PLECS

Tutorial

Using the PLECS DLL Block

Tutorial Version 1.1

www.plexim.com

- Request a PLECS trial license
- Check the PLECS documentation

1 Introduction

The PLECS DLL block is useful for testing custom C control code with a PLECS model. Compared with the C-Script block, C control code can be included in a model without having to conform to the function structure of the C-Script block. The control code can be written using the same file structure that is used for the DSP control code.

In this tutorial you will learn how to generate and debug a DLL file that implements a digital PI controller.

Before you begin Ensure that you have installed Microsoft Visual Studio Community (select the "Desktop development with C++" to be installed and then "Visual C++" development settings) which will be used to generate the DLL. If you wish to generate 64-bit DLLs, you first need to install the Microsoft Windows SDK and .NET Framework 4. This is a free package that allows the creation of applications for 64-bit platforms. During the installation, you can omit the installation of the .NET component. Also ensure the files `DllHeader.h`, `main.c`, `dll_block_1.plecs`, `dll_block_2.plecs` and the folder `Visual studio projects` are all located in your working directory.

Note: A 64-bit DLL will only work within a 64-bit version of PLECS.

Note: This model contains model initialization commands that are accessible from:
PLECS Standalone: The menu **Simulation + Simulation Parameters... + Initializations**
PLECS Blockset: Right click in the **Simulink model window + Model Properties + Callbacks + InitFcn***

2 Creating a DLL File

2.1 Create a project

There are several methods for creating a DLL file. In this tutorial, we will use Microsoft's Visual Studio Community 2022 because it is free and offers debugging capabilities. You will write the code in C, although it should be noted that C++ can also be used.

Your Task:

1 Run Microsoft Visual Studio Community and select the menu **File + New + Project...** Select **Windows Desktop Wizard** as the project type. Enter `pi_controller` as the project "Name" and "Solution name". You can choose any location you want for the project directory since you will need to place the prepared files inside of it. Click **Create**.

2 Under "Application type" select "Dynamic Link Library (.dll)" and under "Additional Options", select "Empty Project". Click **OK**.

> **Note:** You may also right-click on "pi_controller" within the Solution Explorer window, go to **Properties**, and make sure that the "Platform" set as "x64".

3 Copy the file `DllHeader.h`, which is found in the PLECS install directory under `...\include\plecs`, into the project code directory, `...\pi_controller\pi_controller`. Add this file to your project header files by right-clicking on the "Header Files" folder in the "Solution Explorer" window, then choose the **Add + Existing Item**... option. Select `DllHeader.h` and click **Add** . This header file contains macros and function definitions required by the PLECS solver.

4 Copy the file `main.c` from the tutorial reference file directory into the project code directory and add this to your project "Source Files" in the same fashion as described in the previous step. This file contains boilerplate code for interfacing with the PLECS solver.

2.2 Write the control code

You will write your control code in the function *plecsOutput* in *main.c*. This function is automatically called during each sample period. The PI control law is given as follows:

$$i = i + t_\mathrm{s} e \tag{1}$$

$$y = k_\mathrm{p} e + k_\mathrm{i} i \tag{2}$$

where i is the integral action, e is the error, and y is the controller output. The proportional and integral gains, k_p and k_i, along with t_s are passed into the DLL block as external parameters.

The input and output of the DLL block, along with any internal states and external parameters are accessed using the array fields of the *SimulationState* structure. A list of commonly-used fields for a *SimulationState* pointer named *aState* is given below:

- `aState->inputs[n]` - the n_th input to the DLL block.
- `aState->outputs[n]` - the n_th output of the DLL block.
- `aState->states[n]` - the n_th internal state of the DLL block.
- `aState->parameters[n]` - the n_th external parameter.

The *states* field is an array for storing static double variables. Static variables can also be created locally; however, if the solver needs to access a static variable for steady-state analysis or model initialization, the *states* field should be used.

It should be noted that the size of each array, n, must be specified using the fields of the `SimulationSizes` structure. The fields in this structure that set the size of the array fields in *SimulationState* are: `numInputs`, `numOutputs`, `numStates` and `numParameters`.

Your Task:

1 At the top of `main.c`, create global double variables named k_p, k_i and t_s. In the function `plecsStart(struct SimulationState* aState)`, map the external parameters back to their original names. For example:
```
kp = aState->parameters[0];
```

2 Use `#define` statements at the top of `main.c` to map the DLL input to `e` and the DLL output to `y`. Keep in mind that *define* statements don't end with semicolons. For example:
```
#define e aState->inputs[0]
```

3 Also use a `#define` statement to map the first element of the states field to `i`. This will be used to represent the integral action.

