JIDeTEV
ISSN 2591-4219

Utilización de la técnica *Processor-in-the-Loop* para verificar el algoritmo de control digital en hardware DSC de *Texas Instruments*.

Luis C. Hubner [a,b] , Maria J. Martinez [a,b], Aldo J. Benítez [a,b], Roberto E. Carballo [a,b] *

[a] *Facultad de Ingeniería, Universidad Nacional de Misiones (UNaM), Oberá, Misiones, Argentina.*
[b] *GIDE, Universidad Nacional de Misiones (UNaM), Oberá, Misiones, Argentina.*
e-mails: luiscarloshubner9@gmail.com, mm4257222@gmail.com, javierbenitez@fio.unam.edu.ar , robertocarballo@fio.unam.edu.ar

Resumen

En este documento se describe paso a paso como preparar el DSC TMDSDOCK28335 para controlar la simulación de Hardware de un circuito Buck Converter, mostrando como acondicionarlo para la realización de pruebas, generar y modificar el código derivado de este, además de las configuraciones necesarias dentro de PSIM y Code Composer Studio para su correcto funcionamiento, obteniendo así resultados aproximados al circuito físico.

Palabras Clave *– Processor-in-the-Loop, PSIM, TMDSF28335, Simulación de Electrónica de Potencia.*

1 Introducción

Processor-in-the-Loop (PIL), es una técnica de simulación que permite probar el software de control en un microcontrolador, DSC (*Digital Signal Controller*, controlador digital de señales) o procesador, mientras se simula el sistema físico en un entorno de simulación. La aplicación de esta técnica permite desde validar algoritmos de control donde realmente serán implementados para el control de hardware real, hasta la evaluación de condiciones de funcionamiento difíciles de replicar en hardware real, como ser funcionamiento ante fallas en semiconductores.

Este trabajo tiene como objetivo divulgar la experiencia realizada al momento de preparar la simulación PIL, para una posible adopción en cursos como, por ejemplo, Control digital y no Lineal y/o Electrónica de Potencia, donde se realicen laboratorios que utilicen DSCs de *Texas Instruments*, así como también en los proyectos de investigación de la FIO en las áreas de electrónica de potencia y control.

En este documento se presenta la implementación de PIL particularmente en el entorno PSIM, en el cual se esquematiza el circuito para simular el hardware de potencia, mientras que en el bloque PIL se configura para funcionar junto al *Code Composer Studio* (CCS), permitiendo cargar y depurar el código en el DSC TMS320F28335 [1].

2 Pasos para la implementación de la simulación PIL

*Autor en correspondencia

A continuación, se describe cómo configurar y realizar una simulación *Processor-in-the-Loop* utilizando el bloque PIL. El proceso implica los siguientes pasos:

1. Preparar el código para la simulación PIL.
2. Creación del código.
3. Configuración del hardware.
4. Configuración en PSIM.
5. Depurar la simulación PIL con TI *Code Composer Studio*.
6. Visualización de graficas.
7. Preparación de la simulación PIL para múltiples frecuencias de muestreo.

1. Preparar el código para la simulación PIL:

Para el ejemplo del tutorial se utilizará el circuito de un convertidor reductor *(Buck converter)*, según el esquemático de la Fig. 1.

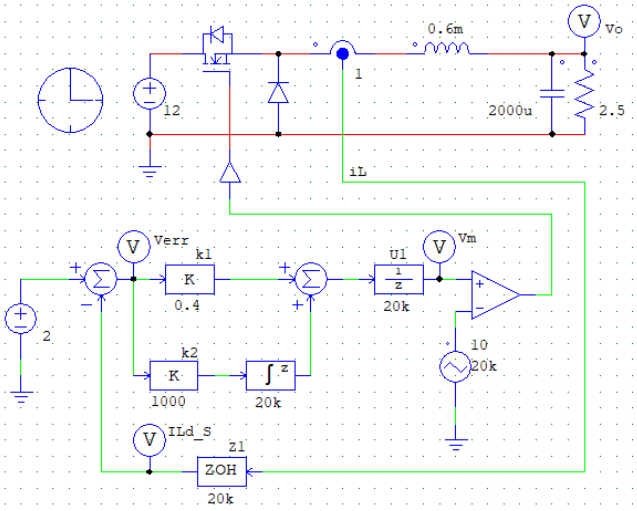

Fig. 1. Circuito *Buck Converter*.

