User's Tutorial to `cereal`

Miloslav Trmač

mitr@volny.cz

User's Tutorial to `cereal`
by Miloslav Trmač

Copyright © 2002 by Miloslav Trmač

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License."

Table of Contents

Chapter 1. Introduction

The `cereal` emulator is an emulation framework featuring an Intel 8051 emulator and powerful abilities to emulate connected peripherals, be it the common ones (i.e. external memory) or any custom peripherals your application uses.

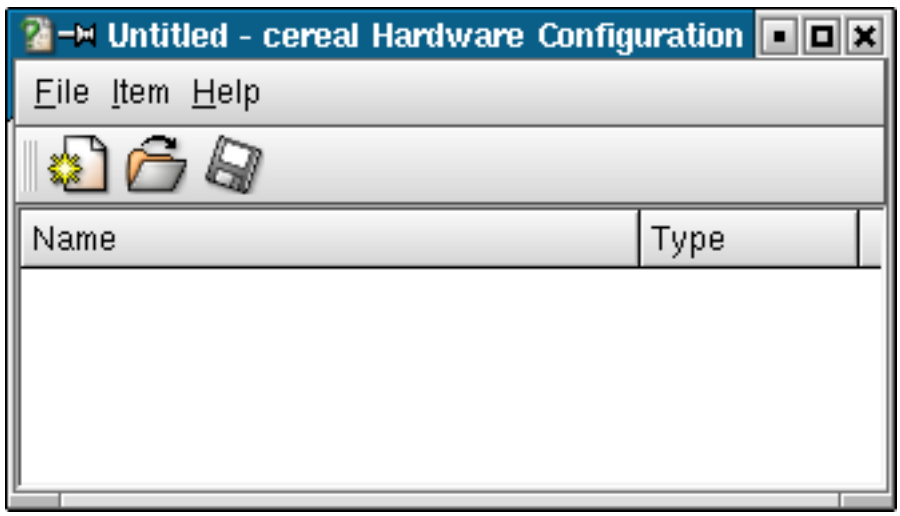
This document will guide you through operating `cereal`. You'll learn how to emulate 8051 programs and how to connect peripherals. Then you'll learn about all `cereal` built-in modules and finally, in the last chapter you'll find description of the command-line utilities provided with `cereal`.

Chapter 2. Emulating Intel 8051

While `cereal` is based on a generic framework able to support any CPU[1], currently only an Intel 8051 core has been implemented. In this chapter you'll learn how to use generic `cereal` features and the 8051 interface.

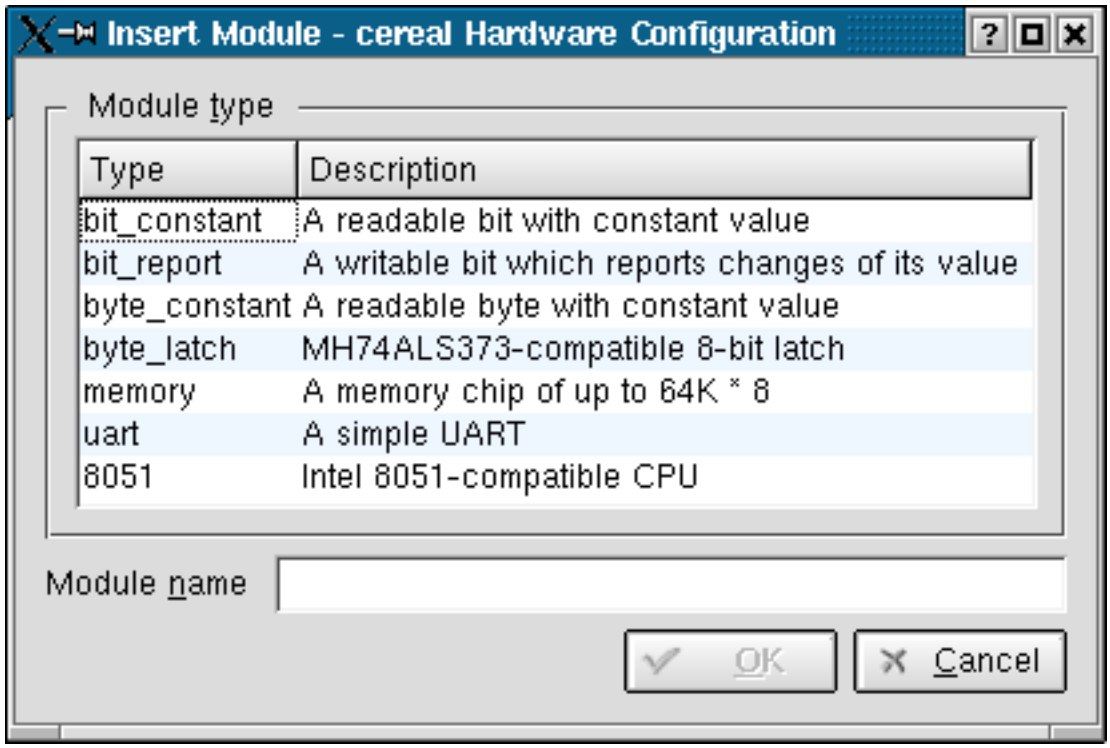
Setting Up Hardware

Because `cereal` can support virtually infinite number of hardware configurations, you need to set up the hardware configuration you want to use. To do this, run **cereal_khwconf**, the `cereal` hardware configuration utility.



