Anything Goes

Saturday, March 9, 2019

VSCode and MSP430 debugging

In process of getting cozy with VSCode and playing with MSP430 via LanchPad here is a simple recipe for tooling my work up.

But first ...

Basic tooling

- Build - uses **msp430-gcc**.
- Debug - uses: **mspdebug** and **msp430-gdb**

All msp430 tolls already prepackaged on multiple Linux distros (I use Mint which is Debian-based hence apt-get is your fried.)

Checking Connection to the Target

```
mspdebug rf2500 "exit"
```

If the output contains Device ID and Chip ID data you're good.

If it can't connect try again with *sudo* in case the problem is with permissions. If it works with sudo, you add udev rules to allow non-root access - plenty of info in the tubes on this (I would advice a reboot after adding rules file).

Sometimes, the hw debugger/drive gets stuck - a reconnect would fix it. Make sure it shows connected with '**lsusb|grep 430**'.

Flashing the image into the device

```
mspdebug rf2500 "prog ./bin/firmware.elf"
```

Starting gdb Debug Server

The server is build into mspdebug it need to be running as gdb will use it to debug.

```
mspdebug rf2500 "gdb"
```

Starting gdb Proper

```
msp320-gdb bin/firmware.elf -ex "target remote :2000"
```

This tells the gdb to load debug info from the given elf file and execute commands to connect to the debug server. Normally, gdb uses this for remote debugin (as in on another PC) here the same is used for embedded systems.

NOTE: each time you disconnect/quit gdb the debug server must be restarted.

Convinience - Make File Targets

In the Ikea Ansluta hacking project (on github here) I have created make targets for continence:

```
make target-flash
make target-startdebug
make target-debug
```

I think it's easier to remember these than particular incantation of commands given above.

The CLI workflow would be:

edit+build with **make**
flash with **make target-flash**
debug with **make target-debug**

Followers

Mga sumusubaybay (2)

Sundin

About Me

B Unknown

View my complete profile

Blog Archive

- ▼ 2019 (5)
  - ▼ March (3)
    - VSCode and MSP430 debugging
    - Ikea Ansluta Hacking - timer and key interrupts
    - Ikea Ansluta Hacking - bang on the pipes (or...LED)
  - ► January (2)
- ► 2018 (5)
- ► 2015 (1)
- ► 2014 (3)
- ► 2011 (1)
- ► 2010 (2)
- ► 2009 (4)

Note that target-flash makes sure the code build is up to date.

VSCode tooling

By adding tasks: build and flash I can Ctrl-Shift-B to build (as normal with VS) and I can also flash directly from the IDE be selecting "Flash" task. These are added to .vscode/tasks.json file manually.

Starting debbugger with just F5 is tooled via .vscode/launch.json. This tells VSCode to use "msp430-gdb" instead of default one and it tells it to connect to remote debug server.

However, I could not make the a **preLaunchTask** to start the msbdebug in the background. This task seem to either exit and kill the mspdebug or the mspdebug is unable to connect despite the rules working file in shell. So for now the debug server must be started manually before entering visual debugging in the IDE.

Posted by Unknown at 2:49 PM

1 comment:

iyegoroff said...

"However, I could not make the a preLaunchTask to start the msbdebug in the background."

To start mspdebug in the background you can add

"debugServerPath": "mspdebug",
"debugServerArgs": "rf2500 \"gdb\""

to launch.json configuration.

February 10, 2021 at 5:05 PM

Post a Comment

Home   Older Post

Subscribe to: Post Comments (Atom)

Simple theme. Powered by Blogger.