Free Development Tools for BCU 1 and BCU 2

Martin Kögler
e9925248@stud4.tuwien.ac.at
http://www.auto.tuwien.ac.at/~mkoegler/

Goal

- Create a set of DFSG free tools to program and configure BCUs for Linux (without ETS)
- No ETS interface
  - no specification available
- No GUI / IDE
  - use of standard C IDEs (KDevelop, Emacs,..)
  - build with makefiles

Parts

- generic assembler/linker for m68hc05
- C-compiler for m68hc05
- libraries / headerfiles for BCU OS
- helper utilties for image building
- download / configuration utilities

Assembler/Linker

- complete GNU binutils (CVS Version) ported to m68hc05
- relaxing supported
- special features
  - generation of unique section names
  - sections can be moved, if a memory region is full
- already done

Debugger/Simulator

- base on GNU gdb
- initially ported for regression test
- simulates only CPU core
- limited backtrace funtionality
- possible future extension:
  - create fake BCU OS to debug BCU applications

GNU GCC

- GCC ported to m68hc05 architecture
  - missing function in the CPU
    - => emulation of some operations
      - arbitrary memory access
      - data stack
      - general purpose registers
      - mul, div, floating point
    - => some expensive operations left out
      - eg. setjmp/longjmp
  - small memory (BCU1: 256 ROM, 18 Byte RAM) limites really useable features

Programming Model

- hardware call stack
  - BCU2 limits to about 4 levels for the user
- data stack
  - max. 256 bytes
  - use one bytes ram as stack pointer
- register
  - use 13 bytes ram (RegB-RegN) as GPR
- memory access
  - only 8 Bit index register
  - => read / write / call subroutine
  - needs 5 Bytes ram

Internals

- Uses two RTL representations
  - high level
    - use only GPRs
    - works on pseudo instruction with 16 / 32 Bit operands
  - low level
    - splitted into target operations (after register allocation)
    - each instruction has a corresponding real instruction or emulation routine
    - some optimizations are redone

Status GCC

- based on development branch gcc 4.0
- is complete and correct
- plattform specific optimization missing
- G++ frontend is partically working
  - eg. no exceptions

EIB Daemon (1/2)

- tool to communicate with EIB bus
- supports different low level interfaces
  - FT1.2 (user mode)
  - EIBnet/IP routing (user mode)
  - TPUART (kernel driver)
  - PEI 16 (kernel driver)
  - experimental
    - TPUART user mode interface
    - PEI 16 user mode interface (not really working)

EIB Daemon (2/2)

- Provides Layer 4 access over TCP/IP or Unix Sockets
- supports Busmonitor mode
- vBusmonitor
  - best effort, cooperative busmonitor mode
- implements Layer 7 and management functions
- simple C Interface for clients
    - each management program about 30 lines

Header Files / Libraries / Helper Utilities

- A prototype for a subset of BCU1 functionality exists
- needs to be rewritten
- features of BCU2 needs to be included
- support for parameter missing
- image patching support
- ...

Questions?

http://www.auto.tuwien.ac.at/~mkoegler/