FISH is a 32bit Flash resident Forth.

FISH V1.7: By Clyde W. Phillips Jr. and A-TEAM FORTH. Copyright 2013-2015

This **FISH** document enumerates and explains a Public Reference Model (RM) **FISH** word set for all ARM Cortex-M 32bit micro-controllers. A Small Reference Model (SRM) may be provided for the smallest targets; NXP 812 Done.

FISH is a modernized F83-ish standard. **FISH** is richly featured: **FLASH_SAVE** of code in Ram, a reserved **RUN** turnkey word, single and multi-line comments and a slimmed down categorized dictionary are some highlights of it's intention to support application development on modern ARM 32bit micro-controllers.

FISH is ideal for learning the best disciplines for *SW Development.* It is *Agile* and supports *Object Oriented* development. **FISH** is an ideal MAKER production tool.

The latest **FISH** release can be found here:

https://www.mediafire.com/folder/6fqkfykcel80s//FISH%20Forth

See the **Fish&Chips.pdf** document to determine which .hex image file to use, along with the UART required, and the Flash and Ram expected, for your particular target. The recommended terminal programs which support text source code file download are specified there.

FISH “how-to” questions will be answered on this forum:

https://groups.yahoo.com/neo/groups/Space-Time-Forth/info

Questions about the implementation, design choices and other request are not for the forum. Email me as per below:

Sorely needed contributions can be facilitated by emailing me at cwpjr02 at gmail.com.

FISH SYNTAX:

FISH syntax is simple: white-space (the space key) separates words and numbers. Processing of the words and numbers begins after Enter is pressed. Backspace and the Delete key are used to correct line entry.

Forth often mutates into various dialects, and **FISH** is no exception. By only releasing and supporting a Core Wordset (*The FISH Reference Model*) it can be counted on as a stable platform for portable code across all FISH targets.

FISH most closely resembles the F83 FORTH model. **FISH** is modernized with ANS standard behavior and words where useful. Most differences are quite minor. If you have used Forth before, then here is a short list of words that have slightly different names from other dialects:
MINUS is now **NEGATE**
DMINUS is now **DNEGATE**
VLIST is now **WORDS**
?TERMINAL is now **?KEY**

FISH Defaults (All Targets):

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**

RUN is a reserved turnkey word. If you have defined ***RUN*** and saved it to Flash **FISH** will boot into your application (***RUN***). Use **ABORT** in your ***RUN*** word for an error condition that wants to restart ***RUN***.

CO is a reserved error handler word. See **EHON** and **EHOFF**

Enter CR and expect the "OK" prompt with current **BASE** printed in decimal. **POFF** turns the prompt off, **PON** turns it back on. See **P** also.

System variable **SYSCLK** contains the processor clock speed of this chip/board running **FISH**. The SYSTICK 24bit timer may be enabled and if so generates an interrupt that increments the **STCTR** variable. The interval of the timer is set by **MS** and **DELAY** and is undefined at power-up. IF the SYSTICK interrupt is disabled with **SYSTICK_IRQ_OFF** the **STCTR** variable will stop being incremented. **SYSTICK_IRQ_ON** will restart the **STCTR** variable.

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. **UARTx_INIT** changes the baud rate.

115200d MYBAUD UART0_INIT *\ This target uses UART0*

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*.

FISH organizes it's 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 current **BASE**.

In this example I know from the prompt that **FISH** is in **DECIMAL** 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.

. (period) inside or at the end of a number creates a double number, that is a 64-bit value (2 32bit cells on the stack). Ex: **12.** and **11.12** both create double numbers. See **D. D+** and **DPL**

, (comma) may be used inside a number to format and create a 32 bit number. Ex: **1010,1001,0111,0011b** and **FF,FE,C2,DCh** create 32bit values.

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

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.
32bit numbers can be formatted with a comma.
FF,FE,C2,DCh 1,269,783,563d
64-bit 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 hardware rules,
a Hard Fault Reset will occur and you will be rebooted (like **BYE**) instead of being "hung".

FLASH Resident Code Libarary:
A separate document may be included that contains a list of **FISH** internal routines resident in FLASH, that are functionally a built-in Code Library. These routines have no Name Field Address (**NFA**) or Link Field Address (**LFA**) and therefore are not seen in the **FISH** dictionary when viewed by **WORDS**.

These routines are called :NONAME (i.e. hidden, header-less) words. The :NONAME routine name in that section are **CFA** names, in *BOLD TYPE* . **CFA** names are listed in the *.sym* document, included in all releases of **FISH**.

Besides :NONAME routines, the .sym file also contains the Ram addresses of **FISH** system variables and buffers, and a complete list of SoC peripherals register addresses!

To use a :NONAME word look up it's **CFA** name in the *.sym* file, for example BELL is listed in the .sym file as follows: (Check your .sym file for the correct address!!)
00000300 t BELL
To use the above information to make a words for use in **FISH** do as follows:

***: Bell 300h EXECUTE ; (BELL** could be used without conflict)*

Many **FISH** definition's in this document include example code in *BOLD ITALICS TYPE-FACE.* You can copy them from here and paste them into your terminal program to learn how to use **FISH**. The examples are in the form of:

(Comments and expected output are inside parenthesis)
: .DOLLARS <# # # 2Eh HOLD (CR - no prompt while compiling!)
#S 24h HOLD #> TYPE ; (now you will see a prompt!)
56668. .DOLLARS ($566.68)

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: VARIABLE and CONSTANT.
Chapter 11: **CREATE** and **DOES>, <BUILDS** is also defined in **FISH**.

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

Currently http://spacetimepro.blogspot.com/ is where you can find custom hardware and projects that run **FISH!**

Other online resources will be listed here or included in a separate document.

----------*GLOSSARY FORMAT*----------

In this glossary stack diagrams are contained within parenthesis, i.e. (--). If there are stack arguments used by a **FISH** word they are within the parenthesis as defined below. Stack items on the left side of the two dashes are arguments the word consumes, and arguments on the right side of the two dashes are arguments the word returns. The top of stack (TOS) is always the rightmost item on either side of the two dashes Stack arguments are in this form.

MSW=Most significant word in double number. LSW is the least significant word.

addr	address of something on the cpu bus
b	8-bit byte.
c	7-bit ASCII character.
n	32bit signed integer.
u	32bit unsigned integer.
x	any 32bit item, not necessarily a number.
value	a number specific to the range allowed in a word.
d	64-bit signed double integer, the signed MSW is on top of the stack.
ud	64-bit unsigned double integer, the MSW is on top of the stack.
f	boolean (all 32bits) flag. 0=false, -1=true.
tf	boolean (all 32bits) true flag=-1.
ff	boolean (all 32bits) false flag=0.
rem quo	used when 2 stack item results are not a double number.
cntstr	addr of a string that starts with a count byte.
nullstr	addr of a string that is terminated by a null.

EXPECT created strings start with a count byte and are null terminated – the null termination is not in the count! The address +1 (after the count byte) of a string created by **EXPECT** equals a null terminated string.

All arithmetic in **FISH** is implicitly 32bit signed integer math (unless otherwise noted), with error and underflow indication unspecified.

Additionally *Cccc* refers to an ASCII string of any case, usually a word's name.

FISH word definitions in this document are defined in this form:

NAME [SYMBOL_FILE_CFA_NAME] (stack diagram)
Text explaining what the **FISH** word **NAME** does, in this case assume it's **EXECUTE** and an example of using it follows.

(Comments and expected output are inside parenthesis)
' FISH CFA EXECUTE (execute **FISH** = types out the sign-on message)

See **' CFA** and **FISH**

----------*Start of **FISH** Glossary Documentation*----------

NAME [SYMBOL_FILE_CFA_NAME] (stack diagram) **Errata on word**

0 [NULL] (--) **IMMEDIATE**
Executed by *Enter* or when input line length maximum is encountered in the **INTER** -pret loop. Returns to **QUIT** loop.
See **TIB** and **EXPECT**

! [STORE] (n addr --)
Write 32bit n to addr. Pronounced "store".
See **C!**

 # [DIG] (d1 -- d2)
Occurs in a colon-definition in the form:
<# # n HOLD #S SIGN #>
Generate from a double number d1, the next ASCII character which is placed in an output string, usually in **PAD**. Result d2 is the quotient after division by **BASE**, and is maintained for further processing. Used between **<#** and **#>**.
Each number = BASE_SV*number+(character-'0')
See **#S HOLD SIGN** and **<#** or **#>** for example code.

#> [EDIGS] (d -- addr count)
Occurs in a colon-definition in the form:
<# # n HOLD #S SIGN #>
Terminates numeric output conversion, leaving the text address and character count suitable for use with **TYPE**.
(In this example 24h is ASCII $ and 2Eh is . <Period>)
: .DOLLARS <# # # 2Eh HOLD (CR - no prompt until this is finished!)
#S 24h HOLD #> TYPE ; (now you will see a prompt!)
56668. .DOLLARS ($566.68)
See **<# # #S HOLD** and **SIGN**

#S [DIGS] (d1 -- d2)
Occurs in a colon-definition in the form:
<# # n HOLD #S SIGN #>
Generates ASCII text in the text output buffer, usually **PAD**, by the use of **#**, until a zero double number d2 results. Used with **#**, **HOLD**, and **SIGN** between **<#** and **#>**.
See **# HOLD SIGN <#** and **#>** for example code.

