The free toolchain for the STM8

Getting into first place when non-free tools had years of head start

Philipp Klaus Krause

February 3, 2018

Table of Contents

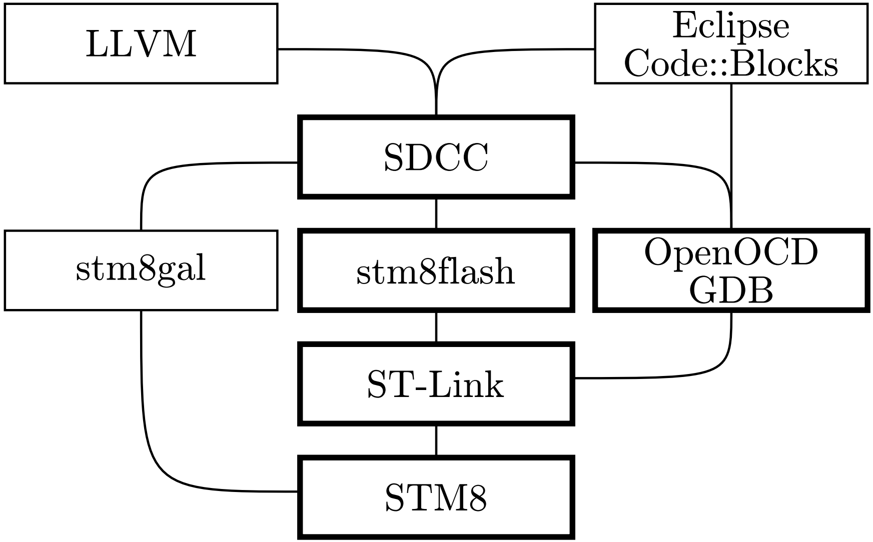
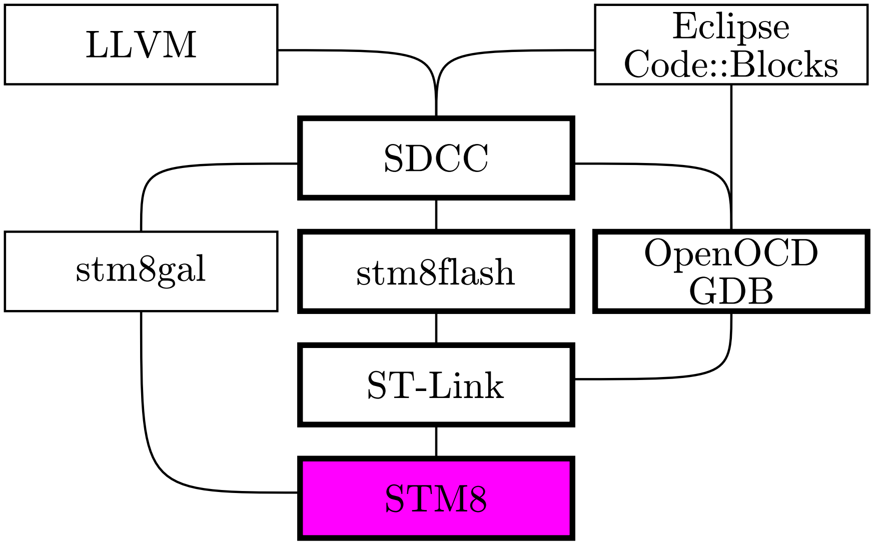


