Coffee Making

Table of Contents

Coffee Making

Fotis Georgatos < gef@ceid.upatras.gr>

V1.0 2004−08−29

One of the most memorable comments about software ever said is whether this or that piece of code can make coffee. Coffee is a world commodity that is second only to oil. Linux DOES make coffee; and it tastes good as well!

1. Introduction

2. Menu

3. Hardware

4. Software

5. Building the Turing Complete Coffee Machine

---

1. Introduction

For a long time humanity has been wondering how could a computer make coffee...

People need coffee to wake up, and stay awake for a long time in front of the computer. It is common wisdom that coding is better at night!

The main trick is interfacing a coffee machine to the computer, so that it can be controlled by software. This HOWTO will show you how to do so.

At first, it demonstrates an ON/OFF switch implemented as an electronic circuit which controls the coffee−machine's power supply. Another chapter will tell you the secrets of building intelligent, Turing Complete suitable, coffee machines!

This HOWTO was initially written as part of a debate in the mailing list linux−greek−users, on whether linux can make coffee or not. It then became an article in our online magazine called magaz. Just in case you wondered, magaz is in Greek and it will surely look like that to you!

Enjoy.

1.1 Copyright

1.2 Disclaimer

Use the information in this document at your own risk. I disavow any potential liability for the contents of this document. Use of the concepts, examples, and/or other content of this document is entirely at your own risk.

All copyrights are owned by their owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark.

Naming of particular products or brands should not be seen as endorsements.

You are strongly recommended to take a backup of your system before major installation and backups at regular intervals.

1.3 Version

The Coffee HOWTO is now called Coffee Making HOWTO and heads for release v1.0, which will first appear somewhere here:

http://fotis.home.cern.ch/fotis/Coffee.html.

It is about time for everyone to know that Coffee Making is just one of the standard features that come for free with *any* Linux distribution. Or, does SCO have a patent on that, too? Gee...

1.4 Translations

You should be able to easily find a translation of this or previous versions of the Coffee Making HOWTO in the following languages:

- Chinese
- Japanese
- Russian
- Indonesian
- Italian
- Polish

and hopefully many more... (Spanish and Portugeuse anyone?)

1.5 Credits

- Ethiopia: Identified as the originating country of coffee.
- Coffee was popular in the Middle East for ages, until a failed invasion of the Turks at the city of Vienna at 1683 left behind sacks with strange brown beans nobody wanted: http://www.vienna.cc/ekaffeeh.htm/.
- Kostas Lialiambis is the one who dared claim he can't make coffee with his Linux box, back then in year 1997.
- Panagiotis Vrionis et al for giving me interesting and humorous notes and let the ball roll in the early days.
- NUMEROUS people on the internet with additions and remarks. Thank you all, really! Even though I might have not replied to your email.

1.6 What do you drink/smoke while writing this?

Well, to the best of my knowledge, this is a dope−free work.

But, I can tell you the secret of the music playing on the background: nearly any song spelled by Zampetas or Mpithikotsis (bouzouki and such).

1.7 Feedback

For your online commentary of your own DIY Coffee Machine steer at http://coffee.sf.net/.

If you still have comments to say, emails get lost these days, so why not send me a postcard with a picture from your great hometown, adding a recommendation of your favourite cafe' in the area?


```
Fotis Georgatos,
Aliartou 32,
TK 11142 Athens,
GREECE
```


PS.

- Your suggestions won't be wasted, I really tend to travel quite a lot these days and might be around before your realize it.
- Yes, I have been in Amsterdam and even lived there for three years; coffee shops generally don't account as cafe's!
- Surprise: I actually prefer chocolate and tea over coffee. ;−)

---

2. Menu

2.1 French

Popular coffee among programmers because it doesn't need a lot of care and its cooking is streamlined; just like commercial software. Its exciting taste has inspired thousands of programmers in writing incredible software, written in the very first hours of a day. M$ Windows for example has been written at 5:00 o'clock in the morning, only thanks to coffee! A similar result is guaranteed.

2.2 Nescafe

Nescafe is a rather strong coffee, made by pouring hot water in a mixture of coffee, sugar and some water. You usually take 1 spoon of coffee and 1 spoon of sugar with just a bit of water, to mix it. In the meantime you should have the water boiling. As soon as the water is hot enough, you mix them all together and preferably add condensed milk. Although you could use something simpler than a coffee−machine to boil the water, I have seen this done so several times...

2.3 Frappe'

A popular variation of the above mentioned coffee. Actually, you don't need a sophisticated coffee−machine, rather a refrigerator for cold water and ice−cubes. It is very popular in South Eastern Europe during the warm summer months.