Main window of **cereal_khwconf**

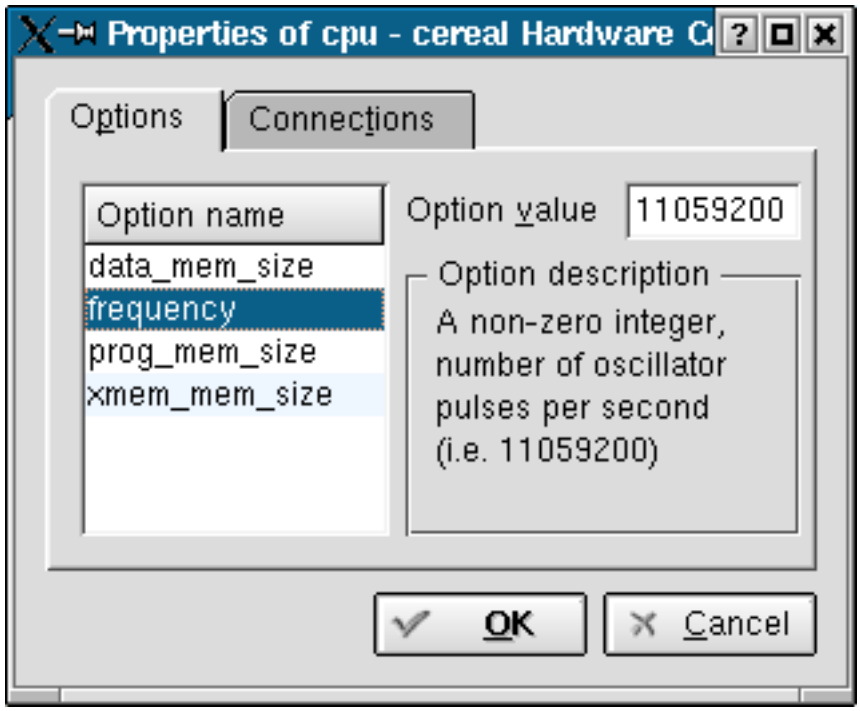
Hardware is represented in `cereal` as a list of *modules* with optional connections. All we want to emulate is a single 8051, so we need to add an 8051 module. Select **Item⟶Insert...**



cereal_khwconf Insert dialog

Select 8051 as *Module type* and name the module **cpu**. After you click **OK**, the **cpu** module appears in the list.

There are many 8051-compatible CPUs differing for example in built-in data memory size and built-in program size. If your application is timing-dependent, you may also want to configure the frequency of CPU clock (although this is useful only if your are also emulating the hardware the CPU is communicating with, such as an UART of the communication partner). To configure the CPU, select **Properties...** in context menu of the module (or just double-click the module).

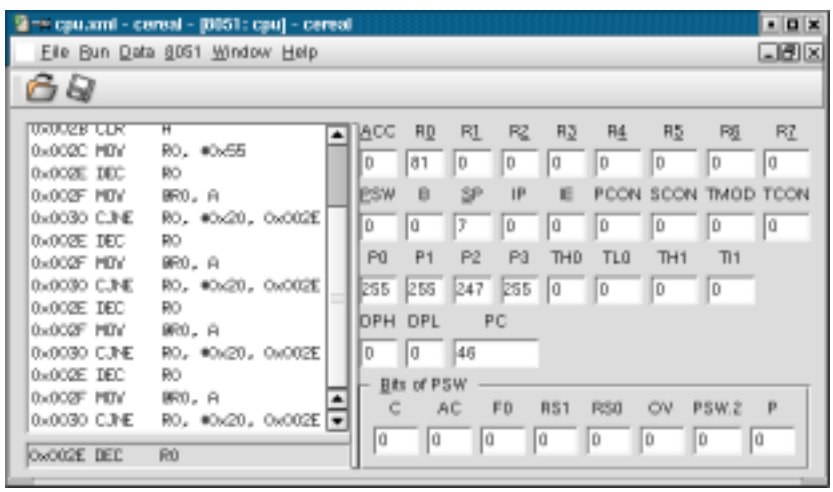


cereal_khwconf 8051 configuration

After setting up the options, save the configuration.

