ISOMAX GLOSSARY

Stack comments use the following notation:

n	a signed 16-bit value, -32768..+32767.
u	an unsigned 16-bit value, 0..65535.
+n	a signed, positive 16-bit value, 0..+32767.
w	a generic16-bit value.
16b	a generic 16-bit value.
addr	an address (16 bits).
c	a character. (Note: stored as 16 bits on the IsoPod™)
8b	a generic 8-bit value. (Note: stored as 16 bits on the IsoPod™)
d	a signed 32-bit value, -2,147,483,648..+2,147,483,647.
ud	an unsigned 32-bit value, 0..4,294,967,295.
wd	a generic 32-bit value.
32b	a generic 32-bit value.
r	a floating-point (real) value.
flag	a logical flag, zero = false, -1 (all ones) = true.

Values on the stack before and after execution of a word are given as follows:

(before --- after)	normal integer data stack
(F: before --- after)	floating-point data stack
(C: before --- after)	compile-time behavior of the integer data stack.

Stack comments in italics also refer to compile-time behavior.

1. Integer Arithmetic

Word	**Stack Effect**	**Description**
*	(w1 w2 --- w3)	Multiplies w2 by w1 and leaves the product w3 on the stack.
*/	(n1 n2 n3 --- n4)	Multiplies n2 by n1 and divides the product by n3. The quotient, n4 is placed on the stack.
*/MOD	(n1 n2 n3 -- n4 n5)	n1 is multiplied by n2 producing a product which is divided by n3. The remainder, n4 and the quotient, n5 are then placed on the stack.
+	(w1 w2 --- w3)	Adds w2 and w1 then leaves the sum, w3 on the stack.
+!	(w1 addr ---)	Adds w1 to the value at addr then stores the sum at addr replacing its previous value.
-	(w1 w2 --- w3)	Subtracts w2 from w1 and leaves the result, w3 on the stack.
/	(n1 n2 --- n3)	Divides n1 by n2 and leaves the quotient n3 on the stack.
/MOD	(n1 n2 --- n3 n4)	Divides n1 by n2 then leaves on the stack the remainder n3 and the quotient n4.
1+	(w1 --- w2)	Adds 1 to w1 then leaves the sum, w2 on the stack.
1+!	(addr ---)	Adds one to the value at addr and stores the result at addr.
1-	(w1 --- w2)	Subtract 1 from w1 then leaves the difference, w2 on the stack.
1-!	(addr ---)	Subtracts one from the value at addr and stores the result at addr.
2*	(w1 --- w2)	Multiplies w1 by 2 to give w2.
2+	(w1 --- w2)	Adds two to w1 and leaves the sum, w2 on the stack.
2-	(w1 --- w2)	Subtracts two from w1 and leaves the result, w2 on the stack.

2/	(n1 --- n2)	Divides n1 by 2, giving n2 as the result.
><	(8b1/8b2 --- 8b2/8b1)	Swaps the upper and lower bytes of the value on the stack.
ABS	(n --- u)	Leaves on the stack the absolute value, u of n.
MAX	(n1 n2 --- n3)	Leaves the greater of n1 and n2 as n3.
MIN	(n1 n2 --- n3)	Leaves the lesser of n1 and n2 as n3.
MOD	(n1 n2 --- n3)	Divides n1 by n2 and leaves the remainder n3.
NEGATE	(n1 --- n2)	Leaves the two's complement n2 of n1.
UM*	(u1 u2 ---ud)	Multiplies u1 and u2 returning the double length product ud.
UM/MOD	(ud u1 --- u2 u3)	Divides the double length unsigned number ud by u1 and returns the single length remainder u2 and the single length quotient u3.

2. Logical and Comparison

Word	**Stack Effect**	**Description**
0<	(n --- flag)	Leaves a true flag if n is less than zero.
0=	(w --- flag)	Leaves a true flag if w is equal to zero.
0>	(n --- flag)	Leaves a true flag if n is greater than zero.
<	(n1 n2 --- flag)	Leaves a true flag on stack if n1 is less than n2.
=	(w1 w2 --- flag)	Returns a true flag if w1 is equal to w2.
>	(n1 n2 --- flag)	Returns a true flag if n1 is greater than n2.
AND	(16b1 16b2 --- 16b3)	Leaves the bitwise logical AND of 16b1 and 16b2 as 16b3.
CLEAR-BITS		Clears bits at addr corresponding to 1s in mask b.
INVERT	(16b1 --- 16b2)	Leaves the one's complement 16b2 of 16b1.
NOT	(flag1 --- flag2)	Leaves the logical inverse flag2 of flag1. flag2 is false if flag1 was true, and vice versa.
OR	(16b1 16b2 --- 16b3)	Leaves the inclusive-or 16b3 of 16b1 an 16b2.
SET-BITS	(b addr ---)	Sets bits at addr corresponding to 1s in mask b.
TOGGLE-BITS	(b addr ---)	Toggles bits at addr corresponding to 1s in mask b.
U<	(u1 u2 ---flag)	Returns a true flag if u1 is less then u2.
XOR	(16b1 16b2 --- 16b3)	Performs a bit-by-bit exclusive or of 16b1 with 16b2 to give 16b3.

3. Double-Precision Operations

Word	**Stack Effect**	**Description**
2CONSTANT <name>	(32b ---)	Creates a double length constant for a <name>. When <name> is executed, 32b is left on the stack.
2DROP	(32b ---)	Removes 32b from the stack.
2DUP	(32b --- 32b 32b)	Duplicates 32b.
2OVER	(32b1 32b2 --- 32b1 32b2 32b3)	32b3 is a copy of 32b1
2ROT	(32b1 32b2 32b3 --- 32b2 32b3 32b1)	Rotates 32b1 to the top of the stack.
2SWAP	(32b1 32b2 --- 32b2 32b1)	Swaps 32b1 and 32b2 on the stack.
2VARIABLE <name>	(---)	Creates double-length variable for <name>. when <name> is executed, its parameter field address is placed on the stack.
D*	(d1 d2 --- d3)	Multiplies d1 by d2 and leaves the product d3 on the stack.
D+	(wd1 wd2 --- wd3)	Adds wd1 and wd2 and leaves the result, wd3 on stack.
D-	(wd1 wd2 --- wd3)	Subtracts wd2 from wd1 and returns the dif- ference wd3.
D/	(d1 d2 --- d3)	Divides d1 by d2 and leaves the quotient d3 on the stack.
D0=	(wd --- flag)	Returns a true flag if wd is equal to zero.
D2/	(d1 --- d2)	Divides d1 by 2 and gives quotient d2.

