QMdbSim Project

A Microchip Device Simulator Framework

Document History

2025.02.xx, Version 0.3.0 – Initial release. Caveat emptor.

About The Project

The QMdbSim Project is a framework for using Microchip micro-controller devices in QSpice simulations. Under the covers, it uses Microchip's software simulator (MDB). Using the framework, developers can easily create and distribute "QMdbSim device driver components" for specific Microchip devices. Others can then use those driver components to load/execute device-specific binary files in QSpice simulations.

The project includes all schematics and code for a PIC16F15213 device. This document provides some minimal information about using the PIC16F15213 example and how to create "drivers" for other Microchip devices.

Note: I assume that you have basic familiarity with QSpice as well as Microchip's MPLabX IDE and micro-controller devices.

Requirements

As mentioned, the project uses the Microchip software simulator (MDB). This tool is installed automatically with the MPLabX IDE. I presume that you have the IDE installed.

The project code executes MDB.bat and expects to find it here:

```
"C:\Program Files\Microchip\MPLABX\v6.20\mplab_platform\bin\mdb.bat"
```

If you don't have MPLabX v6.20 installed or it is installed elsewhere, you'll need to change the path in PIC16F15213.qsch for your version/location.

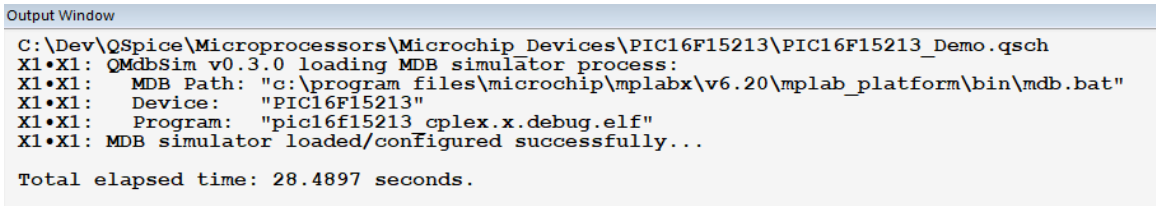
Using The PIC16F15213 Example Device

To test out the PIC16F15213 example, you need only copy the following project files from the repository:

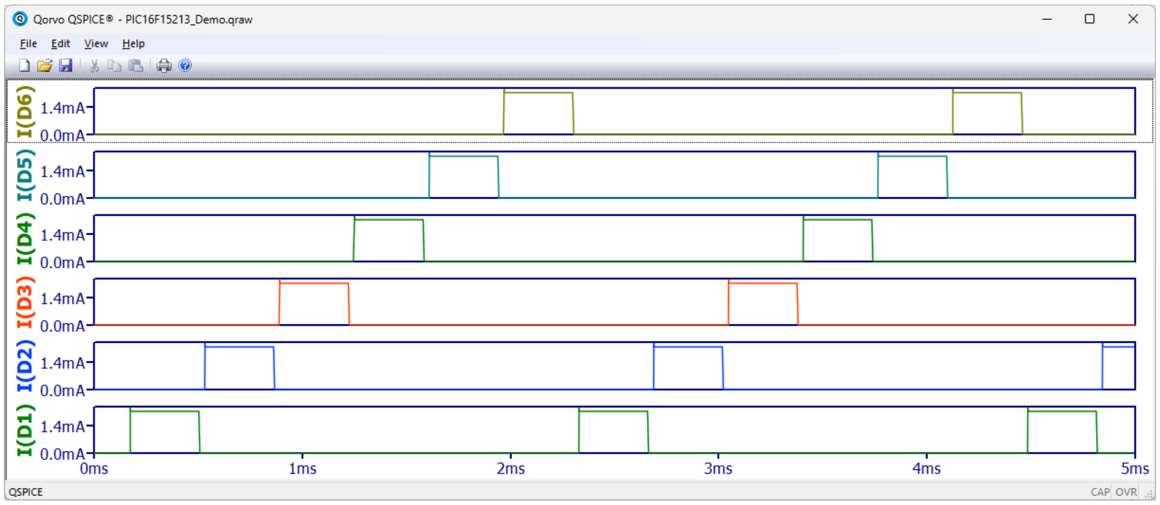
File	Description
PIC16F15213_Demo.qsch	Top-level, user schematic containing the circuitry for an LED Charlie-Plexing project. It contains a hierarchical schematic block for PIC16F15213.qsch.
PIC16F15213.qsch	Device-specific child schematic which contains the C-Block DLL component. It