8051 Emulator Interface

Now you can finally run `cereal` itself, **cereal_kde**. Open the configuration file you have just created and `cereal` will start and automatically load the 8051 UI (user interface) extension. This extension provides an 8051-specific window, which you can open by selecting **8051**⟶**New View**.



`cereal` 8051 UI

All that is left is to load your program. Select **8051**⟶**Load Program** and load your program (in Intel HEX format). If you don't have one, a sample program `addregs.hex` (which adds values of R0, R1, … R7, puts the result to DPTR and loops infinitely) is included with this tutorial.

Now you can start emulating your program, using **Run⟶Step** (**F8**). In the left part, you'll see a disassembling appear. The list contains recently executed instructions, and the separate box below shows next instruction to be executed.

The right part contains most 8051 SFRs (special function registers), all of which you can modify by typing in the desired value and pressing **Enter** or selecting a different register. When you select an 8-bit SFR, the *Bits* box below updates to contains bits of this register, so you can also examine and modify the registers bitwise. This is particularly useful when examining PSW (program status word) or any of the MOD and CON mode and control registers.

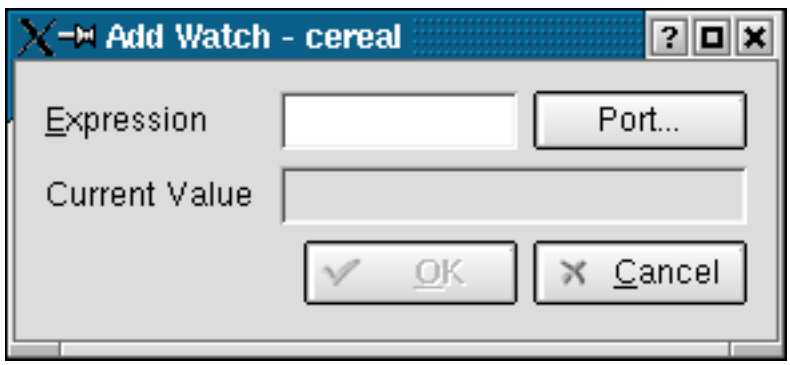
When modifying a register, you can insert any valid `cereal` expression, such as **`1 + 1`**. You'll learn about `cereal` expressions in the next two sections.

> **Note:** You may notice that you can't modify the lowest bit of PSW. That is because this bit contains parity of the accumulator ACC, and `cereal` enforces the right value of the parity bit.

Watches and Breakpoints

If you used the provided `addregs.hex`, you may have had a small difficulty when checking the result. This stems from the fact that the 16b DPTR register is actually composed of two separate 8b registers, DPH and DPL. Therefore you had to calculate the total value by hand.

This problem can be solved using a *watch*. By selecting **Data⟶New Watch Window** you can open a watch window. Then you can add a watch expression by pressing **Insert** or selecting **Insert...** from the context menu.



Add Watch window

We want to see value of 256 * DPH + DPL. Because `cereal` can emulate much more than a simple CPU (in fact, you can easily emulate three CPUs running in parallel), we must tell `cereal` which module should it read the values from. In our case, the module is named **`cpu`** (this is the name given when configuring hardware). Additionally, there is more to the CPU than the registers—you may also want to watch a byte of internal memory, or state of a CPU pin.

This all is contained in the concept of a *port*, which uniquely identifies a module and a variable the module exports. That's why there is the **Port...** button. Click it and in the dialog select module **`cpu`**, space **`sfr`** (*spaces* are groups of ports with similar properties) and port **`DPH`**.

> **Note:** You can either select the port name from the drop-down list, or type it in, or type the numerical index (in our case, DPH has 8051 address 0x83, so the offset is 0x03, as described in the sfr space comment). Using the number is obviously bad idea here, but not

all ports have names—for example **`mem`**, the internal memory space, contains unnamed ports for the whole memory space, which can be specified only using the numerical index.

After selecting the port, click OK and the a port reference (`[cpu/sfr/DPH]`) will appear in the *Expression* box. Append **`* 256 +`** and insert port reference to the DPL register. After clicking **OK**, the watch will be added to the list and updated whenever contents of the registers change.

Note: If you don't like the Port dialog box, you can just type the port reference in, any mistakes you might make will be reported in the *Current Value* field.

Another annoying thing when using the sample program is that in order to see the value, you need to step through almost 70 instructions before you see the result. This certainly disqualifies `cereal` from replacing your calculator application ;-). And because 8051 lacks any HALT or STOP instruction, you can't just let the program run, because simple inspection shows that the program is ended by an infinite loop at address 0x0023. The solution is to create a *breakpoint* which triggers when the program counter reaches 0x0023.

