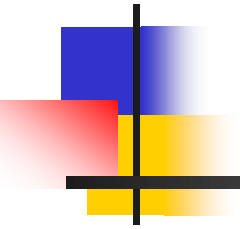
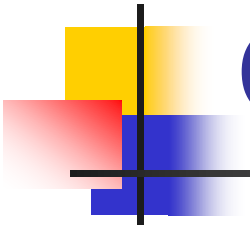


Arduino eForth

Silicon Valley Forth Interest Group

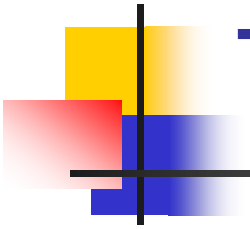
September 24, 2011

Chen-Hanson Ting



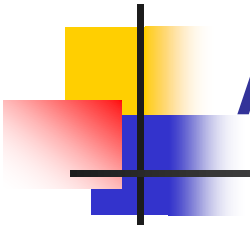
Goals

- A simple Forth system which can be loaded as a Arduino Sketch.
- Must be written in C (Arduino Process) and does not require separated AVR programmer.
- Attractive to beginning Arduino users.



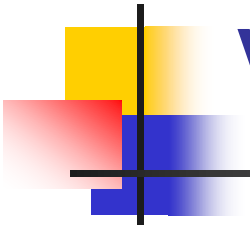
The Fundamental Problem

- C does not have the concept of memory and addresses.
- C does not allow writing into code space in flash memory.
- Arduino bootloader does not support custom flash write operations.



Approach

- Port C-eForth 11 to Arduino.
- Make sure that the Forth interpreter works.
- Add PEEK and POKE so that it is at least somewhat useful.



Virtual Forth Machine

- Virtual Forth Machine with 33 byte code as tokens.
- A token threaded Forth dictionary built by an eForth metacompiler.
- A simple finite state machine runs the Virtual Forth Machine.

Virtual Forth Machine

```
void next(void) { P =
   pgm_read_word(&code[IP++]); jump(); }
void docon(void) { push
   pgm_read_word(&code[P]); P += 1; }
void dolit(void) { push
   pgm_read_word(&code[IP]); IP += 1; next(); }
void dolist(void) { *++R = IP; IP = P; next(); }
void exitt(void) { IP = *R--; next(); }
void execu(void) { *++R = IP; P = top; pop;
   jump(); }
```

Virtual Forth Machine

```
void execu(void) { *++R = IP; P = top; pop; jump(); }
void donext(void)
{ if(*R) { *R -= 1 ; IP = pgm_read_word(&code[IP]); }
   else { IP += 1; R-- ; } next(); }
void qbran(void) { if(top == 0) IP =
    pgm_read_word(&code[IP]);
   else IP += 1;  pop; next(); }
void bran(void) { IP = pgm_read_word(&code[IP]);
    next(); }
void store(void) { data[top>>1] = *S--; pop; }
void at(void) { top = data[top>>1]; }
void cstore(void) { cData[top] = (char) *S--; pop; }
void cat(void) { top = (int) cData[top]; }
void icat(void) { top = (int) gm_read_byte(&cCode[top]); }
void iat(void) { top = pgm_read_word(&code[top]); }
```

Virtual Forth Machine

```
void istore(void) { pop; pop; }
void icstore(void) { pop; pop; }
void rfrom(void) { push *R--; }
void rat(void) { push *R; }
void tor(void) { *++R = top; pop; }
void drop(void) { pop; }
void dup(void) { *++S = top; }
void swap(void) { w = top; top = *S; *S = w; }
void over(void) { push S[-1]; }
void zless(void) { top = (top & 0X8000) LOGICAL ; }
void andd(void) { top &= *S--; }
void orr(void) { top |= *S--; }
void xorr(void) { top ^= *S--; }
void uplus(void) { *S += top; top = LOWER(*S, top) ; }
void nop(void) { jump(); }
void dovar(void) { push P; }
```

Finite State Machine

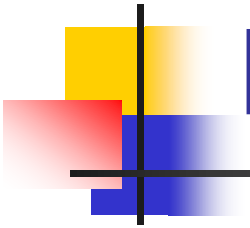
```
void setup()
{
        clock = 0;
        P = 0;
        IP = 0;
        S = stack;
        R = rack;
        top = 0;
        phase = 0;
}
```

Finite State Machine

```
void loop()
{
        phase = clock & 3;
          switch(phase) {
                case 0: fetch_decode(); break;
                case 1: execute(I1); break;
                case 2: execute(I2); break;
                case 3: jump(); break;
            }
          clock += 1;
}
```

Finite State Machine

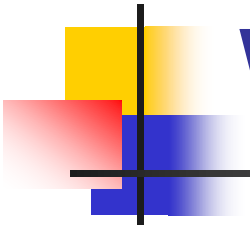
```
void execute(unsigned char icode)
{   if(icode < 33) {
    primitives[icode](); }
   else { Serial.print ("Illegal code=");
    Serial.print(icode, HEX) ;} }

void (*primitives[64])(void) = {
    /* case 0 */ nop,
    /* case 1 */ bye,
    /* case 2 */ qrx,
    /* case 3 */ txsto,
    /* case 4 */ docon,
  …
     /* case 32 */ icat };
```



Forth Dictionary

- It is constructed outside the C compiler by an eForth metacompiler, adopted from eP32 project.
- It is saved and imported to Arduino as a 16-bit data array.
- It can be accessed as a byte array aliased to the word data array.

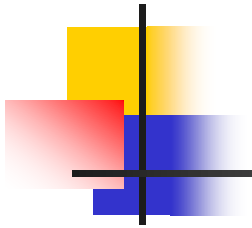
What is it good for?



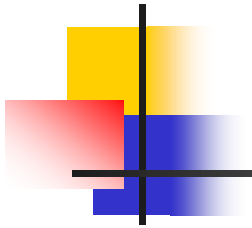
- In the Arduino community, there seems to be a real need for PEEK and POKE to examine I/O registers
- I added these Forth commands:

```
: PEEK C@ ;
: POKE C! ;
```

- If you need a real Forth system on Arduino, get 328eForth system.



Questions?



Thank You.