de cada elemento.

Una vez realizado lo anterior fue necesario asignar cada terminal a utilizar, tanto para la lectura del termopar por el ADC como para la LCD dentro del chip. Para lo anterior se tiene un alto grado de libertad, pero no el mismo que se tendría al realizar una programación en lenguaje C.

Finalmente se compiló el proyecto y el software generó los códigos para la programación del chip automáticamente.

A continuación se presentan los códigos obtenidos del sistema en lenguaje C. Por cuestiones de practicidad no se presentan todos los códigos, solo los más relevantes, sin embargo, siguiendo los pasos descritos anteriormente se puede obtener el mismo resultado que el presentado.

```
//--------------------------------------------------------------------
// C main line
//--------------------------------------------------------------------

#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules

#include "driverdecl.h"
#include "CMXSystem.h"
#include "CMXSystemExtern.h"
#include "TransferFunction.h"

#include "cmx.h"
#include "ProjectProperties.h"
#include "Custom.h"

// Channel includes
// ADC_00 Include
#include "CMX_ADC_CHAN.h"
// LCD_SHARED_02 Include
#include "cmx_lcd_chan.h"

void main( void )
{
   // Initialize Project
   M8C_EnableGInt;                  // Turn on interrupts
   SystemTimer_Start();
```

```
    SystemTimer_SetInterval(SystemTimer_64_HZ);
    SystemTimer_EnableInt();

    // Initialize Channels
    // ADC_00 Initialization
    ADCBUF_Start(3); // Power up ADC Buffer PGA
    ADC_Start(3); // Power up ADC
    AdcScanReset(); // Initialize ADC scanner
    ADC_GetSamples(0); // Turn on GetSamples
    // LCD_SHARED_02 Initialization
    LCDStartSharedChan();
    I2C_CFG &= 0xFC;// Disable I2C in case it's not used.

    // Initialize Variables
            SystemVars.ReadOnlyVars.pse_LCD = 0;
            SystemVars.ReadOnlyVars.pse_entrada = 0;

    // Driver instantiations
CMX_LCDNVU_Instantiate(&pse_LCD);
CMX_TCOUPLEKEXTCJC_Instantiate(&pse_entrada);

    // Custom initization code.
    CustomInit();
    // End Initialize Project

            while(1)
            {
        // Sync loop sample rate
#if ( SAMPLE_DIVIDER  )
        SystemTimer_SyncWait(SAMPLE_DIVIDER, SystemTimer_WAIT_RELOAD);
#endif
                        // update input variables
                        SystemVars.ReadOnlyVars.pse_entrada =
CMX_TCOUPLEKEXTCJC_GetValue(&pse_entrada);

        // Custom Post Input function
        CustomPostInputUpdate();

                        // run transfer function and update output variables
                        TransferFunction();
        // CustomPreOutputUpdate();
        CustomPreOutputUpdate();

                        // set outputs
                        CMX_LCDNVU_SetValue(&pse_LCD, (int)SystemVars.ReadOnlyVars.pse_LCD);
            }
}
//
//  TransferFunction.c
//

#include <m8c.h>
#include "DriverDecl.h"
```

```
#include "CMXSystem.h"
#include "CMXSystemFunction.h"
#include "CMXSystemExtern.h"
#include "FunctionParamDecl.h"

extern void pse_LCD_TransferFunction(void);

void TransferFunction( void)
{
        pse_LCD_TransferFunction();
}

//*****************************************************************************
//*****************************************************************************
//  FILENAME:  calibration.c
//  @Version@
//  `@PSOC_VERSION`
//
//  DESCRIPTION:  This files contains the calibration constansts for the
//             ADC. Currently these values are default values that
//             are not calibrated.
//
//-----------------------------------------------------------------------------
//     Copyright (c) Cypress Semiconductor 2009. All Rights Reserved.
//*****************************************************************************
//*****************************************************************************

#pragma abs_address:0x7FC0
const int CountsPerVolt = 25206;   // ADC gain for 0 to 2.6 volt range.
#pragma end_abs_address

#pragma abs_address:0x7FC2
const int ADC_Offset    = 0;      // ADC offset in counts
#pragma end_abs_address

#pragma abs_address:0x7FC4
// This array of offsets allows for custom calibration
// of each input that uses the mVolts channel. The offset
// will be in the drivers native units.  For the mVolts
// driver it will be in mVolts.  For a temperature driver
// it will be in tenths of degrees, etc.
const int imVolts_Chan_Offset[8] = {0,0,0,0,0,0,0,0};
#pragma end_abs_address

//-----------------------------------------------------------------------------
//  FUNCTION NAME: TCOUPLEKEXTCJC_GetValue(const CMX_TCOUPLEKEXTCJC_ParameterBlock *
thisBLK)
//
//  DESCRIPTION:
//    This function gets the reading from the ADC in mVolts, converts
//    it to degrees C and returns the result.
//-----------------------------------------------------------------------------
```

```
//  ARGUMENTS:
//    thisBLK  => Pointer to ParameterBlock for this instance.
//
//  RETURNS:
//    Interger value of degrees C.
//
//  SIDE EFFECTS:
//
//  THEORY of OPERATION or PROCEDURE:
//-----------------------------------------------------------------------------
int CMX_TCOUPLEKEXTCJC_GetValue(const CMX_TCOUPLEKEXTCJC_ParameterBlock * thisBLK)
{
  return (iGetChanMVolts(thisBLK->InPort) * thisBLK->ScaleFactor);
}
```