En la Fig. 1 se puede observar claramente la etapa de potencia con cables de trazos rojos, mientras que los correspondientes a los de trazos verdes indican la etapa de control, la cual se trasladará posteriormente al código para el DSC.

2. Creación del código:

La creación de código permite que PSIM asista en la configuración de los módulos del DSC objetivo, por lo cual estas configuraciones no son necesarias de realizar "a mano" por el usuario [2].

Para realizar el paso a código, se debe modificar el circuito, en este caso, agregando un conversor A/D luego del sensor de corriente, y reemplazando el comparador por un PWM, y así definir el hardware a utilizar.

Para el convertidor A/D y el PWM se puede obtener de la librería de PSIM F2833x Target, la cual está ubicada en ***"ELEMENTS >> Sim Coder >> F2833x Target"***, el cual, para el convertidor está en ***"F2833x Target >> A/D converter"***, y para el PWM en ***"F2833x Target >> 1-phse PWM"***.

Una vez reemplazadas ambas partes, el circuito queda como se muestra en Fig. 2. La adición de estos elementos es esencial para definir el hardware a utilizar, ya que, sin este, no es posible la generación del código.

Nótese que el convertidor A/D posee un límite de tensión de entrada, el cual para voltaje continuo (DC) es de 0 V a +3 V, y para voltaje alterna (AC) es de -1,5 V a +1,5 V, para este caso se está trabajando con el convertidor en DC, por lo cual debe poseer un límite de tensión de entre 0 V y 3 V.

Además, antes de generar el código se debe configurar el bloque *"Simulation Control"*, en el apartado de *"SimCoder"*, configuramos el *"hardware Target"* en F2833x y el *"CPU Version"* en F28335, como se muestra en Fig. 3.

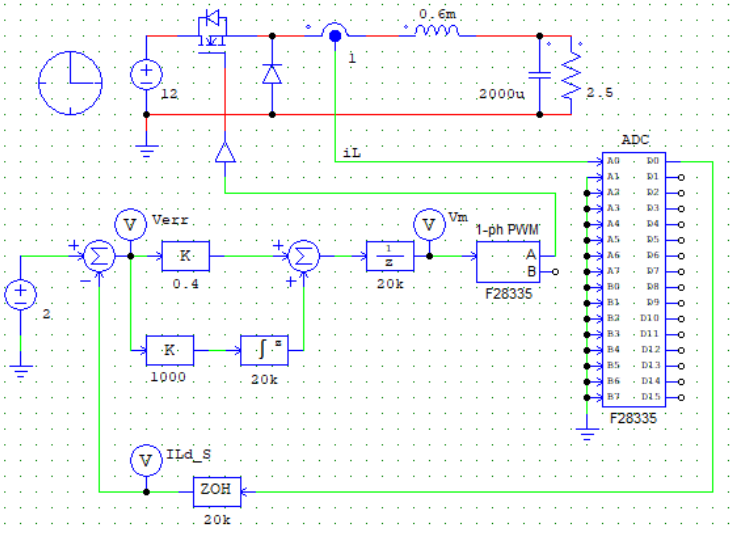

Fig. 2. Circuito modificado.

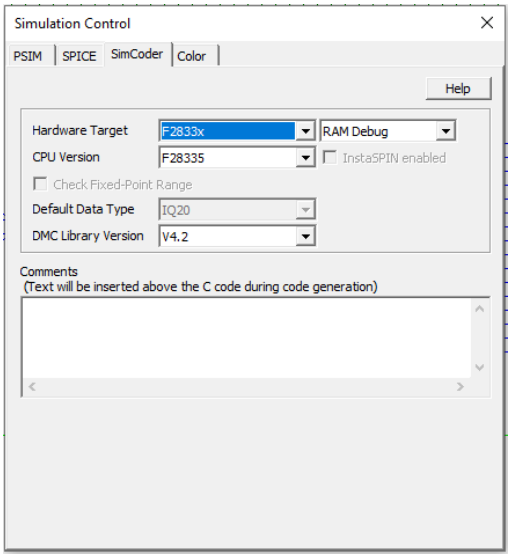

Fig. 3. Configuración del bloque *Simulation Control*.

