# C based eForth on Top of RTOS

SVFIG Meeting on December 19, 2020

Presented by

Masa Kasahara

# Past Projects :

1) Audible Computing: To create an efficient Computer Language based on our natural languages

2) Forth CPU Design: To create a simple, yet usable CPU in FPGA

3) Forth Implementation in C: To utilize the Forth productivity in a simple manner

# Audible Computing:

1) I originally thought of using Morse Code as User Interface since I wanted to create a Morse Code Trainer. If I should create a Morse Code Trainer, why not make something a bit more useful? In Morse code transmission, short and condense expression is used, which seemed to fit Forth vocabulary.

2) I will talk about it in a separate presentation.

# FORTH CPU Design:

1) I studied Dr. Ting's eForth CPU design in FPGA he presented in the past. I would like to implement his new circular stack design, add some real-time related features and digital signal processing capabilities.

2) I will talk about it in a separate presentation.

# FORTH Implementation in C:

1) I studied Dr. Ting's eForth Implementations in C he presented in the past. I would like to use his implementations on top of RTOS (Real Time Operating System).

2) I will talk about the current status in this meeting.

# What is the benefit of FORTH Implementations in C:

1) Brad did a wonderful presentation about it on the Forth Day this year.

2) I think there is another potential application. When you want to add Forth capabilities on an existing system written in C, this makes it easier. In fact, I might need to do it in the near future.

# Logomatic v2 Serial SD Datalogger:

1) I found an abandoned single board computer: https://www.sparkfun.com/products/retired/10216

# Logomatic v2 Serial SD Datalogger:

2) It has a very attractive NXP LPC2148 CPU.
3) It can run FreeRTOS.
4) It has a GNU based C tool chain.
5) It has an interesting USB thumb drive/firmware upload feature.
6) The above feature somewhat derailed the project.

# FreeRTOS Simlulator on Windows:

1) I found a simulator and training manual with extensive examples.

2) It can be used as a test bed.

# ceForth_33.cpp:

1) Dr. Ting's C Implementation uses C extensions under C++.
2) However, his code is 99.9% C.  So, I made an ANSI C version out of it and named it eForth_33-ansi.c.
3) When I tried to integrate it into FreeRTOS, Visual Studio complained.
4) A long story short, I used his ceForth_23.cpp.

# ceForth_23.cpp:

1) Dr. Ting's C Implementation uses C extensions under C++.
2) However, his code is 99.9% C.  So, I made an ANSI C version out of it and named it eForth_23-ansi.c.
3) It worked beautifully.
4) However, Forth Meta compiler is written in eForth.

# ceForth_33.cpp:

1) I learned that the program could be separated into two, namely eForth VM and eFORTH Meta Compiler to create eForth dictionary.
2) So, I stripped eForth VM and created just eForth Meta Compiler and named it cefMETA_23.c.
3) The reason I called it cefMETA_23.c is that the dictionary size is the same as ceForth_23.c.

# Demonstration:

1) Dr. Ting’s ceForth_23 running on top of FreeRTOS Simulator.

# Q & A: