A QUICK TUTORIAL for FISH Forth v1.7
for the ARM Systems on a Chip (SoC's)

by Clyde W. Phillips Jr. and A-TEAM FORTH.

This document details **FISH** specific chips and board(s) supported by **FISH**.

The tools used to install **FISH**, and supported terminals/settings are outlined here.

FISH is simple Forth programming language system (interpreter and compiler).

FISH is a Flash resident Forth Application Development System (ADS) for the ARM Cortex M-series micro-controller Systems on a Chip (SoC). The **FISH** ADS extends Forth by allowing applications to start when your system is turned on or reset.

A terminal program and text files are used for software development in **FISH,** via RS-232 or USB Serial. At power-up **FISH** prints out its status greeting and default prompt.

FISH is ideal for learning *Agile* and *Object Oriented Software Development,* the best disciplines in the industry. **FISH** is an ideal MAKER production tool.

The accompanying **FISH** glossary has examples of using many **FISH** words. This document walks you thru major concepts in **FISH** by example.

FISH is really forgiving; being resident in Flash you cannot hurt it! Be bold and experiment; have fun!

FISH Defaults:

Power Up Reset

FISH issues a SIGNON message when powered up. The message reflects the status of Flash pages used or available and the system model, version and date/time created. Code saved to Flash will be linked into the kernel word set. The SIGNON message and flash page status can be viewed at any time by typing the word **FISH**

Enter CR and expect the "OK" prompt with current **BASE** printed in decimal, EXCEPT in the Small Reference Model, which sacrifices a few words to provide more **FLASH_SAVE** space on the smallest targets.

See **RUN** (below) as the exception where **FISH** will boot into your application. Your application can reconfigure the whole system; clocks, UART etc if need be!

SERIAL UART CONFIGURATION:

At power-up **FISH** communicates via UART protocol at 9600 bps, 8n1, using Xon/Xoff software flow control. **MYBAUD** is used to specify a new baud rate. Depending on the target **UART?_INIT** changes the baud rate, where the **?** Is the UART number used. Type **WORDS** to see which UART your target uses.

```
115200d MYBAUD UART0_INIT       \ Comment NXP target us UART0
115200d MYBAUD UART3_INIT       \ Comment STM target us UART3
```

changes the baud rate to 115,200. Now the new baud rate will be preserved during resets, BUT NOT POWERDOWN.

FISH system words are all capitalized:

FISH word names are case-sensitive, allowing you to define words in any mix of ASCII cases. All pre-defined **FISH** words are upper case ASCII, in *BOLD TYPE* thru out this document. Hexadecimal numbers are upper case *A thru F*. The only lower-case digits in **FISH** are the number **BASE** suffixes; b, d and h.

FISH organizes its words in Word Categories. A word category facility called **WORDCAT** is provided for users to do the same. Type **WORDS**

NUMBERS, BASE and the BASE SUFFIX:

Numeric **BASE** is a central concept in **FISH**.

DECIMAL, **HEX** and **BIN** are three standard words that change the numeric **BASE** in **FISH**.
Press the Enter key and **FISH** will respond with:

*ok, go fish in BASE xxd <=xx is the current value of **BASE** in* **DECIMAL**.

Type **BIN (**which sets **BASE** to Binary) then press Enter:

BIN ok, go fish in BASE 2d

In Binary the only valid numbers are 1 and 0.

Do the same with **HEX** (for Hexadecimal) and **DECIMAL**:

HEX ok, go fish in BASE 16d
DECIMAL ok, go fish in BASE 10d

Hexadecimal numbers use upper case A thru F to represent 10 (A) thru 15 (F).

HEX numbers are a good choice to enter and view memory and register values in your machine. Therefore, some words like **DUMP** only output their info in **HEX** no matter what the **BASE**.