$LEN [NULLSTRLEN] (addr -- addr len)
Count length of null terminated string, leaving addr and len on stack.
Can be used with **TYPE**.

PAD 8 EXPECT (waits for user to type 8 characters)

PAD $LEN TYPE

See **TYPE COUNT** and **PAD**

' [TICK] (-- pfa) **IMMEDIATE**
Used in the form:

' Cccc

When interpreting leaves the parameter field address of the dictionary word Cccc. If the word is not found after a search of CURRENT, an appropriate error message is given. When compiling it is executed inside a colon-definition because it is an **IMMEDIATE** word. Used to compile the address as a literal.

' FISH CFA EXECUTE (Execute **FISH**, same as typing **FISH**)

' WORDS NFA ID. (Print out the name field of **WORDS**)

: Cccc ' FISH CFA EXECUTE ; (Cccc executes **FISH**)

Pronounced "tick".
See **CFA NFA LFA ID.** and **IMMEDIATE**

([PAREN] (--) **IMMEDIATE**
Used in the form:

(Cccc)

Ignore comments within parenthesis. A right parenthesis, “)” ends the comments. The comments may be multi-line but a space or tab is required for any line that is otherwise empty for formatting purposes, else a comment error will be printed. May occur during execution or in a colon-definition. A blank after the leading parenthesis is required.

```
(
Multiline Comments should always end the comment on it's own line.
It especially helps when commenting out code like:
1 PortC_Ack ! \ Bit Bang ACK - A closing ")" here is hard to find...
)
```

See ****

*** [STAR]** (n1 n2 -- n)
Leave the signed product n (32bit) of two (32bit) signed numbers n1 and n2. If the product exceeds 32 bits, it will be truncated.

2 3 * . (6)

***/ [SSLASH]** (n1 n2 n3 -- n)
Leave the ratio n = n1*n2/n3 where all are signed numbers. Retention of an intermediate 64-bit product permits greater accuracy than would be available with the sequence n1 n2 * n3 /

3 2 1 */ . (6)

***/MOD [SSMOD]** (n1 n2 n3 -- rem quo)
Leave the quotient and remainder of the operation n1*n2/n3. A 64-bit intermediate product is used, as it is for */.

3 2 1 */MOD . (0 6)

+ [PLUS] (n1 n2 -- n) SIGNED
Add signed n1 and signed n2, leaving the signed sum.

4 3 + . (7)

+! [PSTORE] (u addr --)
Add u to the value at the address. u can be positive or negative. Pronounced "plus-store".

+LOOP [PLOOP] (n --) **IMMEDIATE**
Occurs in a colon-definition in the form:

n1 n2 DO ... +LOOP

At run-time, **+LOOP** selectively controls branching back to the corresponding DO based on n, the loop index, and the loop limit. The signed increment n is added to the index and the total compared to the limit.

The branch back to **DO** occurs until the new index is equal to or greater than the limit (n1>0), or until the new index is equal to or less than the limit (n1<0). Upon exiting the loop, the parameters are discarded and execution continues ahead.

: print10+loop 10 0 DO I . 2 +LOOP ;

print10+loop (0 2 4 6 8)

See **DO I LOOP** and **LEAVE**

, [COMMA] (n --) ALIGNED
Write n into next **ALIGNED** dictionary memory cell; **DP** is incremented.
See **HERE** and **LATEST**

- [SUB] (n1 n2 -- n)
Leave the difference of n1-n2 as n.

1 2 - . *(-1)* ***2 1 - .*** *(1)*

-DUP [ZNDUP] (n -- n) (if zero)
(n -- n n) (non-zero)
Reproduce n only if it is non-zero. This is usually used to copy a value just before **IF**, to eliminate the need for an **ELSE** part to drop it.

. [DOT] (n --) SIGNED
Print n as a number in the current **BASE**. A trailing blank follows. Pronounced "dot".

7 . (7)

." [DOTQ] (--) **IMMEDIATE**
Type the quoted string, or compile an inline string to be typed at run-time. Used in the form:

." Cccc "

If executed outside a definition, ." will immediately print the text until the final " (double quote). The maximum number of characters is system dependent.

.B [DOTBIN] (n --)
Print n as a number in binary (**BIN**) using **U.** , not affecting **BASE**.

.D [DOTDEC] (n --)
Print n as a number in **DECIMAL**, not affecting **BASE**.

.DS [DOTDICTSPACE] (--)
Prints number of bytes available in dictionary in **DECIMAL**, not affecting **BASE**. See **.VS**, **HERE** and **UP**

.H [DOTHEX] (n --)
Print n as a number in hexadecimal (**HEX**) using **U.** , not affecting **BASE**.

.R [DOTR] (n1 n2 --) SIGNED
Print the signed number n1 in the current **BASE,** right-aligned in a field whose width is n2. No following blank is printed.

.RU [DOTRU] (n1 n2 --) UNSIGNED
Print the unsigned number n1 in the current **BASE,** right-aligned in a field whose width is n2. No following blank is printed.

.S [DOTS] (--)
Non-destructively print items on stack in current **BASE**. TOS> is printed first.

.SB [DOTSBIN] (--)
Non-destructively print items on stack in binary (**BIN**), not affecting **BASE**. TOS> is printed first.

.SD [DOTSDEC] (--)
Non-destructively print items on stack in **DECIMAL**, not affecting **BASE**. TOS> is printed first.

.SH [DOTSHEX] (--)
Non-destructively print items on stack in hexadecimal (**HEX**), preserving **BASE**. TOS> is printed first.

.VS [DOTVARSPACE] (--)
Prints number of bytes available in Ram **VARIABLE** space in **DECIMAL**, not affecting **BASE**.
See **VARALLOT, UP** and **.DS**

/ [SLASH] (n1 n2 -- n) SIGNED
Leave the signed quotient n of n1/n2.
4 2 / . (2)

/MOD [SLMOD] (n1 n2 -- rem quo) SIGNED
Leave the remainder and signed quotient of n1/n2. The remainder has the sign of the dividend.
4 2 /MOD . (0 2)

0 1 2 3 4 [ZERO ONE TWO THREE FOUR] (-- n)
These small numbers are used so often that is attractive to speed up **FISH** by defining them by name in the dictionary as constants.

0< [ZLESS] (n -- f)
Leave a true flag if the number is less than zero (negative), otherwise leave a false flag.
1 0< . (0) ***-1 0< .*** (1)

0= [ZEQU] (n -- f)
Leave a true flag is the number is equal to zero, otherwise leave a false flag.

1 0= . (0) ***0 0= .*** (1)

1+ [ONEP] (n -- n+1)
Add 1 to the top of the stack.

1 1+ . (2)

2+ [TWOP] (n -- n+2)
Add two to the top of the stack.

1 2+ . (3)

2* [TWOSTAR] (n -- n*2)
32bit left-shift by one bit position. (multiply by 2).

4 2* . (8)

2/ [TWOSLASH] (n -- n/1)
32bit arithmetic-shift-right by one bit. Usually used as a “floored” signed divide-by-two.

4 2/ . (2)

4- [FOURM] (n -- n-4)

8 4- . (4)

4+ [FOURP] (n -- n+4)

4 4+ . (8)

2DUP [TDUP] (n2 n1 -- n2 n1 n2 n1)
Duplicate top two stack items. The prefix of 2 convention means an operation on the top two stack items.
See **-DUP**

: [COLON] (--) **IMMEDIATE**
If the dictionary is full, print an error message and abort.
Used in the form called a colon definition:
: Cccc ... ;
Creates a dictionary entry using Cccc as the word name. The definition of the word '...' is compiled into the word until **;** is encountered. The compiling process is done by the text interpreter while **STATE** is 0xC0. **IMMEDIATE** words with the precedence bit P set are executed rather than being compiled.
See **.DS IMMEDIATE** and **WORDS**, which shows a category of FISH words that can be used inside colon definitions.

