C Macros for Ardurino eForth

Forth Day - November 19, 2011

I will present ANSI C macros that can be used to create a Forth dictionary running eForth on the Arduino. The Arduino development system is based on GCC.

Introduction

- Ting's eForth for the Arduino is written in C
- Creating the Forth dictionary in a cogent manner can be difficult
- The C language has a programming facility called macros
- This facility can be used to create the various fields (link, name, code, and parameter) of a Forth word

eForth Word Fields

- Link Field - Used for searching the dictionary for a specific Forth word.
- Name Field - The name of the Forth word. It is a counted string. The first byte is the length of the string and the following bytes are the name.
- Code Field - The “code” that will execute the following parameter fields.
- Parameter Fields - The Forth words or primitives to be executed by the previous code field.

Link Field

The link field is an address of the previously defined Forth word. The exception is the first word in the dictionary. It is set to NULL, indicating the start of the dictionary.

Name Field

The name field identifies the Forth word. It is a counted string, the first byte is the length and the rest of the bytes are the name. For the Audrino architecture, a 16-bit word has to be aligned on an even address. If the name field has an even length, the total name field length will be odd. To correct this an ASCII NULL is appended to the name field.

Code Field

- This field contains the address of the code to execute the following parameter fields.
- Note: Because the Arduino architecture requires 16-bit accesses to be aligned to an even address and the primitives are byte in length, the primitives have to be padded with an ASCII NULL.

Parameter Field

This field contains the address of previously defined Forth words or primitive token.

Link Field C Macros

- The link field macros writes the pointer to the previous defined word's initial code field, CFA.
- LINK_FIELD - Write the pointer to the previous defined Forth word.
- END_WORD - Update the link pointer.

Name Field C Macros

- The name field macros write the name of the following word.
- NAME_FIELD(name) - Write the name as a counted string and, if necessary, pad it to align on a 16-bit boundary.
- FORTH_WORD(name) - Writes the link field and the name field.
- COLON_WORD(name) - Writes the Forth word and the dolist primitive.

Code Field C Macro

- There are a group of code field macros that write 16-bit values to where the current dictionary points to and increments the pointer by two.
- CODE_FIELD(name) - Writes the address of the named code field.

Parameter Field C Macros

- These macros write 16-bit values to the next parameter field.
- PARM_FIELD(name)
- PRIM_FIELD(prim)
- WORD_FIELD(value)

BEGIN - UNTIL C Macro Pair

The BEGIN - UNTIL macro pair make a finite loop. The loop continues if the top of the stack is zero.

BEGIN - AGAIN C Macro Pair

The BEGIN - AGAIN macro pair make a infinite loop.

IF - THEN C Macro Pair

The IF - THEN macro pair make code that is conditionally executed.

IF - ELSE - THEN C Macros

The IF - ELSE - THEN macros make code that is not only conditionally executed but has an alternative part that get executed if the condition is false.

BEGIN - WHILE - REPEAT Macros

The BEGIN - WHILE - REPEAT macros make a finite loop based on the condition just before the WHILE.

FOR - NEXT C Macros

The FOR - NEXT macro make a counted loop based on the top of stack value just before the FOR.

AFT - THEN C Macro Pair

The AFT - THEN macro pair are used inside of a FOR - NEXT pair.

dictionary.c

- #include "dictionary.h"
- CODEword flashDict[] = {
- /* lfa = 0x0000 */ { .s = 0x0000 },
- /* nfa: tmp (0x0002) */
- { .c =   3 }, { .c = 't' }, { .c = 'm' }, { .c = 'p' },
- /* cfa: 0x0006 */
- { /* prim 0x0006: */ .c = 0x04 },
- { /* pad 0x0007 */ .c = '\x00' },

dictionary.c (cont'd)

- { /* prim 0x0008: */ .c = 0x16 },
- { /* pad 0x0009 */ .c = '\x00' },
- { /* 0x000a: */ .s = 0x0100 },

dictionary.c (cont'd)

- /* lfa = 0x0230 */ { .s = 0x021e },
- /* nfa: ROT (0x0232) */
- { .c =   3 }, { .c = 'R' }, { .c = 'O' }, { .c = 'T' },
- /* cfa: 0x0236 */
- { /* prim 0x0236: */ .c = 0x06 },
- { /* pad 0x0237 */ .c = '\x00' },
- { /* code 0x0238: */ .s = _more_R_CFA },
- { /* code 0x023a: */ .s = SWAP_CFA },

dictionary.c (cont'd)

- { /* code 0x023c: */ .s = R_more__CFA },
- { /* code 0x023e: */ .s = SWAP_CFA },
- { /* code 0x0240: */ .s = EXIT_CFA },

dictionary.h

- typedef
- union {
- unsigned char c[2048];
- unsigned short s[1024];
- } CODEword;
- #define tmp_CFA 0x0006
- #define ROT_CFA 0x0236