In this example I press Enter then type **RBASE .** And press Enter:
ok, go fish in BASE 10d
(Notice we are in **DECIMAL** – The address is hard to read!)
RBASE . (268435456 ok, go fish in BASE 10d)

In this example I know I am in **DECIMAL** from the prompt, but I want to see **RBASE** in **HEX** so here is what I do:
RBASE .H (10000000h ok, go fish in BASE 10d)

In this example I know I am in **DECIMAL** from the prompt, but I want to enter a number in **HEX** and see it's value in **BIN** (Binary) so here's how to do it:

2Ah .B (101010b ok, go fish in BASE 10d)

Likewise, if the current **BASE** is **HEX** you can enter **DECIMAL** numbers by using a "d" suffix. If you use **.** It will print out the value in the current **BASE**:
HEX (ok, go fish in BASE 16d)
12d . (C ok, go fish in BASE 16d)

Changing the serial baud rate is a good example of numeric suffix use:
9600d MYBAUD UART0_INIT

This guarantees 9600 is **DECIMAL** and works no matter what **BASE** you are in.

D. . (period) U. .R .RU and **?** print 32bit numbers on the stack in the current base without a **BASE** suffix.

For viewing numbers on the stack in a preferred base use **.D** or **.H** and **.B**

To non-destructively view items in the stack use **.S** or **.SB** , .**SD,** or **.SH**. None of these change the system **BASE**.

Numbers and BASE summary:
Numbers can be entered in three (3) other **BASE**'s regardless of the current **BASE**. A suffix of **b** or **%** for Binary, **d** or **#** for Decimal, and **h** or **$** for Hexadecimal causes the number to be converted in the suffix-specified base without changing the current **BASE** setting.
Numbers can be displayed with or without the **BASE** suffix.
Numbers can be entered with or without the **BASE** suffix.
Double numbers are created by placing a period in the number.
-1F.h 1.0 -2000.00d (are all double numbers – use ***D.*** to view)

If a number suffix is used it must be the last character!

Hard Fault Reset (HFR):
If you end up feeding **FISH** code that violates the target hardware rules, a Hard Fault Reset will occur and you will be rebooted (like **BYE**) instead of being "hung".

For example try this: ***1 1 !*** \ *Works on NXP, STM allows this!*

FISH SYNTAX:

FISH syntax is simple: white-space (a space key) separates words and numbers. Processing of the words and numbers begins after Enter is pressed.

WORD beginning a line you should type and then press Enter:

**** and
(are comments – see the glossary**)**

FISH
This word prints out the Flash memory status and the sign on message. It will re-link words saved in FLASH to the dictionary, if they exist. See **FISH_ONLY**

WORDS
This word lists all words defined, in a few categories to aid in recalling relevant words to use.

MYWORDS
This word lists only the words that you have created in RAM or saved in flash. Make yourself a word:

: MyFirstWord CR ." yippie!" CR ;

After this type in ***MYWORDS -*** Should list **MyFirstWord**
After this type in ***MYFirstWord*** - Should type yippie! on it's own line because of the bracketing **CR**'s.

.DS
This word will show you how much dictionary space is available. No matter what base you are in it will leave that base intact while displaying the available space in **DECIMAL**.

.VS
This word will show you how much RAM Variable space is available. No matter what base you are in it will leave that base intact while displaying the available space in **DECIMAL**.

To create a variable (with any name)

VAR TESTvar

VAR takes no argument and allocates 4 bytes in ram for the variable. When you type ***TESTvar*** the address of the variable is placed on the stack. Initializing a variable is usually done like this:

100 TESTvar ! *(Initialized it to 100 in whatever* ***BASE*** *you are in!)*

FORGET

FORGET TESTvar

removes all Words after Word name (including word name), reclaims any **VAR** space allocated and prints out **.VS** and **.DS**.

FLASH_FORGET
removes all Words saved in flash and prints out **.VS** and **.DS**.. Words in Ram are left alone.

FLASH_SAVE
relocates words in RAM to a flash page. RAM is now available for new words.

