GDB Debugging for Andes Corvette-F1-N25

Outline

1. Setup GDB for Andes Corvette-F1-N25
    i. Start GDB Server
    ii. Connect to GDB Server
    iii. Find the Debug Target Executable with Arduino
2. Simple GDB Guide
    i. Load an Executable
    ii. Common Commands

Prerequisite

- Have a working Arduino setup for Andes Corvette-F1-N25

1. Setup GDB for Andes Corvette-F1-N25

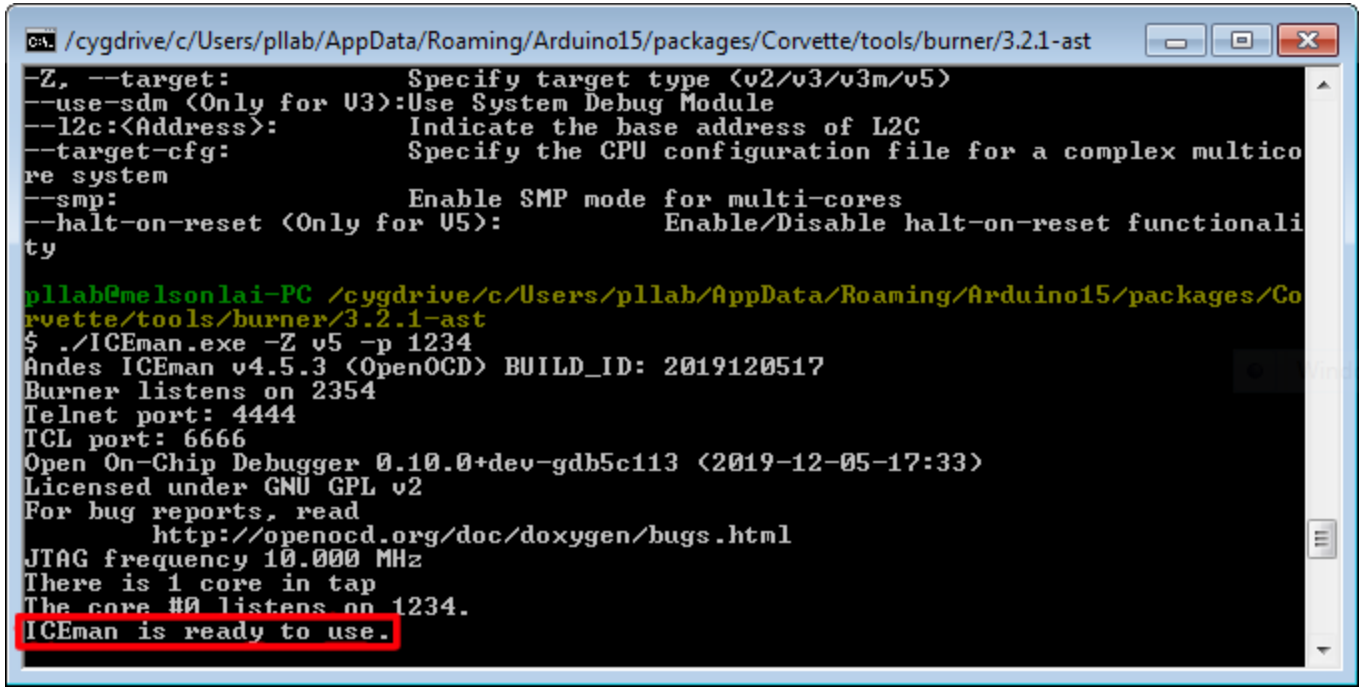
1-1. Start GDB Server

1. Start a Cygwin shell by executing `C:\Users\${USERNAME}\AppData\Roaming\Arduino15\packages\Corvette\tools\nds32le-elf-mculib-v5\3.2.1-ast\cygwin\Cygwin.bat`
2. In the Cygwin shell:
   i. `cd /cygdrive/c/Users/${USERNAME}/AppData/Roaming/Arduino15/packages/Corvette/tools/burner/3.2.1-ast/`
   ii. Run `./ICEman.exe -Z v5 -p 1234`

> The ICEman command starts a GDB server at `localhost:1234`. If port 1234 is already taken, just name your own la.