; [SEMI] (--) **IMMEDIATE**
Terminate compilation of a COLON (**:**) definition and return **STATE** to zero, the interpretation **STATE**.
See **:**

< [LESSTHAN] (n1 n2 -- tf/ff) SIGNED
Leave a true flag if n1 is less than n2 otherwise leave a false flag.
1 80 < . *(1)* ***80 1 < .*** *(0)*

<# [BDIGS] (--)
Setup for pictured numeric output formatting using the words:
<# # #S SIGN #>
The conversion is done on a double number producing text at PAD.
(In this example 24h is ASCII $ and 2Eh is . <Period>)
: .DOLLARS <# # # 2Eh HOLD (no prompt until this I finished!)
#S 24h HOLD #> TYPE ; (now you will see a prompt!)
56668. .DOLLARS ($566.68)
See **# #S #> HOLD** and **SIGN**

<BUILDS [BUILDS] (--) **CREATE** can be used instead.
Used within a colon definition:

: Cccc <BUILDS ... DOES> *...* ***;***

Each time Cccc is executed, <BUILDS defines a new word with a high-level execution procedure. Executing Cccc in the form:

Cccc nnnn

uses **CREATE** to create a dictionary entry for nnnn with a call to the **DOES>** part for nnnn. When nnnn is later executed, it has the address of its parameter area on the stack and executes the words after DOES> in Cccc. **<BUILDS** and **DOES>** allow run-time procedures to written in high-level rather than in assembler code.

: DCONST <BUILDS , , DOES> DUP @ . 4 + @ . ;
-1 -1 DCONST DNEGATIVEONE
DNEGATIVEONE D. *(* -1 *)*

See **DOES>** and **CREATE**

= [EQUAL] (n1 n2 -- f)
Leave a non-zero true flag if n1=n2; otherwise leave a false flag.

2 1 = . (0) ***3 3 = .*** (1)

> [GREATERTHAN] (n1 n2 -- f) SIGNED
Leave a true flag if n1 is greater than n2 otherwise a false flag.

-1 1 > . (0) ***1 -1 > .*** (1)

>R [TOR] (x --) R:(–- x)
Remove an item from the computation stack and place it on top of the return stack. Use must be balanced with **R>** in the same definition. Pronounced to-r.

(In this example 3 would print 3 times if **>R** and **R>** weren't used)

: ShowRstackUsage 3 DUP . >R .S R> . ;
ShowRstackUsage (3 3)

See **R>** and **R**

? [QUES] (addr --)
Print the value contained at the address in free format according to the current base using **DOT**. Equivalent to "**@ .**"

?ADDR [QADDR] (addr --)
If addr not aligned by 4 bytes, then issue message and **ABORT**.

?ERROR [QERROR] (f cntstr --)
If f is true, then **TYPE** error message (cntstr) and **ABORT**.

?KEY [QKEY] (-- f)
Return zero unless there is a char in the UART's RX FIFO – the char is not consumed.

```
: Wait_For_Key BEGIN ?KEY UNTIL ;  ( Execute until key is entered. )
Wait_For_Key  ( Press a key to exit this word )
    KEY .H   ( Get the key, see and forget it. )
```

See **KEY** and **UART0_LSR**

@ [AT] (addr -- n)
Get the 32bit contents of addr and place the contents on TOS.
Pronounced “fetch”.

ABORT [ABORT) (--)
Clear the stacks and enter the execution state. Does not reset the dictionary. If ***RUN*** is defined in Ram or Flash execute it first.
See **COLD** and **BYE**

ABS [ABS] (n -- u)
Leave the absolute value of n as u.
See **DABS**

AGAIN [AGAIN] (--) **IMMEDIATE**
Used in a colon-definition in the form:

```
BEGIN ... AGAIN
```

At run-time, **AGAIN** forces execution to return to corresponding **BEGIN**. There is no effect on the stack. Execution cannot leave this loop (unless **R> DROP** is executed). At compile time, **AGAIN** compiles the :NONAME **[BRAN]** instruction with an offset back to **BEGIN** (from **HERE** at the time to addr).
See **BEGIN** for example.

ALIGN [ALIGN] (--) DPANS94:
Pads the current DP with 0FFh until aligned. Used in **,** reflected by **HERE**.
Use when DP may be unaligned and the next value to be compiled needs to be aligned, for example after an odd amount **ALLOT**'ed or odd number of **C,** usage.
See **ALIGNED C,** and **ALLOT**

ALIGNED [ALIGNED] (addr -- aligned-addr) DPANS94
Modify addr if necessary to ensure it is aligned to a 32bit boundary.
See **ALIGN**

ALLOT [ALLOT] (n --) SIGNED
Add the signed number to the dictionary pointer **DP**. May be used to reserve or de-allocate dictionary space.
CAUTION! **ALLOT** and **C,** are the only words that can allot uneven amounts.
NOTE: you can use **ALIGN** after **ALLOT** or **C,** to insure **DP** is aligned.
See **ALIGN ALIGNED VARALLOT DP HERE** and **LATEST**

AND [AND] (x1 x2 -- x)
Leave the bit-wise logical AND of x1 and x2.
110b 10b AND .B (10)
See **OR NOT ASR LSR** and **LSL**

ANDBITS [ANDBITS] (addr value --)
Logical **AND** the value-bits with the contents of addr using read, modify, write method.
(Create and initialize variable **TEST_ANDBITS**)
VARIABLE TEST_ANDBITS 110b TEST_ANDBITS !
(Use **ANDBITS** to modify **TEST_ANDBITS**)
TEST_ANDBITS 10b ANDBITS
(View results)
TEST_ANDBITS @ .B (10)
See **CLRBITS** and **SETBITS**

ASR [ASR] (sn count -- sn') SIGNED
Shift sn (sign-extended) right by count. Valid count values are 0 to 31.
-3 2 ASR . (-1)
See **AND OR NOT LSR** and **LSL**

BASE [BASE_SV] (-- addr)
A system variable containing the current number base used for input and output conversion. The current **BASE** is reflected in decimal in the system prompt.
See **BIN DECIMAL HEX** and **DIGIT**

BEGIN [BEGIN] (-- addr n) **IMMEDIATE**
Occurs in a colon-definition in form:

BEGIN ... UNTIL
BEGIN ... AGAIN
BEGIN ... WHILE ... REPEAT

At run-time, **BEGIN** marks the start of a sequence that may be repetitively executed. It serves as a return point from the corresponding **UNTIL**, **AGAIN** or **REPEAT**. When executing **UNTIL**, a return to **BEGIN** will occur if the top of the stack is false; for **AGAIN** and **REPEAT**, a return to **BEGIN** always occurs.

: Wait_For_Key BEGIN ?KEY UNTIL ; (Execute until key is entered.)
Wait_For_Key (Press a key to exit this word)
KEY .H (Get the key, see and forget it.)

See **AGAIN UNTIL WHILE** and **REPEAT**

BIN [BIN] (--)
Set the numeric conversion base to two (binary). The current **BASE** is reflected in decimal in the system prompt. **0** and **1** are the only valid digits in binary.

BIN (ok, go fish in BASE 2d)

See **DIGIT HEX** and **DECIMAL**.

BL [BLANK] (-- n)
Push ASCII space character (20h) to TOS.

BLANKS [BLANKS] (addr count --)
Fill an area of memory beginning at addr with blanks (20h).
See **ERASE FILL PAD** and **DUMP**

BYE [BYE] (--)
Cold-restart with serial initialization in last baud rate set by **MYBAUD**. If **MYBAUD** not used the deafult baud rate will be used. Executes **ABORT**
See **COLD** and **ABORT**.

C! [CSTORE] (n addr --)
Store the 8-bit byte in the 8 least significant-bits of n at addr.
See **C@**

C, [CCOMMA] (n --)
Store the least-significant 8-bits of n into the next available dictionary byte, advancing the dictionary pointer.
>>> CAUTION: **ALLOT** and **C,** are the only words that can allot uneven amounts.
NOTE: Use **ALIGN** after **ALLOT** or **C,** when alignment is needed.
See **, ALIGNED** and **ALIGN**

C@ [CAT] (addr -- b)
Leave the 8-bit contents of addr on the stack as a 32 but number.
Pronounced “C-Fetch” or “Char Fetch”.
See **C!**

CFA [CFA] (pfa -- cfa')
Convert the Parameter Field Address (**PFA**) of a definition to its Code Field Address (**CFA**). **CFA**'s are the execution entry point for words.
(**' FISH** leaves the **PFA** of **FISH** on the stack)
' FISH CFA EXECUTE (types out the sign-on message)
See **PFA** and **LFA**

CLRBITS [CLRBITS] (addr value)
Logical **AND** then **NOT** the value-bits with the contents of addr using read, modify, write method.
(Create and initialize variable **TEST_CLRBITS**)
VARIABLE TEST_CLRBITS 110b TEST_CLRBITS !
(Use **ANDBITS** to modify **TEST_CLRBITS**)
TEST_CLRBITS 10b CLRBITS
(View results)
TEST_CLRBITS @ .B (100)
See **ANDBITS** and **SETBITS**

CMOVE [CMOVE] (addr1 addr2 count --)
Move the specified count of bytes beginning at addr1 to addr2. The contents of from_addr is moved first, proceeding toward high memory.
See **FILL**

COLD [COLD] (--)
Reset **FISH** system and execute **ABORT**, which execute ***RUN*** if saved in Flash, else restarts the outer interpreter, **QUIT**.
>>> CAUTION: all user words in RAM are forgotten.
See **ABORT BYE** and **FISH_ONLY**

CONSTANT [CONSTANT] (n --) **CONSTANT and CON** defined.
A CONSTANT-defining word used in the form:

n CONSTANT Cccc

to create word Cccc, with its parameter field containing n. When Cccc is later executed, it will push the value of n to the stack.