A continuación se presenta el diseño del circuito del sistema (Figuras 2 y 3), no se ha podido realizar una simulación adecuada dado que los software dedicados no cuentan con las librerías adecuadas.

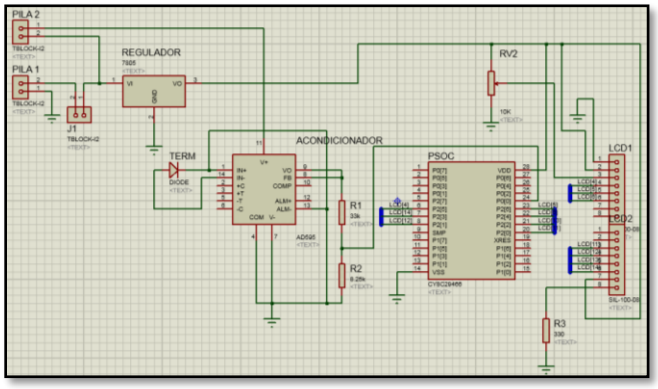

Figura 2: Circuito esquemático de sistema de medición de temperatura.

En el circuito esquemático, de izquierda a derecha, se puede apreciar la alimentación, para lo cual se utilizarán dos pilas de 9V, un regulador de voltaje 7805, un diodo para las terminales del termopar, el acondicionador de señal AD595, el divisor de tensión mostrado en publicaciones anteriores para el ADC del PSoC®, el PSoC® CY8C29466 y la LCD. Cabe destacar que el PSoC® no es un elemento para simulación.

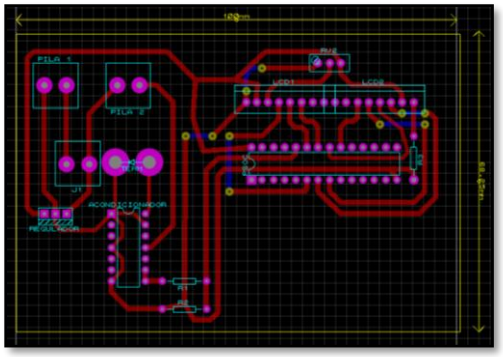

Figura 3: Diseño de PCB para sistema de medición de temperatura.

Por último se presenta el circuito físicamente montado sobre el contenedor final (Figuras 4, 5 y 6).