Breakpoints are conditions that are periodically evaluated and stop emulation when they *trigger*. In `cereal`, breakpoints are ordinary expressions which trigger whenever they evaluate to a non-zero value. Breakpoints are defined in the same way as watches, except that you use a Breakpoints window (opened by **Data⟶New Breakpoint Window**). Enter the expression **`[cpu/misc16/PC] == 0x0023`** (of course, you can use the Port dialog to insert reference to PC). Now you can just use **Run⟶Run** (**F9**) to run through the whole program and get instant results.

Note: When a breakpoint triggers, it is automatically deactivated until it evaluates to a zero value at least once. This allows you to easily break program execution on a leading edge of a signal (such as the ALE (Address Latch Enable) output from the CPU). Breaking program execution on falling edge of a signal can be accomplished by negating the breakpoint condition.

`cereal` Expressions Compendium

You have seen how to use expressions for watches, breakpoints and modification of 8051 registers. This sections aims to be a complete reference of what you can and cannot do using `cereal` expressions.

An expression is build from operands and operators, which may optionally be separated by white space. All expressions are evaluated in the widest signed integral type your compiler supports (which is at least 64 bits wide). The expressions are based on C programming language, so if you know C, most of the description should sound familiar.

Valid *operands* include:

Numeric constants

Numeric constants can be expressed in decimal, octal (with leading **`0`**) or hexadecimal (with leading **`0x`**).

Character constants

Character constants have value defined by the character set used by your OS. A character constant is formed by enclosing a single character in apostrophes, like this: **'A'**. Instead of a single character, you can also use a C-like escape sequence:

\'	The apostrophe character itself (you can't use **"'** to represent it).
\", **\?**	The **"** and **?** characters. These escape sequences are provided only for compatibility with C.
\a	Alarm (BEL) character.
\b	Backspace (BS) character.
\f	Form-feed (FF) character.
\n	New-line (LF) character.
\t	Horizontal tabulator (HT) character.
\r	Carriage-return (CR) character.
\v	Vertical tabulator (VT) character.
ooo	Character with given code (in octal, up to three digits).
\x*x...*	Character with given code (in hexadecimal, any number of digits)
****	The \ character itself (single \ character is recognized as a start of an escape sequence).

Port references

Port references are in the form **[*module/space/port*]** or **[c:*module/space/port*]**. In most cases, there is the **Port...** button available for easy port reference creation. The meaning of the **[c:**... form will be explained in the following section.

Valid *operators*, listed in order of decreasing precedence (can be overridden by using parentheses **(** and **)**):

- Unary operators

+ *a*	Unary plus	Unchanged value of *a*
- *a*	Unary minus	Value of *a* negated
! *a*	Logical NOT	1 if *a* is 0, 0 otherwise

- Multiplicative operators

a ***** *b*	Multiplication	*a* multiplied by *b*
a **/** *b*	Division	*a* divided by *b*, rounded toward zero

a **%** *b*	Modulus	Remainder of dividing *a* by *b*. Note that when *a* is negative, the remainder is negative too.

- Additive operators

a **+** *b*	Addition	*a* added to *b*
a **-** *b*	Subtraction	*b* subtracted from *a*

- Bitwise shift operators

a **<<** *b*	Bitwise shift left	*a* shifted to the left by *b*bits
a **>>** *b*	Bitwise shift right	*a* shifted to the right by *b* bits. Note that results of shifting negative numbers right may vary between different computer architectures.

- Relational operators

a **<** *b*	Less than	1 if *a* is less than *b*, 0 otherwise
a **>** *b*	Greater than	1 if *a* is greater than *b*, 0 otherwise
a **<=** *b*	Less than or equal	1 if *a* is less than or equal to *b*, 0 otherwise
a **>=** *b*	Greater than or equal	1 if *a* is greater than or equal to *b*, 0 otherwise

- Equality operators

a **==** *b*	Equal	1 if *a* is equal to *b*, 0 otherwise
a **!=** *b*	Not equal	1 if *a* is not equal to *b*, 0 otherwise

-

a **&** *b*	Bitwise AND	Each bit of result is 1 if the corresponding bits of both *a* and *b* are 1, 0 otherwise

a **^** *b*	Bitwise XOR	Each bit of result is 1 if the corresponding bits of *a* and *b* are not equal, 0 otherwise

a **|** *b*	Bitwise OR	Each bit of result is 1 if at least one of the corresponding bits of *a* and *b* is 1, 0 otherwise

a **&&** *b*	Logical AND	Result is 1 if both *a* and *b* are non-zero, 0 otherwise