8 CONSTANT EIGHT EIGHT . (8)

See **VARIABLE**

COUNT [COUNT] (addr1 -- addr2 n)
The byte at addr1 contains the count of the text starting at addr+1. **COUNT** leaves the byte-count n and the text address addr2 (addr1+1). **COUNT** is usually followed by **TYPE**. User Strings may be up to 255 characters.
See **$LEN** and **TYPE**

CR [CR] (--)
Transmit a carriage return (ASCII 0Dh) and line feed (ASCII 0Ah) to the terminal.
CRS [CRS] (n --)
Transmit n carriage returns (ASCII 0Dh) and line feeds (ASCII 0Ah) to the terminal.
CREATE [CREATE] (--) Maximum word length is 31 characters.
A defining word used in the form:

CREATE Cccc

Create a word in the Dictionary named Cccc. Executing the created word returns the address following itself, which is **HERE**. The returned address points to nothing until some companion data or code is made to be the object of the address. IF **ALLOT** is used after **CREATE**, **HERE** points to after the allotted memory.
If the dictionary is full, then **CREATE** will print an error message and **ABORT**.

CREATE Cccc (Cccc is now a Word, like a CONSTANT with no value.)
HERE .H (Show next available dictionary address)
Cccc .H (Will be the same – where you can put data or code)
80d ALLOT (Allocate 80 Bytes. Cccc returns the beginning of the array)

Use with **DOES>** inside a definition.
See **<BUILDS DOES> LATEST DP CURRENT** and **HERE**

CURRENT [CURRENT_SV] (-- addr of CURRENT)
Leave the address of the system variable (**CURRENT**), which contains the address of the **NFA** of the last word in the dictionary. **CURRENT** is searched every time the interpreter looks for a word.

CURRENT ? LATEST . (2 numbers printed that are the same)

See **LATEST**

D+ [DPLUS] (d1 d2 -- d) SIGNED
Leave the signed double number sum of two signed double numbers.

D. [DDOT] (d --)
Print a signed double number from a 64-bit value. The high-order 32 bits are topmost on the stack. Conversion is performed according to the current **BASE**. A blank space follows.
Pronounced D-dot.
See **DOT**

D.R [DDOTR] (d n --) SIGNED
Print a signed double number d right-aligned in a field n characters wide.

DABS [DABS] (d -- ud)
Leave the absolute value ud of a signed double number.
See **ABS**

DBASE [DBASE] (-- addr)
Return base addr of user words defined in the dictionary, that are located in RAM memory until saved.
See **DUMP**

DECIMAL [DECIMAL] (--)
Set the numeric conversion **BASE** to ten. The current **BASE** is reflected in decimal in the system prompt.
DECIMAL (ok, go fish in BASE 10d)
See **HEX** and **BIN**

DELAY [DELAY] (n value --) 0 0 **DELAY** \ stops timer and it's interrupts.
Delay clock cycles indicated by value, repeating n times. The ARM SYSTICK 24 bit timer is used, with an interrupt handler that increments the **STCTR** variable every time value countdown reaches 0. **DELAY** sets **STCTR** to negated n, and returns when **STCTR** reaches 0. If n=0 just initialize SYSTICK reload value.
To calculate and execute a given delay: (Do this if you change **SYSCLK**!)
1 SYSCLK 100d / 1 - **DELAY** (Value of 100 = 10 ms delay)
See **MS SYSCLK STI_ON STI_OFF** and **STCTR**

DIGIT [DIGIT] (c base -- n 1) (ok)
(c base -- 0 (bad)
Converts the ASCII character c (using **BASE**, c must be valid in **BASE**) to its binary equivalent n, accompanied by a true flag. If the conversion is invalid, **DIGIT** leaves only a false flag.

DLITERAL [DLITERAL] (-- d) **IMMEDIATE**
If compiling, then compile a 64-bit literal created within a definition, in the form:
: Cccc [(create 64bit value] DLITERAL ;
Does nothing if interpreted. This definition is **IMMEDIATE** so that it will execute during a colon definition. This means the compilation must be suspended by **[** , allowing for compile-time calculation of the value, the resume compiling by **]** , and **LITERAL** compiles this value in the definition.
: Cccc [3. 2. D+] DLITERAL ; (***Cccc*** computes 5 while compiling!)
Cccc D. (5)
See **LITERAL IMMEDIATE [COMPILE] [** and **]**

DNEGATE [DNEGATE] (d1 -- d2) RENAMED --> DMINUS to **DNEGATE**
Convert d1 to its two's-complement d2.
1. DNEGATE D. (-1)
-1. DNEGATE D. (0)
See **NEGATE**

DO [DO] (n1 n2 --) **IMMEDIATE**
Occurs in a colon-definition in the form:
n1 n2 *DO ... LOOP*
n1 n2 DO ... +LOOP
At run-time, **DO** begins a sequence with repetitive execution controlled by a loop limit n1 and an index with initial value n2. Upon reaching **LOOP** the index is incremented by one. **+LOOP** is incremented by another inline index argument, explained in the example below. Until the updated index equals or exceeds the limit, execution loops back to just after **DO**. When the index is equal or exceeded execution continues after the **LOOP** or **+LOOP**. Within a loop, **I** will copy the current value of the loop-index to the stack for use as variable in the construct.
: print10 10 0 DO I . LOOP ;
print10 (0 1 2 3 4 5 6 7 8 9)
: print10+loop 10 0 DO I . 2 +LOOP ;
print10+loop (0 2 4 6 8)
See **I LOOP +LOOP** and **LEAVE**

DOES> [DOES] (pfa --)
Occurs in a colon-definition in the form:

CREATE *... DOES> … \ ANS*
<BUILDS ... DOES> … \ F83

This construct is used to make other words that execute the words you place after the **DOES>**. Used in combination with **CREATE** or **<BUILDS**. When the **DOES>** part executes, it begins with the address of the first parameter of the new word on the stack. This allows interpretation using this area, or its contents. Typical uses include assemblers, multidimensional arrays, and compiler generation.

(Create a word that creates constants that print their value.)
: .CONST CREATE , DOES> @ . ;
(7 is compiled in the new word by the comma above)
7 .CONST SEVEN
(The words after **DOES>** execute with the PFA of seven on the stack)
SEVEN (7)

See **<BUILDS** and **CREATE**

DP [DP_SV] (-- addr of **DP**)
A system variable, the dictionary pointer, which contains the address of the next free memory in the dictionary. The value may be read by **HERE.**

HERE .H (Prints the address in **DP** in hexadecimal)

See **: , ALLOT and CALLOT**

DPL [DPL_SV] (-- addr of **NDPL**)
A system variable (**NDPL**) containing the number of digits to the right of the decimal-point of the last double number input. It may also be used to hold the output column location of a decimal-point in user-generated formatting. The default value on single number input is -1.

DROP [DROP] (n --)
Discard n from the top of the stack.
See **DUP SWAP ROT** and **OVER**

DUMP [DUMP] (addr n --)
Print n lines of the addr and 4 columns of 4 bytes each in hexadecimal.
addr must be a aligned by 4 or an error message will be issued. FISH is “Little Endian”, so addresses will be coherent in **DUMP** but character strings will be backwards in each column. Press any key to abort a **DUMP**.

DBASE 5 DUMP (Prints five lines of the start of the dictionary in ram.)

DUP [DUP] (x -- x x)
Duplicate the item on the top of the stack.
See **DROP SWAP ROT** and **OVER**

ELSE [ELSE] (--) **IMMEDIATE**
Occurs within a colon-definition in the form:
IF ... ELSE ... ENDIF
At run-time, the **ELSE** section of words until **ENDIF** are executed if the argument to **IF** is false.
: True? IF ." YES" ELSE ." NO" ENDIF ;
1 True? (Prints YES)
0 True? (Prints NO)
See **IF ENDIF** and **THEN**

EMIT [EMIT] (c --)
Transmit a ASCII character to the terminal. **OUT** is incremented for each character output. Character values of 0 to 0FFh are allowed versus **TYPE** which outputs ASCII characters from 0 to 07Fh..
FBh EMIT (FBh is a ASCII CHECKMARK symbol)
See **TYPE UART0_TX UART0_LSR** and **OUT**

ENCLOSE [ENCL] (addr1 c -- addr1 n1 n2 n3)
The text scanning primitive used by **WORD**. From the text address addr1 and an ASCII delimiting character c, is determined the byte offset to the first non-delimiter character n1, the offset to the first delimiter after the text n2, and the offset to the first character past the enclosed text. This procedure will not process past an ASCII 'NULL', treating it as an unconditional delimiter. NULL terminated strings are provided by EXPECT.
See **EXPECT**