Para generar el código en PSIM, se debe dirigir a ***"Simulate >> Generate Code"*** para así generar un código en C y los archivos necesarios para el *"Code Composer Studio V12"*, el código generado se alojará en una carpeta con la misma dirección que el archivo de PSIM, la cual poseerá el mismo nombre que este, pero con la terminación *"(C code)"*.

Una vez generado el código, se debe identificar las variables que interactúan con el PSIM, siendo estas:

- Desde el PSIM: variables en la salida del convertidor A/D.
- Para el PSIM: variables de entrada del generador PWM y cualquier otra variable interna.

Solamente es necesario realizar el cambio de dos cosas en el código:

1. Todas las variables de interfaz deben de definirse como globales.

2. Las instrucciones que asignan valores a la salida de ADC deben de ser comentadas en el código, esto es debido a que no hay ninguna entrada en el puerto de ADC real, y los valores de las variables de interfaz deben provenir todas del PSIM.

Con esto, también se debe de tener en cuenta que las salidas del generador PWM no pueden ser variables de interfaz, ya que las señales de salida del PWM son señales reales que aparecen en los puertos de salida del hardware y no pueden ser enviados a PSIM, sino que deben de generarse dentro de este a través de un circuito.

Del código, se pueden identificar las siguientes variables de interfaz: la salida de ADC como fADC1, la entrada al PWM (Vm) como fSUMP1, y la tensión de error (Verr) como fSUM1

Utilizando un editor de texto para abrir el archivo **.c**, se deben realizar las modificaciones a este, teniendo en cuenta que las variables de interfaz se deben de inicializar en cero (0), por lo que se deben identificar las siguientes variables, la salida de ADC como fADC1, la entrada al PWM (Vm) como fSUMP1, y la tensión de error (Verr) como fSUM1, con esto, se muestra a continuación el código luego de los cambios:

```
DefaultType	fADC1=0, fSUM1=0, fSUMP1=0;
DefaultType	fGblUDELAY1 = 0;
DefaultType	fGblILd_S = 0;
DefaultType	fGblVerr = 0;
DefaultType	fGblVm = 0;
interrupt void Task()
{
	//DefaultType	fUDELAY1, fSUMP1, fB1, fP2, fP1, fSUM1, fZOH1, fADC1, fVDC2;
	DefaultType	fUDELAY1, fB1, fP2, fP1, fZOH1, fVDC2;
	PS_MaskIntr(M__INT13);
	fUDELAY1 = fGblUDELAY1;
	//fADC1 = PSM_AdcGetValBySoc(0) * (1.0 * 3.0 / 4096); // Get DC value from ADC_A0.
	fVDC2 = 2;
	fZOH1 = fADC1;
	fSUM1 = fVDC2 - fZOH1;
	fP1 = fSUM1 * 0.4;
	fP2 = fSUM1 * 1000;
	{
		static DefaultType out_A = 0, in_A = 0.0;
		fB1 = out_A + 0.5/20000 * (fP2 + in_A);
		out_A = fB1;in_A = fP2;
	}
	fSUMP1 = fP1 + fB1;
	fGblUDELAY1 = fSUMP1;
#ifdef	_DEBUG
	fGblILd_S = fZOH1;
#endif
#ifdef	_DEBUG
	fGblVerr = fSUM1;
#endif
```

```
#ifdef    _DEBUG
          fGblVm = fGblUDELAY1;
#endif
PS_ExitTimer1Intr();
}
```

Las modificaciones se encuentran resaltadas en amarillo, las cuales son:

- las variables fADC1, fSUMP1, fSUM1 fueron definidas como globales, y sus definiciones anteriores fueron comentadas.
- se comenta la inicialización de la variable fADC1.

Como siguiente paso, se compila el código modificado. En CCS se debe ir a ***"Project >> Import Legacy CCS v3.3 Project"*** y de la carpeta donde se aloje el código, seleccionar al archivo **.pjt** e importarlo.

Tener en cuenta que la importación de este archivo debe de realizarse una sola vez, ya que se convertirá automáticamente a la versión v12.3 (o versión superior) y el archivo de proyecto podrá abrirse directamente, por más que el código fuente sea modificado posteriormente.