D<	(d1 d2 --- flag)	Leaves a true flag if d1 is less than d2; otherwise leaves a false flag.
D=	(wd1 wd2 --- flag)	Returns a true flag if wd1 is equal to wd2.
DABS	(d --- ud)	Returns the absolute value of d as ud.
DCONSTANT <name>	(32b ---)	Creates a double length constant for a <name>. When <name> is executed, 32b is left on the stack. Same as 2CONSTANT.
DDROP	(32b ---)	Removes 32b from the stack. Same as 2DROP.
DDUP	(32b --- 32b 32b)	Duplicates 32b. Same as 2DUP.
DMAX	(d1 d2 --- d3)	Returns d3 as the greater of d1 or d2.
DMIN	(d1 d2 --- d3)	Returns d3 as the lesser of d1 or d2.
DMOD	(d1 d2 --- d3)	Divides d1 by d2 and leaves the remainder d3.
DNEGATE	(d1 --- d2)	Leaves the two's complement d2 of d1.
DOVER	(32b1 32b2 --- 32b1 32b2 32b3)	32b3 is a copy of 32b1. Same as 2OVER.
DROT	(32b1 32b2 32b3 --- 32b2 32b3 32b1)	Rotates 32b1 to the top of the stack. Same as 2ROT.
DSWAP	(32b1 32b2 --- 32b2 32b1)	Swaps 32b1 and 32b2 on the stack. Same as 2SWAP.
DU<	(ud1 ud2 --- flag)	Returns a true flag if ud1 is less than ud2.
DVARIABLE <name>	(---)	Creates double-length variable for <name>. when <name> is executed, its parameter field address is placed on the stack. Same as 2VARIABLE.
S->D	(n --- d)	Sign extend single number to double number.

4. Floating-point Operations

Word	**Stack Effect**	**Description**
2**X	(F: r1 -- r2)	Raise 2 to the r1 power giving r2.
D>F	(d --) (F: -- r)	R is the floating-point equivalent of d.
e	(F: -- r1)	Put natural value e (=2.718282) on the floating-point stack as r1.
F!	(addr --) (F:r --)	Store r at addr.
F*	(F:r1 r2 -- r3)	Multiply r1 by r2 giving r3.
F**	(F:r1 r2 -- r3)	Raise r1 to the r2 power giving r3.
F+	(F:r1 r2 -- r3)	Add r1 to r2, giving r3.
F,	(F:r --)	Store r as a floating-point number in the next available dictionary location.
F-	(F:r1 r2 -- r3)	Subtract r2 from r1, giving r3.
F/	(F:r1 r2 -- r3)	Divide r1 by r2, giving r3.
F0<	(F:r --) (-- flag)	flag is true if r is less than zero.
F0=	(F:r --) (-- flag)	flag is true if r is equal to zero.
F2*	(F:r1 -- r2)	Multiply r1 by 2 giving r2.
F2/	(F:r1 -- r2)	Divide r1 by 2 giving r2.
F<	(F:r1 r2 --)(-- flag)	flag is true if r1 is less than r2.
F>D	(F:r --)(-- d)	Convert r to d.
F@	(addr --)(F: -- r)	r is the value stored at addr.
FABS	(F:r1 -- r2)	R2 is the absolute value of r1.
FALOG	(F:r1 -- r2)	Raise 10 to the power r1, giving r2.
FATAN	(F:r1 -- r2)	R2 is the principal radian whose tangent is r1.
FATAN2	(F:r1 r2 -- r3)	R3 is the radian angle whose tangent is r1/r2.
FCONSTANT <name>	(F:r --)	Define a constant <name> with value r.
FCOS	(F:r1 -- r2)	r2 is the cosine of the radian angle r1.
FDEPTH	(-- +n)	+n is the number of values contained on separate floating point stack.

FDROP	(F:r--)	Remove r from the floating-point stack.
FDUP	(F:r -- r r)	Duplicate r.
FEXP	(F:r1 -- r2)	Raise e to the power r1, giving r2.
FLN	(F:r1 -- r2)	R2 is the natural logarithm of r1.
FLOAT+	(addr1 -- addr2)	Add the size of a floating-point value to addr1.
FLOATS	(n1 -- n2)	n2 is the size, in bytes, of n1 floating-point numbers.
FLOG	(F:r1 -- r2)	R2 is the base 10 logarithm of r1.
FLOOR	(F:r1 -- r2)	Round r1 using the "round to negative infinity" rule, giving r2.
FMAX	(F:r1 r2 -- r3)	r3 is the maximum of r1 and r2.
FMIN	(F:r1 r2 -- r3)	r3 is the minimum of r2 and r3.
FNEGATE	(F:r1 -- r2)	r2 is the negation of r1.
FNIP	(F:r1 r2 -- r2)	Remove second number down from floating-point stack.
FOVER	(F:r1 r2 -- r1 r2 r1)	Place a copy of r1 on top of the floating-point stack.
FROUND	(F:r1 -- r2)	Round r1 using the ";round to even"; rule, giving r2.
FSIN	(F:r1 -- r2)	R2 is the sine of the radian angle r1.
FSQRT	(F:r1 -- r2)	R2 is the square root of r1.
FSWAP	(F:r1 r2 -- r2 r1)	Exchange the top two floating-point stack items.
FTAN	(F:r1 -- r2)	R2 is the tangent of the radian angle r1.
FVARIABLE <name>	(--)	Create a floating-point variable <name>. Reserve data memory in the dictionary sufficient to hold a floating-point value.
LOG2	(F:r1 -- r2)	R2 is the base 2 logarithm of r1.
ODD-POLY	(F: -- r1)(addr --)	Evaluate odd-polynomial giving r1.
PI	(F: -- r1)	Put the numerical value of pi on the floating- point stack as r1.
POLY	(F: -- r1)(addr --)	Evaluate polynomial giving r1.
S>F	(n--)(F: -- r)	R is the floating-point equivalent of n.
SF!	(addr --)(F:r --)	Store the floating point number r as a 32 bit IEEE single precision number at addr.
SF@	(addr --)(F: -- r)	Fetch the 32-bit IEEE single precision number stored at addr to the floating-point stack as r in the internal representation.