ENDIF [ENDIF] (addr 2 --) **IMMEDIATE**
Occurs within a colon-definition in the form:
IF ... ENDIF
IF ... ELSE ... ENDIF
At run-time, **ENDIF** serves s the destination of the branch from **IF** or **ELSE**., depending on the argument passed to the **IF**.
: True? IF ." YES" ELSE ." NO" ENDIF ;
1 True? (Prints YES)
0 True? (Prints NO)
See also **IF ELSE** and **THEN**

ERASE [ERASE] (addr n --)
Clear a region of memory to zero from addr to addr+n.
PAD 8 ERASE (Fill 8 bytes of **PAD** with zero)
PAD 2 DUMP (Display 2 lines of **PAD** with 8 zeros)
See **FILL PAD** and **DUMP**

ERROR [ERROR] (cntstr --)
Notify the user of a fatal error by printing out the counted string, then execute **ABORT** to restart the system.
See **ABORT** and **?COMP**

EXECUTE [EXECUTE] (cfa --) RENAMED--> EXECUTE to **EXECUTE**
Execute the definition whose Code Field Address (**CFA**) is on the stack. The code field address is also called the compilation address.
' WORDS CFA EXECUTE (execute **WORDS**, same as typing **WORDS**)
See the Symbol file for all **NFA**s and **CFA**s in **FISH**, and the :NONAME section of this document whose names are CFA labels in the symbol file.
See **PFA** and **CFA**

EXPECT [EXPECT] (addr count --)
Wait for user input, characters from the terminal to the buffer starting at addr+1, until a "return" or the count of characters have been received. One or more nulls are added at the end of the text creating a null-terminated string. A count byte is placed at addr, the count being derived by $LEN. The count byte limits strings to 255 characters long. The backspace and Delete key erases previous character making **EXPECT** a useful TextBox primitive. Use **COUNT** with **TYPE** for strings saved this way. **$LEN** can be used with addr+1.
PAD 8 EXPECT (waits for user to type 8 characters or CR is entered)
PAD COUNT TYPE (Print what you typed that was saved in **PAD**)
See **PAD $LEN COUNT NUMBER** and **TYPE**

FORGET [FORGET] (--)
Executed in the form:
FORGET Cccc
Deletes the definition named Cccc from the dictionary in RAM, along with all entries that physically follow it. **VARIABLE** space is reclaimed. Prints **.VS** and **.DS**.. NOTE: Cannot be used to forget words in Flash.
See **FENCE .VS .DS** and **FLASH_FORGET**

FENCE [FENCE_SV] (-- addr of **FENCE**)
A system variable (**FENCE**) containing an address below which **FORGET**ting is trapped. To forget below this point the user must alter the contents of **FENCE**.
See **FORGET MYWORDS** and **FLASH_FORGET**

FILL [FILL] (addr quan b --)
Fill memory at the address with the specified quantity of bytes b.
PAD 8 33h FILL (Fill 8 bytes of **PAD** with 33h)
PAD 2 DUMP (Display 2 lines of **PAD** with 8 33h's)
See **ERASE PAD** and **DUMP**

FISH_ONLY [FISH_ONLY)] (--)
Reset **DP** and **UP** to original power-up dictionary settings. Effectively **FORGET**s all ram definitions, and removes links to words saved in Flash, until **COLD**, **BYE**, **FISH** or a power cycle re-links them. Useful in text file download debugging.
See **FISH FORGET FLASH_FORGET DP** and **UP**

FLASH_FORGET [FLASH_FORGET] (--)
Erase all Flash Pages (if any). If there are words in RAM **FLASH_FORGET** will not reclaim **VARIABLE** space or **FORGET** the words in Ram. Execute **FISH_ONLY** to reclaim **VARIABLE** space allocated by words in Flash if needed.
See **FLASH_SAVE** and **FISH_ONLY**

FLASH_SAVE [FLASH_SAVE] (--)
Save **WORDS** and **VARIABLE**'s to one of several Flash Pages if available.
A message is printed if there is nothing to save, or the last flash page has been used.
NOTE: After a power cycle the **VARIABLE**'s saved will still be allocated, but not set to known values, so words saved in Flash must re-initialized their **VAR**s:
VARIABLE Set? : SetVARs 4 Set? ! ; (Initialize Set? To 4)
Set? ? (Not likely to be initialized since it hasn't been initialized yet!)
(RUN will now print out message at power up)
: RUN SetVARs CR ." Set? Is now initialized to " Set? ? CR ;
FLASH_SAVE (Save **Set? SetVARs** and **RUN** to flash)
ABORT (RUN will print out message)
(Now turn power off and on and RUN will print out message)
See **FLASH_FORGET** and **FISH_ONLY**

HERE [HERE] (-- addr)
Leave the address of the next available dictionary location, stored in the system variable **DP**.
See **DP** and **LATEST**

HEX [HEX] (--)
Set the numeric conversion base to sixteen (hexadecimal). The current **BASE** is reflected in decimal in the system prompt.
HEX (ok, go fish in BASE 16d)
See **DECIMAL** and **BIN**

HOLD [HOLD] (c --)
Occurs in a colon-definition in the form:
<# # n HOLD #S SIGN #>
Used in colon definition between **<#** and **#>** to insert an ASCII character into a pictured numeric output string.
2Eh HOLD (will create a decimal point in the output number.)
See **SIGN # #S #> HOLD** and **<#** for example code

I [I] (-- n)
Copy the value on top of the return stack to the parameter stack. Used within a **DO LOOP** and **DO +LOOP** to copy the current loop index to the stack.
See **R DO LOOP +LOOP**

ID. [IDDOT] (nfa --)
Print a dictionary name from its name field address (**NFA**). **NFA is** limited to 31 characters.
LATEST ID. (Prints the name of the last dictionary name created)
See **LATEST** and **NFA**

IF [IF] (f --) **IMMEDIATE**
Occurs is a colon-definition in form:
IF ... ENDIF
IF ... ELSE ... ENDIF
At run-time, **IF** selects execution based on a Boolean flag. If the flag is true (non-zero), execution continues ahead thru the **IF** part. If flag is false (zero), execution skips till just after **ELSE** to execute the false part. After either part, execution resumes after **ENDIF**. **ELSE** and its false part are optional; if **ELSE** is missing, false execution skips to just after **ENDIF**.
: True? IF ." YES" ELSE ." NO" ENDIF ;
1 True? (Prints YES)
0 True? (Prints NO)
See also **ELSE ENDIF** and **THEN**

IMMEDIATE [IMMED] (--)
Occurs after a colon-definition in form:

: Cccc ; **IMMEDIATE**

Mark the most recently made definition (Cccc) **IMMEDIATE** so that when encountered at compile time, it will be executed rather than being compiled. This method allows definitions to handle unusual compiling situations, rather than build them into the fundamental compiler. The user may force compilation of an immediate definition by preceding it with **[COMPILE]**.

(Make Cccc an immediate word)
: Cccc CR ." I'm an immediate word!" CR ; IMMEDIATE
: watch_this Cccc FISH ; (Note Cccc executes while compiling this!)
watch_this (Note that **Cccc** isn't in **watch_this**, **FISH** is!)

See **[COMPILE] [** and **]**

IN [IN_SV] (-- addr of IN)
A system variable (**IN**) containing the byte offset within the current input text buffer (**TIB**) from which the next text will be accepted. **WORD** uses and adjusts the value of **IN**. **IN** is initialized to 1 to skip count byte.
See **EXPECT TIB** and **OUT**

IRQS_RESUME [CMSIS_ENABLE_IRQS] (–) CMSIS __enable_irq();
IRQS_SUSPEND [CMSIS_DISABLE_IRQS] (–) CMSIS __disable_irq();
These are not in the 812 Small Reference Model (SRM) Wordset.

KEY [KEY] (-- c)
Wait for the next terminal key struck, and put the ASCII value c on the stack.
LATEST [LATEST] (-- addr)
LATEST leaves the name field address (**NFA**) of the last word defined, pointed to by **CURRENT**.

(Print the name of the last definition created in the dictionary)
LATEST ID.

See **CURRENT HERE** and **DP**

LEAVE [LEAVE] (--)
Force termination of a **DO** loop the next time **LOOP** or **+LOOP** is executed.

(This example will leave the loop after the loop index = 4)
: NotAllLoop 10 0 DO I . I 4 = IF LEAVE THEN LOOP ;
NotAllLoop (0 1 2 3 4)

LFA [LFA] (pfa -- lfa)
Convert the parameter field address (**PFA**) of a word in the dictionary to its link field address (**LFA**).

' FISH (leaves **FISH**'s **PFA**)
LFA (LFA contains the NFA of the next word in the dictionary!)
@ ID. (Get that NFA and print it!)

See **PFA LFA** and **CFA**

LITERAL [LITERAL] (--) **IMMEDIATE**
If compiling, then compile a 32bit literal created within a definition, in the form:

: Cccc [(create 32bit value] LITERAL ;

Does nothing if interpreted. This definition is **IMMEDIATE** so that it will execute during a colon definition. This means the compilation must be suspended by **[** , allowing for compile-time calculation of the value, then resumed compiling by **]** , and then **LITERAL** will compiles this value into the definition.

