What's this?

An interface for Medtronic CGM devices.

I've created this document to help you build and configure your MMCommander for your Medtronic device.

Oopss.. I nearly forgot. As you can imagine, Medtronic is not related to this project in any way and the author (me) assumes no responsibility for any damage caused by o with any of the information found in this repository. **USE IT AT YOUR OWN RISK**.

What do I need?

First of all, if you want to use NightScout, apart from the MMCommander you'll need an OTG-capable android phone and an OTG cable (micro-USB male to USB **female**). Check the NightScout webpage and the "CGM in the cloud" Facebook group for more info about this.

The MMCommander is composed of three things:

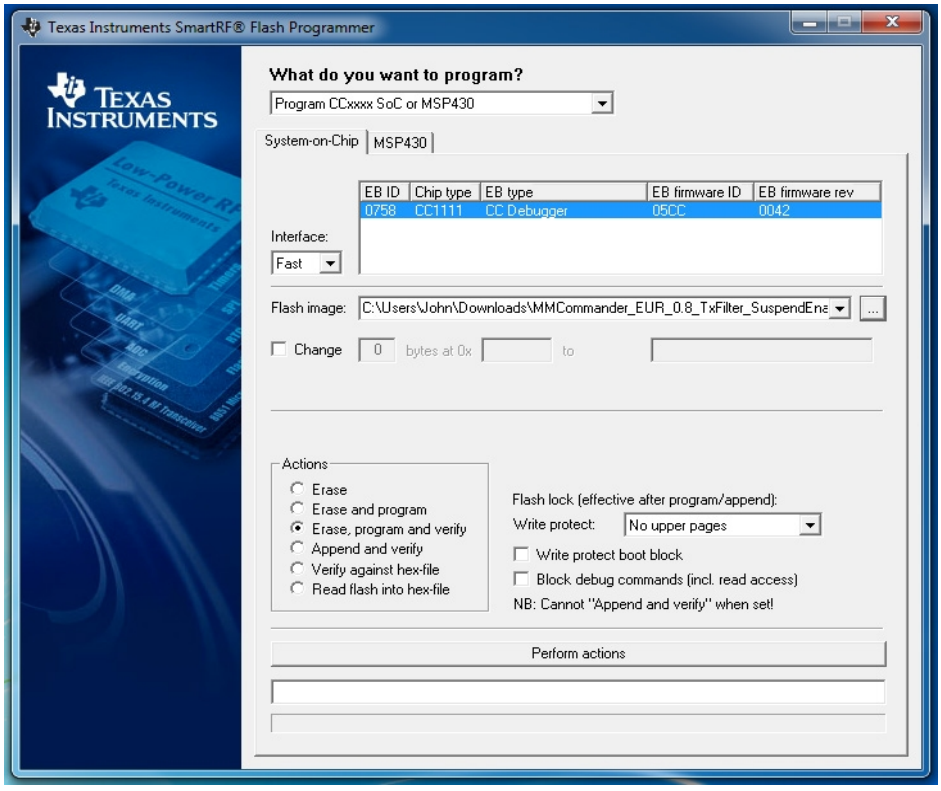
- A CC1111 USB EMK868-915 from Texas Instruments (http://www.ti.com/tool/cc1111emk868-915)
- One CC Debugger to be able to program the CC1111 board. (http://www.ti.com/tool/cc-debugger)
- And the software found in this repository

The cost of the whole MMCommander pack is not too high (at the time I'm writing this) :

- $49 for the CC1111 USB on TI's website.
- $49 for the CC Debugger on TI's website (which you'll use only to program the CC1111 and, after that, you can share with others if you want).