Seguidamente se debe definir el archivo de configuración de destino correspondiente al hardware del DSC utilizado, el cual se puede crear de la siguiente manera:

1. Abrir CCS e ir a la pestaña "Ver"*(View)* en la barra de menú superior.
2. Seleccionar "Configuraciones de destino" *(Target Configuration)* en el menú desplegable.
3. Hacer clic en el botón "Nuevo"*(New Target Configuration File)* en la esquina superior izquierda de la ventana "Configuraciones de destino"*(Target Configurations).*
4. Nombrar al archivo de configuración, como "F2833x.ccxml", y guárdarlo en una ubicación adecuada.
5. Seleccionar *"Texas Instruments XDS100v2 USB Emulator"* en el menú desplegable "Conexión" *(connection)* y hacer clic en "Siguiente".
6. Seleccionar "TMS320F2833x" en el menú desplegable "Dispositivo" *(Board or Device)* y hacer clic en "Finalizar" *(Save).*

Una vez que se haya creado el archivo F2833x.ccxml, se puede seleccionar como archivo de configuración de destino al importar un proyecto CCS v12.3 heredado o al compilar un proyecto existente.

Una vez completado los pasos anteriores, se necesita saber si se encuentra instalado XDAIS, estando este estándar diseñado para permitir que múltiples algoritmos coexistan en un sistema y compartir recursos, este es necesario para proyectos entre *Code Composer Studio* y un DSC porque define las reglas y directrices para la creación de algoritmos.

A continuación se procede a verificar que XDAIS esté instalado, para esto, ejecutar CCS y dirigirse a ***Window >> Preferences >> Code Composer Studio >> Products***, con lo cual aparecerá una lista de productos en *Discovered Products*, si se encuentra a XDAIS es porque está instalado, en caso contrario, este software es distribuido de forma gratuita por *Texas Instruments*, y se lo puede hallar fácilmente en internet, por lo cual, se siguen los siguientes pasos:

1. Descargar la versión más reciente de XDAIS que sea compatible con el sistema desde la página de descarga de XDAIS en el sitio web de *Texas Instruments*.
2. Descomprimir el archivo descargado con Zip y coloca la carpeta descomprimida en la carpeta raíz de TI.
3. Abrir *Code Composer Studio* e ir a ***Window >> Preferences >> Code Composer Studio >> Products***.
4. Hacer clic en el botón *Add* junto a *Product discovery path* y navegar hasta la ubicación donde se instaló el producto.
5. Una vez que se haya agregado la nueva ruta, en *Discovered Products* presionar *Refresh,* con esto, aparecerá la pestaña *Install Discovered Products*, en la cual se puede seleccionar a XDAIS, y luego presionar *Install*.
6. Reiniciar CCS.

Para compilar, en CCS, ir a ***"Project >> Build All"*** para construir el proyecto. Esto generará el archivo ejecutable de hardware **.out** en la carpeta "Nombre de Archivo (C *code*)/ *RamDebug*". Tener en cuenta que el archivo ejecutable **.out** también se puede compilar en *RamRelease y FlashRelease*.

3. Configuración en PSIM:

Para preparar el esquemático para la simulación PIL, primero asegurarse de que la ruta de la carpeta CCS sea compatible. Si CCS está instalado en la carpeta "c: \ TI \ ccs1230" para CCS v12.3 o cualquier versión similar, no es necesario hacer nada ya que PSIM ya ha incluido estas dos carpetas. Si CCS está instalado en una carpeta distinta a estas dos carpetas, debe agregarse a la ruta de PSIM yendo a ***"Options >> Set Path"*** y agregar la carpeta CCS a la ruta de búsqueda de PSIM.

El siguiente paso es modificar el circuito original, eliminando el circuito de control, excepto el generador PWM y el bloque de retardo de unidad U1, y reemplazándolo con un bloque PIL de la biblioteca PSIM en ***"Elements >> Control >> PIL Module >> PIL Block"***. La razón por la que el bloque de retardo de unidad está presente es porque el bloque PIL no incluye ningún retardo. Además, se utiliza un limitador con un rango de 0 a 3V en la entrada del bloque PIL. Esto es para simular el limitador incorporado del convertidor A/D en caso de que la entrada supere el límite.

Una vez que se coloca el PIL *Block*, entrar en sus atributos, y configurarlos como se ve en la Fig. 4.

