Programming the DMCC in C

Task

This tutorial will teach you how to write your first program on a dual motor control cape (**DMCC**) through the **BeagleBone** microcontroller. The **DMCC** is a stackable board that can control up to two motors. This tutorial has **6 steps** that should take no longer than 15 minutes to complete. After completing these steps you should be able to program your **DMCC** and use it in your various projects.

Motivation

This project will teach you to control your open source board, manipulate your machinery in the ways required by your project. The **DMCC** can control motors of up to 7 amps, so there is a lot you can do with it once you have this knowledge.

Audience

This tutorial is for **BeagleBone** users who would like to use the **DMCC** board for their projects. Audience members should be familiar with C, navigating a linux terminal, and accessing SSH. They should also be experienced enough to perform basic tasks like calling methods found in the **DMCC** library.

Requirements

You will access to a **BeagleBone** through either SSH Secure Shell or through direct USB connection. You will also need internet on your **BeagleBone**. You will also need to download the **DMCC** library from **github**. If you do not have a active internet connection you will need to get the library code from **github** into the **BeagleBone** from https://**github**.com/Exadler/**DMCC**_Library.

Caution

You may see warnings if you do not have an active internet connection when trying to download the library code from **github**. If this happens reestablish your internet connection and try to download the library again. You also may receive errors if you are trying to talk to a **DMCC** that is not connected to the **BeagleBone**. If this happens, reconnect or try a different board number.

Step 1: Download the DMCC Library Code

In this step, you will retrieve the DMCC Library code from github to begin programming on the BeagleBone microcontroller for the DMCC (Dual Motor Control Cape). There are two options to retrieve the code. Choose which one that you prefer, and if one method fails try the other.

STEP REQUIREMENTS: You will need an active internet connection or a way to get the DMCC Library code from github for this step

1) Get the code from github
   - **Case 1:** Type the command

     `git clone git://github.com/Exadler/DMCC_Library`

```
root@beaglebone:~/prod_v1# git clone git://github.com/Exadler/DMCC_Library
Cloning into DMCC_Library...
remote: Counting objects: 14, done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 14 (delta 5), reused 14 (delta 5)
Receiving objects: 100% (14/14), 13.00 KiB, done.
Resolving deltas: 100% (5/5), done.
root@beaglebone:~/prod_v1#
```

Figure 1: Downloading the Library Code from github

2) Navigate to the directory

`cd DMCC_Library` **OR** `cd DMCC_Library-master`

```
root@beaglebone:~/prod_v1# cd DMCC_Library
root@beaglebone:~/prod_v1/DMCC_Library# ls
DMCC.c  DMCC.h  Makefile  getCurrent.c  getQEI.c  setMotor.c  setPID.c
root@beaglebone:~/prod_v1/DMCC_Library#
```

Figure 3: Navigating to the DMCC Directory

After successfully downloading the base library code to program the DMCC (Dual Motor Control Cape) and navigating to the directory which holds the code, you will be able to proceed to **Step 2** *(Creating Your First DMCC Program)* implementing the DMCC library.

Step 2: Creating Your First DMCC Program

In this step, we will go through creating a file for your first DMCC Program. This step will be a basic building block for all of your future DMCC Programs.

1) Create a new file or edit a current file with your preferred editor (replace FILE_NAME)

```
vim FILE_NAME.c   OR   nano FILE_NAME.c
```

2) Include the following directories by adding the following to the beginning of the file

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "DMCC.h"
```

3) Start writing code in the file by creating a main function after the included directories

```
int main(int argc, char *argv[]){return 0;}
```

4) In the main function, start the session, add your code, then end the session

```
int session = DMCCstart(boardNum);
YOUR_CODE_HERE
DMCCend(session);
```

   a. Specify which board number you want to talk to in
      - You will get errors if you are talking to a board that is disconnected

```
Error: No version number found
Error in write address 0x02
```

      - Refer to www.exadler.com/dmcc to find the board number you should use for some particular BeagleBone board

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "DMCC.h"

int main(int argc, char *argv[]){
    int session = DMCCstart(0);

    //-----YOUR_CODE HERE-----//

    DMCCend(session);
    return 0;
}
```

Figure 4: Sample Code Using the DMCC Library

After creating your code, you will then need to compile it to ensure that your program is correctly written (has no syntax or compilation errors). You will also need to create an output file so that you can run your program. How to do this will be explained in **Step 3** *(Compiling Your Code Using a Makefile).*

Step 3: Compiling Your Program Using a Makefile

In the step, you will be updating the Makefile so that you can compile your program. This is a standard way to compile your code in Linux, but this step will allow you to compile all your programs with a simple command.