: Cccc [3 2 *] LITERAL ; (***Cccc*** computes 6 while compiling!)
Cccc . (6)

See **DLITERAL IMMEDIATE [COMPILE] [** and **]**

LOOP [LOOP] (addr n --) **IMMEDIATE**
Occurs in a colon-definition in the form:

n1 n2 *DO ... LOOP*

At run-time, **LOOP** selectively controls branching back to the corresponding **DO** based on the loop index and limit. The loop index is incremented by one and compared to the limit. The branch back to **DO** occurs until the index equals or exceeds the limit; at that time, the parameters are discarded and execution continues ahead.

: print10 10 0 DO I . LOOP ;
print10 (0 1 2 3 4 5 6 7 8 9)

See **DO I +LOOP** and **LEAVE**

LSL [LSL] (n count -- n')
Logical (zero-extended) shift left by count. Valid count values are 0 to 31.

1 1 LSL . (2)

See **AND OR NOT LSR** and **ASR**

LSR [LSR] (n count -- n')
Logical (zero-extended) shift right by count. Valid count values are 0 to 31.

8 1 LSR . (4)

See **AND OR NOT LSL** and **ASR**

M* [MSTAR] (n1 n2 -- d) SIGNED
A mixed-precision math operation which leaves the double number signed product of two signed single numbers.

2 2 M* D. (4)
-2 2 M* D. (-4)
2 -2 M* D. (-4)

See ***

M/ [MSLASH] (d1 n1 -- srem32 squo32) SIGNED
A mixed-precision math operator which leaves the signed remainder and signed quotient from double number d1, and divisor n1.
NOTE: Result is not a Double Number but is two signed 32bit integers.
NOTE: Double numbers are created by placement of a period in the number.

3. 2 M/ . . (1 1)
-3. 2 M/ . . (-1 -1)
3. -2 M/ . . (-1 1)

See ***/*** and ***/MOD***

MAX [MAX] (n1 n2 -- max) SIGNED
Leave the greater of two (signed) numbers.

-2 2 MAX . (2)

MIN [MIN] (n1 n2 -- min) SIGNED
Leave the smaller of two (signed) numbers.

-2 2 MIN . (-2)

MOD [MOD] (n1 n2 -- modulus)
Leave the remainder of n1/n2, with the same sign as n1.

2 3 MOD . (2)
2 2 MOD . (0)
-2 3 MOD . (-2)
2 -3 MOD . (2)

MS [MS] (n --)
Delay n milliseconds. SYSTICK reload value is set to 1 millisecond. If you change the system clock do not use until fixed in later release. Use **DELAY**.
See **DELAY STI_ON STI_OFF STCTR** and **SYSCLK**

MYBAUD [MYBAUD] (baudrate --)
Must be used before using **UART0_INIT** in the form of:
115200d MYBAUD
Because of the way **FISH** handles hard-fault resets, this word provides a mechanism to preserve user baud rate despite faults. **UARTx_INIT**, **BYE** or a power cycle will change the current baud rate to the baud rate specified by **MYBAUD**.
See **UARTx_INIT**

MYWORDS [MYWORDS] (--)
Print only user defined words, including those saved in Flash.
See **WORDS** and **FENCE**

NEGATE [NEGATE] (n1 -- n2) RENAMED --> MINUS to **NEGATE**
Leave the Two's complement of a number.
4 NEGATE . (-4)
-4 NEGATE . (4)
See **DNEGATE**

NFA [NFA] (pfa -- nfa)
Convert the parameter field address (**PFA**) of a definition to its name field address (**NFA**).
(**' FISH** leaves the **PFA** of **FISH** on the stack)
' FISH NFA ID. (types the **NFA** of **FISH**)
See **PFA CFA** and **LFA**

NOOP [NOOP] (--)
A do nothing word that performs no operation.

NOT [NOT] (x1 -- x2)
Leave the bitwise logical NOT of x1 as x2.
3 NOT . (-4)
-4 NOT . (4)
See **AND OR** and **XOR**

NUMBER [NUMBER] (addr -- d)
Convert a counted character string at addr (with a null following) to a signed double number, using the current numeric base. If a decimal point is encountered, its position will be given in DPL, but no other effect occurs. If numeric conversion is not possible, an error message will be given.

PAD 3 EXPECT (<cr> enter 123)
PAD NUMBER D. (123)

See **WORD**

OR [OR] (x1 x2 -- x)
Leave the bitwise logical OR of x1 and x2 as x.

101b 010b OR .B (111)
1000b 101b OR .B (1101)

See **AND XOR** and **NOT**

OUT [OUT_SV] (-- addr of OUT)
A system variable (**OUT**) that contains a value incremented by **EMIT**. The user may examine **OUT** for output display formatting, typically by **EMIT**-ing of further spaces to reach a certain line position to format output.

OVER [OVER] (nl n2 -- nl n2 n1)
Copy the second stack value, placing it as the new TOS.

1 2 .S (TOS> 2 1)
OVER .S (TOS> 1 2 1) ***DROP DROP DROP*** (Remove example items)

See **DUP SWAP ROT** and **OVER**

PAD [PAD_SV] (-- addr of PAD)
Leave the address of **PAD** on the stack. **PAD** is typically at least 82 bytes long. **PAD** may be used in programs as a temporary buffer.

PAD 8 EXPECT (waits for user to type 8 characters or CR is entered)
PAD COUNT TYPE (Print what you typed that was saved in **PAD**)
PAD 8 33h FILL (Fill 8 bytes of **PAD** with 33h)
PAD 2 DUMP (Display 2 lines of **PAD** with 8 33h's)

See **ERASE FILL EXPECT TYPE** and **DUMP**

PFA [PFA] (nfa -- pfa)
Convert the Name Field Address (**NFA**) of a compiled definition to its Parameter Field Address (**PFA**)

. (**' FISH** leaves the **PFA** of **FISH** on the stack)
' FISH CFA EXECUTE (types out the sign-on message)

See **CFA** and **LFA**

R [R] (-- x)
Copy the top of the return stack to the parameter stack. Can only be used inside a colon definition. Use must be balanced with **>R** in the same definition.

(In this example 3 is printed 3 times.)

: ShowRstackUsage 3 DUP . >R R . R> . ;

ShowRstackUsage (3 3 3)

See **>R** and **R>**

R> [RFROM] (-- x)
Remove the top item from the return stack and leave it on the parameter stack. Can only be used inside a colon definition. Pronounced r-from.
See **>R** and **R**

REVW [REVW] (n -- n)
Reverse bytes in 32bit n.

RBASE [RBASE] (-- addr)
Return base addr of RAM. Do not modify values there!
See **DBASE**

REPEAT [REPEAT] (addr1 n1 addr2 n2 --) **IMMEDIATE**
Used within a colon-definition in the form:

BEGIN ... WHILE ... REPEAT

At run-time, **REPEAT** forces an unconditional branch back to just after the corresponding **BEGIN**. Compiles a branch back to **BEGIN**, and computes a branch offset for **WHILE**.

: BeginTest (0 1 --) BEGIN ." Begin " WHILE ." While " REPEAT CR ;

(The argument are successively used by **WHILE**)

0 1 BeginTest (Begin While Begin)

1 0 BeginTest (Begin)

See **BEGIN** and **WHILE**

ROT [ROT] (nl n2 n3 -- n2 n3 nl)
Rotate the top three values on the stack, bringing the third to the top.

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

ROT .S (TOS> 1 3 2) ***DROP DROP DROP*** (Remove example items)

See **.S SWAP OVER DUP** and **DROP**

RP@ [RPAT] (-- addr)
Place the address of the top of the return stack on the parameter stack, as it was before RP@ was executed.

S->D [STOD] (n -- d) SIGNED
Sign-extend a single number, leaving a signed double number.
Could be defined as **: S->D DUP 0< NEGATE ;**
1 S->D D. (1)
-1 S->D D. (-1)
See **D+ D.R DABS** and **DNEGATE**

SETBITS [SETBITS] (addr val --)
Logical **OR** the value-bits with the contents of addr using read, modify, write method.
(Create and initialize variable **TEST_SETBITS**)
VARIABLE TEST_SETBITS 110b TEST_SETBITS !
(Use **SETBITS** to modify **TEST_SETBITS**)
TEST_SETBITS 10001b SETBITS
(View results)
TEST_SETBITS @ .B (10111)
See **CLRBITS** and **ANDBITS**

SIGN [SIGN] (n d -- d)
Occurs in a colon-definition in the form:
<# # n HOLD #S SIGN #>
Place an ASCII "-" sign just before a converted numeric output string in the text output buffer when n is negative. n is discarded but double number d is maintained. Must be used between **<#** and **#>**.
(In this example 24h is ASCII $ and 2Eh is . <Period>)
: .SIGNEDDOLLARS (n d --)
<# # # 2Eh HOLD #S 24h HOLD SIGN #> TYPE ; (PRESS ENTER)
-1 56668. .SIGNEDDOLLARS (-$566.68)
See **<# # #S #>** and **HOLD**

SP@ [SPAT] (-- addr)
Return the address of the current stack position to the top of the stack, as it was before SP@ was executed.