1-1. Start GDB Server (cont.)

- You should see that `ICEman is ready to use.` after a while



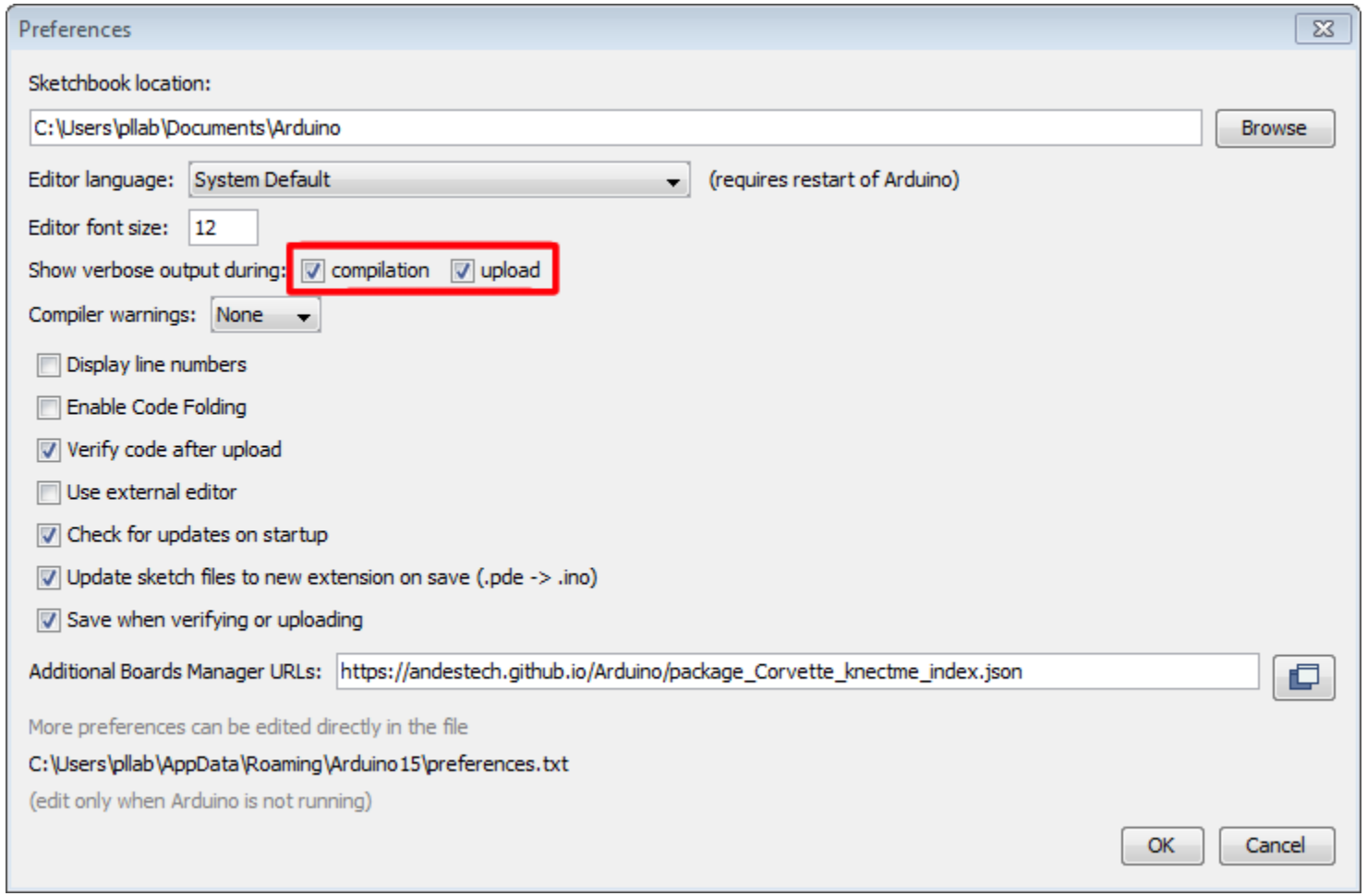
1-2. Connect to GDB Server

1. Start another Cygwin shell by executing `C:\Users\${USERNAME}\AppData\Roaming\Arduino15\packages\Corvette\tools\nds32le-elf-mculib-v5\3.2.1-ast\cygwin\Cygwin.bat`
2. In the Cygwin shell:
   i. `cd /cygdrive/c/Users/${USERNAME}/AppData/Roaming/Arduino15/packages/Corvette/tools/nds32le-elf-mculib-v5/3.2.1-ast/bin/`
   ii. Run `./riscv32-elf-gdb.exe`
3. In the GDB shell, run `target remote localhost:1234` to connect to the GDB server

> If you changed the port, remember to use the correct one.

1-3. Find the Debug Target Executable with Arduino

- Enable verbose output for compliation and upload in Arduino preference



1-3. Find the Debug Target Executable with Arduino (cont.)

- In Arduino, click `Sketch > Verify / Compile` to compile the code. You should now see the commands run by Arduino
- Among the log, you should find the final ELF filename as something like `C:\Users\${USERNAME}\AppData\Local\Temp\build5286355239941495947.tmp/assembly.cpp.elf`
  - The `build5286355239941495947.tmp` directory is a random/hash name following the pattern of `build[0-9]+\.tmp`

1-3. Find the Debug Target Executable with Arduino (cont.)

assembly | Arduino 1.6.5

File Edit Sketch Tools Help

assembly | codegen.S

```
extern "C" void codegen() asm ("codegen");

void setup() {
  // put your setup code here, to run once:
  pinMode(LED1, OUTPUT); // LED1 = 26
  pinMode(LED2, OUTPUT); // LED2 = 27
  pinMode(LED3, OUTPUT); // LED2 = 28
  pinMode(LED4, OUTPUT); // LED2 = 29
}

void loop() {
  // put your main code here, to run repeatedly:
  codegen();
}
```