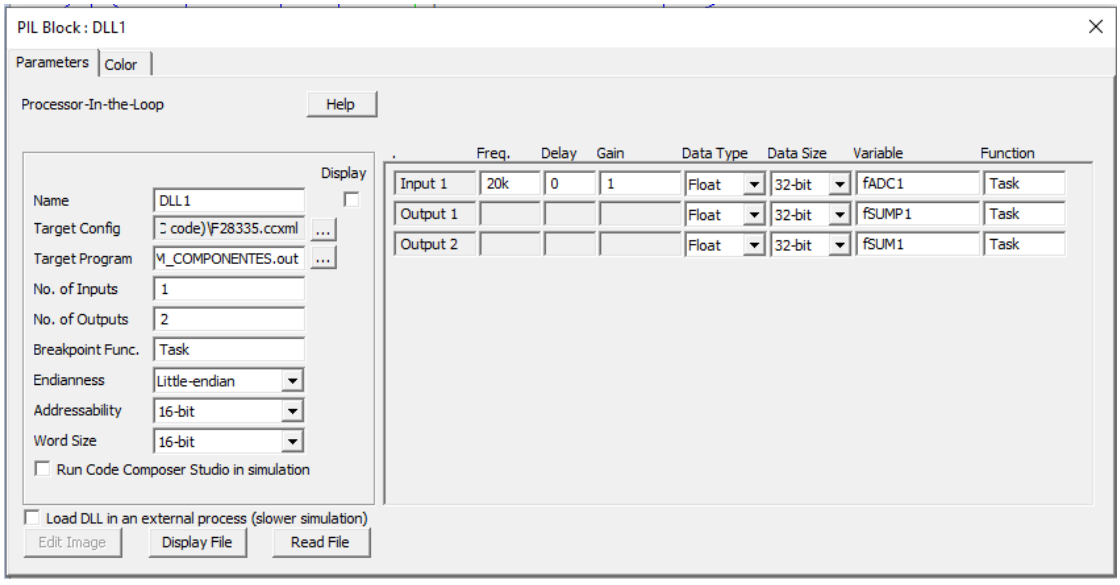

Fig. 4. Configuración de los atributos del bloque PIL.

Cuando se selecciona la opción ***Run Code Composer Studio in simulation***, esperar hasta que se lance CCS antes de continuar. Además, no se debe cerrar el CCS en medio de la simulación PIL. Para depurar el código en CCS, en PSIM, seleccionar ***Simulate >> Pause Simulation*** para poner la simulación PIL en espera. Luego, seguir el código en CCS. Se debe tener en cuenta que los valores enviados de PSIM al DSC permanecerán sin cambios cuando la simulación PIL esté en espera. Para reanudar la simulación, en PSIM, seleccionar ***Simulate >> Restart Simulation***.

Cuando se selecciona la opción ***Run Code Composer Studio in simulation***, si CCS no se inicia y si se observa la ventana de recursos de *Eclipse Platform* en su lugar, como se muestra Fig. 6, para abrir CCS ir a *Window >> Open Perspective* y seleccionar CCS *Debug*.

El esquemático del circuito modificado se muestra en la Fig. 5.

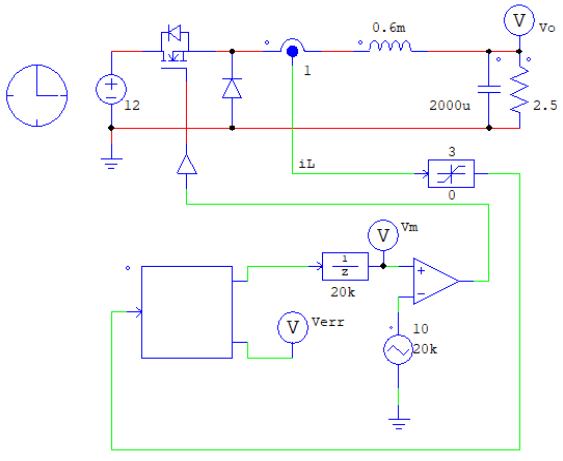

Fig. 5. Circuito *Buck Converter* con PIL *Block* implementado.

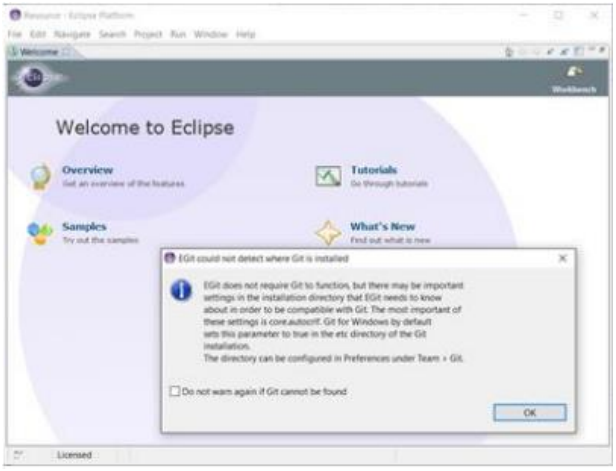

Fig. 6. Ventana de *Eclipse Platform*.

