

Ubuntu 下使用 Code::Blocks+OpenOCD+GDB

搭建 Atmel MCU 開發偵錯環境

Nuttx Fan now 於 2019-05-25 19:51:10 發佈

閱讀量 887 收藏 1

點贊數 1

分類專欄：[ARM](#) 文章標籤：[codeblocks](#) [OpenOCD](#) [GDB](#) [Atmel](#) [SAMV71](#)

版權

ARM 專欄收錄該內容

在 [Ubuntu](#) 下搭建 Atmel MCU 開發環境

- [1、準備工作](#)
 - [安裝 Code::Blocks](#)
 - [下載 arm-none-eabi-gcc 編譯器](#)
 - [安裝 OpenOCD 工具](#)
- [2、組態 Code::Blocks](#)
 - [建立一個新的 Debugger 組態](#)
 - [組態 Compiler](#)
- [3、start.atmel.com 下載官方例程](#)
- [4、在 Code::Blocks 新建工程並匯入 atmel 官方例程](#)

1、準備工作

作為一個重度 Linux 系統使用者，平時開發相關的工作都是在 Ubuntu 下進行，最近有個項目需要用到 Atmel 32 位的 Cortex-M7 MCU，所以花時間研究下如何在 Ubuntu 下搭建起 ARM MCU 的開發偵錯環境！如果你也是想使用 **Code::Blocks** 進行 ARM MCU 相關的開發工作，可以參照這篇部落格所列出的步驟指引搭建起開發偵錯環境。

安裝 Code::Blocks

Code::Blocks 是作者非常喜歡和推薦的一款多平台的軟體開發 IDE，尤其對 C/C++ 的支援做的特別好，整個軟體對電腦資源的消耗很少，輕便快捷，程式設計師最看中的程式碼補齊和搜尋功能絲毫不弱於 Source Insight 和 vs code。

