Configurando e usando a placa STM32F4 H407

Martin Vincent Bloedorn
Patrick José Pereira

12 de Fevereiro de 2014

Conteúdo

1 Introdução

Esse documento tem como objetivo fornecer os passos necessários para o setup da placa STM32-H407F [1] em um PC rodando Linux. Serão dadas também instruções para geração de projetos para a placa através de templates fornecidos.

Observações:

- O documento prevê o uso de um adaptador JTAG *BusBlaster* FINISHME, no entanto, qualquer adaptador baseado no CI FT2232 (ou em geral, suportado pelo OpenOCD) deve funcionar, necessitando eventuais alterações em alguns comandos.
- Os comandos do shell apresentados são previstos para Debian/Ubuntu. Alguns comandos podem variar em outras distros.
- Os links para pastas neste documento são relativos e partem do pressuposto que o documento se encontra em FINISHME.

2 Setup do ambiente

2.1 Instalando o OpenOCD

O OpenOCD (*Open on-chip Debugger*, [2]) é uma ferramenta para debug, gravação e inspeção de processadores (*targets*) através de uma ferramenta de interface (JTAG, ISP, etc). Ele será usado para upload e debug da placa, primeiramente pelo terminal, e depois diretamente da IDE Eclipse.

Configurando o OpenOCD no sistema:

1. Instalando/atualizando algumas bibliotecas e ferramentas necessárias.

   ```
   $ sudo apt-get install build-essentials libusb-dev libftdi-dev git
   ```

2. Apesar de disponível no repositório do Ubuntu, é recomendável utilizar a versão mais recente do OpenOCD. Para tal, clone o repositório do projeto usando o Git:

   ```
   $ git clone git://repo.or.cz/openocd.git
   ```

3. Navegue para a pasta transferida e execute o *bootstrap*.

   ```
   $ cd openocd && ./bootstrap
   ```

4. Os módulos a serem compilados no OpenOCD podem ser escolhidos durante a execução do *configure*. Para tal, execute

   ```
   $ ./configure --disable-werror --enable-buspirate --enable-ft2232_libftdi
       --enable-jlink --enable-stlink --enable-ti-icdi --enable-
       usb_blaster_libftdi --enable-ulink
   ```

   Mais módulos podem ser adicionados manualmente se necessário for. Eventuais erros que surgem nessa etapa costumam estar relacionados à bibliotecas faltantes.

5. Compile o código, e instale a aplicação.

   ```
   $ make && sudo make install
   ```

2.2 Lançando o OpenOCD e conectando à placa

Primeiramente, conecte a placa STM32F4 H407 à uma fonte de alimentação (6 a 16V), o JTAG Busblaster à porta USB e conecte um ao outro via um cabo flat de 20 vias. O JTAG deve aparecer no sistema como um dispositivo FT2232.

```
$ lsusb
        ...
        Device 00630401: ...
```

O diretório padrão de instalação do OpenOCD é o `/usr/local/share/openocd`. Nele, fica a pasta `scripts`, que contem as definições das interfaces e targets suportados. Para lançar o OpenOCD, utiliza-se um comando da forma

```
$ openocd -f [script de interface] -f [script de target]
```

Onde o flag `-f` especifica um arquivo de configuração a ser lido. Considerando a pasta atual como o diretório padrão do OpenOCD, com a placa STM32F4 H407 e com um adaptador JTAG Busblaster, o comando se torna então

```
$ openocd -f scripts/interface/busblaster.cfg -f scripts/target/stm32f4x.cfg
```

Este comando pode precisar de `sudo` para ser executado corretamente. Utilizar `sudo` em comandos triviais não é recomendado, e a solução para tal problema é apresentada na seção 2.2.1.

O comando pode opcionalmente ser lançado como um shell script, o que evita ter que digitá-lo a cada sessão. Um exemplo de script simples é dado abaixo (ele pode ser posteriormente executado com `sh nomedoscript.sh`).

```
#!/bin/sh
openocd -f /usr/local/share/openocd/scripts/interface/busblaster.cfg -f /usr/
    local/share/openocd/scripts/target/stm32f4x.cfg
```

Após execução do comando, o OpenOCD começará a rodar e (muito provavelmente) detectará a interface e o target, produzindo um output que terminará em algo como

```
Info : JTAG tap: stm32f4x.cpu tap/device found: 0x4ba00477 (mfg: 0x23b, part: 0
    xba00, ver: 0x4)
Info : JTAG tap: stm32f4x.bs tap/device found: 0x06413041 (mfg: 0x020, part: 0
    x6413, ver: 0x0)
Info : stm32f4x.cpu: hardware has 6 breakpoints, 4 watchpoints
```

Isso indica que a placa foi corretamente detectada. O OpenOCD ficará então rodando naquele terminal, e está pronto para ser utilizado.

2.2.1 Rodando o OpenOCD sem *sudo*

2.3 Usando o OpenOCD local e remotamente

Seção incompleta! Proceder com cuidado!