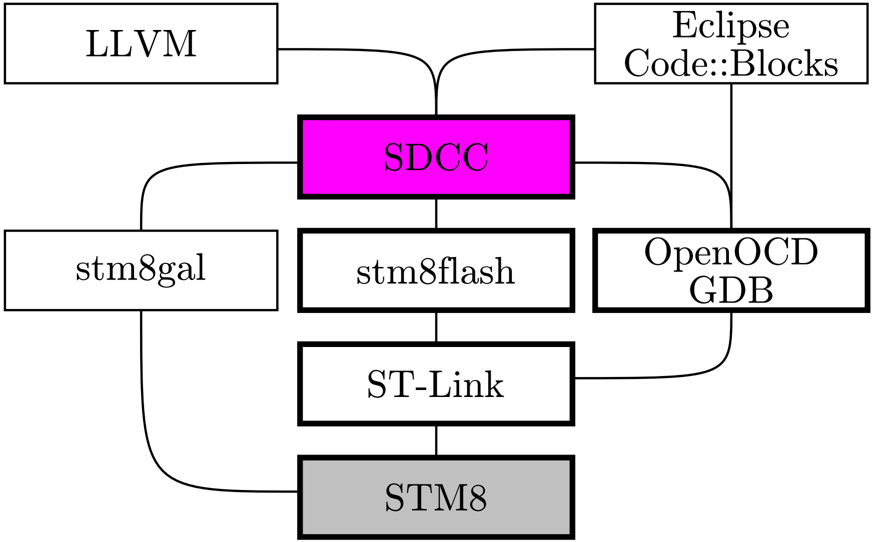
Table of Contents



STM8 microcontrollers

- High-performance 8-bit core
- Rich set of peripherals similar to STM32
- C-friendly architecture
- Up to 6 KB of RAM, 128 KB of flash
- Cheap
- Billions made

Table of Contents



What is SDCC?

- Standard C compiler (ANSI C89, ISO C90, ISO C99, ISO C11)
- Freestanding implementation or part of a hosted implementation
- Supporting tools (assembler, linker, simulator, ...)
- Works on many host systems (GNU/Linux, Windows, Mac OS, NetBSD, FreeBSD, OpenBSD, Hurd, ...)
- Targets various 8-bit architectures (MCS-51, DS80C390, Z80, Z180, Rabbit 2000, Rabbit 3000A, GBZ80, TLCS-90, HC08, S08, STM8, PIC)
- Has some unusual optimizations that make sense for these targets (in particular in register allocation)

Optimal Register Allocation in Polynomial Time

- Register allocator based on graph-structure theory
- Optimal register allocation in polynomial time
- Flexible through use of cost function
- Provides substantial improvements in code quality
- But slow for architectures with many registers
- Compilation speed / code quality trade-off:
  –max-allocs-per-node

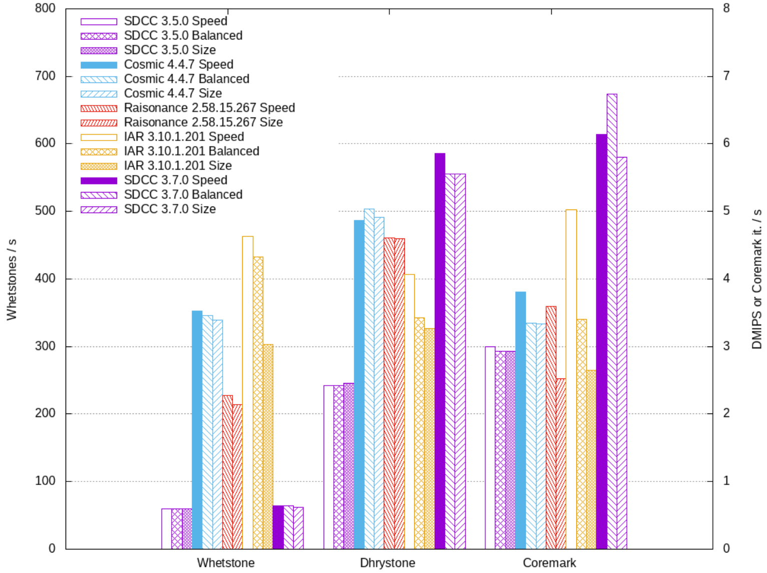
Bytewise Register Allocation and Spilling

- Decide on the storage of variables bytewise
- Decide for each individual byte in a variable whether to store it in memory or a register
- Consider any byte of any register as a possible storage location

