Halitosis, a universal emc machine configuration wizard?

Very short readme:

What?:
A gui wizard to configure emc hal for any generalised serial cnc machine: that is it guides the user to enter required parameters for a machine (joint geometry, joint drive control, joint sensors) and creates a complete hal configuration along with virtual control panel and gui simulation.
This is my attempt to create a universal emc configuration wizard, it will not be a successful attempt as i dont have time to make it so, but it may be a good start point for someone else to continue to improve the idea??

Why?:
I got three robot arms on ebay for 85 quid and needed to control them (none came with the controllers), i decided to use emc and some l293 chips to drive the motors via the parallel port.
It seemed like a good idea to create something to
help me generate the emc kinematics and hal config files (this was of course a fucking stupid idea as i guess writing config files would have taken 4 hours, whereas i have wasted probably 40 hours on this gui system)

When?:
I got the robot arms in July 2010, this readme was started august 25th 2010, i guess the code will be released within a week..

How?:
Write some python code, read the emc docs, ask some questions on the #emc irc channel, solder up some electronic circuits.

Current state of development

I am not sure i should be releasing this code to the public as it may reflect badly on my skills at programming. That said this is really only a hack project and almost every aspect of it has been learnt from scratch over the last month and a half in my spare time, so I tell myself people will understand the totally nuts code in some sections of program.

Its really just a sketch and proof of concept, and many ideas have been started then discarded yet remnants are still left in the code.

The code has got to a stage where it pretty much does what i need which is configure etchservo type drives and quadrature type stepper drives.

It all really needs a refactor, but this being open source i doubt you will get any improvement unless someone else picks up this project or someone sponsors its development or sends me some nice free hardware that i have to integrate with the code.

Download

Releases should be named in the format: halitosis-yyyy-mm-dd.tgz

though i get the feeling this may be the only release?

The latest version will always be linked to as halitosis_latest.tgz

Todo

There is so much that i should have done but have not, years ago i would probably have done this in half the time and twice as well, but i seem unable to concentrate in my old age.

Basically the code is a hack and a learning experience for python and robotics, both of which i have never studied or used before, so please forgive errors.

A quick list:

- buttons to test each joint drive (jog forward, jog reverse, move x text input)
- Fix the dhkins.c kinmatics driver
- fix the machine simulation creation code
- make all the code more object orientated such as servo.py and stepper.py should be subclasses of a general paramentry type at present stepper was originally coded then servo copy pasted modified to base class and servo subclass...
- simplify the code, it way to log and confusing at the moment.
- refactor and rethink everything now i know more
- fix all the bugs
- fix the callbacks
- fix the sensors stuff
- implement testing of drivers from within the program (use halrun?)
- seperate out all the text and lists to different files (keep code seperate from data) e.g. code: stepper.py, data stepper_data.py
- possibly seperate out the config writing function to different source files from the gui handling functions?
- bla bla bla....

The code purports to do more than it really does, such as creation of the simulated machine is totally brain dead, most of the stepgen types are not implemented, etc...

If anyone fixes these please send in some patches.