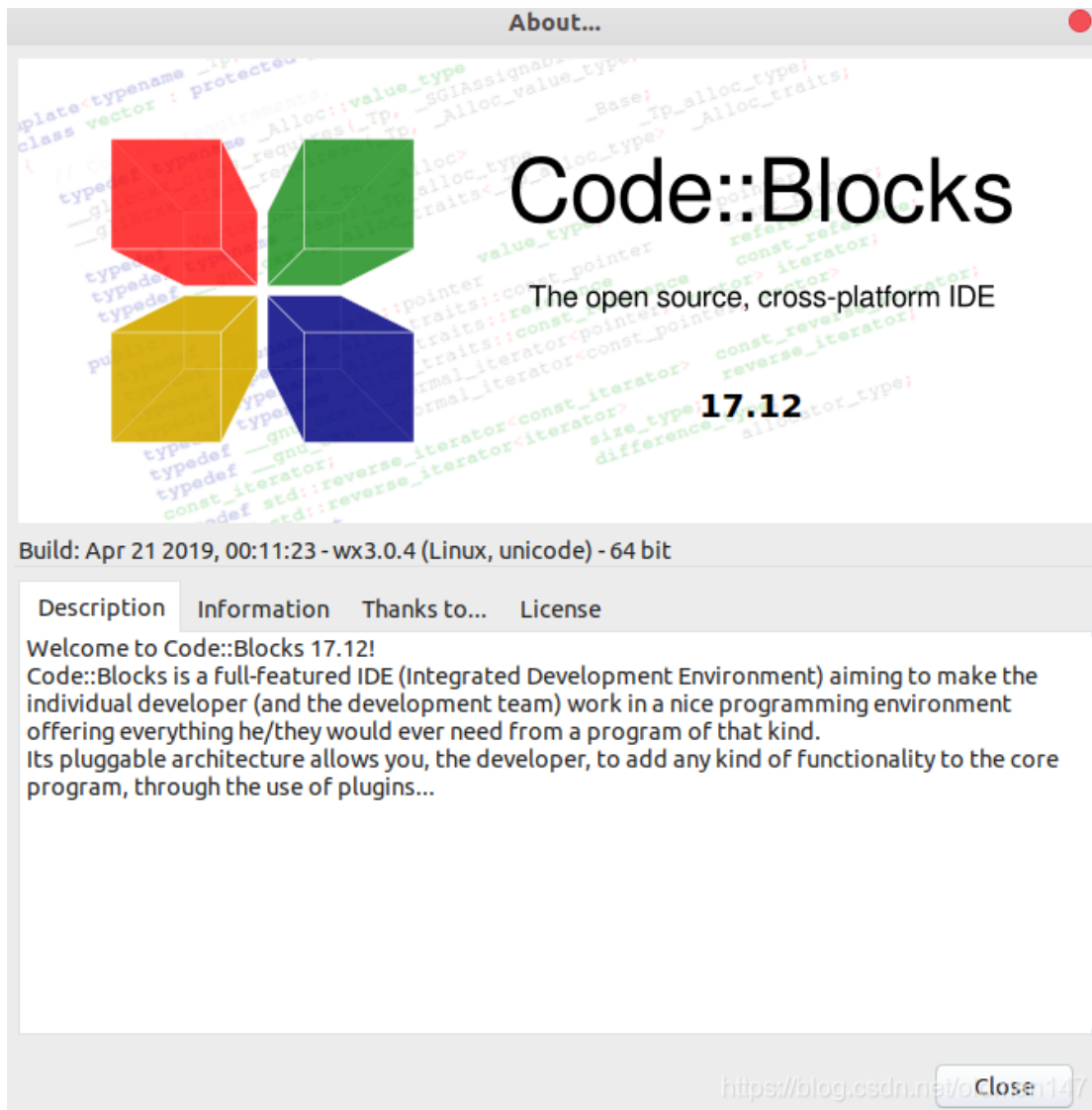
作者使用的 **Code::Blocks** 版本是 17.12，自己去官網下載原始碼手動編譯安裝的，動手能力強的可以嘗試手動編譯下，或者省事的點就在命令終端輸入 `sudo apt-get install codeblocks`。(Ubuntu 默認支援的版本是 16.x 的舊版本)

如果需要更新版本的，可以嘗試：

```
$sudo add-apt-repository ppa:damien-moore/codeblocks-stable
```

```
$sudo apt update
```

```
$sudo apt install codeblocks codeblocks-contrib
```



下載 arm-none-eabi-gcc 編譯器

直接去 ARM 官網就可以下載，記得下載 for Linux 的版本。

<https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads>

演示時使用的是 2018-q4 版本：

[Version 8-2018-q4-major Linux 64-bit](#)

將下載後的編譯器工具包解壓到使用者目錄下，比如作者將工具解壓到 `/home/kevin/opt` 目錄

安裝 OpenOCD 工具

OpenOCD（**Open On-Chip Debugger**）開源片上偵錯程式，是一款開放原始碼軟體，最初是由 Dominic Rath 同學還在大學期間發起的（2005 年）項目。**OpenOCD** 旨在提供針對嵌入式裝置的偵錯、系統程式設計和邊界掃描功能。