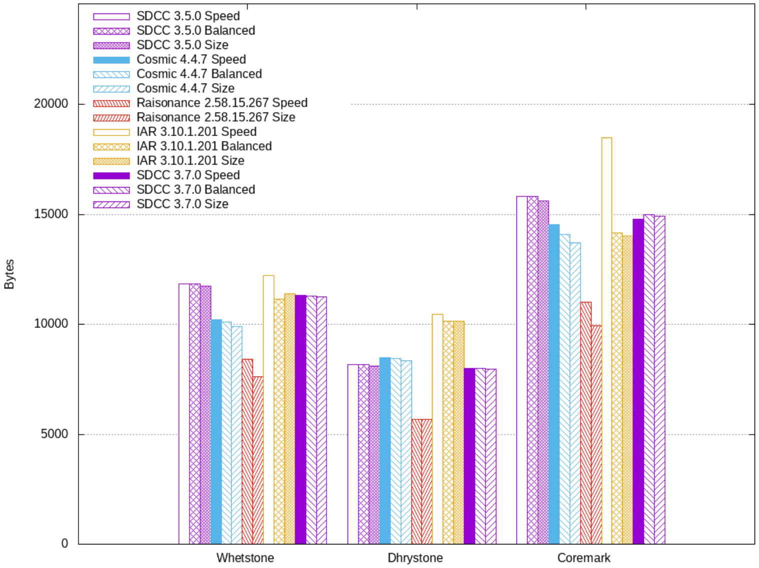
Regression testing

- Regression testing of nightly snapshots
- ≈ 10000 tests compiled and executed on simulator
- Tests mostly from fixed bugs and from GCC
- Targets architectures: MCS-51, DS390, Z80, Z180, GBZ80, Rabbit 2000, Rabbit 3000A, TLCS-90, HC08, S08, STM8
- Host OS: GNU/Linux, Windows, MacOS
- Host architectures: i386, x86_64, ppc, arm

SDCC vs. non-free compilers: Benchmark scores



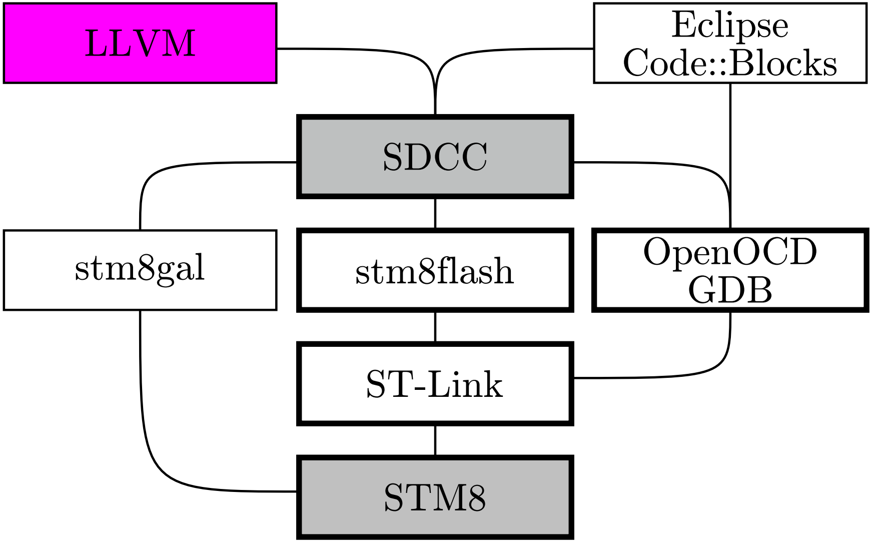
SDCC vs. non-free compilers: Code size



SDCC vs. non-free compilers: Summary

	Standard compl.	Code speed	Code size	OS support	License
SDCC 3.5.0	+	-	-	+	+
Cosmic	○	○	○	○	-
Raisonance	-	-	+	-	-
IAR	+	○	-	-	-
SDCC 3.7.0	+	+	○	+	+