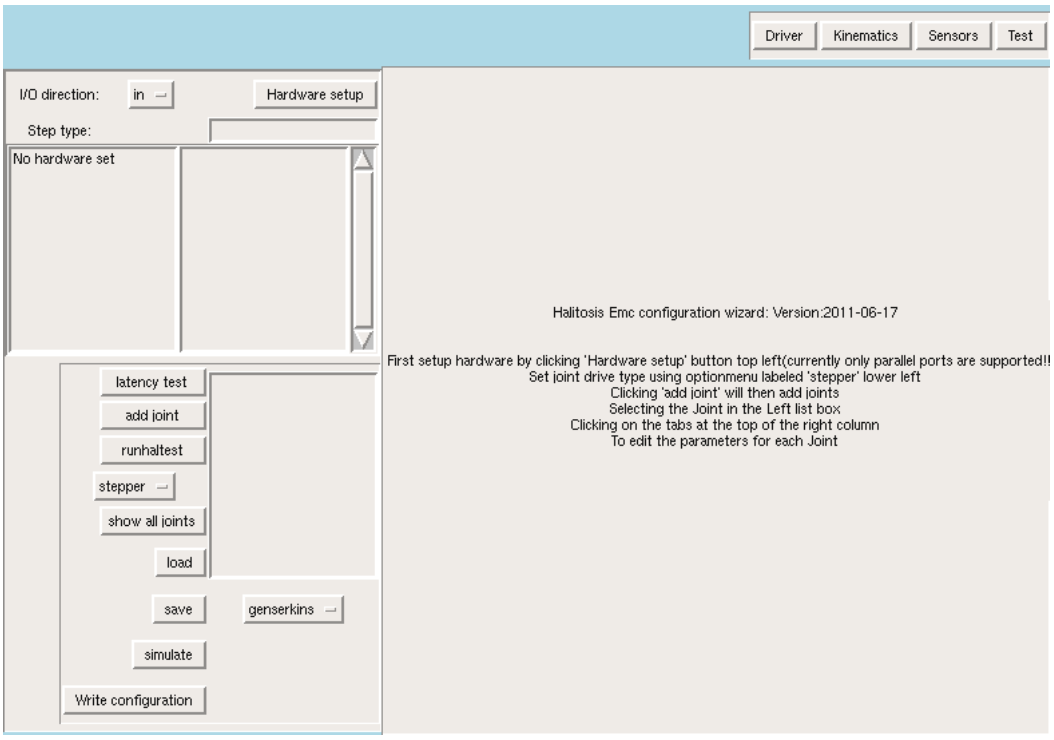
Instructions

basic instructions are:

run "halitosis"

click buttons and configure things.....

More detailed instructions:



Halitosis Emc configuration wizard: Version:2011-06-17

First setup hardware by clicking 'Hardware setup' button top left(currently only parallel ports are supported!!
Set joint drive type using optionmenu labeled 'stepper' lower left
Clicking 'add joint' will then add joints
Selecting the Joint in the Left list box
Clicking on the tabs at the top of the right column
To edit the parameters for each Joint

the first screen shot shows the general layout:
top right is the per joint configuration type options:
clicking on these buttons will change the information (driver, kinematic, sensor) in the window pane below.

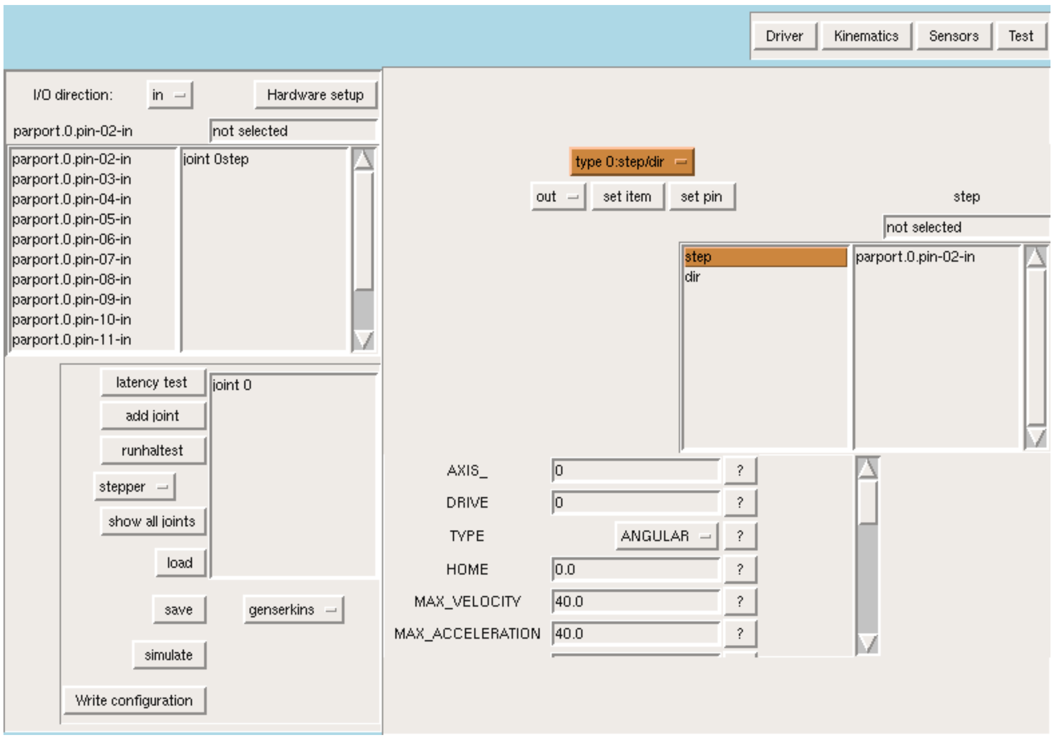
The left side of the window is the hardware input/output listbox the option menu can be set to in or out to show the pins available and their settings. The "Hardware" button selects the hardware driver configuration and clicking on this allows configuration in the right hand window pane. Once Hardware is setup the listbox will display all available pins for IO.