	should not be changed (except for the MDB path mentioned above).
PIC16F15213.dll	The compiled device-specific C-Block code.
PIC16F15213_CPlex.X.debug.elf	The PIC binary "hex file" compiled with MPLabX for debugging. You can recompile the code for release (*.hex) if desired – either *.hex or *.elf files will work with the simulator.
PIC16F15213_CPlex.zip	Source code for PIC16F15213_CPlex.X.debug.elf. (Not required for simulation but you probably want it.)

Open PIC16F15213_Demo.qsch and run a simulation. Be patient – on my rather average laptop, the simulation takes about 30 seconds to complete. (The first 10 seconds appear to be just initializing the simulator for the device and program.)



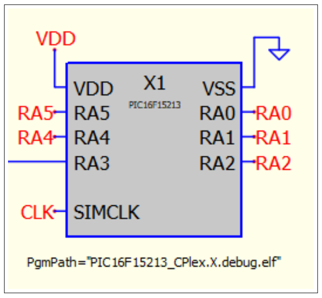
```
C:\Dev\QSpice\Microprocessors\Microchip Devices\PIC16F15213\PIC16F15213_Demo.qsch
X1•X1: QMdbSim v0.3.0 loading MDB simulator process:
X1•X1:   MDB Path:  "c:\program files\microchip\mplabx\v6.20\mplab_platform\bin\mdb.bat"
X1•X1:   Device:    "PIC16F15213"
X1•X1:   Program:   "pic16f15213_cplex.x.debug.elf"
X1•X1: MDB simulator loaded/configured successfully...

Total elapsed time: 28.4897 seconds.
```

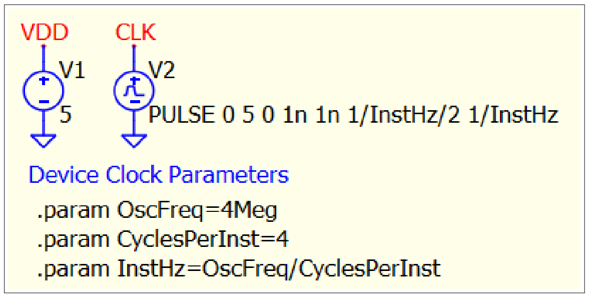


You can now change the PIC device code and recompile in MPLabX to produce a revised binary. Just change the X1 PgmPath attribute to match your binary file location/name if changed.

You can, of course, also make any changes you like to the top-level schematic. But you'll need to know a few things….

Internal Oscillators





Device Clock Parameters
.param OscFreq=4Meg
.param CyclesPerInst=4
.param InstHz=OscFreq/CyclesPerInst

X1 has the same pins as an actual PIC16F15213 with an added “SIMCLK” pin that doesn’t exist on the physical device. We feed that with a clock (V2) and its frequency is set with a few .param statements.

This is necessary because the simulator (MDB) doesn’t run in real-time. That is, while the physical device has an internal oscillator that can be configured for various frequencies, the simulator has no way to map PIC internal oscillator time to QSpice simulation time.

The SIMCLK frequency is the instruction frequency. That is, the above assumes that the PIC binary configures the internal oscillator for 4MHz. On this device, an instruction takes 4 cycles to complete. So V2 is driven at 1MHz.

You’ll need to set the OscFreq parameter to match the internal oscillator speed configured in your PIC code (i.e., configuration bits). Note also that the physical PIC can dynamically change internal oscillator speeds – that’s not currently supported in the simulation code.