1) In the *all* option, add an option name that you want to compile your file with

   `all: …, …, OPTION_NAME`

2) Create an option and add *YOUR_FILENAME.c*, *DMCC.c, DMCC.h* to check when to compile

   `OPTION_NAME: YOUR_FILENAME.c DMCC.c DMCC.h`

3) Add the gcc command on the next line by (replace OPTION_NAME with chosen name)

   `$(CC) -o OPTION_NAME YOUR_FILENAME.c DMCC.c`

```
CC = gcc -Wall

all: getQEI setMotor getCurrent setPID example

getQEI: getQEI.c DMCC.c DMCC.h
	$(CC) -o getQEI getQEI.c DMCC.c

setMotor: setMotor.c DMCC.c DMCC.h
	$(CC) -o setMotor setMotor.c DMCC.c

getCurrent: getCurrent.c DMCC.c DMCC.h
	$(CC) -o getCurrent getCurrent.c DMCC.c

setPID: setPID.c DMCC.c DMCC.h
	$(CC) -o setPID setPID.c DMCC.c

example: example.c DMCC.c DMCC.h
	$(CC) -o example example.c DMCC.c
```

Figure 5: Updating the included Makefile

Now that you have compiled your program, you need to run your file and call the command to compile your file from the command line. **Step 4** describes how you can go about running your program.

Step 4: Running Your DMCC Program

Once you have your file, you have to run it from the shell on the BeagleBone. This step will tell you a step-by-step instruction set on how to run your file.

1) Compile your program – either case will work to compile your program. It is recommended that you use Case 2 however, to ensure that no file that you use has errors.
   - **Case 1: Compile only your newly created file or program**

```
make OPTION_NAME
```

```
root@beaglebone:~/prod_v1/DMCC_Library# make example
gcc -Wall -o example example.c DMCC.c
root@beaglebone:~/prod_v1/DMCC_Library#
```

Figure 6: Compiling Your Program

OR

   - **Case 2: Compile all files in the directory**

```
make
```

```
root@beaglebone:~/prod_v1/DMCC_Library# make
gcc -Wall -o getQEI getQEI.c DMCC.c
gcc -Wall -o setMotor setMotor.c DMCC.c
gcc -Wall -o getCurrent getCurrent.c DMCC.c
gcc -Wall -o setPID setPID.c DMCC.c
gcc -Wall -o example example.c DMCC.c
root@beaglebone:~/prod_v1/DMCC_Library#
```

Figure 7: Compiling All Programs in the Directory

2) Run your file

```
./OPTION_NAME
```

```
root@beaglebone:~/prod_v1/DMCC_Library# ./example
```

Figure 8: Running Your Program

After running your file, as an additional option, you can pass in arguments from the command line. This allows the user to change input values to their program in real time. In **Optional Step 5**, we explain how to do this.

Optional Step 5: Getting Command Line Arguments into Your Program

As an option, you can get arguments from the command line so that you can quickly make calls to your program with different variables without editing and re-compiling your program.

a. Open your file again using your preferred editor
   `vim FILE_NAME.c` **OR** `emacs FILE_NAME.c` **OR** `nano FILE_NAME.c`
b. argv[] contains all the arguments from the command line. So, get the arguments by getting the value of the array at a certain index.
   i. To get an integer argument use the atol(argv[#])
      - E.g. to get one argument from the user

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "DMCC.h"

int main(int argc, char *argv[]){
    if (argc != 2){
        return 1;
    }
    int boardNum = atol(argv[1]);

    int session = DMCCstart(0);

    //-----YOUR_CODE HERE-----//

    DMCCend(session);
    return 0;
}
```

Figure 9: Adding Code to Get Arguments From the Command Line

c. Re-compile your code with the new arguments
   `make` **OR** `make OPTION_NAME`
d. In the command line, to run the function

```
root@beaglebone:~/prod_v1/DMCC_Library# ./example 1
```

Figure 10: Running Your Function with Arguments