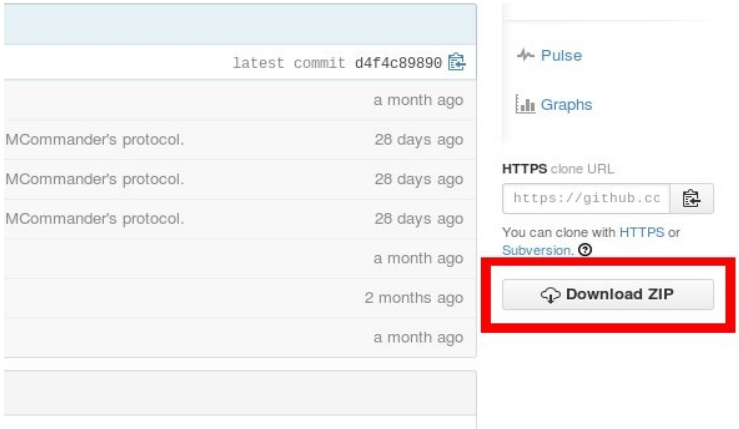
Now that I have the hardware... What do I do now?

Now that you have the hardware, you have to install (in a Windows machine) the software that provides Texas Instruments to program the CC1111. It's called SmartRF Flash Programmer and looks like this.



Is it working?. Well... lets download the MMCommander image file.

If you're not used to download info from Github I would recommend to download the complete repository in a ZIP file. Go to the repository's main page (http://github.com/jberian/mmcommander) and click on the “Download ZIP” button on the right side of the page.



Decompress the ZIP file and, in the “hex” folder, you'll find two HEX files:

- MMCommander_EUR_0.87_NoTx.hex
- MMCommander_US_0.87_NoTx.hex

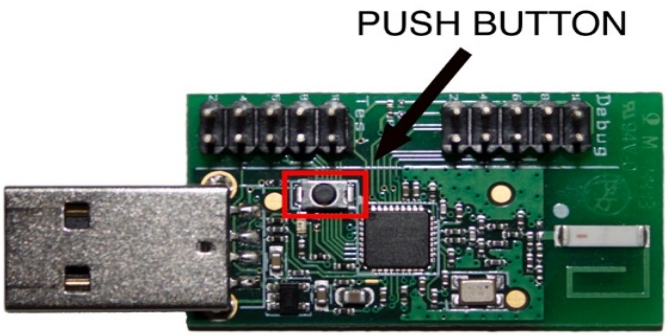
These files are the result of compiling the sources for US and EUR frequencies WITHOUT transmission enabled. This is the safest implementation possible and using these binaries you won't be able to affect in any way your pump or sensor. You'll be a passive listener and will be able to listen to your sensor and glucometer. When configuring the android-uploader for Medtronic you'll see options like obtaining the calibration factor from the pump or history retrieving... these options won't work since you won't be able to talk to your pump or Medtronic receiver, ok?.

If you want to do more testing or enable other features which for you need to enable the transmission part, install the IAR Embedded Workbench you'll find with the kit you just bought, ask for a 30 day license and compile the sources with the options you need enabled. Check the configuration header file for that. Be careful when doing tests.

The MMCommander, in any version with transmission enabled, is capable of receiving and TRANSMITTING any string. What would happen if someone hacks your smartphone/computer and sends commands to your pump?... for instance... a bolus command. Would be dangerous, right?. That's why I implemented what I called the "transmission filter". This filter won't let you (or your hacker) send some certain types of commands so that its use is safer.

And what's the suspend support?. Yes... It's a transmission filter that will allow you to send suspend commands. This will enable you/us to implement a Medtronic's "Low Glucose Suspend" if you need it.

And what happens if I want to have the Tx Filter but, in certain cases, I want to send "forbidden" commands?. No problem. The CC1111 USB board has a push button and a green led.



If you press the push button for a couple seconds the led will turn on and then the filter will be disabled. Whenever you have the green led on you'll be able to send whatever you want. After that, if you want the filter back, just press the push button again for a couple seconds until the led turns off.

Then… Which one do I need?. If you need to transmit I would recommend to use the one with suspend support for your zone, if not, the one without TX. Again… at your own risk, remember?