The Pane below the Hardware IO list box is the main joint creation and selection area. A new joint can be created by clicking the "add joint" button, but the type of drive needs to be selected first by selecting "stepper" or "servo". Once a joint is created it will be listed and can be selected in the listbox pane on the right of the buttons.

Once all joints have been created and configured the "complete" button should be clicked to save all the emc hal configurations and the program will then exit.

the hal configuration will be written to files named "test"

Drive:
each joint has a drive (servo or stepper motors) that requires hardware in/out configuration normally this is just output pins on the parallel port



I/O direction: in

parport.0.pin-02-in: not selected

```
parport.0.pin-02-in    joint 0step
parport.0.pin-03-in
parport.0.pin-04-in
parport.0.pin-05-in
parport.0.pin-06-in
parport.0.pin-07-in
parport.0.pin-08-in
parport.0.pin-09-in
parport.0.pin-10-in
parport.0.pin-11-in
```

joint 0

stepper

genserkins

type 0:step/dir

out

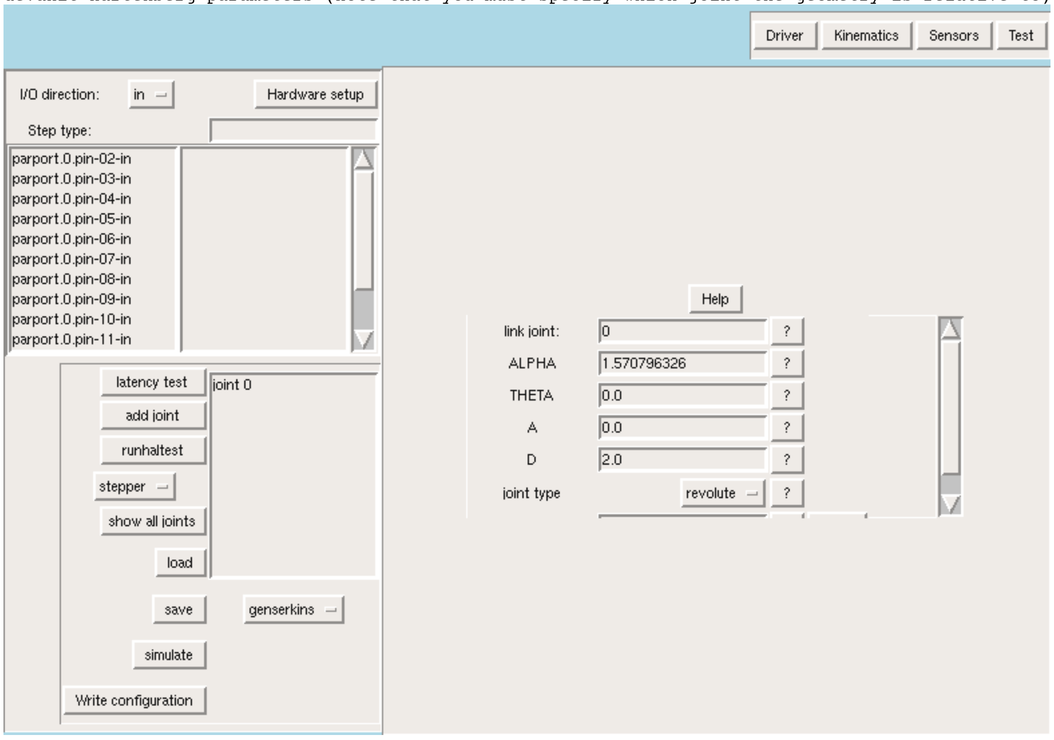
step: not selected

```
step    parport.0.pin-02-in
dir
```