4. Configuración del hardware:

La única configuración de hardware necesaria para la simulación PIL es conectar el hardware del controlador a la computadora. En este ejemplo, un kit de experimentación de TI está conectado a la computadora a través de un cable USB, como se muestra Fig. 7.

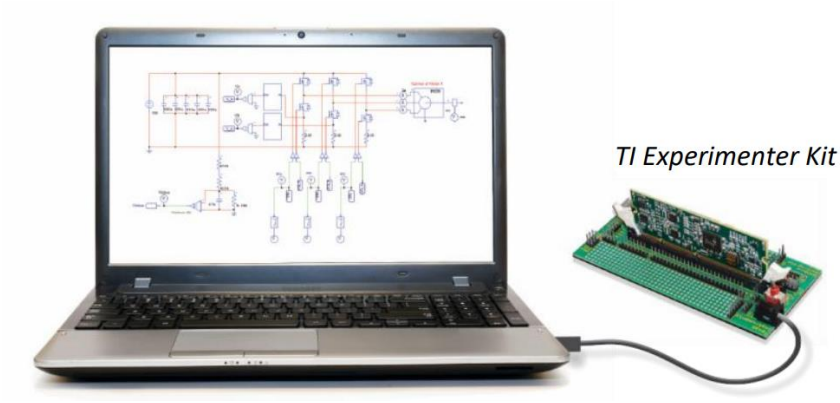

Fig. 7. Conexión de kit DSC a la computadora por USB.

Una vez que el hardware del controlador esté conectado, se debe ejecutar la simulación en PSIM seleccionando ***Simulate*** **>>** ***Run Simulation***. PSIM establecerá conexión con el hardware. Si la conexión se establece correctamente, se verá el cuadro de diálogo según la Fig. 8.

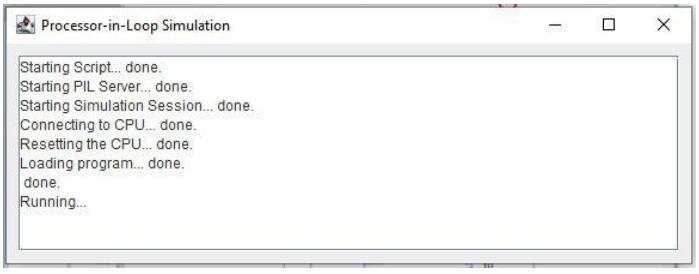

Fig. 8. Cuadro de dialogo de *Processor-in-Loop Simulation*.

No se debe cerrar este cuadro de diálogo durante la simulación. Se podrán mostrar formas de onda durante o al final de la simulación de la misma manera que se haría en una simulación regular de PSIM.

5. Depurar la simulación PIL con TI *Code Composer Studio*:

El código en el DSC se puede depurar con *Code Composer Studio* durante la simulación PIL. Para depurar el código en el ejemplo del convertidor *buck*, asegurarse de que la casilla de verificación *"Run Code Composer Studio in simulation"* esté marcada.

Hacer clic en ***Run PSIM simulation*** para iniciar la simulación. Aparecerá una ventana emergente igual al de la Fig. 9:

Fig. 9. Ventana emergente.

CCS se iniciará. Hacer clic en Aceptar para cerrar la ventana emergente y comenzará la simulación PIL. Mientras se ejecuta la simulación, en PSIM, pausar la simulación haciendo clic en el icono ***Pause Simulation***, como se muestra en Fig. 10.

Fig. 10. Indicación del icono *Pause Simulation*.

Luego regrese a CCS. En caso de que aparezca el mensaje:

No se puede encontrar un archivo fuente **.c**.

Ubique el archivo o edite la ruta de búsqueda de origen para incluir su ubicación.