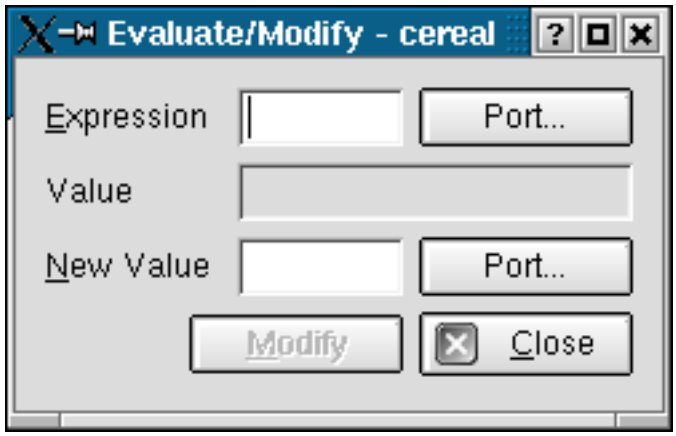
a **||** *b*	Logical OR	Result is 1 if at one of *a* and *b* is non-zero, 0 otherwise

a **?** *b* **:** *c*	Condition	*b* if *a* is non-zero, *c* otherwise

Note that unlike in C, all expressions are completely evaluated and the **&&**, **||** and **? :** operators have no shortcut semantics. Another difference from C is that the **~** (bitwise negation) operator is not included. The reason is that `cereal` expressions have no associated type and the result would be bitwise negation in the integral type expressions are evaluated with, which is probably not what you want. Instead of **~**, use explicit bitwise XOR: to negate an 8-bit value *x*, use ***x*** **^ 0xFF**.

Evaluate/Modify

The last feature of the `cereal` UI is the *Evaluate/Modify* dialog box. Invoke it by selecting **Data⟶Evaluate/Modify...** (**F4**).



Evaluate/Modify window

The Evaluate/Modify dialog box can be used to compute value of an expression you don't need as a watch (this is the *real* replacement of your calculator application). Just enter the expression and the value will be computed as you type.

The dialog box can be also used for modification of memory (or generally port) values that are not available in the 8051 interface by referencing the port in the *Expression* box and writing the new value to the *New Value* box.

The string you enter to the *New Value* box can of course be any valid `cereal` expression. If you feel bored, set *Expression* to **`[cpu/sfr/ACC]`**, *New Value* to **`[cpu/sfr/ACC] + 1`** and hold the **Enter** key for a while.

When modifying a port value, you can use any valid `cereal` expression in the *Expression* box just as in the *New Value* box. Thus almost the same effect as in the previous paragraph can be achieved by setting *Expression* to **`[cpu/sfr/ACC] - 1`** and *New Value* to **`[cpu/sfr/ACC]`**. `cereal` contains a minimal equation solver, which allows you to modify values of simple expressions without computing the solution yourself.

The equation solver is quite a trivial piece, so about the best you can expect it to solve is a linear equation, it can't solve **`[cpu/sfr/ACC] * [cpu/sfr/ACC] = 4`**. The general rule is that the expression to be modified can only contain a single port reference, together with the fact that `cereal` refuses to modify the port value if there are multiple possible solutions.

Of course, changing the register values in the 8051 window is done using the same mechanism and modifying the value in the *ACC* edit box is equivalent to doing it in the Evaluate/Modify dialog box.

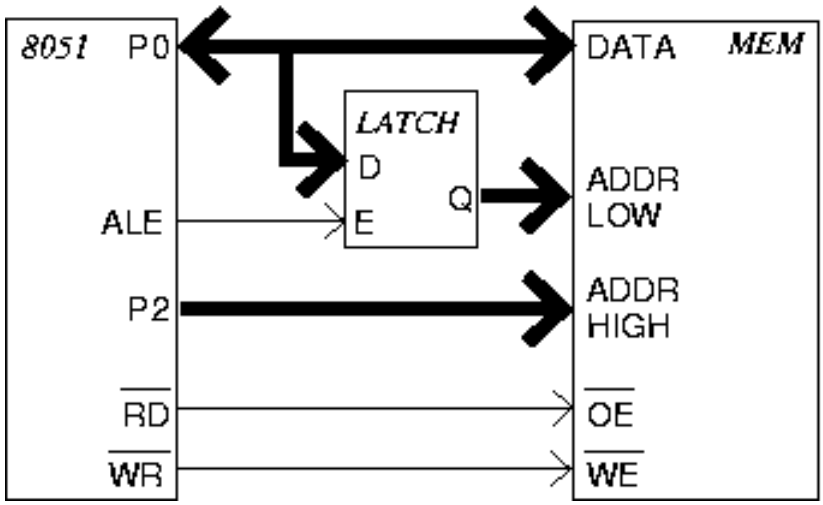
If you know the 8051 architecture a bit, you'll recall that the R0, ..., R7 registers are not registers with fixed addresses at all. R0, ..., R7 stand for internal memory locations either 0, ... 7 or 8, ..., 15 or 16, ..., 23 or finally 24, ... 31 according to RS1 and RS0 bits of PSW. This is still possible to implement using `cereal` expressions. For example, the expression for value of R0 is **`([c:@/sfr/PSW] & 0x10) == 0 ? (([c:@/sfr/PSW] & 0x08) == 0 ? [@/mem/0] : [@/mem/8]) : (([c:@/sfr/PSW] & 0x08) == 0 ? [@/mem/16] : [@/mem/24])`**, where `@` represents the module name. This expression is much more complicated than what the rule above allows for expressions that the solver can handle (five different port references, one of them repeated three times). But we know that we want to modify one of the **`[@/mem/x]`** values and not the **`[@/sfr/PSW]`** value. Moreover, depending on the value of **`[@/sfr/PSW]`**, one simple expression (which can be handled by the solver) is selected. In this expression, we have helped the solver by inserting the **`c:`** (meaning *constant*) prefix to the PSW port references. This tells the solver that these ports are not to be modified. The solver takes these port references as if it were constants and with a bit of constant expression evaluation it can reduce the expression to the **`[@/mem/x]`** form, which can be handled easily.