AXIS_	0
DRIVE	0
TYPE	ANGULAR
HOME	0.0
MAX_VELOCITY	40.0
MAX_ACCELERATION	40.0

Kinematics:

each joint has geometry ralative to another joint on the machine this geometry is defined using devanit hartenberg parameters (note that you must specify which joint the geometry is relative to)



I/O direction: in

Step type:

```
parport.0.pin-02-in
parport.0.pin-03-in
parport.0.pin-04-in
parport.0.pin-05-in
parport.0.pin-06-in
parport.0.pin-07-in
parport.0.pin-08-in
parport.0.pin-09-in
parport.0.pin-10-in
parport.0.pin-11-in
```

joint 0

stepper

genserkins

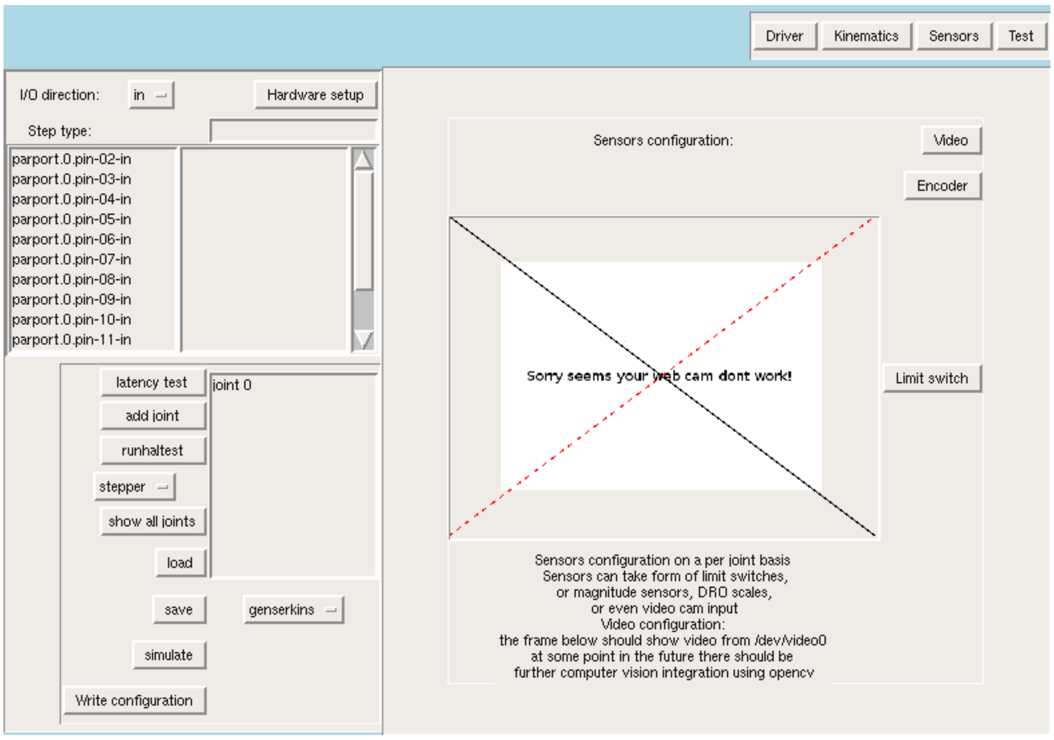
link joint:	0
ALPHA	1.570796326
THETA	0.0
A	0.0
D	2.0
joint type	revolute