5. Stack Operations

Word	**Stack Effect**	**Description**
-ROLL	(n ---)	Removes the value on the top of stack and inserts it into the nth place from the top of stack.
>R	(16b ---)	Removes 16b from user stack and place it onto return stack.
?DUP	(16b --- 16b 16b), (0 --- 0)	Duplicates 16b if it is a non-zero.
DEPTH	(--- +n)	Returns count +n of numbers on the data stack.
DROP	(16b ---)	Removes 16b from the data stack.
DUP	(16b --- 16b 16b)	Duplicates 16b.
OVER	(16b1 16b2 --- 16b1 16b2 16b3)	16b3 is a copy of 16b1.
PICK	(+n --- 16b)	Copies the data stack's +nth item onto the top.
R>	(--- 16b)	16b is removed from the return stack and placed onto the data stack.
R@	(--- 16b)	16b is a copy of the top of the return stack.
ROLL	(+n ---)	Removes the stack's nth item and places it onto the top of stack.
ROT	(16b1 16b2 16b3 --- 16b2 16b3 16b1)	Rotates 16b1 to the top of the stack.
RP!	(--)	Initializes the bottom of the return stack.

RP@	(-- addr)	addr is the address of the top of the return stack just before RP@ was executed.
SP!	(--)	Initializes the bottom of the parameter stack.
SP@	(--- addr)	addr is the address of the top of the parameter stack just before SP@ was executed.
SWAP	(16b1 16b2 --- 16b2 16b1)	Exchanges positions of the top two items of the stack.

6. String Operations

Word	Stack Effect	Description
-TRAILING	(addr +n1 --- addr +n2)	Counts +n1 characters starting at addr and subtracts 1 from the count when a blank is encountered. Leaves on the stack the final string count, n2 and addr.
."	(---)	Displays the characters following it up to the delimiter " .
.(	(---)	Displays string following .(delimited by) .
COUNT	(addr1 --- addr2 +n)	Leaves the address, addr2 and the character count +n of text beginning at addr1.
PCOUNT	(addr1 --- addr2 +n)	Leaves the address, addr2 and the character count +n of text beginning at addr1 in Program memory.

7. Terminal I/O

Word	Stack Effect	Description
?KEY	(--- flag)	True if any key is depressed.
?TERMINAL	(--- flag)	True if any key is depressed. Same as ?KEY.
CR	(---)	Generates a carriage return and line feed.
EMIT	(16b ---)	Displays the ASCII equivalent of 16b onto the screen.
EXPECT	(addr +n ---)	Stores up to +n characters into memory beginning at addr.
KEY	(--- 16b)	Pauses to wait for a key to be pressed and then places the ASCII value of the key (n) on the stack.
PTYPE	(addr +n ---)	Displays a string of +n characters from Program memory, starting with the character at addr.
SPACE	(---)	Sends a space (blank) to the current output device.
SPACES	(+n ---)	Sends +n spaces (blanks) to the current output device.
TYPE	(addr +n ---)	Displays a string of +n characters starting with the character at addr.

8. Numeric Output

Word	Stack Effect	Description
#	(+d1 --- +d2)	+d1 is divided by BASE and the quotient is placed onto the stack. The remainder is converted to an ASCII character and appended to the output string toward lower memory addresses.
#>	(32b --- addr +n)	Terminates formatted (or pictured) output string (ready for TYPE).
#S	(+d --- 0 0)	Converts all digits of an entire number into string.
(E.)	(F:r --)(-- addr +n)	Convert the top number on the floating-point stack to its character string representation using the scientific notation. Addr is the address of the location where the character string representation of r is stored, and +n is the number of bytes.
(F.)	(F:r --)(-- addr +n)	Convert the top number on the floating-point stack to its

		character string representation using the fixed-point notation. Addr is the address of the location where the character string representation of r is stored, and +n is the number of bytes.
.	(n ---)	Removes n from the top of stack and displays it.
.R	(n +n ---)	Displays the value n right justified in a field +n characters wide according to the value of BASE.
<#	(---)	Starts a formatted (pictured) numeric output. Terminated by #> .
?	(addr ---)	Displays the contents of addr.
BASE	(--- addr)	Leaves the address of the user variable containing the numeric numeric conversion radix.
D.	(d ---)	Displays the value of d.
D.R	(d +n ---)	Displays the value of d right justified in a field +n characters wide.
DECIMAL	(---)	Sets the input-output numeric conversion base to ten.
E.	(--)(F:r --)	Convert the top number on the floating-point stack to its character string representation using the scientific notation.
F.	(F:r --)(--)	Print the top number on the floating-point stack on the screen using fixed-point notation.
F?	(addr --)	Display the floating-point contents stored at addr.
HEX	(---)	Sets the numeric input-output conversion base to sixteen.
HOLD	(char ---)	Inserts character into a pictured numeric out- put string.
PLACES	(n ---)	Set the number of decimal places (digits to the right of the radix point) displayed by E. and F.
SIGN	(n ---)	Appends an ASCII "; - "; (minus sign) to the start of a pictured numeric output string if n is negative.
U.	(u ---)	Displays the unsigned value of u followed by a space.
U.R	(u +n ---)	Displays the value of u right justified in a field +n characters wide according to the value of BASE.

9. Numeric Input

Word	Stack Effect	Description
CONVERT	(+d1 addr1 --- +d2 addr2)	Converts an input string into a number.
FNUMBER	(+d1 addr1 -- +d2 addr2)	Converts an input string into a number.
NUMBER	(addr --- d)	Converts the counted string at addr to d according to the value of BASE .