2.4 Freddo

This is a complicated one, read coffee−faq (read further information)

2.5 Espresso

Espresso is a very strong, italian sort of coffee. You serve it in small cups (You ask why? See chapter: Overdose Symptomes) with one or two pieces of lump sugar. To produce a good espresso you need fresh grinded coffee beans, water, lump sugar and a special machine. These machines boil the water and press the very hot steam through the grinded coffee beans. You can buy a super−duper−automatic machine for a lot of money. But a low cost machine is usable, too.

OK., lets start. Fill water in your machine. Let it become hot. In the meantime fill about 1 teespoon of coffeepowder in the filterhandle of your machine. Press the coffeepowder down. Not too much. Now the water is at the right temperature. Attach the filterhandle to the machine and let the machine work. After about 30 seconds you can serve a delicate, hot espresso. It is fine after a good meal. You feel good and can code for a few more hours.

2.6 Cappuccino

(See also chapter: Espresso) If you have a more profi−like machine, you can use it, to froth milk with it. You need this feature to make a creamy sort of coffee. It is easy to prepare. Put some frothed milk in a coffee pot and fill it up with espresso. Then decorade with some chocolade flakes. That's it.

---

3. Hardware

A generic diagram could look like this:

```
---------   0-5V  ---------   ~220V   ----------------
|  PC    |====>====|Circuit|==========|Coffee-Machine|
---------         ---------           ----------------
```

The concept is that we take a controling voltage from the computer, which drives an electrically isolated circuit with a Relay or Triac.

You must choose a Relay circuit, if you have a coffee−machine greater than 200W. You can use a triac−based one if your coffee machine isn't high power.

All circuits presented are tested, but the results and risks are YOUR OWN RESPONSIBILITY. If you have no experience with electronics you should NOT try building it on these, otherwise you may get a bad one...

You should be very careful while experimenting with 220V, and using an appropriate fuse is absolutely advisable.

3.1 Driving voltage 0−5V from the computer

Here is a simple example to get a voltage 0−5V from the parallel port of the computer.

```
       Back View          -----     Pin 10 - ACK
       Male DB-25         |   |     Pin  9 - D7
       Connector          |   |                            Pin 2 - D0
                          v   v                            v   Pin 1 -
~Strobe
        ____________________________________________________________
       /                                                            \
       \     13   12   11   10    9    8    7    6    5    4    3    2    1     /
        \                                                          /
         \     25   24   23   22   21   20   19   18   17   16   15   14      /
          \______________________________________________________/
```

Pin 1 is Strobe (inverse logic)

Pins 2−9 is DATA BUS's signals, exactly what was written to the parallel port's latches with an OUTB command.

Pin 10 is the acknowledge signal (ACK), controlled by you, so that you can produce an interrupt to the CPU.

Pins 18−25 are short−circuited and this is the ground (GND).

In detail:

```
<= in   DB25    Cent    Name of         Reg
=> out  pin     pin     Signal          Bit     Function Notes
------  ----    ----    --------        ---     -----------------------------
=>        1       1     -Strobe         C0-     Set Low pulse >0.5 us to send
=>        2       2     Data 0          D0      Set to least significant data
=>        3       3     Data 1          D1      ...
=>        4       4     Data 2          D2      ...
=>        5       5     Data 3          D3      ...
=>        6       6     Data 4          D4      ...
=>        7       7     Data 5          D5      ...
=>        8       8     Data 6          D6      ...
=>        9       9     Data 7          D7      Set to most significant data
<=       10      10     -Ack            S6+ IRQ Low Pulse ~ 5 uS, after accept
<=       11      11     +Busy           S7-     High for Busy/Offline/Error
<=       12      12     +PaperEnd       S5+     High for out of paper
<=       13      13     +SelectIn       S4+     High for printer selected
=>       14      14     -AutoFd         C1-     Set Low to autofeed one line
<=       15      32     -Error          S3+     Low for Error/Offline/PaperEnd
=>       16      31     -Init           C2+     Set Low pulse > 50uS to init
=>       17      36     -Select         C3-     Set Low to select printer
==      18-25   19-30,  Ground
```

3.2 Controlling with a Relay

The straight−forward circuit one can build is:

```
                             Vcc
                              |
                              +------+
                              |    __|__
                            Relay   /^\  Diode 1N4002
                             Coil  /---\
                              |      |
                              +------+
                              |
                           | /
                 4.7K    B |/  C
parallel port >-\/\/\/\/---|        NPN Transistor: BC547A or 2N2222A
data pi                    |\  E
                           | \
                             V
parallel port >--------------+
ground pin                   |
                          Ground
```

Connect Vcc with the same voltage as the relay type (usually 5 or 12V). Obviously, the relay's specifications should be scaled for your coffee−machine.