Now that you know which one is the one you need, get the HEX file and let's go back to the SmartRF Flash programmer and program the CC1111.

First connect the CC Debugger to the CC1111 debug port using the ribbon cable that comes with the CC Debugger and plug both USB connectors to your computer. Pay close attention to the red line on the ribbon cable marking pin 1: connect the cable as you can see in the following picture.

The CC Debugger has a led. If it's red, press the reset button on the CC Debugger and it should turn green. If not, something is not correctly connected. Check the ribbon cable, its orientation, and that both USB ports have to be connected to the computer. Take a look to the previous picture to check.

Note: some people have reported that their cables were incorrectly assembled and they had to connect the flat cable with the red line facing the opposite side. Don't be afraid to test, you won't break anything.

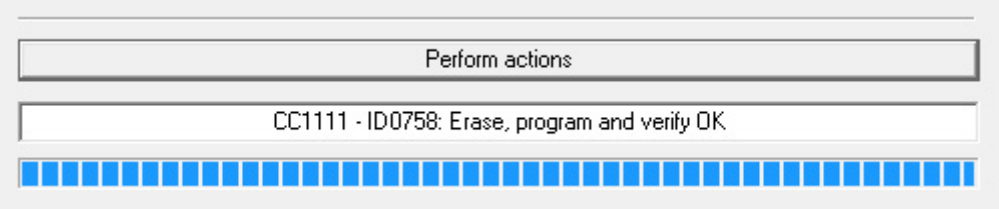
Now everything is connected, the led on the CC Debugger is green... It's time to configure the Flash programmer. Everything should be configured as the picture in which I showed you how it was like, remember?

Check list:

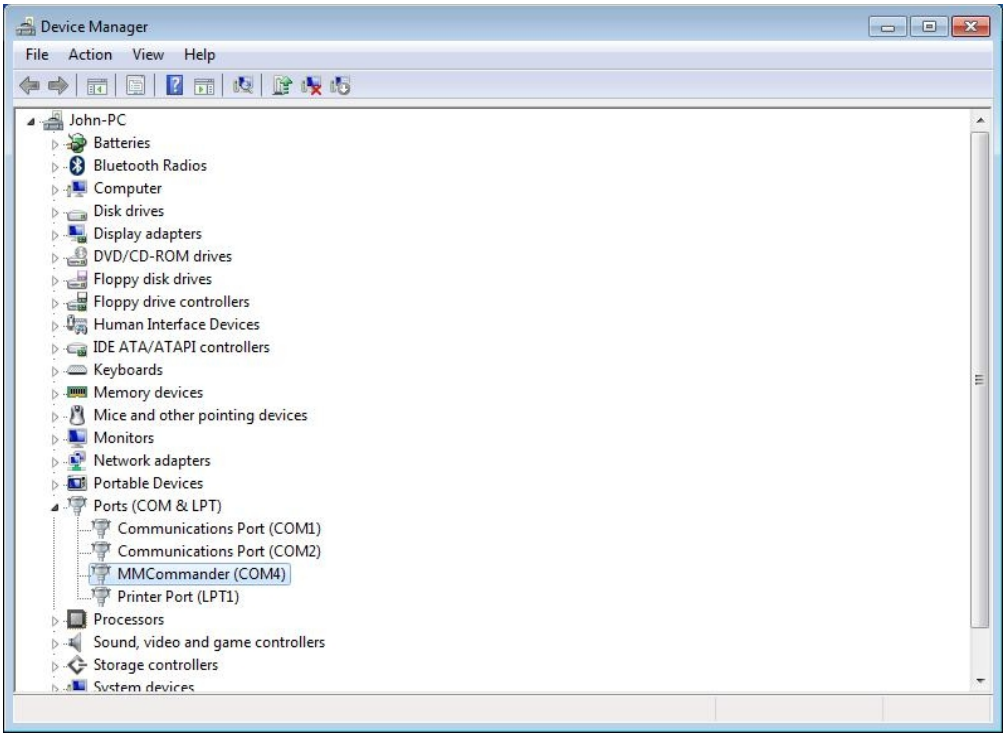
- We want to program a CCxxxx SoC.
- System-on-Chip tab selected
- If everything is correctly connected you should see a CC1111 in the devices list.
- In "Flash image" click the button with the 3 dots and selected the HEX file you have downloaded.
- In actions: Erase, program and verify.