SPACE [SPACE] (--)
EMIT a space character, ASCII 20h.

SPACES [SPACES] (n --)
Transmit n ASCII blank spaces (20h) to the output device.
8 SPACES (Moves cursor 8 spaces)

STATE [STATE_SV] (-- addr of **CSTATE**)
A system variable (**CSTATE**) containing the compilation state. A non-zero value indicates compilation. The value itself will be 0C0h while compiling and zero when interpreting. 0C0h must be used to execute **IMMEDIATE** words while compiling colon definitions.

STCTR [STCTR] (-- addr of **STCTR**)
A system variable (**STCTR**) containing a value incremented by the ARM SYSTICK 24bit counter interrupt handler.
See **DELAY MS SYSTICK_IRQ_ON SYSTICK_IRQ_OFF** and **SYSCLK**

SYSTICK_IRQ_OFF [SYSTICK_IRQ_OFF] (–) 0 0 **DELAY** is equivalent.
Turn SYSTICK Interrupt off. This will stop incrementing **STCTR**.
See **SYSTICK_IRQ_ON DELAY MS STCTR** and **SYSCLK**
This is not in the 812 Small Reference Model (SRM) Wordset.

SYSTICK_IRQ_ON [SYSTICK_IRQ_ON] (–) **MS** and **DELAY** turns this on.
Turn SYSTICK Interrupt on. This will resume incrementing **STCTR**.
See **SYSTICK_IRQ_OFF DELAY MS STCTR** and **SYSCLK**
This is not in the 812 Small Reference Model (SRM) Wordset.

SWAP [SWAP] (xl x2 -- x2 x1)
Exchange the top two items on the stack.
1 2 .S (TOS> 2 1)
SWAP .S (TOS> 1 2)
See **DUP DROP ROT** and **OVER**

SYSCLK [SYSCLK] (-- value)
Return the system clock frequency in hertz.
SYSCLK .D (Prints out system clock frequency in decimal)
See **DELAY MS** and **STCTR**

THEN [THEN] (--) **IMMEDIATE**
An alias for **ENDIF**.
See **IF ELSE** and **ENDIF**

TIB [TIB_SV] (-- addr of **TIB**)
Leave the address of the system text input buffer **TIB** on the stack. Typically **TIB** is at least 82 bytes long.
See I**N EXPECT** and **WORD**

TYPE [TYPE] (addr count --)
EMIT count bytes in a string (only ASCII characters from 0 to 07Fh) from addr thru addr+count. **OUT** is incremented. **EMIT** allows character values from 0 to 0FFh to be output, but **TYPE** does not.

(Print FISH NFA, whose count has extra bits, so clear then to get count)
' FISH NFA COUNT 1Fh AND TYPE (Prints ***FISH NFA***)

See **ID. COUNT $LEN OUT** and **EMIT**

U. [UDOT] (value --) UNSIGNED
Print out value in current **BASE** as an unsigned value.

HEX -2 . (-2)
-2 U. (FFFFFFFE)
DECIMAL

See **D.** and **.**

U< [ULESSTHAN] (u1 u2 -- f) UNSIGNED
Leave a true flag if u1 is lower than u2, otherwise leave a false flag.

-1 -2 U< . (0)
-1 2 U< . (0)
1 2 U< . (1)

See **<**

UARTx_INIT [UARTx_INIT] (--) **x** is board dependent.
Change Baud Rate of the terminal according to value set with **MYBAUD**, using 8 bit characters, no parity and 1 Stop Bit, with the software flow control mechanism named **XON XOFF**. To change the baud rate, **MYBAUD** must be used first. This is to preserve your baud during any reset. **UARTx_INIT** will set the baud rate to the system **DEFAULT_BAUD** unless **MYBAUD** is used.
See **MYBAUD**

UARTx_RX [UARTx_RX] (-- addr) **x** is board dependent.
UART RX register. This is the serial terminal character input register.
See **UARTx_RX** and **UARTx_LSR**

UARTx_TX [UARTx_TX] (-- addr) **x** is board dependent.
UART TX register. This is the serial terminal character output register.
See **UARTx_RX** and **UARTx_LSR**

UARTx_LSR [UARTx_LSR] (-- value) **x** is board dependent.
UART Line Status Register. This is the serial terminal status register. Returns bits indicating character available, ready to transmits and error conditions.
See **UARTx_TX** and **UARTx_RX**

UNTIL [UNTIL] (f --) **IMMEDIATE**
Occurs within a colon-definition in the form:
BEGIN ... f UNTIL
At run-time, **UNTIL** controls the conditional branch back to the corresponding **BEGIN**. If f is false, execution returns to just after **BEGIN**; if true, execution continues ahead to the next word after **UNTIL**.
(**?KEY** returns zero (false flag) until a key is pressed, then a true flag)
: TestUntil BEGIN ?KEY UNTIL ; (will execute until a key is pressed }
KEY DROP (Remove the key pressed)
See **BEGIN** AND **AGAIN**

UP [UP_SV] (-- addr of **UP**) ALIGNED
A system variable (**UP**), the **VARIABLE** and **VARALLOT** pointer, which contains the address of the next free memory available for **VARIABLE** and **VARALLOT**. The value is read by **.VS** and altered by **VARIABLE** and **VARALLOT**.
See **.VS**

VARIABLE [VARIABLE] (--) **VARIABLE and VAR** defined.
A defining word used in the form:
VARIABLE Cccc
When **VARIABLE** is executed, it creates the definition Cccc with its cell space un-initialized. When Cccc is executed, its cell space address is put on the stack, and then used by **@** or *!*. If the memory space for variables is full, **VARIABLE** will print an error message and **ABORT**.
NOTE: **VARIABLE**'s cell space is allocated in RAM.
VARIABLE MyVar (Create a variable)
7 MyVar ! (Initialize MyVar to 7)
MyVar @ . (7)
See **.VS UP VBASE** and **VARALLOT**

VARALLOT [VARALLOT] (n -- addr) ALIGNED
Add n * 32 bits to the systems ram variable space pointer **UP**. Used to allocate space in RAM for **VARIABLE**'s and other data structures. If the memory space for variables is full, **VARALLOT** will print an error message and **ABORT**.
See **.VS** and **VBASE**

VBASE [VBASE] (-- addr)
Return the starting address of the Ram Variable Space.
(Print the base address of the ram where variables are stored)
VBASE @ .H
See **.VS** and **UP**

WHILE [WHILE] (f --) **IMMEDIATE**
Occurs in a colon-definition in the form:
BEGIN ... WHILE ... REPEAT
At run-time, **WHILE** selects conditional execution based on boolean flag f. If f is true (non-zero), **WHILE** continues execution of the true part thru to **REPEAT**, which then branches back to **BEGIN**. If f is false (zero), execution skips to just after **REPEAT**, exiting the structure.
: BeginTest (0 1 --) BEGIN ." Begin " WHILE ." While " REPEAT CR ;
(The argument are successively used by **WHILE**)
0 1 BeginTest (Begin While Begin)
1 0 BeginTest (Begin)
See **BEGIN** and **REPEAT** :

WORD [WORD] (c --)
Parse the text in **TIB** , until a delimiter c is found. Move the character string to **HERE**, with a count byte and 2 nulls at the end. Leading occurrences of c are ignored. **IN** is incremented (**IN** is initialized to 1 to skip count byte). Usually used inside a definition.
: UseWORD 0Dh WORD CR ." Your word is " HERE COUNT TYPE CR ;
UseWORD MyWord! (Prints Your word is MyWord!)
See **IN**

WORDCAT [WORDCAT] (--)
Creates a Word Category **NFA** and **LFA** that cannot be searched for, but is displayed by **WORDS** and **MYWORDS** to label a group of Words. You can define a group of Words and then add a category name with **WORDCAT**. Spaces are allowed in Word Categories:

: MyAPP ;
WORDCAT MY APP WORDS:
(**MY APP WORDS:** will be printed at the beginning of a new line)
(**MyAPP** will be indented on the next line)
MYWORDS (Prints **MY APP WORDS: then MyAPP)**
WORDS (Prints **MY APP WORDS: MyAPP** and then **FISH** words)

WORDCAT must be on a line of it's own. The colon at the end is the **FISH** convention for **WORDCAT**'s.

WORDS [WORDS] (--) RENAMED --> VLIST to **WORDS**
List the names of all the definitions in the dictionary.
See **MYWORDS**

XOFF [XOFF] (--)
Send ASCII XOFF character to the terminal.
See **XON**

XON [XON] (--)
Send ASCII XON character to the terminal.
See **XOFF**

XOR [XOR] (xl x2 -- x)
Leave the bitwise logical exclusive-or of x1 and x2 as x.