Hacer clic en el botón ***Locate File…*** y ubicar el archivo **.c** en la carpeta correcta. Con el archivo **.c** cargado, la ventana CCS aparecerá como se muestra en Fig. 11:

Se podrá establecer el punto de interrupción, y entrar o salir del código, y depurar el código de la misma manera que sí lo hiciera en una implementación de un controlador real. Es posible inspeccionar los valores de las variables y los registros.

Una vez que se haya terminado la depuración en CCS, volver a PSIM y hacer clic en ***Restart Simulation*** para reanudar la simulación PIL.

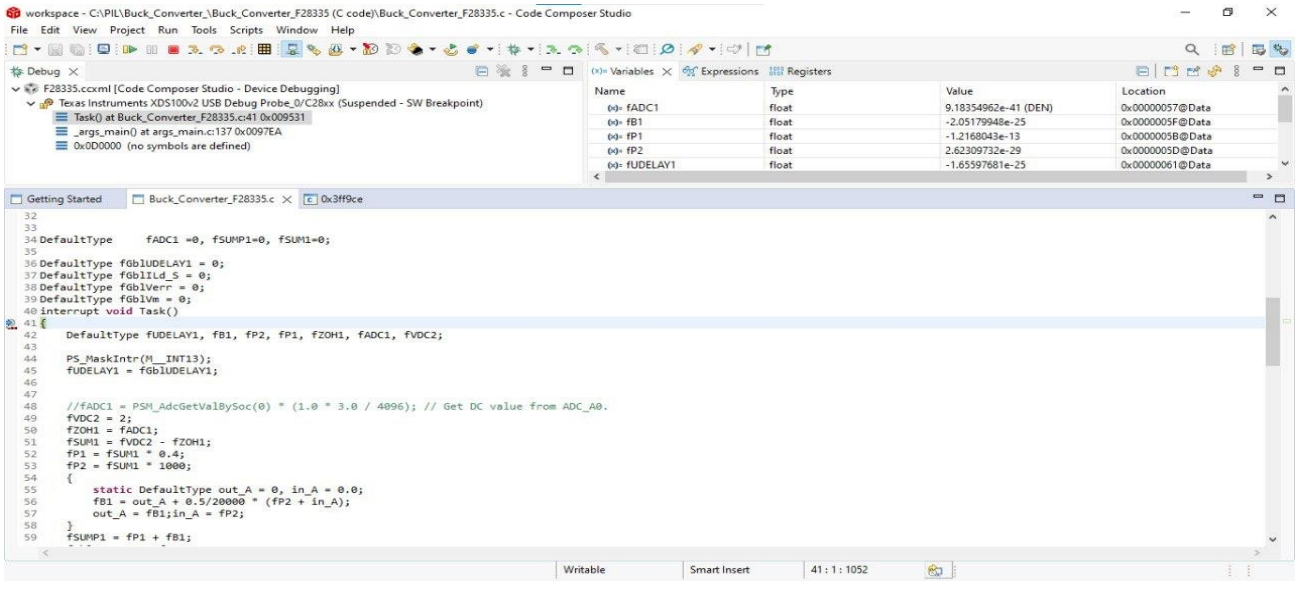

Fig. 11. Ventana de CCS con la simulación PIL en proceso.

6. Visualización de graficas:

Al terminar la simulación PIL, se abrirá la ventana de *simview*, seleccionar *Add All*, para agregar todas las variables que graficar, obteniendo así la Fig. 12.

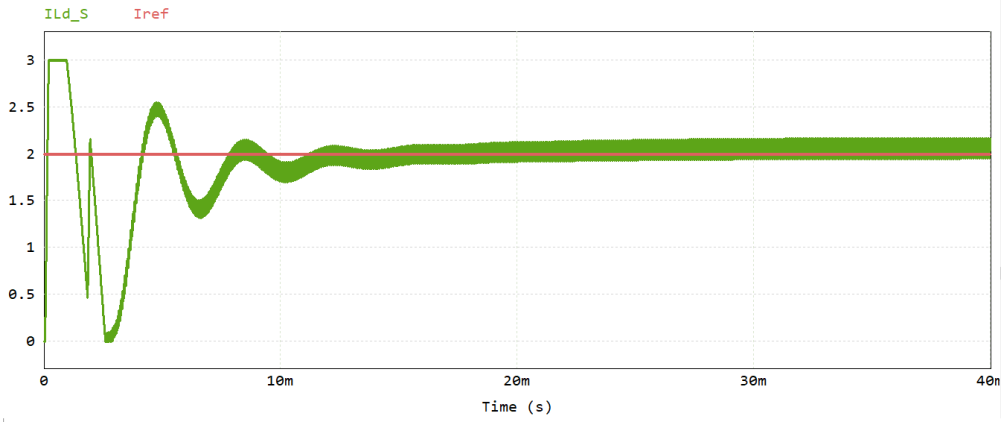

Fig. 12. Grafica de variables circuito *Buck Converter* con bloque PIL.