Some CC Debbugers are shipped with an old FW version and the Smart RF Flash programmer will ask you if you want to update it. You can do it if you want, but it's not mandatory. I would only do it if you have problems programming the CC1111. Up to you.

After configuring the programmer, click on "Perform actions" and wait until you see this:



At this moment your computer should detect a new unknown device when you plug it in. Install the MMCommander driver found in “Windows_Driver” and check the device manager in your control panel. You should see a new COM port like the one in the next image.



In Windows 8, you won't be able to install the driver directly because it's not signed. Don't worry. You can skip this step. It's only necessary if you plan to develop with it on a windows PC. Just plug the MMCommander in your USB port and press the pushbutton for a couple seconds. If it the led turns on and, after pressing again another couple seconds, it turns off it's ready ;-).

Congratulations!. You have a new MMCommander!.

Now continue the process configuring the data backend as described in the NightScout website (http://www.nightscout.info/wiki/welcome/configuring-the-data-backend) .

Once the data backend is ready, install the Android Uploader for Medtronic (https://github.com/arbox0/MedtronicUploader) instead of the one for Dexcom (the one in the NightScout website) and, after that, continue the process for the Azure website, again, as described in the NightScout website (http://www.nightscout.info/wiki/welcome/monitor-cgm-web).

REMEMBER, USE IT AT YOUR OWN RISK

Some notes on configuring the Medtronic Uploader

You have set up the mongo database and installed the Medtronic Uploader... how do I configure the uploader?.

Let's do a step by step configuration...

1. Close the uploader app if it's opened and connect the MMCommander with the OTG cable to the phone. You should see a pop-up asking you what to do with it. Select the Nightscout app and it should open automatically. If not, something is wrong with your setup.
2. The uploader is running. Open the preferences menu and configure the uploader:
    - mmol/L -> Select the unit you want to use.
    - Type -> Select Medtronic CGM
    - Pump ID -> The ID on the back of your pump
    - Glucometer ID -> The same on the glucometer
    - Sensor ID -> The ID on the back of the enlite transmitter. (Also configured in your pump)
    - Calibration Type -> Choose “Manual”.
    - Glucose Value Source -> Choose “Medtronic Sensor”
    - Activate the “MongoDB uploadin”
    - MongoDB URI -> Put here the address you got when you created your database.
    - Collection Name -> write “cgm”

- Glucometer Data Collection Name -> write “gcvalues”
- Wifi Hack -> Set it to OFF
- 2 Days at Startup -> Set it to ON
- Disclaimer -> You have to accept it and set it to YES. The uploader won't start the service until you accept it.

3. Go back to the main screen. You should see dashes, a green text saying “CGM Service Started” and another text saying “Uncalibrated”.
4. Each time the uploader receives data from the sensor you'll see a text line in the log (lower part of the screen) that says: “Medtronic CGM Message: sensor data value received”. Don't do anything else until you see that message for the first time. If you don't see that line in 10 minutes go back and repeat the configuration. The transmitter sends a message each 5 minutes (more or less).
5. You have received the first data from the sensor. It's time to calibrate.. Select “Instant Calibration” and enter the BG value display on your pump. After that, on the main screen, you should see your actual BG and that it's calibrated.
6. At this point, your mongo database should start recording data from your uploader. If you want, you can check it looking at the number of documents stored in the “cgm” collection. Each time a sensor message is received the number of documents should increase (one unit).
7. As far as I know, you need to have at least 3 or 4 values before the azure website starts displaying info.