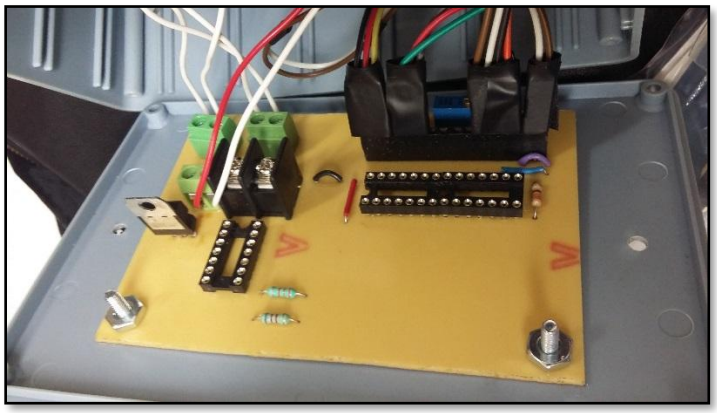

Figura 4: PCB ensamblada en contendor del sistema.

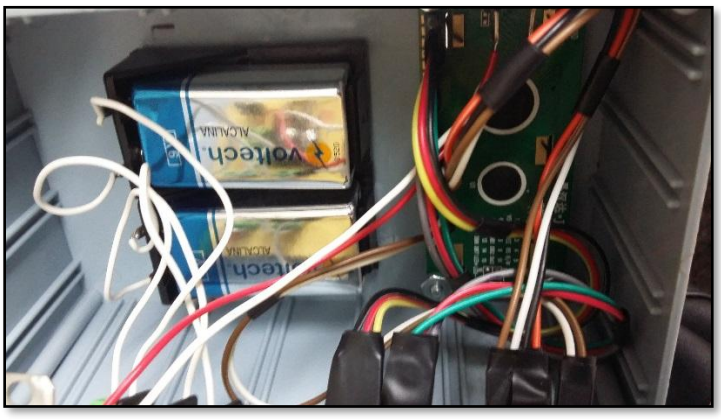

Figura 5: Alimentación y LCD para despliegue de datos montada sobre contenedor del sistema.

Figura 6: Sistema final encendido.

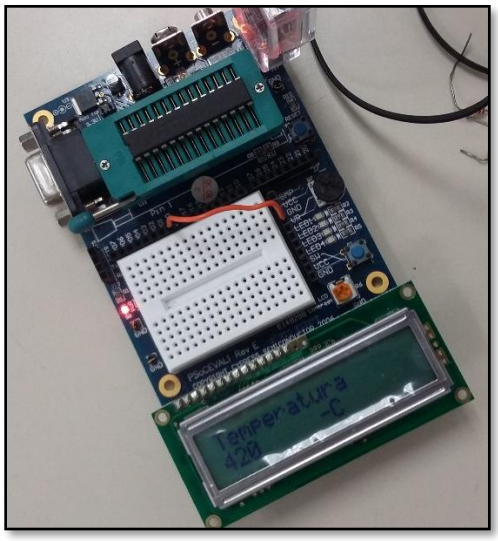

Figura 7: Pruebas usando tarjeta de desarrollo de Cypress Semiconductor®.

Conclusiones

El sistema aún no ha podido ser implementado experimentalmente ya que se deben hacer adecuaciones sobre la superficie sobre la que será aplicado, sin embargo, se han realizado pruebas usando la tarjeta de desarrollo de Cypress Semiconductor® (Figura 7), con lo que se ha podido determinar que el sistema funciona correctamente. En dichas pruebas se ha utilizado un potenciómetro como variador de voltaje para la entrada del ADC y el despliegue de la información es presentado en una LCD.

Se ha desarrollado un sistema de medición de temperatura utilizando un termopar tipo K y un PSoC® como controlador con la intención de conocer la temperatura sobre la superficie que

alojará un sustrato. Lo anterior será aplicado en el proceso de depósito de materiales para la generación de películas delgadas. El proyecto será ampliado posteriormente, por lo que serán necesarias las herramientas que ofrece el PSoC®.

Referencias

Christiano, M. (27 de Agosto de 2015). *All About Circuits*. Obtenido de All About Circuits: http://www.allaboutcircuits.com/projects/getting-started-with-psoc/

Cypress Semiconductor. (14 de Agosto de 2013). *Cypress Perform*. Obtenido de Cypress Perform: http://www.cypress.com/blog/psoc-creator-news-and-information/psoc-101-what-psoc

Tapia Farías, J. (2016). *PSOC-CHILE*. Obtenido de PSOC-CHILE: http://psoc-chile.es.tl/Psoc.htm