4 6 XOR . (2)
3 0 XOR . (3)

See **AND NOT ASR LSR** and **LSL**

[[LBRAC] (--)
Used in a colon-definition in form:

: xxx [words] more ;

Suspend compilation by setting **STATE** to zero (0C0h is compiling). The words after [are executed, not compiled. This allows calculation or compilation exceptions before resuming compilation with] .

: Cccc [3 2 *] LITERAL ; (***Cccc*** computes 6 while compiling!)
Cccc . (6)

See **STATE LITERAL** and **]**

\ [BACKSLASH] (--)
After a trailing space treat the rest of the line as a comment.
See PAREN **(**

[COMPILE] [BCOMP] (--) **IMMEDIATE**
Compile next word in definition even if it is an **IMMEDIATE** word that would otherwise execute during compilation. Used in a colon-definition in the form:

: Cccc [COMPILE] FISH ;
Cccc *(Acts like FISH i.e. Prints what FISH prints)*

See **[** and **]**

] [RBRAC] (--) **IMMEDIATE**
Resume compilation, by setting **STATE** back to 0C0h (zero is interpreting). Setting **STATE** to 0C0h is used in **INTER** to execute **IMMEDIATE** words while compiling.

: Cccc [3 2 *] LITERAL ; (***Cccc*** computes 6 while compiling!)
Cccc . (6)

See **[** and **STATE**

---------- *:NONAME (i.e. header-less) words* ----------

This is a partial list of **FISH** internal routines in FLASH with the **FISH** kernel. A separate file may be supplied with more information. This is not complete as some routines are tied to internal conventions. The routines listed have no Name Field Address (**NFA**) or Link Field Address (**LFA**) and therefore are not seen in the FISH dictionary when viewed by **WORDS**.

The names in BOLD in this section are to be found in the symbol table file. Associated with each name will be a number. That number is the **CFA** address of the routine, for use with **DUMP** and **EXECUTE**.

To use a :NONAME word look up it's **CFA** address in the *.sym* file, for example BELL is listed in the .sym file as follows: (Check your .sym file for the address!!)
000037F0 t BELL
To us the above information to make a words for use in **FISH** do as follows:
: Bell 37F0h EXECUTE ; *(BELL could be used without conflict)*

:NONAME_names are in **BOLD TYPE** in this section.

Format of this section:
noNFAname (--) As seen in .sym file.

BELL (--) **EMIT** 07h = terminal BELL sound.
BSOUT (--) **EMIT** 08h = terminal Backspace.
SIGNON (--) print signon message.
DICTSPACE (-- n) calculate and push the available dictionary space.
RVSPACE (-- n) calculate and push RAM **VARIABLE**'s available space.
FPADDR (-- addr) contains addr of next free flash page.
FPCURR (-- addr) contains flash_save's flash page DP.
FPUSER (-- addr) contains flash_save's flash page UP. (**VARIABLE**'s)

LIT (-- n) compiler internal
Within a colon-definition, **LIT** is automatically compiled before each 32bit literal number encountered in input text. Later execution of **LIT** causes the contents of the next dictionary address (the **LITERAL** item) to be pushed to the stack.
See **LITERAL**

BRAN (--) COMPILER INTERNAL
The run-time procedure to unconditionally branch. An in-line offset is added to the interpretive pointer IP to branch ahead or back. Compiled by **ELSE**, **AGAIN**, **REPEAT**.

ZBRAN (f --) COMPILER INTERNAL
The run-time procedure to conditionally branch. If f is false (zero), the following in-line parameter is added to the interpreter pointer to branch ahead or back.

Compiled by **IF**, **UNTIL**, and **WHILE**.

PDOTQ (--) COMPILER INTERNAL
The run-time procedure compiled by **."** which transmits the following in-line text to the output device.
See **."**

XPLOOP (n --) COMPILER INTERNAL
The run-time procedure compiled by **+LOOP**, which increments the loop index by n and tests for loop completion.
See **+LOOP**

XDO (Limit Index --) COMPILER INTERNAL
(Limit = addr+cnt Index = addr --)
The run-time procedure compiled by **DO** which moves the loop control parameters to the return stack.
See **DO**

XLOOP (--) COMPILER INTERNAL
The run-time procedure compiled by **LOOP** which increments the loop index and tests for loop completion.
See **LOOP**

PNUMBER (0 0 addr1 -- d addr2)
(0 0 addr1 -- LSW MSW addr2=addr1+chars)
Convert the ASCII text beginning at addr1+l with regard to **BASE** into a 64-bit unsigned number d. addr2 is the address of the first un-convertible digit, usually a space (20h) or **NULL**. Used by **NUMBER**.

PFIND (addr1 addr2 -- pfa b tf) (ok)
(addr1 addr2 -- ff) (bad)
Searches the dictionary starting at the name field address addr2, matching to the text at addr1. Returns the Parameter Field Address (**PFA**), length byte (+80h) of the name field, and a Boolean true flag for a good match. If no match is found, only a Boolean false flag is left.
See **DFIND**

;CREATE (--) Error handler during word compilation.
Used in **; CONSTANT** and **VARIABLE** to reset **CSDP** System **VARIABLE** which is used to support a feature called Auto Forget, that forgets words that have errors during compilation, reclaiming dictionary space.

GOTO (cfa --) Used in **DOES>**
Redirect threaded execution to CFA in another hi-level word.
>>> CAUTION!!! **GOTO** is a system word, that is exposed only for use by the

bold and brave! Use it only at your own risk. The stack and return stack must be in the same level of nesting where you jump from to where you go to!

NEXT (--)
This is the inner interpreter that uses the interpretive pointer IP to execute compiled **FISH** definitions. It is not directly executed. It jumps to the address pointed to by the address pointed to by the 'W' CPU register. W points to the code field of a definition which contains the address of the code which executes for that definition. This usage of indirect threaded code is a major contributor to the power, portability, and extensibility of **FISH**. Locations of IP and W are computer-specific.

QEXEC (--)
Issue an error message if not executing.

QPAIR (n1 n2 --)
Issue an error message if n1 does not equal n2. A message is issued when compiler conditionals do not match.

QSTACK (--)
Issue an error message and **ABORT** if the stack is out of bounds according to the value in **INITSO**.

BACK (addr --) ALIGNED
Calculate the backward branch offset from **HERE** to addr and compile it into the next available dictionary memory address.

CSP_SV (-- addr of **CSP**)
A system variable temporarily storing the stack pointer position in **CSP**, for compilation error checking.

HLD_SV (-- addr of **NHLD**)
A system variable that holds the address of the latest character of text created during numeric output conversion.

INTERPRET (--)
The outer text interpreter which sequentially executes or compiles words or numbers from the input stream depending on the system variable **STATE**. If the input text cannot be found after searching the dictionary, starting at **CURRENT**, conversion to a number according to the current base is attempted. If successful, a 32bit number will be placed on the stack. If a decimal point is found in the number, a double number will be placed on the stack, and the decimal point position is saved in **DPL**, until another number is entered. If input is not a word or number, an error message will issued and **ABORT** executed.
See **DFIND WORD NUMBER DPL** and **NULL**

QUIT (--)
Clears the return stack, stops compilation, and returns control to the interpreter, **INTER**. No message is given.

INITRO_SV (-- addr of INITRO)
Return the address of the system variable that contains the initial value of the return stack pointer, I**NITRO**.
See **RPSTO**

RPSTO (--)
Initialize the return stack pointer from system variable **INITR0**. Pronounced (RP-store).

INITSO_SV (-- addr of INITSO)
Return the address of the system variable that contains the initial value for the parameter stack pointer, **INITSO**.
See **SPSTO**

SPSTO (--)
Initialize the stack pointer from the system variable INITSO. Pronounced "SP-store".

TOGGLE (addr b --)
Complement **(FLIP)** the byte-contents of addr by the bit-pattern b.

TRAVERSE (addr1 n -- addr2)
Move across a variable-length Name Field in the dictionary. addr1 is the address of either the length byte, or the last letter. If n=1, the motion is toward NFA/hi memory. If n=-l, the motion is toward CFA/low memory. The addr2 resulting, is the address of the other end of the name.

These are not **CFA** labels:
TXRDY_SUBR:
XOFF_SUBR:
XON_SUBR:
Appendix: *FISH differences from other Forths:*

Not supported:
(;CODE) PSCOD
USER, VOC-LINK, CONTEXT, VOCABULARY, DEFINITIONS, and all BLOCK/SCREEN SUPPORT.

Renamed words:
MINUS is now **NEGATE**
DMINUS is now **DNEGATE**
VLIST is now **WORDS**
?TERMINAL is now **?KEY**
VAR and **VARIABLE** are both defined.
CON and **CONSTANT** are both defined.

Modified words:
KEY EMIT's **XON** to assure input is available.
EXECUTE EMIT's **XOFF** to allow for long winded words!
: error handling of failed word compilation resets DP to the end of last valid word defined, reclaiming space.
<BUILDS ... DOES> is the same as **CREATE … DOES>**