10. Memory Operations

Word	Stack Effect	Description
!	(16b addr ---)	Stores 16b at addr.
2!	(32b addr ---)	Stores 32b at addr.
2@	(addr --- 32b)	Returns 32b from addr.
@	(addr --- 16b)	Replaces addr with its 16b contents on top of the stack.
@!	(16b addr ---)	Stores 16 at address pointed to by addr.
@@	(addr --- 16b)	Replaces addr with 16b, 16b is contents of address pointed to by addr.
BLANK	(addr u ---)	Sets u bytes of memory beginning at addr to the ASCII code for space (decimal 32).

C!	(c addr ---)	Stores the character c into addr.
C@	(addr --- c)	Fetches the character c contents from addr.
CMOVE	(addr1 addr2 u ---)	Moves towards high memory the u bytes at ad- dresses addr1 and addr2.
CMOVE>	(addr1 addr2 u ---)	Moves u bytes beginning at addr1 to addr2.
D!	(32b addr ---)	Stores 32b at addr. Same as 2!
D@	(addr --- 32b)	Returns 32b from addr. Same as 2@
EE!	(16b addr ---)	Stores 16b into addr in EEPROM.
EEC!	(16b addr ---)	Stores the least significant byte of 16b into addr in EEPROM.
EEMOVE	(addr1 addr2 u ---)	Moves towards high memory the u bytes at addresses addr1 and addr2. addr2 should be in EEPROM.
EEERASE	(addr ---)	Erase one page of Data Flash memory at addr.
ERASE	(addr u ---)	Sets u bytes of memory to zero, beginning at addr.
EXCHANGE	(w1 addr --- w2)	Fetches contents w2 from addr, then stores w1 at addr. (Exchanges w1 for w2 at addr.)
FILL	(addr u c ---)	Fills u bytes, beginning at addr, with byte pattern c.
P!	(16b addr ---)	Stores 16b into Program memory at at addr.
P@	(addr --- 16b)	Fetches the 16b contents from Program memory at addr.
PC!	(c addr ---)	Stores the character c into Program memory at addr.
PC@	(addr --- c)	Fetches the character c contents from Program memory at addr.
PF!	(16b addr ---)	Stores 16b into addr in Program Flash ROM.
PFERASE	(addr ---)	Erase one page of Program Flash memory at addr.
PFMOVE	(addr1 addr2 u ---)	Moves the u locations from Program RAM at addr1, to Program Flash at addr2.
TOGGLE	(addr b --)	Toggles setting of bits with mask b at addr.

11. Memory Allocation

Word	**Stack Effect**	**Description**
,	(16b ---)	Stores 16b into a word at the next available dictionary location.
?AVAIL	(---)	Prints an error message if insufficient RAM or Flash memory space is available.
ALLOT	(w ---)	Reserves w bytes of dictionary space.
AVAIL	(--- n)	Returns number of locations remaining in Data RAM memory.
C,	(c ---)	Stores the character c into a byte at the next available dictionary location.
CELL+	(addr1 --- addr2)	Add the size of one cell to addr1, giving addr2.
EEAVAIL	(--- n)	Returns number of locations remaining in EEPROM (Data Flash) memory.
EXRAM	(---)	Enable external RAM. (for future use)
FLOAT+	(addr1 --- addr2)	Add the size of one floating-point number to addr1, giving addr2.
FLOATS	(n1 --- n2)	Returns the number of memory locations n2 used by n1 floating-point numbers.
HERE	(--- addr)	Leaves the address of the next available dictionary location.
P,	(w ---)	Stores 16b into a word at the next available location in Program memory.
PALLOT	(n ---)	Reserves n bytes of dictionary space in Program memory.
PAVAIL	(--- n)	Returns number of locations remaining in Program RAM memory.
PC,	(c ---)	Stores the character c into a byte at the next available location in Program memory.

PF,	(n ---)	Stores 16b into a word at the next available location in Program Flash ROM.
PFAVAIL	(--- n)	Returns number of locations remaining in Program Flash memory.
PHERE	(--- addr)	Leaves the address of the next available dictionary location in Program memory.

12. Program Control

Word	**Stack Effect**	**Description**
+LOOP	(n ---) *(C: sys ---)*	Increments the DO LOOP index by n.
AGAIN	(---) *(C: sys ---)*	Affect an unconditional jump back to the start of a BEGIN-AGAIN loop.
BEGIN	(---) *(C: --- sys)*	Marks the start of a loop.
DO	(w1 w2 ---) *(C: --- sys)*	Repeats execution of words between DO LOOPs and DO +LOOPs, the number of times is specified by the limit from w2 to w1.
ELSE	(---) *(C: sys1 --- sys2)*	Allows execution of words between IF and ELSE if the flag is true, otherwise, it forces execu- tion of words after ELSE.
END	(flag ---) *(C: sys ---)*	Performs the same function as UNTIL . See UNTIL .
EXECUTE	(addr ---)	Executes the definition found at addr.
EXIT	(---)	Causes execution to leave the current word and go back to where the word was called from.
I	(--- w)	Places the loop index onto the stack.
IF	(flag ---) *(C: --- sys)*	Allows a program to branch on condition.
J	(--- w)	Returns the index of the next outer loop.
K	(--- w)	Returns the index of the second outer loop in nested do loops.
LEAVE	(---)	Forces termination of a DO LOOP.
LOOP	(---) *(C: sys ---)*	Defines the end point of a do-loop.
REPEAT	(---) *(C: sys ---)*	Terminates a BEGIN...WHILE...REPEAT loop.
THEN	(---) *(C: sys ---)*	Marks the end of a conditional branch or marks where execution will continue relative to a cor- responding IF or ELSE .
UNTIL	(flag ---) *(C: sys ---)*	Marks the end of an indefinite loop.
WHILE	(flag ---) *(C: sys1 --- sys2)*	Decides the continuation or termination of a BEGIN...WHILE...REPEAT loop.

13. Compiler

‘ <name>	(--- addr)	Returns <name>'s compilation address, addr.
(	(---)	Starts a comment input. Comment is ended by a) .
: <name>	(--- sys)	Starts the definition of a word <name>. Definition is terminated by a ; .
:CASE	(n ---) *(C: --- sys)*	Creates a dictionary entry for <name> and sets the compile mode.