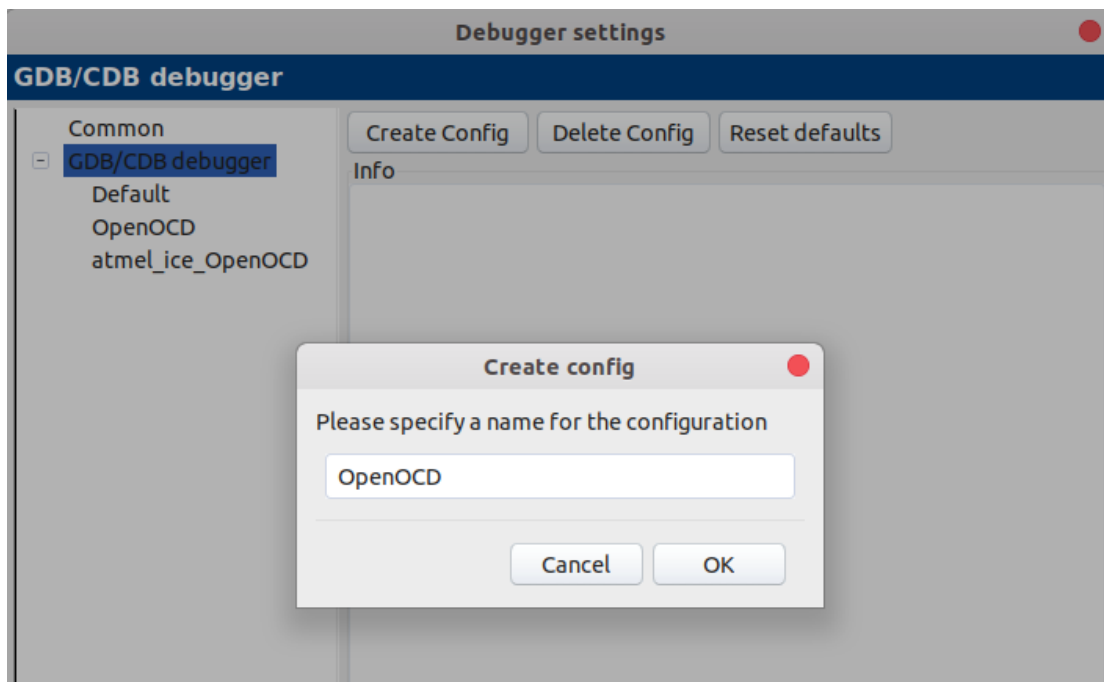
OpenOCD 的功能是在模擬器的輔助下完成的，模擬器是能夠提供偵錯目標的電訊號的小型硬體單元。模擬器是必須的，因為偵錯主機（運行 **OpenOCD** 的主機）通常不具備這種電訊號的直接解析功能。在 Linux 主機環境下，我們可以用 **GDB + OpenOCD** 就可以進行程式碼的下載和偵錯。

說道這裡得致敬下 10 幾年前國內大牛做的一個非常好用的偵錯程式叫 **H-JTAG**，可惜的是後來停止更新。在當時那個 ARM 偵錯工具非常稀缺的年代，**H-JTAG** 的出現還是幫助到了很多早期 ARM 開發人員。