Barmen, tend to put the relay AFTER the transistor, at the emitter (E) pin instead of the collector (C) pin. This is bad practice because it biases the transistor badly, and might result in bad coffee :−). Diode 1N4002 is useful to protect the transistor from the relay's currents. If you don't use it the transistor will sooner become dark and smelly...

3.3 Controlling with TRIAC #1

If you only want a simple circuit, you can use Motorola's triac driver MOC301[012], together with a general purpose TRIAC like SC141D. This method has the advantage that you don't need any extra power supply.

For non−inductive loads, this is the circuitry:

```
      270     1 +-------+ 6    180
+5v -VAVAVA-----+       +----VAVAVA-----+-------------- Line Hot
              2 |  MOC  |               |
TTL in ---------+ 3012  +nc            VA  SC141D
                |       | 4           / |
              nc+       +------------/  |
                +-------+               +----\/\/\/---- Line Neutral
                                              LOAD
```

If you are going to work with 220V, try to obtain a 3021. Inductive loads should be used in conjuction with bypass capacitors, better consult *Motorola Application Note AN−780*. Coffee−machines are mainly resistive loads and not inductive (like a motor), but who knows what yours is?

3.4 Controlling with TRIAC #2

```
+5VDC
|    180                      180            2.2k
+---/\/\/\----+-----+   +----/\/\/-+--/\/\/\---+-------> 120V
              |    1|   |6         |           |         Hot
              |    +=====+         |           | MT1
```

```
              |    | MC  | TRIAC   |           +-+
              |    | 3032| Driver  |         G | | TRIAC
              |    +=====+         |          /| |
              \    2|   |4         |         / +-+
       2N3904  |----+   |          |         |  | MT2
              /     |   +--------- | -------+  |
             V      \              |         |  |
             |      /              |         \  |
             |      \ 43    .01u  ---   10k /  |
             |      /       500V  ---       \  |
             |      |              |         /  |
             +------+              |         |  |          Neutral
             |                     +--------+--+---o    o--> 120V
             /                                      load
 >-/\/\--|  2N3904
             \
              V
              |
             ---
            ///

This design is for 120V. You should change resistors accordingly for 220V.
```

Circuit description:

The MC3032 is an optoisolator TRIAC driver. The 180−ohm resistor sets the current for the LED emitter in the optoisolator. Change the value of this resistor − if necessary − to get a reasonable current (e.g., 15 mA).

Note that you cannot test this circuit without a load. The TRIAC will not switch unless connected to an AC voltage source, so you can't test it for simple switching without applying AC and a load. Note the 500V rating on the .01 capacitor.

---

4. Software

4.1 Programming

You will have to build an executable that will take the following steps:

- Get permission to use I/O address space, by calling kernel, with the call **ioperm**: eg *ioperm(BASE, range ,1);*
- Perform an out request instruction, to set the 0−5V voltage to the parallel port, eg *outb(1, BASE);*
- Wait enough time so the coffee is made. It would be nice if that time is read by looking at the command line.
- Then it will turn off the coffee−machine: outb(0 , BASE);
- Before ending it should give back the parallel port with an *ioperm(BASE, range, 0);*

Change BASE = 0x3bc for /dev/lp0, 0x378 for /dev/lp1, and 0x278 for /dev/lp2; range=8.

It would be useful if you had that program setuid, so that everybody can drink coffee! You BOFH!

4.2 Device driver

Just read kernel hacker's guide, to implement a device driver; you might also do it in user space. Please compile it as a module, so that we won't need a kernel compile in every update. Then write:

```
echo cappuccino >/dev/coffee
```

And you will have a hot cup of coffee in minutes! Remember to give the right permissions to /dev/coffee, depending on whether you want only root making coffee or not.

The advantage of this method is that it supports feedback from the coffee−machine by using the ACK of parallel port and such, so that smart coffee−machines can produce an interrupt when ready.

Do it yourself, after reading the excellent book of Alessandro Rubini and Jonathan Corbet Linux Device Drivers and studying the Cross Reference Linux source code repository.

4.3 Connecting with the Internet

If you have implemented the controlling program in C (see above), you just have to write a CGI script to turn ON and OFF the coffee−machine or pass along more complex instructions. You should write some nice webpages, explaining how to make coffee, and put them on an **apache** web server...