Uma vez rodando, o OpenOCD se torna um servidor, que pode ser acessado via `telnet`. O servidor escuta a porta 4444 para interface com o usuário, e a porta 3333 para interface com o GDB (*GNU Debugger*). A placa pode sere então acessada e gravada manualmente via terminal.

```
$ telnet localhost 4444
```

Listando o target conectado, parando a execução de um eventual programa, limpando a flash e gravando algum *hex* já compilado e disponível.

```
> targets
> halt
> flash erase_sectors 0 0 0
> flash write_image main.hex
> reset run
```

Mais informações sobre os comandos do OpenOCD estão em [2] e [3]. Por ser um acesso via telnet, o OpenOCD pode ser acessado remotamente (ex., para compartilhar um hardware conectado em um único PC), por exemplo com `$ telnet 192.168.0.125`, para o caso de dois computadores conectados em uma mesma rede local.

- Instalar Eclipse e Eclipse-CDT, com sudo apt-get install eclipse eclipse-cdt

```
place your source code here
```

- Instalar Zylin Embedded CDT Plugin no Eclipse, (Help , Install New Software), http://opensource.zylin.com/zylincdt .
- Instalar toolchain GCC-ARM Embedded de https://launchpad.net/ terry.guo/+archive/gcc-arm-embedded
-

2.4 Instalando o Toolchain para compilação

Para instalar a versão do GCC necessária para a compilação dos programas, rode os comandos abaixo. Eles descompactam o toolchain em `/usr/bin`, o que exige o uso de `sudo`. Caso isso não seja possível ou desejado, pode-se instalar o toolchain em qualquer outro lugar do sistema, conquanto que ele esteja no `PATH` do Bash (o fazendo acessível para Makefiles).

```
$ cd /usr/bin/
$ sudo wget https://sourcery.mentor.com/sgpp/lite/arm/portal/package8734/public
    /arm-none-eabi/arm-2011.03-42-arm-none-eabi-i686-pc-linux-gnu.tar.bz2
$ sudo tar xjfv arm-2011.03-42-arm-none-eabi-i686-pc-linux-gnu.tar.bz2
$ sudo rm *.tar.bz2
```

Opcionalmente, ao invés de baixar o pacote TAR com `wget`, ele também está salvo no reposiório, em `/birotor/components/software/doc/ARM/Base para desenvolvimento e ferramentas`.

2.5 Configurando a IDE Eclipse para o desenvolvimento

Seção incompleta! Proceder com cuidado!

Primeiramente vamos instalar a IDE.

```
$ sudo apt-get install eclipse-cdt
```

Após instalando, vamos criar uma pasta para iniciar os trabalhos de programação.

```
$ cd
$ mkdir Workspace
```

Agora temos uma pasta para onde poderemos iniciar a mágica dos embedded systems. Copie os arquivos da pasta zipada XXXXXXXXXX - nome da pasta zipada que tem os exemplos marotos do Martien - para a pasta Workspace que criamos.

O Próximo passo é criar o projeto a partir de um Makefile criado.
Com o Eclipse aberto: File → New → Project → C/C++ → Makefile Project with Existing Code.

Selecione a pasta que contenha o Makefile do projeto e em seguida click em: finish → Workbench.
Devera ver o projeto no lado direito da IDE, agora iremos iniciar a parte de configuração do Eclipse.

Na barra superior, vamos configurar para aparecer os atalhos do nosso projeto, para uma melhor utilização. Todas as configurações a seguir são do ambiente gráfico utilizado na maquina de trabalho com o ARM, não necessário para a utilização do ARM, mas para tornar o tutorial mais dinâmico. Primeiramente, Window → customize perspective...:

- Tool Bar Visibility.
  Selecione: File, C/C++ Element Creation, Debug, Breakpoints, Launch, Search, Editor Presentation, Navigate e help.

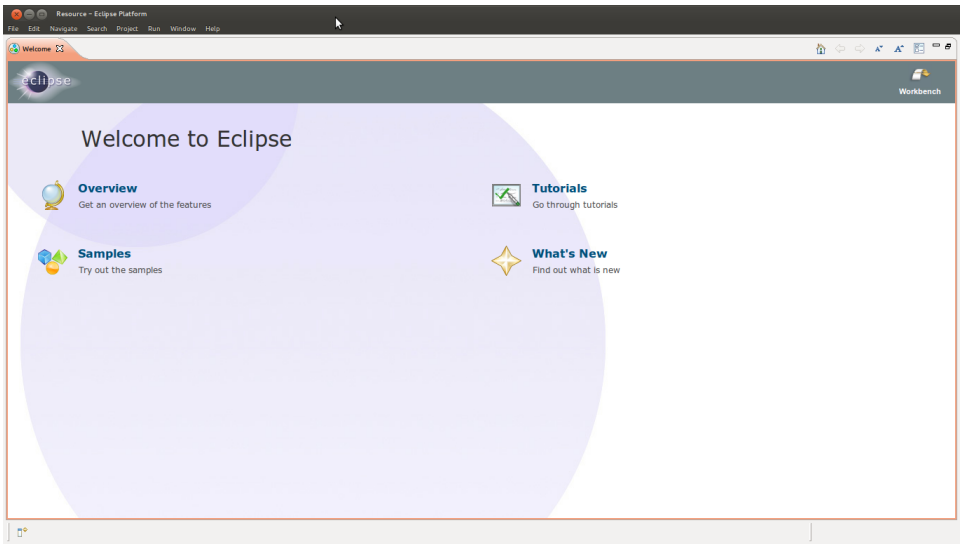

Figura 1: IDE Eclipse aberta.