Sensors:

each joint has sensors linked to it these may be

- position encoders,
- limit switches,
- or other sensors like video camera sensors

(I have not really bothered with looking at this as i dont have time, but the video webcam view should work).
I have left it in place for other to experiment with the use of opencv (open computer vision)



Sensors configuration on a per joint basis
Sensors can take form of limit switches,
or magnitude sensors, DRO scales,
or even video cam input
Video configuration:
the frame below should show video from /dev/video0
at some point in the future there should be
further computer vision integration using opencv

I have also started messing with the genserkins driver to allow correct and full dh parameter configuration and motion controlled by emc. Of course that is totally unfinished as i did not really need it for the robots i was controlling.

Distribution and support

Basically this is an unfinished half assed piece of shit, but it did what i wanted and you are getting to use it for free so you should be grateful, your job to repay me for my generosity is to improve what i have left you or give something else of use to the world.

I hope this software will help someone else, but i really would prefer that others dont waste my time asking for help, you are welcome to try and get help if you see me on irc but please dont email unless you have bug fixes or other useful additions.

Please ignore spelling and grammatical errors as english is not my first language.

Development overview:

please ignore most of this rambling bullshit!

- Its written in python (because thats what most other emc tools are written in) which i had never used before starting this project

- It is about 30 hours programming and 10 hours thought and research so far. all development was done in the standard emc ubuntu8.04 live distro installed to a vmware virtual machine.

- The concept is that the wizard shows lists of connections required and only allows the user to connect pins correctly.

- it configures the machine on a joint by joint basis, in terms of kinematics, driver, and geometry and any number of joints can be added

- it automatically creates all relevant emc configuration (virtual controls, simulated machine, machine kinematics, servo or stepper hardware controls)

- it should be extensible for new control types as i tried to create parameter entry screens that take lists of parameternames and value ranges all configuration parameters are defined as lists.

Slightly more in depth overview of the extent of code development:

The idea was to create a complete universal emc configuration wizard, unfortunatly i dont have time to complete the idea, currently I have only completed the parts relevant to the machines i am working on.

• the servo config is really only setup for the etch servo type configuration, and includes encoder input whcih should be moved to the sensors code.
• the stepper config is really only setup for type 2 quadrature stepper drives
• the sensors config is not really setup at all, it is really only a skeleton for future work
• the hardware config is really only setup for parallel configuration
• the kinematics works 'almost' but the simulation code creation is totally brain dead

source code

the code is divided into a number of different files:

- halitosis   (the main program loads all the parts and makes the magic happen)
- hardware.py   (defines the gui that configures the computer io hardware pins and writes emc configs)
- servo.py   (defines the parameters for a servo drive)
- stepper.py   (defines the parameters for a stepper drive)
- sensors.py   (defines the parameters for sensors (video, limit switches, encoders))
- paramentry.py (base class for creating gui parameter entry pages other classes derive from it)
- EmcGuiprocs.py (defines some general gui procedures that are used in the program)
- kinematics.py   (defines the kinematics that are used in the genserkins defs and also creates a vismach machine simulation model
- dhkins.c   (this is totally unfinished and really unstarted the idea was to genralise genserkins for revolute and prismatic joints adding the theta variable)
- software_install.sh   (this is just a script that installs some software that i thought would be interesting and relevant to e