;	(sys ---)	Terminates a colon-definiton.
;CODE	(---) *(C: sys1 --- sys2)*	Terminates a defining-word. May only be used in compilation mode.
AUTOSTART <name>	(addr ---)	Prepare autostart vector at addr which will cause <name> to be executed upon reset. Note: addr must be on a 1K address boundary.
CODE	(--- sys)	Creates an assembler definition.
CODE-INT	(--- sys)	Creates an assembler definition interrupt routine.
CODE-SUB	(--- sys)	Creates an assembler definition subroutine.
COMPILE	(---)	Copies the compilation address of the next non- immediate word following COMPILE.
CONSTANT <name>	(16b ---)	Creates a dictionary entry for <name>.
DOES>	(--- addr) *(C: ---)*	Marks the termination of the defining part of the defining word <name> and begins the definition of the run-time action for words that will later be defined by <name>.
EEWORD	(---)	Moves code of last defined word from the Program RAM memory to the Program Flash memory.
END-CODE	(sys ---)	Terminates an assembler definition.
FORGET <name>	(---)	Deletes <name> from the dictionary.
IMMEDIATE	(---)	Marks the most recently created dictionary entry as a word that will be executed immediately even if FORTH is in compile mode.
IS <name>	(16b ---)	Creates a dictionary entry <name> for the constant value 16b. Same as CONSTANT.
RECURSE	(---)	Compile the compilation address of definition currently being defined.
UNDO	(---)	Forget the latest definition regardless of smudge condition.
USER <name>	(n ---)	Create a user variable.
VARIABLE <name>	(---)	Creates a single length variable.
\	(---)	Starts a comment that continues to end-of-line.

14. Compiler Internals

Word	**Stack Effect**	**Description**
;S	(---)	Stop interpretation.
<BUILDS	(---)	Creates a new dictionary entry for <name> which is parsed from the input stream.
<MARK	(--- addr)	Leaves current dictionary location to be resolved by <RESOLVE .
<RESOLVE	(addr ---)	Compiles branch offset to location previously left by <MARK .
>BODY	(addr1 --- addr2)	Leaves on the stack the parameter field address, addr2 of a given field address, addr1.
>MARK	(--- addr)	Compiles zero in place of forward branch offset and marks it for future resolve.
>RESOLVE	(addr ---)	Corrects branch offset previously compiled by >mark to current dictionary location.
?BRANCH	(flag ---)	Compiles a conditional branch operation.
?COMP	(--)	Checks for compilation mode, gives error if not.
?CSP	(--)	Checks for stack integrity through defining process, gives error if not.
?ERROR	(flag n --)	If flag is true, error n is initiated.
?EXEC	(--)	Checks for interpretation mode, gives error if not.

?PAIRS	(n1 n2 --)	Checks for matched structure pairs, gives error if not.
?STACK	(---)	Checks to see if stack is within limits, gives error if not
[	(---)	Places the system into interpret state to execute non-immediate word/s during compilation.
[']	(--- addr) *(C: ---)*	Returns and compiles the code field address of a word in a colon-definition.
[COMPILE]	(---)	Causes an immediate word to be compiled.
]	(---)	Places the system into compilation state.] places a non-zero value into the user variable STATE.
ATO4	(--- n)	Returns address of subroutine call to high level word as indicated in R0 register.
BRANCH	(---)	Compiles an unconditional branch operation.
CFA	(pfa --- cfa)	Alter parameter field pointer address to code field address.
CREATE <name>	(---)	Creates a dictionary entry for <name>.
DLITERAL	(32b ---)	Compile a system dependent operation so that when later executed, 32b will be left on the stack.
FIND	(addr1 --- addr2 n)	Obtains an address of counted strings, addr1 from the stack. Searches the dictionary for the string.
FLITERAL	(F:r --)	Compile r as a floating point literal.
INTERPRET	(---)	Begins text interpretation at the character indexed by the contents of >IN relative to the block number contained in BLK, continuing until the input stream is exhausted.
LATEST	(--- nfa)	Leaves name field address (nfa) of top word in CURRENT.
LFA	(pfaptr --- lfa)	Alter parameter field pointer address to link field address.
LITERAL	(16b ---)	Compile a system dependent operation so that when later executed, 16b will be left on the stack.
NFA	(pfaptr - nfa)	Alter parameter field pointer address to name field address.
PFAPTR	(nfa --- pfaptr)	Alter name field address to parameter field pointer address.
QUERY	(---)	Stores input characters into text input buffer.
SMUDGE	(---)	Toggles visibility bit in head, enabling definitions to be found.
TASK	(---)	A dictionary marker null word.
TRAVERSE	(addr n --- addr)	Adjust addr positively or negatively until contents of addr is greater then $7F.
WORD	(char --- addr)	Generates a counted string until an ASCII code, char is encountered or the input stream is exhausted. Returns addr which is the beginning address of where the counted string are stored.

15. Error Processing

Word	**Stack Effect**	**Description**
ABORT	(---)	Clears the data stack and performs the function of QUIT .
ABORT”	(flag ---) (C: ---)	If flag is true, message that follows "; is dis- played and the ABORT function is performed. If flag is false, the flag is dropped and execu- tion continues.
COLD	(---)	Cold starts FORTH.
ERROR	(--)	Begins error processing.
MESSAGE	(n --)	Prints error message # n.
QUIT	(---)	Clears the return stack, stops compilation and returns control to current input device.