NOTE: Code saved in Flash must explicitly initialize their variables.

```
VAR Times
: InitVars 3 Times ! ;   ( Initialize Times to 3 )
```

RUN is reserved as a turnkey word. It is used to run your application at power up. You can test it in Ram with **ABORT** before saving to Flash like this:

```
: RUN BEGIN CR ." Running! – press any key to exit" ?KEY UNTIL ;
ABORT            ( Executes RUN - Press any key to exit )
FLASH_SAVE       ( RUN is now in Flash )
```

When saved to flash it will run at power-up, or by typing **COLD** or **BYE**
Use **FLASH_FORGET** to start over again!

BYE is **COLD** with initialization of the serial port to the last baud set default (9600) or by use of **MYBAUD**, with the system default of 8bits, no parity, 1 start bit and software flow control using the XON/XOFF protocol.

ABORT resets the interpreter and executes RUN if it is defined.

Writing FISH software

Test your work interactively in the terminal screen. Copy good work and paste it to a text file. The terminal program can now be used to download your good code from the text file.

Do not try to download buggy/untested code. **FISH** will try to process the code and the errors - one after another – you will get you confused. FISH may hang, requiring a reset. Test your code interactively and then save it to a file.

That's why we advise testing in the terminal and only downloading known-good code. But that is also the beauty of Forth. Interactive testing is easy.

I f you are modifying code in your text file and downloading it, use **FISH_ONLY** at the beginning of your file to forget all previous code and start fresh with this download. Add definitions one at time and test them before adding more!

FISH STYLE:

This methodology is applicable to factoring a word thru a full application:

1) Create Ideal code for reference:
   - This is the Top Down Vision, the outer structure:
   - Can be used as a demo with dummy output.

2) Write real code to flesh out the ideal code:
   Don't commit conceptually to the outer structure:
   Let the bottom up low-level code modulate the outer structure.

3) Repeat until you reach a good solution:
   Let this sit; walk away, come back, review and rethink:

4) Consider if what you have is good enough to survive a future of improvements and re-use.

MATH:

FISH does math using a system known as RPN (Reverse Polish Notation), which does not need parentheses for compound expressions. Numbers are put on The Stack, then math operators use the numbers on The Stack and replace them with results.

Algebraic expressions such as "((1+2)-3)=" are written in RPN for **FISH** as

1 2 + 3 - .

See **+ - /** and **/MOD** in the Glossary, which contains many examples.
And just try things out. Just keep in mind that numbers will go to the stack.
The most recently entered numbers will be the first ones operated on.

1 2 3 .S (TOS> 3 2 1)

After a while it will become second nature. Boolean Logical operations like **AND** and **OR** and **NOT** operate on all 32 bits. **ASR LSR** and **LSL** are shift operations.

Some 64-bit operators are available. See **D+**, **D.** etc. in the Glossary.

NOTE:
If You Are New To Forth: an excellent overview and online tutorial for the Forth language can be found at:
http://forth.com/starting-forth/

The introduction is a whirlwind tour/history of FORTH.
Chapter 1 & 2 of Starting FORTH matches **FISH** exactly.
Chapter 3 is an old FORTH standard editor and is not used in **FISH**. With **FISH** you can use whatever editor you like since your PC/terminal simply downloads text files you make with your own editor..
Chapter 8: **FISH** uses **VAR** for VARIABLE and **CON** for CONSTANT.
Chapter 11: **FISH** uses **<BUILDS** instead of **CREATE**. **CREATE** is slightly different - See the **FISH** glossary.
By now you may know that **FISH** uses **WORDS** instead of VLIST (Vocabulary list).

Forth in micro-controller development is discussed in this article:
http://www.forth.org/lost-at-c.html

Currently Mediafire.com and http://spacetimepro.blogspot.com/ is where you can find the latest **FISH** Reference Models.

Mediafire.com: https://www.mediafire.com/folder/6fqkfykcel80s/FISH_Forth

FISH user support is at Space-Time-Forth. This is for discussion about coding using the **FISH** Reference Model.

Email me with other **FISH** questions or to contribute financially to **FISH** at:
cwpjr02@gmail.com \ pizza always welcome!

Other **FISH** online resources will be listed here in subsequent versions of this document.