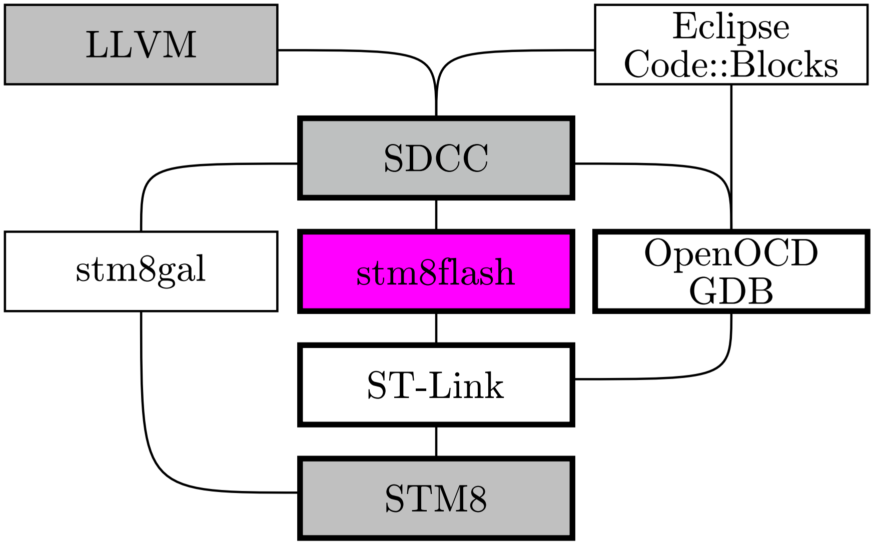
Table of Contents



LLVM+SDCC

- Uses LLVM C front- and backend to produce C code to be compiled with SDCC
- Code compiled with LLVM+SDCC can be mixed with C code compiled with SDCC
- Allows languages other than C
- Enables high-level optimizations
- Experimental, many issues remaining

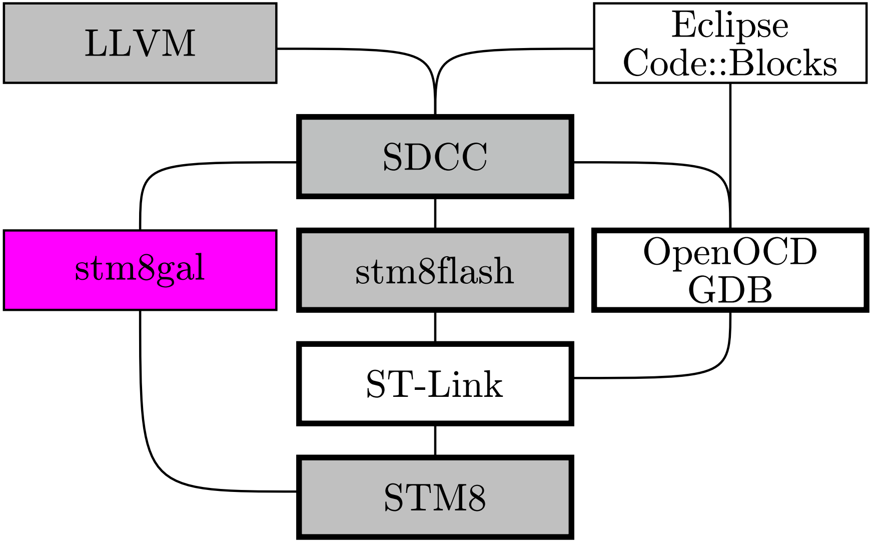
Table of Contents



stm8flash

- Allows reading and writing of flash, eeprom, option bytes
- Supports all known STM8 variants
- Uses ST-Link and ESP-STLINK

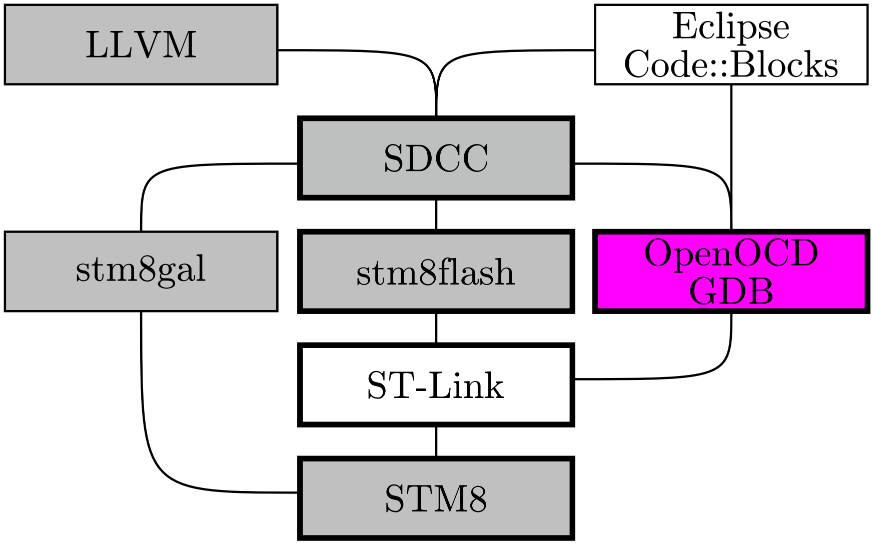
Table of Contents



stm8gal

- Allows reading and writing of flash
- Does not need ST-Link, directly uses STM8 on-chip UART
- Uses bootloader (not available on all STM8)
- Needs non-free blob for some STM8

