FORTHtalk

WARNING: Under development and not fully tested. Not all exceptions are captured so may crash unexpectedly.

A Python shell for communicating with AVR (e.g Arduino) Forth based systems via serial communications. Primarily developed to work with **flashforth** (Mikael Nordman) but some features should work with other AVR Forth's over serial communications with a few tweaks. Developed on Linux, so may need modifying for other OS's and only used so far with Arduino (ATmega328P).

Acknowledgements: Inspired by amforth-shell.py (Keith Amidon) and ff-shell.py (Mikael Nordman) but pretty much written from the ground up.

Brief list of commands:

#send	Uploads a file to the Forth system
#include	Same as #send
#require	Same as #send
#comp	Compiles a list of required files and sends them to the Forth system
#file	Analyses a file for words that need other files to be uploaded
#defs	Searches the pathList for files that have definitions
#path	Adds a path to the pathList
#warm	Initiates a warm start. Same as sending 'warm' directly to the Forth system
#empty	Sends 'empty' to **flashforth** and removes user defined words from definedWords
#list	Shorthand for '#words list'
#words	Default is to send 'words' to the Forth system and save these in definedWords
#find	Find a word or words in the definedWords list
#last	Copies of last lines received from the Forth system
#stats	Prints out free memory statistics after interrogating the Forth system
##	Exits the program (Keyboard command only)

Configuration of the serial port and speed is currently made by modifying the forthtalk.py file. The program looks for a file called config.ftk in the current working directory and can load any intial forthtalk commands from there as it starts up, and/or send Forth code to the Forth system. A sample config.ftk file is provided.

The preferred extension for Forth files is '.frt' which is appended to file requests if no extension is specified.

forthtalk commands start with an '#'. Commands can be entered directly from the keyboard or can be embedded in files. The '#' must be the first character on the line, or the third character following a backslash (i.e. in a comment line). E.g.
#send filename
\ #send filename
would result in the #send command being executed. Valid commands are stripped out by forthtalk and not sent to the serial port. It is a matter of preference whether commands are included in comment lines. Obviously if another program such as minicom or similar is sending a file to a Forth system, then putting the command after a backslash would ensure they are ignored.

Keyboard entered lines for flashforth

forthtalk pre-processes lines to substitute ATmega register names with literals. Only one file is included here but the file (slighly modified) is pulled from the amforth project device files. For devices other than ATmega328p look for amforth repositiories (/avr8/devices/your_device name) and extract the relevant python file. This needs to be specified in place of 'device328p' in the import list at the top of the forthtalk.py file.

forthtalk also converts upper case hex literals to lower case. **flashforth** only accepts lower case hex, but many forth files use upper case. IMHO upper case hex in files looks better!

The command list is table driven and can be easily extended or modified.

Commands

#send filename (synonyms: **#include**, **#require**) Sends the specified file to the Forth system. If an extension is specified it will be used, but if no extension is specified '.frt' will be appended to the filename. If a path is specified then the path is used. If no path is specified then the file will be searched for in the current working directory and then in the paths list (see #path). The first matching file found is sent. Paths are searched in the order they are added to the path list.

Each line of the file is processed for '#' commands which are executed; surplus whitespace & comments which are stripped; register names (see device328p.py) which are substituted by literals, and upper case hex literals which are converted to lower case.

#path [pathname] Adds a path to the path list. No check is made on whether this is a valid path. If no path is provided it prints the current path list.

#stats Sends a simple set of forth commands and prints the free ram, eeprom & flash e.g.
Free ram : 1535 bytes
Free eeprom : 1011 bytes
Free flash : 21643 bytes

#words [get/list/user/alpha] Only the first letter of a parameter is actually required e.g. #words a = #words alpha
#words with no parameters, sends 'words' to the Forth system and saves the received words in a Python list: 'definedWords'.
#words get As above, but also prints the Python list 'definedWords'
#words list Just prints the current list of words stored in 'definedWords' without updating it.
#words user Prints the user defined words after 'marker'. (flashforth has a core set of words and user defined words)
#words alpha Prints the stored list in alphabetical order
#list Shorthand for' #words list'

#find *word* Searches the Python list 'definedWords' for the the 'word' and reports 'Found' or 'Not found'

#defs Searches the paths in the path list (see #path) for files with a '.frt' extension. It then analyses the files for forth code which defines new words. This is assumed to be words following a word which terminates in a colon. e.g. ':', 'portpin:', 'as:', etc. The new words are stored in a dictionary

(wordFiles) with the name of the file they were found in. It then prints out a list of the words found.

#file *filename* Analyses a file and tries to determine if all the words used to define a new word are either in the definedWords list or the wordFiles list. If all the required words are already defined then the 'compileFiles' list will only contain the specified filename. If however, some of the words are not currently defined but found in the wordFiles list, the correcponding files are added to the 'compileFiles' list which are recusively searched for other undefined words. The compileFiles list is then printed. Nothing is sent to the Forth system. E.g.
#file star-slash
--- Analysing file .../Forth/forthwords/star-slash.frt ---
--- Analysing file .../Forth/forthwords/star-slash-mod.frt ---
--- Analysing file .../Tech/Forth/forthwords/m-star.frt ---
--- Analysing file .../Forth/forthwords/sm-slash-rem.frt ---
Compile files: ['star-slash.frt', 'star-slash-mod.frt', 'm-star.frt', 'sm-slash-rem.frt']

#comp *filename* As #file except that the files in compileFiles are also sent to the Forth system in reverse order, i.e. from right to left as they appear in the list.
NOTE: This is an alternative approach to including '#send filename' within a file. E.g. m-star.frt could include a line '#send sm-slash-rem' which would pause the sending of m-star.frt while sm-slash-rem.frt is sent. flashforth ignores words which are already defined so there is no harm including such commands.

#warm (flashforth specific) Sends CTRL-O to the **flashforth** system which is the same as sending 'warm'.

#empty (flashforth specific) Sends 'empty' to the flashforth system and deletes all words in the Python 'defnedWords' list back to 'marker', just leaving the core words in the list.

#last Primarily for debugging, it prints the last (default=10) lines received from the Forth system. These lines are stored in the Python list 'lastLines' which is also used by other commands to process responses from the system. E.g. commands such as '#stats' and '#words' analyse text in the lastLines list.