Para contrastar con los resultados de simulación obtenidos solamente en PSIM, donde todos los bloques de control son implementados como en la Fig. 1, en la Fig. 12 se presentan las mismas variables graficadas. Notar que aparece en el resultado de la Fig. 12 una saturación, la cual es introducida para modelar el efecto del conversor A/D que realizaría el DSC si se implementara físicamente.

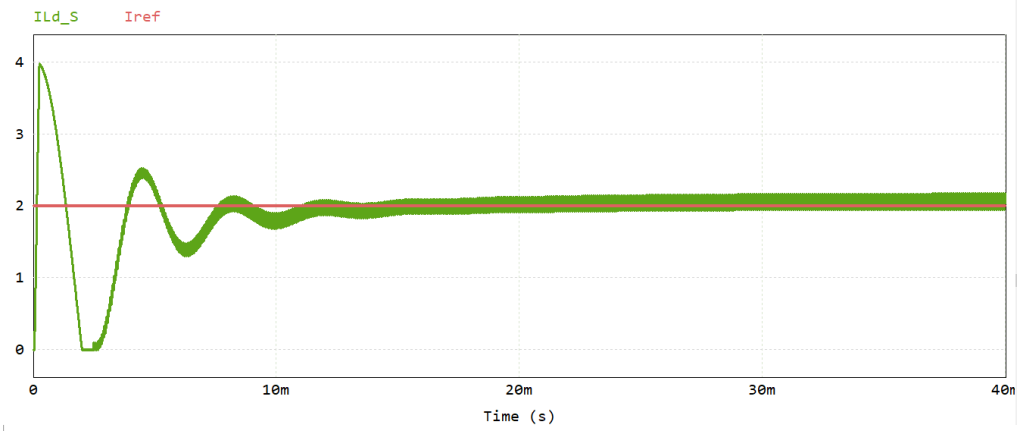

Fig. 13. Grafica de variables circuito *Buck Converter* de la Fig. 1.

Observando las gráficas anteriores se puede apreciar leves diferencias en régimen transitorio, pero en régimen permanente los resultados son iguales a los del circuito sin el bloque PIL.

7. Preparación de la simulación PIL para múltiples frecuencias de muestreo:

En trabajos futuros se explorará el análisis e implementación de estrategias de control con múltiples frecuencias de muestreo, por lo que a continuación se detalla como configurar la simulación PIL para estos casos.

Se recomienda que, cuando haya múltiples frecuencias de muestreo, se muestreen todas las variables de entrada con la frecuencia de muestreo más alta y luego se reduzca la velocidad de muestreo de las variables seleccionadas después del conversor A/D. Por ejemplo, si hay dos entradas con una a 10 kHz y la otra a 1 kHz, muestrear ambas a 10 kHz y luego reducir la velocidad de muestreo de la segunda entrada a 1 kHz a través de un retardo de orden cero a 1 kHz.

3 Conclusiones

En este trabajo se validó el procedimiento para configurar una simulación PIL mediante PSIM y el DSC TMS320F28335 de *Texas Instruments*, detallando cada uno de los pasos a seguir. Se espera que esta herramienta pueda ser utilizada por materias que requieren laboratorios para programar con este DSC, así como también en las líneas de investigación actual de la FIO.

4 Referencias

[1] "Tutorial: *Processor-In-The-Loop Simulation*", POWERSIM , Bergen, Noruega, Jun, 2019.

[2] "*Auto Code Generation for F2833x Target.pdf"*, POWERSIM , Bergen, Noruega, Version 2020a, May, 2020. [Online]. *Available*: *Simcoder Manual* (powersimtech.com)

[3] "C2000™ DIMM100 *Experimenter's Kit Overview*", *Texas Instruments*, Dallas, Texas, Estados Unidos, October 2019. [Online]. Available: TMS320C2000 DSC Experimenter Kit Overview (Rev. I) (ti.com)