...LAMP technology (Linux, Apache, MySQL, [Perl|Python|PHP]), will help you to build a perfect user−customizable coffee system!

At some time in the future when the applications get rather complex, you might want to extend on the basis of Flow−Based Programming: http://www.jpaulmorrison.com/fbp/. What a great match for a great Coffee Machine!

---

5. Building the Turing Complete Coffee Machine

Do you pine for the nice old days, when men were men and build their own coffee machines?

This chapter is about assembling a smart, intelligent!, coffee machine. It will be a computer designed with a von Neumann architecture, comprised of a CPU, ROM/RAM and I/O and will also be suitable for generic use, a.k.a. Universal Turing Machine.

5.1 An adequate assembly language

Unlike other complex, but popular, systems that are either CISC or RISC, our machinery will be MISC: Mono−Instruction Set Computer!

Alas, our processor will understand just one command and yet, given enough memory and time, it is able to perform any action that your 3GHz Pentium IV could do, or just simulate it alltogether; It can solve any computable problem by running simple code like this:

```
        SBN     $mem1, $addr1
        SBN     $mem2, $addr2
```

```
SBN     $mem3, $addr3
SBN     $mem4, $addr4
SBN     $mem5, $addr5
SBN     $mem6, $addr6
[...]
```

The magic command is called SBN $mem, $addr (Subtract and Branch if Negative), and does take the value of a memory cell $mem, subtract it from the accumulator (A is the only available register in this architecture) and store it back to the accumulator and memory $mem *: [mem] <= A <= A−[mem]*. If the result is negative, and only then, it jumps to the designated address $addr. If $addr points to the next command, there is no conditional jump. Now, with that command at hand you can subtract, add, zero memory addresses, move bytes around, multiply, compare and so on, so forth. What's best of all, you can easily build an optimizing compiler.

Voila. This is a great system for any Turing Complete problems plus, it is even simpler in coding than the original Turing Machine!

5.2 Hardware and interfacing

The great thing with this innovative MISC processor is that you need 0 bits to store the opcode of your commands. This makes your CPU much, much simpler: you only have to read a couple of operands each time. You might wish to extend the capabilities of your processor by extending the SBN instruction to 3 or 4 operands, so that it can directly load and store data from main memory. This is an exercise left to the reader; kudos to google.

The CPU diagram looks like this:

```
<========= ADDRESS BUS ==============>
       =                 =
       =  +---------+    =
       =  | CONTROL |    =
   +---------+  +-----------------+
   | ALU & A |  | Program Counter |
   +---------+  +-----------------+
       =  |  LOGIC  |    =
       =  +---------+    =
       =                 =
<=========== DATA BUS ===============>
```

Now, all you have to do is just hook together some memory chips, for example by recycling static cache RAM from old 386 PCs, an ALU and a few glue components. You may pick one of TTL or CMOS for logic gates and latches; I'm a CMOS guy, but this really depends on your favourite flavor. You may build an 8, 16, 32, 64 bit or whatever width system you need. Just in case, for larger word widths, I have found preferable building the ALU with pre−programmed 27128 EPROMS instead of the harder−to−find 74x181s. Look around for a carry propagation unit, too.

The monolithic nature of this system allows only memory−mapped I/O, and requires special design provisions for bidirectional interfacing, but nothing more peculiar than what is seen in older−generation systems. AGC, the computer that drove Apollo 11 mission to the moon was making use of such techniques, so it should be sufficient in this case, too.

Note that the data bus has to be exactly as wide as the address bus, that implies that the notion of a byte is only applicable to 8 bit coffee machines, which you will eventually find that it is more of a feature than a bug. You will be surprised with what a coffee you can have for 8 or 16 bits bus! It is really a general−purpose piece of hardware, built for peanuts.

5.3 Software

Such a pure system will make a good fit together with the, famous for embedded systems controlling, FORTH programming language. The major prerequisite for doing so is to have a stack mechanism, which in this case can be constructed by a counter combined together with a memory pool.

If you want to claim a serious coffee development platform, C portability is an absolute must nowadays. Your choices might be hacking one of gcc, lcc or sdcc, which with proper tweaking at the back−ends will be able to spit out the specially crafted MISC assembly code. One day you might even want to rewrite another language like C, forget the D letter − it is taken already, so do not make again the same mistakes with your compiler please: http://www.gnu.org/software/gcc/projects/beginner.html

Just in case you thought of writing your own compiler, please read in advance about flex, yacc and just a little bit of related theory. In particular you will quickly appreciate Noam Chomsky's taxonomy on languages:

- regular grammars (the abstract formalism for regular expressions),
- context free grammars (any BNF−described language is such)

so on, so forth. Read ahead on Computability Theory.