Notes

1. The code currently does not support data paths wider than 16 bits, but it is simple to extend it up to number of bits of widest integral data type your machine supports.

Chapter 3. Connecting External Devices

The main strength of `cereal` (and, indeed, the very reason of its existence) is the ability to emulate sets of connected devices as a group and support of comparatively easy adding of new devices. In this chapter you'll see this ability demonstrated by connecting an external 64K × 8 memory chip to the 8051 CPU. To be able to use 8051 MOVX instructions, we need to connect port P2 to higher 8 memory address pins, port P0 to memory data pins. Lower 8 bits of memory address are also provided at port P0, and are supposed to be latched by the ALE signal. The whole situation is depicted in the following schema:



In `cereal`, this is represented as a set of three modules, representing the CPU, the latch and the memory chip, respectively. Unsurprisingly, all of them are available as `cereal` built-in modules. Each of these modules exports the "pins of the emulated chip". We need to connect the modules: for example, we want to assure that when the 8051 module changes value on the /RD pin, the memory module receives this as a change of its /OE pin. Similarly, we want that when the memory module reads current status of its /OE pin, it actually reads the value of the /RD pin of the 8051 module.

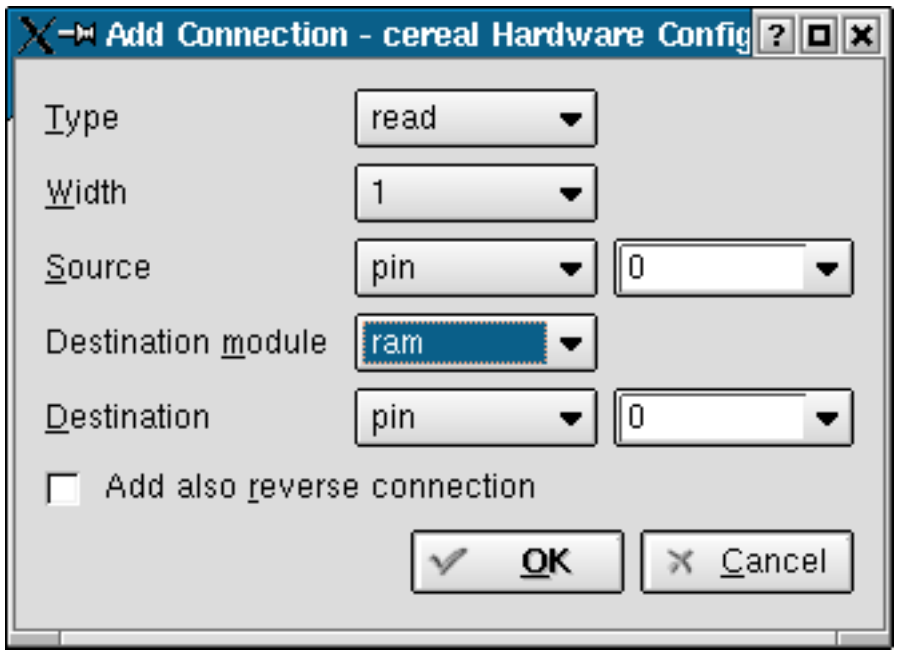
The two mentioned relationships are called *connections* in `cereal` and are configured using the `cereal` hardware configuration utility. Run **cereal_khwconf** and insert three modules: **`cpu`** of type 8051, **`latch`** of type byte_latch and **`ram`** of type memory. *Don't forget* to set memory data_size to 65536!

We'll start with the above mentioned examples: we want to create an *write* connection from **`[cpu/pin/~RD]`** to **`ram/pin/~OE`** and a *read* in the reverse direction. This situation (read/write in opposite directions) naturally occurs quite often, so you can create only one of the connections and if possible, `cereal` will create the reverse one automatically.