- Menu Visibility.
  Selecione: File, Edit, Navigate, Project, em Run → Step Into Selection, Window e help.
- Command Groups Availability.
  Selecione: Annotation Navigation, Breakpoints, Build Configuration, C/C++ Coding, C/C++ Editor Presentation, C/C++ Element Creation, C/C++ Navigation, C/C++ Open Actions, C/C++ Search, Cheat Sheets, Convert Line Delimiters, Debug, Editor Navigation, Editor Presetation, External Tools, Keyboard Shortcuts, Launch, Make Actions, Open Files, Remote..., Resource Navigation e Search.
- Shortcuts. Selecione: C/C++.

Agora vamos configurar o Eclipse para compilar o nosso projeto.
Importando o Makefile, já temos todos os caminhos de compilação configurados, seria interessante dar uma olhada para entender a configuração do repositório.
Primeiramente, precisamos de uma ferramenta para o Eclipse, chamada de Zylin, podendo ser facilmente instalada.. (Help → Install New Software → Add.., preenchendo os campos; Name: ZylinCDT e Location: http://opensource.zylin.com/zylincdt, Ok → Finish e prossiga até reiniciar o Eclipse).
Com o Eclipse aberto novamente, click com o botão direito encima da pasta do projeto e em seguida, Debug As e Debug Configurations.

Com o Debug Configurations aberto, entre na ultima opção de Zylin Embedded debug (Native) e em: C/C++ Applications selecione o arquivo *.elf da pasta que esta o seu projeto. Caso o arquivo *.elf não existir, devera compilar manualmente ou utilizando makefile do projeto.
Na aba Debugger, edite o GDB debugger para: arm-none-eabi-gdb. Agora na aba Commands..

- Se estiver no pc hospedeiro:

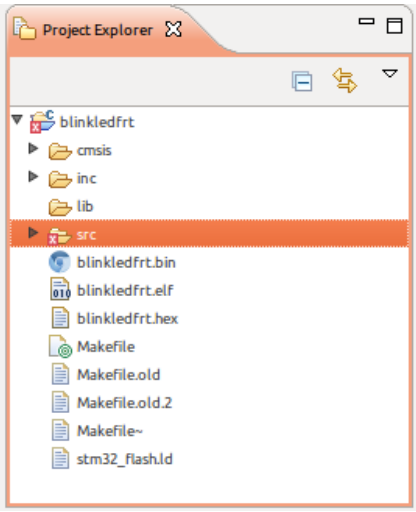

Figura 2: Project Explorer.

```
target remote localhost:3333
monitor reset halt
monitor flash write_image erase /caminho/completo/ate/o/arquivo.elf
monitor reset halt
```

- Se estiver no pc do acesso remoto:

```
Fazer comandos
```

Referências

[1] Olimex, *STM32-H407 development board - User's Manual*, Revision C, February 2013.

[2] OpenOCD, *User's Guide*, acessado em 27/09/2013.

[3] OpenOCD, *Developer's Manual*, acessado em 27/09/2013.

[4] E. H. Norman *Japan's emergence as a modern state* 1940: International Secretariat, Institute of Pacific Relations.

[5] Bob Tadashi Wakabayashi *Anti-Foreignism and Western Learning in Early-Modern Japan* 1986: Harvard University Press.

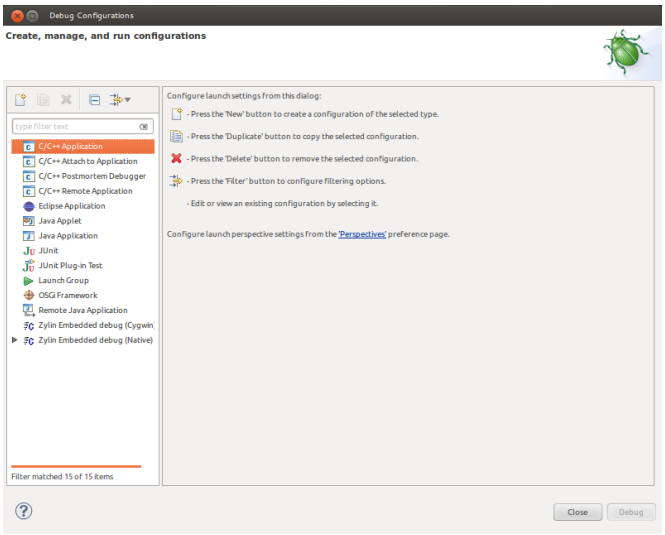

Figura 3: Debug Configurations.