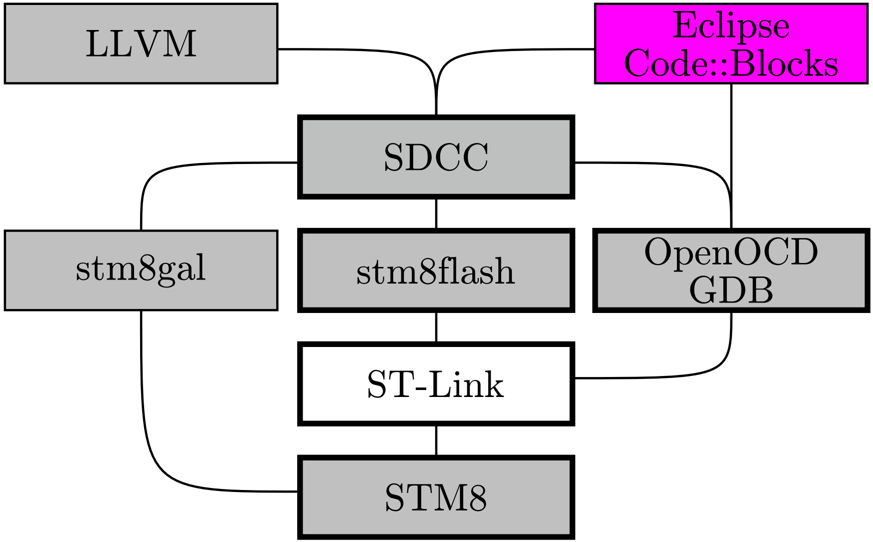
Table of Contents



OpenOCD and GDB

- Patches to OpenOCD and GDB for STM8 and ST-Link support
- Allows on-target debugging

Table of Contents



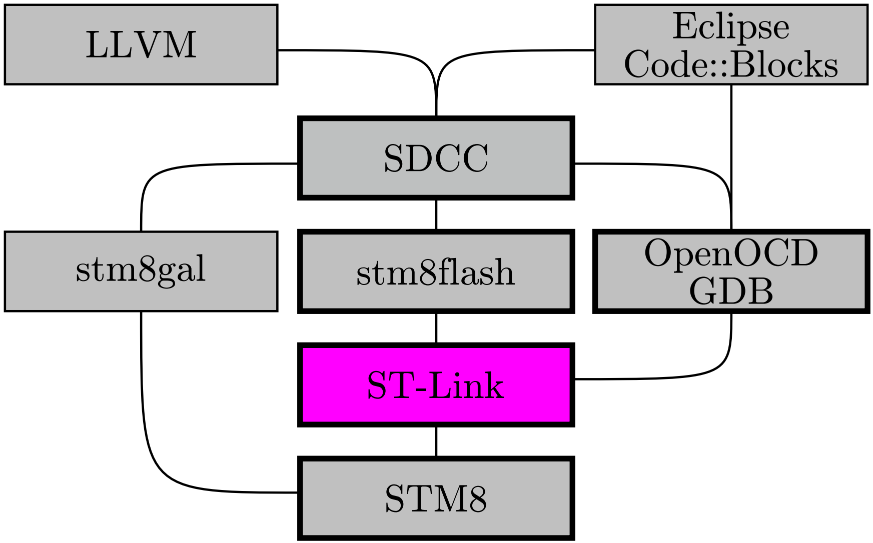
Eclipse

- eclipseSDCC plugin from 2006
- Plugin needs updating for current SDCC and Eclipse
- Needs work from someone familiar with Java and Eclipse
- Debugging support works with Eclipse without plugin

Code::Blocks

- Has support for current SDCC
- Needs testing and improvement
- Needs debugging support

Table of Contents



ST-Link

- Originals by ST, many cheap clones available
- All variants use non-free firmware
- Free alternatives: ESP-STLINK, UPROG2 support memory read / write, but not debugging

TODO

- SDCC needs developers
- Fix SDCC bugs
- Improve SDCC further in standard compliance, optimizations, debug info, etc
- Make LLVM+SDCC useable
- Get GDB changes integrated upstream
- Improve IDE integration
- Port RTOSes
- Replace non-free ST-Link firmware and bootloader, support free replacements in tools