4 In the function `plecsSetSizes(struct SimulationSizes* aSizes)`, define the sizes for the array fields of *SimulationState*. For example:
```
aSizes->numParameters = 3;
```

5 Implement Eqs. `eq:integral` and `eq:output` in the output function, `plecsOutput`.

2.3 Compile and run

Your Task:

1 Compile the code into a DLL using **Build + Build Solution**.

2 Copy the model `dll_block_1.plecs` from the tutorial reference file directory into the same project directory as the DLL, `...\x64\Debug`.

3 Open the PLECS model, look under the mask of the PI controller **Ctrl + U** and place a PLECS DLL block from the component library under “Control” in the sub-librarie “Functions & Tables”, between the Signal Inport and Outport blocks and connect the appropriate terminals. Configure the block with the parameters shown in Fig. 1.

4 Run the simulation and compare the output voltage and current with the demo model from the PLECS Demo Model library, *Buck Converter with Peak Current Control*.

At this stage, your model should be the same as the reference model, `dll_block_2.plecs` and your control code should be be similar to the `main.c` file in the reference `\Visual studio\pi_controller\pi_controller\main.c`.

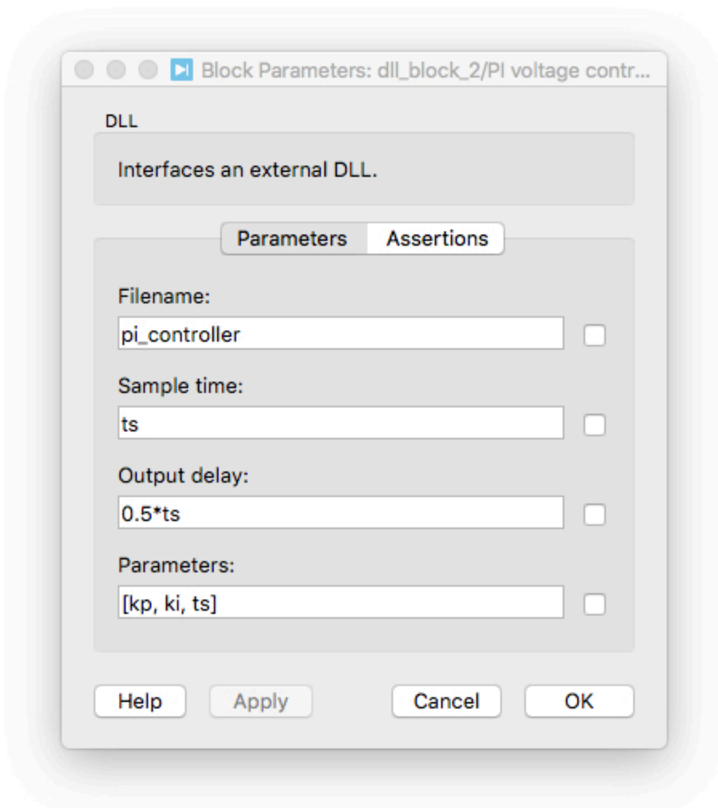

Figure 1: DLL block parameters

3 Debugging a DLL File

You can also debug your control code while it is running in a PLECS model. This can be done directly inside Visual Studio.

Your Task:

1. Attach the DLL to the PLECS process by selecting `PLECS.exe` with the title of your model under **Debug + Attach to Process...**. Under **Attach to:** select the automatic option and make sure **Native code** is selected, otherwise select it manually.
2. In `main.c`, create a breakpoint at the line `y = kp*e + ki*i` by clicking on the gray vertical bar to the left of the line.
3. Run the PLECS simulation.
4. In the “Watch” window at the bottom, select the “Autos” tab and notice the “inputs” and “outputs” fields of the “aState” struct. These fields represent `e` and `y`. Step through the simulation using the “F5” key and observe these values changing.

Revision History:

Tutorial Version 1.0	First release
Tutorial Version 1.1	Adapting the tutorial to 64-bit

How to Contact Plexim:

☎	+41 44 533 51 00	Phone
	+41 44 533 51 01	Fax
✉	Plexim GmbH Technoparkstrasse 1 8005 Zurich Switzerland	Mail
@	`info@plexim.com`	Email
	`http://www.plexim.com`	Web