16. System Variables

Word	Stack Effect	Description
#TIB	(--- addr)	Returns the address of the user variable that holds the number of characters input.
>IN	(--- addr)	Leaves the address of the user variable >IN which contains the number of bytes from the beginning of the input stream at any particular moment during interpretation.
BLK	(--- addr)	Leaves the address of the user variable contain- ing the the number of block that is currently being interpreted.
CONTEXT	(--- addr)	Returns the address of a user variable that determines the vocabulary to be searched first in the dictionary.
CURRENT	(--- addr)	Returns the address of the user variable specifying the vocabulary into which new word definitions will be entered.
DP	(--- addr)	Put Dictionary Pointer address on stack.
DPL	(--- addr)	Returns the address of the user variable con- taining the number of places after the frac- tional point for input conversion.
EDELAY	(--- addr)	Put EEPROM programming delay variable onto the stack.
EDP	(--- addr)	Put EEPROM memory pointer onto the stack.
FENCE	(--- addr)	System variable which specifies the highest address from which words may be compiled.
FLD	(--- addr)	Returns the address of the user variable which contains the value of the field length reserved for a number during output conversion.
FSP	(-- addr)	User variable holds floating-point stack pointer.
FSP0	(-- addr)	User variable holds initial value of floating- point stack pointer.
PAD	(--- addr)	Puts onto stack the starting address in memory of scratchpad.
PDP	(--- addr)	System variable which holds the address of the next available Program memory location.
PFDP	(--- addr)	System variable which holds the address of the next available Program Flash memory location.
R0	(-- addr)	Returns the address of the variable containing the initial value of the bottom of the return stack.
S0	(--- addr)	Returns the address of the variable containing the initial value of the bottom of the stack.
seed	(--- addr)	Place the variable on the stack.
SPAN	(--- addr)	Returns the address of the user variable that contains the count of characters received and stored by the most recent execution of EXPECT .
STATE	(--- addr)	Returns the address of the user variable that contains a value defining the compilation state.
TIB	(--- addr)	Returns the address of the start of the text- input buffer.
UABORT	(-- addr)	User variable points to ABORT routine.
WARNING	(--)	User variable controls error handling.

17. System Constants

Word	Stack Effect	Description
B/BUF	(--- n)	Number of characters in a block storage buffer (not used).
BL	(--- 32)	Puts the ASCII code for a space (decimal 32) on the stack.
C/L	(--- n)	Maximum number of characters per line.
FALSE	(--- flag)	Returns a false flag (zero).

ISOMAX	(--- n)	Returns the current IsoMax version number.
TRUE	(--- flag)	Returns a true flag (all bits '1').

18. IsoPod Control

Word	Stack Effect	Description
DINT	(---)	Disable CPU interrupts. *Warning: disables IsoMax and may disable serial I/O.*
EINT	(---)	Enable CPU interrupts.
HALFSPEEDCPU	(---)	Switch IsoPod CPU to 20 MHz clock. All timing functions (baud rate, PWM output, etc.) operate at half speed.
FULLSPEEDCPU	(---)	Switch IsoPod CPU to normal 40 MHz clock.
RESTORE-RAM	(---)	Restores system and user RAM variables from Data Flash.
SAVE-RAM	(---)	Copies system and user RAM variables to Data Flash.
SCRUB	(---)	Erases Data Flash and user's Program Flash, empties the dictionary, and restores system variables to their default values.

19. Debugging

Word	Stack Effect	Description
.S	(---)	Display stack contents without modifying the stack.
DUMP	(addr u ---)	Displays u bytes of data memory starting at addr.
F.S	(--)	Display the contents of the floating-point stack without modifying the stack.
FLASH	(---)	Launch the Flash memory programmer. (unused)
ID.	(nfa ---)	Print <name> given name field address (NFA).
PDUMP	(addr u ---)	Displays u bytes of Program memory starting at addr.
WORDS	(---)	Lists all the words in the CURRENT vocabulary.

20. Object Oriented Programming

Word	Stack Effect	Description
.CLASSES	(---)	Display all defined objects and classes. Same as WORDS.
BEGIN-CLASS <name>	(--- sys1)	Defines a class <name>, and begins the "private" definitions of the class.
END-CLASS <name>	(sys2 ---)	Ends the definition of class <name>.
NO-CONTEXT	(---)	Clears the object context, and hides all private methods.
OBJECT <name>		Defines an object <name> which is a member of the currently active class.
OBJREF	(--- addr)	System variable holding the address of the currently active object.
PUBLIC	(sys1 --- sys2)	Ends the "private" and starts the "public" definitions of the class.
SELF	(--- addr)	Returns the address of the currently active object.

21. IsoMax State Machines

Word	Stack Effect	Description
WITH-VALUE n	(--- sys)	Specifies ‘n’ to be used as tag value to be stored for this state.
AS-TAG	(sys ---)	Ends a tag definition for a state.
END-MACHINE-CHAIN	(sys ---)	Ends definition of a machine chain.
MACHINE-CHAIN <name>	(--- sys)	Starts definition of a machine chain <name>.
.MACHINES	(---)	Prints a list of all INSTALLed machines.
PERIOD	(n ---)	Changes the running IsoMax period to ‘n’ cycles.
ISOMAX-START	(---)	Initializes and starts IsoMax. Clears the machine list and starts the timer interrupt at the default rate of 50000 cycles.
NO-MACHINES	(---)	Clears the IsoMax machine list.
ALL-MACHINES	(---)	Execute, once, all machines on the IsoMax machine list.
UNINSTALL	(---)	Removes the last-added machine from the list of running IsoMax machines.
INSTALL <name>	(---)	Adds machine <name> to the list of running IsoMax machines.
MACHINE-LIST	(--- addr)	System variable pointing to the head of the IsoMax installed-machine list.
SCHEDULE-RUNS <name>	(sys ---)	Specifies that machine chain <name> is to be performed by IsoMax. This overrides the INSTALL machine list.
CYCLES	(--- sys)	Specifies period for SCHEDULE-RUNS; e.g., EVERY n CYCLES SCHEDULE-RUNS name.
EVERY	(--- sys)	Specifies period for SCHEDULE-RUNS; see CYCLES.
STOP-TIMER	(---)	Halts IsoMax by stopping the timer interrupt.
TCFAVG	(--- addr)	System variable holding the average IsoMax processing time.
TCFMIN	(--- addr)	System variable holding the minimum IsoMax processing time.
TCFMAX	(--- addr)	System variable holding the maximum IsoMax processing time.
TCFALARMVECTOR	(--- addr)	System variable holding the CFA of a word to be performed when TCFALARM is reached. Zero means “no action.”
TCFALARM	(--- addr)	System variable holding an “alarm limit” for TCFOVFLO. Zero means “no alarm.”
TCFOVFLO	(--- addr)	System variable holding a count of the number of times state processing overran the allotted time.
TCFTICKS	(--- addr)	System variable holding a running count of IsoMax timer interrupts.
IS-STATE?	(addr --- f)	Given state address “addr”, returns true if that is the current state in the associated state machine.
SET-STATE	(addr ---)	Makes the given state “addr” the current state in its associated state machine.
IN-EE	(---)	Moves code of last defined CONDITION clause from the Program RAM memory to the Program Flash memory.
TO-HAPPEN	(addr ---)	Makes given state “addr” execute on the next iteration of the IsoMax machine. Same as NEXT-TIME.
NEXT-TIME	(addr ---)	Makes given state “addr” execute on the next iteration of the IsoMax machine.
THIS-TIME	(addr ---)	Makes given state “addr” execute on this iteration of the IsoMax machine, i.e., immediately.

