CREOLE FORTH FOR EXCEL

Joseph M. O'Connor

SVFIG Forth Day

November 2016

SOME HISTORY

- Creole Forth started out as a Delphi component
- Inspired by Norman Smith's UNTIL
- Used successfully as a macro/scripting language in many projects.
- Because Delphi is not very popular, management was never very happy with its use.
- This inspired me to consider transferring the language to a more widely used platform.

MICROSOFT EXCEL

- An indispensable, and probably the most popular Microsoft Office product.
- Has a built-in macro language (VBA).
- Available ‘everywhere’, at least in a corporate environment.
- The layout of an Excel worksheet is preadapted for the Forth dictionary.
- Has great similarities to the Delphi drag-and-drop visual development environment.

WHY A LANGUAGE BUILT ON TOP OF VBA

- Excel already has a powerful built-in language
- So the question is, why build another on top of it?
- Here are a few reasons:
- 1. To combine a systems and a scripting language.
- 2. The Forth methodology of writing an application-specific instruction set is exceptionally productive.
- 3. Painless parameter passing via a stack or stacks.
- 4. Easy extensibility.

CREOLE FORTH FOR EXCEL – ITS TRACK RECORD.

- Initial version was released in early August 2016.
- In that time I have created three applications with it:
- 1. Optum Track Tools. This is a quasi client/server app that would interact with Cygwin and perform various administration tasks on a Linux server.
- 2. An XML import/export organizer. This was developed using the MVC (Model-View-Controller) pattern and helps manage the 2-way transfer of records from a Windows flashcard application (Supermemo) to one on the Android (Anymemo).
- 3. Issue Log Organizer. Sets up a Table of Contents and a Master Issues Log interface with buttons and hyperlinks. These buttons then import data from the linked spreadsheets.

IMPLEMENTATION DETAILS

- At the GitHub repository is a spreadsheet CFExcel1.xls.
- Compatible with Excel 2003.
- Opens to a workbook with three tabs:
- 1. CreoleForthInitPage
- 2. GlobalDS
- 3. Dictionary

CREOLE FORTH INIT PAGE

- Holds information about different Forth “bundles”.
- A bundle is defined as a combination of the VBA code, GlobalDS page, and Dictionary page.
- In the current version, there are 4 possible Forth bundles at a maximum.
- Only one bundle can be active at a time.
- There is currently one Forth bundle in the project, and in most cases you would probably not want or need more.

THE GLOBALDS PAGE

- Holds the major external data structures.
- Has five “stacks” and other associated data structures such as the InputArea and PAD.
- You put code in the InputArea box and hit the ‘Submit’ button and it will execute the code.

THE DICTIONARY PAGE

- Holds the dictionary and associated fields.
- Data structures are mostly column ranges.
- Parameter Field is essentially everything to the right of those fields, although it gets a bit more complicated than that.

EXECUTION OF CREOLE FORTH

- The values in the input area are split based on the space delimiter and placed into the ParsedInputField.
- Each value is looked up in the dictionary based on the list of vocabularies on the Vocabulary stack.
- In this case FORTH is searched first, and if the search fails, ONLY is searched.
- If a match is found, the word is executed.
- If no match is found in any of the vocabularies on the vocabulary stack, the value is pushed onto the data stack.

TYPES OF WORDS

- Simple Primitives. These point directly to public methods in VBA class modules in the project.
- Colon definitions. These are words compiled with : (colon). Addresses are compiled into the parameter field with CompileColon and executed with DoColon.
- Defining words. Defined with CREATE or CREATE/DOES> combination.
- Compiling words. Have separate primitives for compile-time and run-time and are used for branching/looping.

SOME DIFFERENCES FROM "NORMAL" FORTHS

- No attempt is made to convert a value to an integer before placing on the stack. If a plain string is found, it goes on there.
- Strings can also be compiled into definitions as literals.
- Some words were renamed because the canonical Forth names caused bad reactions with Excel.

UNDER THE HOOD

- The methods of the primitives all take a single parameter, which is a GlobalSimpleProps object.
- The GlobalSimpleProps object is a list of properties that pass information between themselves and the GlobalDS, Dictionary, and CreoleForthInitPage pages.
- Most of the work is done through named ranges.

UNDER THE HOOD 2

- When a word is looked up in the dictionary, it is looked up by its fully qualified name.
- The fully qualified name consists of what's in its Name Field, a period, and the vocabulary it was defined in.

- When words are resolved down to the primitive level, they're executed by the CallByName module as below.
- CallByName objClassModule, sCodeField, VbMethod, poGSP
- poGSP is the single GlobalSimpleProps object being passed as a parameter.

UNDER THE HOOD 3

COMPILATION

- Creole Forth has no state variable.
- Compilation starts when the IMMEDIATE vocabulary is pushed onto the stack.
- IMMEDIATE words are first in the search order.
- High-level definitions are built in PAD.
- Each word has its index looked up in the dictionary, and its associated compile action placed next to it.
- A “Smudge Flag” prevents accidental recursion.
- After this is done, the results are fed back into the interpreter.
- A word that has a COMININPF action will be compiled into the parameter field.
- Words with an EXECUTE action (compiling words) will be executed.
- Literals are tagged with a COMPLIT action and are treated accordingly.
- Compilation terminates with the execution of DoSemi.
- The colon compiler is able to compile help into its definition. The commenting policy is to put the stack comment first, followed by the single-line comment just before the definition.

CONTROL STRUCTURES

- IF-THEN-ELSE
- BEGIN-UNTIL.
- DO-LOOP and relatives are still under development.

ORGANIZATION METHODOLOGY

- Try not to mix the “base code” already in the project with application-specific code.
- Add a new “Main” form to call new subroutines.
- Define a new class module or set of class modules for new primitives.
- FORGET is available, but CTRL-SHIFT-R and CTRL-SHIFT-C offer an easier alternative. (blow away and rebuild dictionary).
- A typical application might acquire 30-40 primitives and 10-20 high-level definitions.

TO DO

- “Redefined” functionality in high-level definitions.
- More control structures.
- Better/more consistent commenting.
- List compiler to allow multiple arguments in a single cell on the stack.
- VOCABULARY defining words – vocabulary words right now are all hard-coded primitives.
- Prefilter/Postfilter stack support.
- POSTPONE.
- Recursion.

QUESTIONS?