> **Note:** The other two port types, *display* and *modify* are rarely used. They are the way the user interface reads and changes the port values. The difference is sometimes important, for example just reading a status register may clear an interrupt line. For example of creating display or modify connections, see description the sfr_ext space of the 8051 module in the `cereal` Built-In Modules chapter.

To add the connection, bring up the Properties dialog for **`cpu`**, choose the *Connections* tab and click on **New...**. By opening the Properties dialog for the **`cpu`** module, you have implicitly defined the source module. Now select **`write`** in *Type*, **`1`** in *Width* (single-bit connection), **`pin`** and **`~RD`** in *Source*, **`ram`** in **`Destination module`** and

finally **pin** and **~OE** in *Destination*. Note that the *Add also reverse connection* checkbox is automatically checked. By clicking **OK** you create the connection.



Add Connection window

Similarly you can continue adding other connections. Admittedly it is a bit lengthy, but you only need to do it once—not at all in this case, because the result, `8051_mem.xml` is available with this tutorial.

> **Note:** When trying to connect this yourself, you could have noticed that the precise configuration cannot be created, because the P0.*x* ports cannot be connected for writing twice (to the latch and to the memory). The solution would be to add 8 "bit_tee" modules, but this has been sidestepped by using the coincidence that the ram module always reads the values from the CPU itself and therefore doesn't need the write connection. This is obviously a bad kludge, but it works and I didn't have to program the bit_tee module, insert it 8 times and create 8 more connections.

To test this module, you can use enclosed `8051_mem.hex`, which negates all bytes of the external memory.

> **Note:** If you use /PSEN instead of /RD in the above example, the CPU will be able to fetch instructions from the external memory (make sure to note the function of the EA pin). In that case, the next instruction disassembling in the 8051 window is not available, because there is no way for the 8051 UI to get the value from the external memory other than to emulate the access just for the purpose of the disassembling, which of course doesn't happen in the real world.

Chapter 4. `cereal` Built-In Modules

With `cereal` comes the all-important 8051 module and a few quite simple modules which should provide you with a starting point for creating your hardware configuration as well as your own modules.

You can write additional modules, which can be either built-in (if you recompile `cereal`), or dynamically loaded, in which case you need to point environment variable `CEREAL_MODULE_DIR` to a directory containing the modules.

The 8051 module

Need I say more? Emulates most of Intel 8051-compatible CPU. Unsupported features include the PD (Power Down) and IDL (Idle) bits of PCON. Also, the module has its own, built-in clock (i.e. you can't emulate clever tricks with stopping the CPU clock), although this is not terribly hard to fix (it will just slow down the emulation even more).

The module has only one unexpected feature: XMEM is a built-in "external" memory, a built-in memory accessed using the MOVX instruction. This occurs quite often in practice, i.e. Atmel 89S8252 has a built-in EEPROM accessed using MOVX. This cannot be always emulated by emulating a "true" external memory (as demonstrated in the previous chapter), because using the "true" external memory uses 19 pins of the CPU, while accessing the built-in EEPROM does not, and at least 18 of them can be used for other purposes. Therefore applications that use this EEPROM and also use some of the pins cannot be emulated without having XMEM.

Table 4-1. 8051 Module Options

Name	Description
data_mem_size	Size of internal data memory, one of 128, 192 and 256.
frequency	A non-zero integer, number of oscillator pulses per second (i.e. 11059200).
prog_mem_size	Size of internal program memory, a power of two in range [64, 65536].
xmem_mem_size	Size of XMEM (described above), a power of two less than or equal to 65536, or 0 to use "regular" external memory.
load_hex	A hidden option which causes the module to load the Intel HEX file provided as a parameter.

Table 4-2. 8051 Module Spaces

Name	Width	Description
bit	1	Equivalent to 8051 bit addresses.
pin	1	Contains emulated 8051 pins.
mem	8	The 8051 internal data memory.
prog	8	The 8051 internal program memory.

Name	Width	Description
sfr	8	Contains the SFR address space (0x80 . . . 0xFF). Because port addresses are 0-based, this is mapped to range 0x00 . . . 0x7F.
sfr_ext	8	A mirror of the sfr space for the purpose of extending the 8051 emulator. To emulate an additional SFR, create a separate module for it implementing all four port types (read, write, display modify) that make sense for the SFR, and connect it to the appropriate sfr_ext port. Then any connection your SFR in the sfr space will be forwarded to the module you connected to the sfr_ext space.
prev_insn	8	Contains the last executed instruction. Used in the KDE UI plug-in.
next_insn	8	Contains the next executed instruction, if not using external memory. Used in the KDE UI plug-in.
misc16	16	Contains the program counter and its value at previous instruction.

The bit_constant Module

This module has a single port that can be read by other modules and modified by the user.

Table 4-3. bit_constant Module Spaces

Name	Width	Description
bit	1	Contains the bit.

The bit_report Module

This module has a single port that can be written by other modules and reports changes of its value.