THEN-STATE	(sys3 ---)	Ends the CAUSES clause.
CAUSES	(sys2 --- sys3)	Specifies actions to be taken when the CONDITION clause is satisfied.
CONDITION	(sys1 --- sys2)	Specifies the logical condition to be tested for a state transition.
IN-STATE	(--- sys1)	Specifies the state to which the following condition clause will apply.
ON-MACHINE <name>	(---)	Specifies the machine to which new states and condition clauses will be added.
APPEND-STATE <name>	(---)	Adds a new state "name" to the currently selected machine.
MACHINE <name>	(---)	Defines a new state machine "name".
CURSTATE	(--- addr)	System variable used by the IsoMax compiler.
ALLOC	(n --- addr)	Allocate "n" locations of state data and return its address "addr".
RAM	(--- addr)	System variable which holds an optional address for IsoMax state data allocation. If zero, IsoMax state data will use the dictionary for state data.

22. I/O Trinaries

Word	Stack Effect	Description
AND-MASK n	(--- sys)	Specifies 'n' to be used as the AND mask for output.
AT-ADDR addr	(--- sys)	Specifies the address 'addr' to be used for input or output.
CLR-MASK n	(--- sys)	Specifies 'n' to be used as the Clear mask for output.
DATA-MASK n	(--- sys)	Specifies 'n' to be used as the Data mask for input.
DEFINE <name>	(--- sys1)	Begin the definition of an I/O or procedural trinary.
END-PROC	(sys2 ---)	Ends a PROC definition.
FOR-INPUT	(sys ---)	Ends an input trinary definition.
FOR-OUTPUT	(sys ---)	Ends an output trinary definition.
PROC	(sys1 --- sys2)	Defines an I/O trinary using procedural code.
SET-MASK n	(--- sys)	Specifies 'n' to be used as the Set mask for output.
TEST-MASK n	(--- sys)	Specifies 'n' to be used as the Test mask for input.
XOR-MASK n	(--- sys)	Specifies 'n' to be used as the XOR mask for output.

23. Loop Indexes

Word	Stack Effect	Description
LOOPINDEX <name>	(---)	Define a loop-index variable <name>. <name> will then be used to select the variable for one of the following index operations.
START	(n ---)	Set the starting value of the given loop-index variable.
END	(n ---)	Set the ending value of the given loop-index variable.
STEP	(n ---)	Set the increment to be used for the given loop-index variable.
RESET	(---)	Reset the given loop-index variable to its starting value.
COUNT	(--- flag)	Increment the loop-index variable by its STEP value. If it passes the END value, reset the variable and return a true flag. Otherwise return a false flag.
VALUE	(--- n)	Return the current value of the given loop-index variable.
LOOPINDEXES	(---)	Select LOOPINDEXES methods for compilation.

24. Bit I/O

Word	Stack Effect	Description
PE0 PE1 PE2 PE3 PE4 PE5 PE6 PE7 PD0 PD1 PD2 PD3 PD4 PD5 PB0 PB1 PB2 PB3 PB4 PB5 PB6 PB7 PA0 PA1 PA2 PA3 PA4 PA5 PA6 PA7 GRNLED YELLED REDLED	(---)	Select the given pin or LED for the following I/O operation.
OFF	(---)	Make the given pin an output and turn it off.
ON	(---)	Make the given pin an output and turn it on.
TOGGLE	(---)	Make the given pin an output and invert its state.
SET	(flag ---)	Make the given pin an output and set it on or off as determined by flag.
GETBIT	(--- 16b)	Make the given pin an input and return its bit value.
ON?	(--- flag)	Make the given pin an input and return true if it is on.
OFF?	(--- flag)	Make the given pin an input and return true if it is off.
?ON	(--- flag)	Return true if the pin is on; do not change its direction (works with input or output pins).
?OFF	(--- flag)	Return true if the pin is off; do not change its direction (works with input or output pins).
IS-OUTPUT	(---)	Make the given pin an output.
IS-INPUT	(---)	Make the given pin an input. (Hi-Z)
I/O <name>	(16b addr ---)	Define a GPIO pin <name> using bit mask 16b at addr.
GPIO	(---)	Select GPIO methods for compilation.

25. Byte I/O

Word	Stack Effect	Description
PORTB PORTA	(---)	Select the given port for the following I/O operation.
GETBYTE	(--- 8b)	Make the given port an input and return its 8-bit contents as 8b.
PUTBYTE	(8b ---)	Make the given port an output and write the value 8b to the port.
IS-OUTPUT	(---)	Make the given port an output.
IS-INPUT	(---)	Make the given port an input. (Hi-Z)
I/O <name>	(addr ---)	Define a GPIO port <name> at addr.
BYTEIO	(---)	Select BYTEIO methods for compilation.

26. Serial Communications Interface

Word	Stack Effect	Description
SCI1 SCI0	(---)	Select the given port for the following I/O operation.
BAUD	(u ---)	Set the serial port to “u” baud. If HALFSPEEDCPU is selected, the baud rate will be u/2.
RX?	(--- u)	Return nonzero if a character is waiting in the receiver. If buffered, return the number of characters waiting.
RX	(--- char)	Get a received character. If no character available, this will

		wait.
TX?	(--- u)	Return nonzero if the transmitter can accept a character. If buffered, return the number of characters the buffer can accept.
TX	(char ---)	Send a character.
RXBUFFER	(addr u ---)	Specify a buffer at addr with length u is to be used for receiving. u must be at least 5. If u=0, disables receive buffering.
TXBUFFER	(addr u ---)	Specify a buffer at addr with length u is to be used for transmitting. u must be at least 5. If u=0, disables transmit buffering.
SCIS	(---)	Select SCIS methods for compilation.