Weak Pull-Ups

The PIC16F15213 supports weak pull-ups on some pins. Control over pull-ups is configured in the binary code and can by changed dynamically in the code. Until someone convinces me that it’s truly important, it’s not implemented in the simulations. You can add pull-ups in your top-level schematic to match the (static) configuration in your device code for “better” electrical simulation. (See below.)

Peripherals

Microchip devices have “integrated peripherals” such as ADC, DAC, SPI, I2C, USB, etc. The MDB simulator may or may not support a given peripheral for a given device. If MDB doesn’t support it, well, this project doesn’t support it. See the References section for a link to Microchip’s list of peripherals supported by MDB.

According to that list, MDB supports the following peripherals for the PIC16F15213: CCP1/2, EEFlash, INT/IOC, FVR, OSC, PORTA, PPS, PWM3/4, Pull-Up, TMR0-2, UART1, WDT. (I was disappointed that ADC isn’t in that list.)

Electrical Fidelity

The simulation doesn’t attempt to accurately replicate the electrical behavior of a physical device. PIC16F15213.qsch contains the electrical circuitry seen by QSpice. You can change that (for example, to add internal pull-up resistors) but the simulator is a “digital-behavior tool” and not really suitable for electrical analysis. That is, you can’t use the QSpice simulation to accurately measure currents on the device pins.

Creating Other Microchip Devices

Creating other Microchip devices isn’t hard using the QMdbSim framework. Well, it’s as easy as I can make it….

You'll definitely need a modern C++ compiler. I've provided project files for Microsoft Visual Studio (2022 Community Edition). If you use these, you'll be able to use the MSVS debugger to run simulations in a terminal window. (See resources???)

First, download the rest of the files from the project repository. Here are the steps to create a "PICXXX" device:

- Rename PIC16F15213.qsch to PICXXX.qsch and edit it. This schematic provides the electrical mapping between the component code and the top-level schematic (here, PIC16F15213_Demo.qsch).
    - For every tri-state GPIO pin, you'll need three QSpice component ports (in/out/ctrl) and a GPIO_Pin tri-state "connector."
    - For input-only device pins, see RA3.
    - Change the DLL name attribute to PICXYZ and generate a template from QSpice to get the "data[]" mappings to use later.
    - Change the DLL name attribute to PICXXX.
- Rename PIC16F15213.cpp to PICXXX.cpp.
    - Edit the evaluation function name to "picxxx".
    - Copy the "data[]" mappings from above into the evaluation function.
    - Replace all "PIC16F15213" literals with "PICXXX".
    - In the evaluation function initialization, edit the pin/port/name mappings to match the new device. It looks like this:

```
// add all pin/port/name mappings to list
inst->mdb.addPinPortMap("RA0", &RA0_I, &RA0_O, &RA0_C);
inst->mdb.addPinPortMap("RA1", &RA1_I, &RA1_O, &RA1_C);
inst->mdb.addPinPortMap("RA2", &RA2_I, &RA2_O, &RA2_C);
inst->mdb.addPinPortMap("RA3", &RA3_I);
inst->mdb.addPinPortMap("RA4", &RA4_I, &RA4_O, &RA4_C);
inst->mdb.addPinPortMap("RA5", &RA5_I, &RA5_O, &RA5_C);
```

I might have missed a step or two but that should be pretty much all that you need to do.

Please let me know if I missed something or if you have questions.

References

My GitHub QSpice Repository

https://github.com/robdunn4/QSpice

Microchip

MPLabX IDE

PIC16F15213/14/23/24/43/44 Low Pin Count Micro-Controllers

Microchip Debugger (MDB) User's Guide (one)

Microchip Debugger (MDB)User’s Guide (another one)

Simulator Peripheral Support By Device

Microsoft

Visual Studio Downloads

Conclusion

I hope that you find this project to be useful or – at least – interesting. Please let me know if you find problems or suggestions for improving the code or this documentation.

--robert