Table 4-4. bit_report Module Options

Name	Description

Name	Description
name	A string the module prepends to its value change reports.

Table 4-5. bit_report Module Spaces

Name	Width	Description
bit	1	Contains the bit.

The byte_constant Module

This module has a single port that can be read by other modules and modified by the user.

Table 4-6. byte_constant Module Spaces

Name	Width	Description
byte	8	Contains the byte.

The byte_latch module

This module emulates a 8-bit latch. If somebody cares, it was created using documentation of Czech chip MH74ALS373.

Table 4-7. byte_latch Module Spaces

Name	Width	Description
pin	1	Contains the pins of the latch.

The memory Module

This module emulates a static memory of up to 64K $\times$ 8.

Table 4-8. memory Module Options

Name	Description
data_size	A power of two less than or equal to 65536, capacity of the memory module in bytes. The module ignores unneeded address lines.

Table 4-9. memory Module Spaces

Name	Width	Description
pin	1	Contains the memory pins.

Name	Width	Description
data	8	Contains the data kept in the memory.

The uart Module

Emulates a very simple UART. Connect RXD and the module will write any received data to RX, write data to TX and the module will transmit it on TXD. After completion it will set RX_done or TX_done, respectively, to 1 (stays 1 until cleared by writing a 0).

Table 4-10. uart Module Options

Name	Description
baud_rate	An integer, baud rate the UART is using.
data_bits	A non-zero integer less than or equal to 13, number of bits in one data unit.

Table 4-11. uart Module Spaces

Name	Width	Description
pin	1	Contains the UART pins.
data	16	Contains the received and transmitted data units.

Chapter 5. `cereal` Command-Line Utilities

In the course of writing `cereal`, two command-line utilities were created. They are used in the automatic testsuite created during `cereal` development, but they may be useful also to you.

cereal_disasm

The **cereal_disasm** utility is a simple 8051 disassembler. When invoked with arguments, it treats them as file names of Intel HEX files, loads them all in the order given on the command line. When given no arguments, it reads one Intel HEX file from standard input instead. Then it outputs the disassembling to standard output.

cereal_text

The **cereal_text** utility is a text interface to the `cereal` emulation framework, which is able to do most of what is available using the GUI. It can be used for automatic regression testing not only of `cereal` itself, but also of your applications.

The user interface is admittedly crude. It expects commands on the standard input and writes results to the standard output. An empty line means to repeat the previous command.

Table 5-1. cereal_text commands

Synopsis	Description
`help` ***`command`*** or ***`command`*** **`--help`**	Gives short help on usage of *`command`*.
`mod_new` ***`type name`***	Creates a new module *`name`* of type *`type`*.
`mod_delete` ***`name`***	Deletes module *`name`*
`option` ***`module option`***	Displays current value of option *`option`* of module *`module`*.
`option` ***`module option`*** *`value`*	Sets option *`option`* of module *`module`* to *`value`*.
`connect_1` ***`type port_1 port_2`***	Connects for *`type`* bit port *`port_1`* to *`port_2`*.
`mod_rename` ***`old new`***	Renames module *`old`* to *`new`*
`breakpoint` ***`expression`***	Creates a new breakpoint on *`expression`*
`bp_list`	Lists defined breakpoints, along with their ID numbers. If a breakpoint currently evaluates to nonzero, it is marked by a "+" character.
`bp_del` ***`id`***	Deletes breakpoint with ID *`id`*.
`print` ***`expression`***	Prints current value of *`expression`*.
`set` ***`dest = src`***	Sets port referenced in *`dest`* so that *`dest`* == *`src`*.

Synopsis	Description
step ***condition...***	Runs one step of emulation. If there is a list of *condition*s specified, runs until one of them is met. Valid conditions are error, warning, insn (end of instruction) and breakpoint.
setup_load ***file***	Loads hardware setup from *file*
setup_save ***file***	Saves hardware setup to *file*
state_load ***file***	Loads emulation state from *file*
state_save ***file***	Saves emulation state to *file*

Appendix A. GNU Free Documentation License

Copyright © 2000 by Free Software Foundation, Inc.

Free Software Foundation, Inc.
59 Temple Place, Suite 330,
Boston,
MA 02111-1307
USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each

Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A

Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B

List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C

State on the Title Page the name of the publisher of the Modified Version, as the publisher.

D

Preserve all the copyright notices of the Document.

E

Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F

Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G

Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H

Include an unaltered copy of this License.

I

Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J

Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K

In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L

Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M

Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N

Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version .

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and dispbibute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document. If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with trans-

lations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation[1] may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/[2].

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

Addendum

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME.
>
> Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License[3], to permit their use in free software.

Notes

1. http://www.gnu.org/fsf/fsf.html

2. http://www.gnu.org/copyleft
3. http://www.gnu.org/copyleft/gpl.html