5.4 A minor criticism on the Turing Machine

Because of the way a Turing Machine works (see for that http://plato.stanford.edu/entries/turing−machine/), it is a very complicated device to program, and debug at the end of the day. The reason is, that its behavior is a sequential process that is completely determined by the following parameters:

- (1) the State the machine is in,
- (2) the Symbol (or number) on the square it is scanning, and
- (3) a Table of Instructions

The major contemporary disadvantage of the Turing Machine (TM) is that it is of sequential nature, which implies that only a particular range of problems can be mapped to it in a straightforward way. TMs are suitable for problems that are described well on a serial storage medium (tapes) and don't make use of indexes for data reference. This is in contrast to the Coffee Machine (CM) that can handle any Random Access algorithms as well (with no compromise of simplicity).

Add to this, that TMs impose a very high and unnecessary complexity on item (3) in favor of keeping (1) and (2) simple. And just in case you don't agree that the so called Table of Instructions gets trully overwhelmed, have you ever tried to write a compiler for a Turing Machine? A system that isn't easily programmable and is hard to debug, should be considered a seriously questionable system, at least as far as Computer Engineering (!= CS) is concerned. For instance, try to simulate the Coffee Machine with a Turing Machine and vice versa. Hey, if you still disagree, show me the code.

Bottom Line: The Coffee Machine (CM), is a much better model for the von Neumann architecture and has a O(1) relationship with what is standard practice of weighting algorithms, in the current form of complexity theory.

6. Overdose symptoms

- excitement
- nervousness
- insomnia
- tachycardia or cardiac arhythmia
- restlessness
- Hypersensibility to light
- Annoyance in respect with various audio stimuli
- gastrointestinal disturbance

7. Expansions

- All hardware and software described here, can be expanded so that it will support toast, beaf, applepies, etc.
- Cluster with 8 coffee−machines. This will let you have coffee even when the first one gets off. Of course there will be a perfomance hit.
- Parallel vector coffee−machine will be a future release.
- If you want the maximum automation you'll need more circuits and sensors, so that you can control water flow, temperature, coffee quantity etc.
- In the near future we will implement SNMP features.
- Serial coffee−machine at 115Kbps.

8. Further Information

- http://db.uwaterloo.ca/~alopez−o/Coffee/coffaq.html This is most known Internet's **Coffee−FAQ**
- http://faculty.washington.edu/chudler/caff.html Caffeine and effects of on the Nervous System
- http://www.gardfoods.com/coffee/coffee.coffee.htm A pretty comprehensive, if informal, history of humans' interaction with the coffee plant. People have been chewing coffee berries in Africa for 100,000 years or so. Coffee was definitely growing in Yemen, where it is not native (so must have been planted), in 525 AD.
- http://www.faqs.org/rfcs/rfc2324.html RFC2324: Hyper Text Coffee Pot Control Protocol (HTCPCP/1.0)
- http://www.faqs.org/rfcs/rfc2325.html RFC2325: Definitions of Managed Objects for Drip−Type Heated Beverage Hardware Devices using SMIv2
- http://www.tldp.org/HOWTO/mini/IO−Port−Programming Programming of I/O ports under popular operating system Linux.
- http://www.ee.washington.edu/circuit_archive/circuits/F_ASCII_Schem_PC.html A lot of circuits in ASCII. Some of them are for parallel port.
- http://www.citr.auckland.ac.nz/~james/parport.html Whatever you wanted to learn about a parallel port and didn't dare to ask.
- http://en.tldp.org/LDP/khg/HyperNews/get/khg.html How to write your own device drivers. Come on, go ahead!
- http://www.hut.fi/Misc/Electronics/circuits/parallel_output.html Tomi Engdahl's web page is a *must see* for everyone who enjoys electronics.
- http://dir.yahoo.com/Computers_and_Internet/Internet/Devices_Connected_to_the_Internet/Coffee_Machines/ Coffee−machines on−line. Unfortunately, there are no benchmark tests.

- http://www.cs.su.oz.au/~bob/Coffee/index.html This coffee−machine offers only cappuccino. It has to be upgraded!
- http://einstein.et.tudelft.nl/~janssen/ Hot coffee from the the The Netherlands.
- http://www.cl.cam.ac.uk/coffee/coffee.html The Trojan Room Coffee Machine
- http://www.menet.umn.edu/coffeecam/ CoffeeCAM

The list of links in this chapter, often becomes outdated, therefore you might wish to use the excellent Way−Back Machine to find them again: http://www.archive.org/