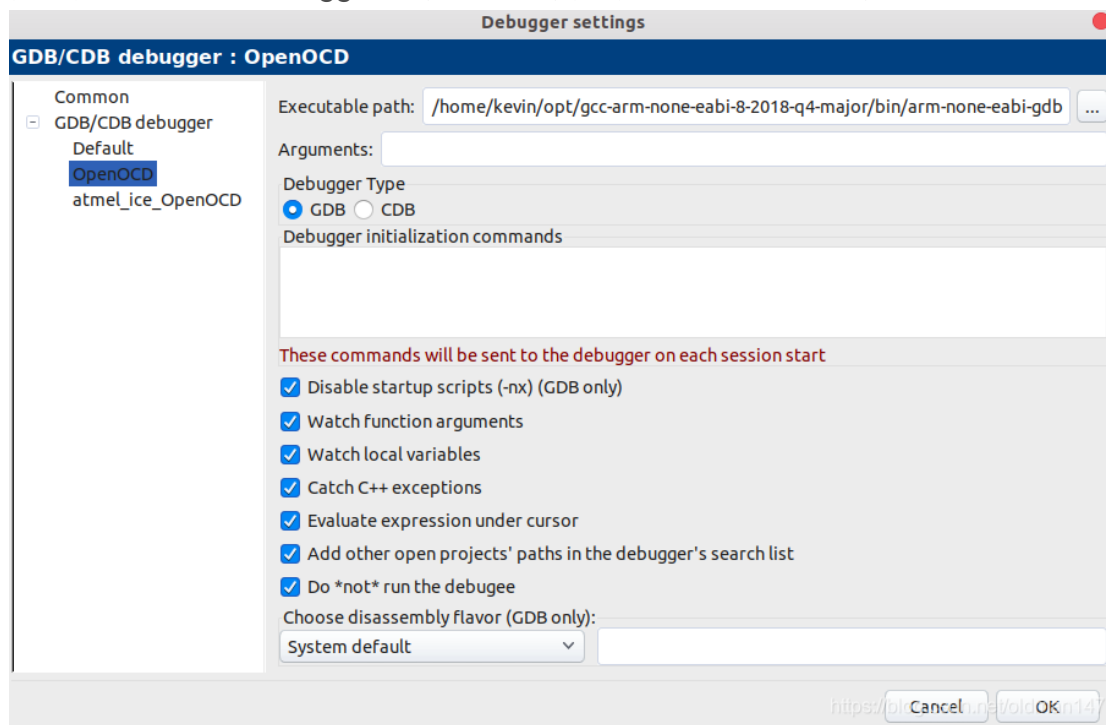
2、組態 Code::Blocks

建立一個新的 Debugger 組態

打開 **Code::Blocks**，選擇 **Settings -> Debugger.**，然後選擇 **Create Config**，新建立一個名叫組態 OpenOCD 的組態，點選 OK 保存新的組態。




接著對新建立的 Debugger 進行組態，需要指定 GDB 工具路徑：




組態 Compiler


打開 **Code::Blocks**，選擇 **Settings -> Compiler.**，在 **Selected Compiler** 下拉框裡面選擇 **GNU GCC Compiler for ARM**，我們需要組態下 **arm-none-eabi-gcc** 編譯工具路徑等內容，**Debugger** 選擇新建立的 **OpenOCD**，最後點選 **OK** 保存組態。




Global compiler settings



Valgrind settings



Profiler settings



Batch builds

Global compiler settings

Selected compiler

GNU GCC Compiler for ARM

Set as default

Compiler settings Linker settings Search directories Toolchain executables Custom variables

Compiler's installation directory

/home/kevin/opt/gcc-arm-none-eabi-8-2018-q4-major/bin

NOTE: All programs must exist either in the "bin" sub-directory of this path, or in any of the "Additional Paths"

Program Files Additional Paths

C compiler: arm-none-eabi-gcc

C++ compiler: arm-none-eabi-g++

Linker for dynamic libs: arm-none-eabi-g++


Linker for static libs: arm-none-eabi-ar

Debugger: GDB/CDB debugger : OpenOCD


Resource compiler:

Make program: make -j4


另外還可以組態系統編譯器標頭檔尋找路徑：




Global compiler settings



Valgrind settings



Profiler settings



Batch builds

Global compiler settings

Selected compiler

GNU GCC Compiler for ARM

Set as default Copy Rename

< Compiler settings Linker settings Search directories Toolchain executables Custom variables

Compiler Linker Resource compiler

Policy:

/home/kevin/opt/gcc-arm-none-eabi-8-2018-q4-major/arm-none-eabi/include

3、start.atmel.com 下載官方例程

[Atmel 官網例程線上組態網站 start.atmel.com](http://start.atmel.com)，可以線上組態所需要的 MCU 外設，外設驅動參數等資訊，用起來還是非常方便，關鍵組態的工程最後下載支援多種整合開發環境，比如它官方的 Atmel Studio 7，通用的 IAR/Keil，還有 Linux 使用者可以用的 GCC Makefile。

由於 Code::Blocks 直接支援匯入使用者 **Makefile** 檔案進行編譯（敲下小黑板劃重點-這點很重要），這樣很多開放原始碼專案可以直接通過此類的方法進行組態匯入

Code::Blocks 中進行編譯和偵錯模擬。原始碼具體的下載過程就請參考後面提供的視訊連結。

4、在 Code::Blocks 新建工程並匯入 atmel 官方例程

整個操作流程請參考以下優酷網視訊連結：

[視訊教學](#)