Done compiling.

```
C:\Users\pllab\AppData\Roaming\Arduino15\packages\Corvette\tools\nds32le-elf-mculib-v5\3.2.1-ast/bin/build.bat C:\Users\pllab\AppData\Roaming\Arduino15\packages\Corvette\tools\nds32le-elf-mculib-v5\3.2.1-ast/bin/riscv32-elf-ar rcs
C:\Users\pllab\AppData\Local\Temp\build5286355239941495947.tmp/core.a C:\Users\pllab\AppData\Local\Temp\build5286355239941495947.tmp\wiring_tone.cpp.o

C:\Users\pllab\AppData\Roaming\Arduino15\packages\Corvette\tools\nds32le-elf-mculib-v5\3.2.1-ast/bin/build.bat C:\Users\pllab\AppData\Roaming\Arduino15\packages\Corvette\tools\nds32le-elf-mculib-v5\3.2.1-ast/bin/riscv32-elf-ar rcs
C:\Users\pllab\AppData\Local\Temp\build5286355239941495947.tmp/core.a C:\Users\pllab\AppData\Local\Temp\build5286355239941495947.tmp\WMath.cpp.o

C:\Users\pllab\AppData\Roaming\Arduino15\packages\Corvette\tools\nds32le-elf-mculib-v5\3.2.1-ast/bin/build.bat C:\Users\pllab\AppData\Roaming\Arduino15\packages\Corvette\tools\nds32le-elf-mculib-v5\3.2.1-ast/bin/riscv32-elf-ar rcs
C:\Users\pllab\AppData\Local\Temp\build5286355239941495947.tmp/core.a C:\Users\pllab\AppData\Local\Temp\build5286355239941495947.tmp\WString.cpp.o
C:\Users\pllab\AppData\Roaming\Arduino15\packages\Corvette\tools\nds32le-elf-mculib-v5\3.2.1-ast/bin/build.bat C:\Users\pllab\AppData\Roaming\Arduino15\packages\Corvette\tools\nds32le-elf-mculib-v5\3.2.1-ast/bin/riscv32-elf-g++ -Os
-lgloss -lgcc -lc -lstdc++ -lm -nostdlib -nostartfiles -mcmodel=large -static -TC:\Users\pllab\AppData\Roaming\Arduino15\packages\Corvette\hardware\Corvette-F1-N25\1.6.5\cores\arduino/sys/board/ae250.ld
-IC:\Users\pllab\AppData\Roaming\Arduino15\packages\Corvette\hardware\Corvette-F1-N25\1.6.5\cores\arduino -IC:\Users\pllab\AppData\Roaming\Arduino15\packages\Corvette\hardware\Corvette-F1-N25\1.6.5\cores\arduino/sys/arch
-IC:\Users\pllab\AppData\Roaming\Arduino15\packages\Corvette\hardware\Corvette-F1-N25\1.6.5\cores\arduino/sys/inc -LC:\Users\pllab\AppData\Local\Temp\build5286355239941495947.tmp
C:\Users\pllab\AppData\Local\Temp\build5286355239941495947.tmp/loader.c.o C:\Users\pllab\AppData\Local\Temp\build5286355239941495947.tmp/interrupt.c.o C:\Users\pllab\AppData\Local\Temp\build5286355239941495947.tmp\codegen.S.o
C:\Users\pllab\AppData\Local\Temp\build5286355239941495947.tmp\assembly.cpp.o C:\Users\pllab\AppData\Local\Temp\build5286355239941495947.tmp/core.a -o C:\Users\pllab\AppData\Local\Temp\build5286355239941495947.tmp/assembly.cpp.elf
C:\Users\pllab\AppData\Roaming\Arduino15\packages\Corvette\tools\nds32le-elf-mculib-v5\3.2.1-ast/bin/build.bat C:\Users\pllab\AppData\Roaming\Arduino15\packages\Corvette\tools\nds32le-elf-mculib-v5\3.2.1-ast/bin/riscv32-elf-objcopy -O
binary -R .note -R .comment -S C:\Users\pllab\AppData\Local\Temp\build5286355239941495947.tmp/assembly.cpp.elf C:\Users\pllab\AppData\Local\Temp\build5286355239941495947.tmp/assembly.cpp.bin
```

14 Corvette-F1-N25 on COM1

1-3. Find the Debug Target Executable with Arduino (cont.)

- To use the filename in GDB, convert it to the Cygwin convention:

  i. Replace `c:\` with `/cygdrive/c/`

  ii. Replace all backward slashes `\` with forward slashes `/`

> E.g. we get `/cygdrive/c/Users/${USERNAME}/AppData/Local/Temp/build5286355239941495947.tmp/assembly.cpp.elf` for the previous filename

2. Simple GDB Guide

2-1. Load an Executable

- In the GDB shell, run `file ${FILENAME}` to load a new debug target file
  - `y` if prompted by `A program is being debugged already. Are you sure you want to change the file? <y or n>`

2-2. Common Commands

- `run` : Run the debug target file
- `list [<symbol>]` : List the source code around `<symbol>`

  > You may want to try `list codegen`

- `break <loc>` : Setup a breakpoint at `<loc>`
- `step` : Continue execution till the next line/instruction
- `next` : Like `step` , but proceeds through subroutine calls
- `info registers` : Print the contents of registers
- `x <mem_loc>` : Examine the content of memory at `<mem_loc>`

Thanks