27. Serial Peripheral Interface

Word	**Stack Effect**	**Description**
SPI0	(---)	Select the given port for the following I/O operation.
MBAUD	(n ---)	Set the SPI port to n Mbaud. n must be 1, 2, 5, or 20, corresponding to actual rates of 1.25, 2.5, 5, or 20 Mbaud. All other values of n will be ignored and will leave the baud rate unchanged.
LEADING-EDGE	(---)	Receive data is captured by master & slave on the first (leading) edge of the clock pulse. (CPHA=0)
TRAILING-EDGE	(---)	Receive data is captured by master & slave on the second (trailing) edge of the clock pulse. (CPHA=1)
ACTIVE-HIGH	(---)	Leading and Trailing edge refer to an active-high pulse. (CPOL=0).
ACTIVE-LOW	(---)	Leading and Trailing edge refer to an active-low pulse. (CPOL=1).
LSB-FIRST	(---)	Cause data to be sent and received LSB first.
MSB-FIRST	(---)	Cause data to be sent and received MSB first.
BITS	(n ---)	Specify the word length to be transmitted/received. n may be 2 to 16.
SLAVE	(---)	Enable the port as an SPI slave.
MASTER	(---)	Enable the port as an SPI master.
RX-SPI?	(--- u)	Return nonzero if a word is waiting in the receiver. If buffered, return the number of words waiting.
RX-SPI	(--- 16b)	Get a received word. If no word is available in the receive buffer, this will wait. In `MASTER` mode, data will only be shifted in when a word is transmitted by `TX-SPI`. In this mode you should use `RX-SPI` immediately after `TX-SPI` to read the data that was received.
TX-SPI?	(--- u)	Return nonzero if the transmitter can accept a word. If buffered, return the number of words the buffer can accept.
TX-SPI	(16b ---)	Send a word on the SPI port. In `MASTER` mode, this will output 2 to 16 bits on the MOSI, generate 2 to 16 clocks on the SCLK pin, and simultaneously input 2 to 16 bits on the MISO pin.
RXBUFFER	(addr u ---)	Specify a buffer at addr with length u is to be used for receiving. u must be at least 5. If u=0, disables receive buffering.
TXBUFFER	(addr u ---)	Specify a buffer at addr with length u is to be used for transmitting. u must be at least 5. If u=0, disables transmit buffering.
SPI	(---)	Select SPI methods for compilation.

28. Timers

Word	Stack Effect	Description
TD2 TD1 TD0 TC3 TC2 TC1 TC0 TB3 TB2 TB1 TB0 TA3 TA2 TA1 TA0	(---)	Select the given timer for the following I/O operation.
ACTIVE-HIGH	(---)	Change output & input to normal polarity, 1=on. For output, `PWM-OUT` will control the *high* pulse width. For input, `CHK-PWM-IN` will measure the width of the *high* pulse. The reset default is `ACTIVE-HIGH`.
ACTIVE-LOW	(---)	Change output & input to inverse polarity, 0=on. For output, `PWM-OUT` will control the *low* pulse width. For input, `CHK-PWM-IN` will measure the width of the *low* pulse.
ON	(---)	Make the given pin a digital output and turn it on.
OFF	(---)	Make the given pin a digital output and turn it off.
TOGGLE	(---)	Make the given pin a digital output and invert its state.
SET	(flag ---)	Make the given pin a digital output and set it on or off as determined by flag.
ON?	(--- flag)	Make the given pin a digital input and return true if it is on.
OFF?	(--- flag)	Make the given pin a digital input and return true if it is on.
GETBIT	(--- 16b)	Make the given pin a digital input and return its bit value.
?ON	(--- flag)	Return true if the timer input pin is on; do not change its mode.
?OFF	(--- flag)	Return true if the timer input pin is off; do not change its mode.
SET-PWM-IN	(---)	Start time measurement of an input pulse. The duration of the next high pulse will be measured (or low pulse if ACTIVE-LOW).
CHK-PWM-IN	(--- u)	Returns the measured duration of the pulse, in cycles of a 2.5 MHz clock, or zero if not yet detected. Only the first non-zero result is valid; successive checks will give indeterminate results.
PWM-PERIOD	(u ---)	Set PWM period to u cycles of a 2.5 MHz clock. u may be 100-FFFF hex.
PWM-OUT	(u ---)	Outputs a PWM signal with a given duty cycle u, 0-FFFF hex, where FFFF is 100%. `PWM-PERIOD` must be specified before using `PWM-OUT`.
TIMER <name>	(addr ---)	Define a timer <name> at addr.
TIMERS	(---)	Select TIMERS methods for compilation.

29. PWM Output Pins

Word	Stack Effect	Description
PWMB5 PWMB4 PWMB3 PWMB2 PWMB1 PWMB0 PWMA5 PWMA4 PWMA3 PWMA2 PWMA1 PWMA0	(---)	Select the given pin for the following I/O operation.
PWM-PERIOD	(+n ---)	Set PWM period to +n cycles of a 2.5 MHz clock. n may be 100-7FFF hex. This will affect all PWM outputs in the group (A or B).

PWM-OUT	(u ---)	Outputs a PWM signal with a given duty cycle u, 0-FFFF hex, where FFFF is 100%. `PWM-PERIOD` must be specified before using `PWM-OUT`.
ON	(---)	Make the given pin a digital output and turn it on.
OFF	(---)	Make the given pin a digital output and turn it off.
TOGGLE	(---)	Make the given pin a digital output and invert its state.
SET	(flag ---)	Make the given pin a digital output and set it on or off as determined by flag.
?OFF	(--- flag)	Return true if the pin is on.
?ON	(--- flag)	Return true if the pin is off.
PWM <name>	(16b1 16b2 n addr ---)	Define a PWM output pin <name> using configuration pattern 16b1, bit pattern 16b2, and channel n, at addr.
PWMOUT	(---)	Select PWMOUT methods for compilation.

30. PWM Input Pins

Word	Stack Effect	Description
ISB2 ISB1 ISB0 FAULTB3 FAULTB2 FAULTB1 FAULTB0 ISA2 ISA1 ISA0 FAULTA3 FAULTA2 FAULTA1 FAULTA0		Select the given pin for the following I/O operation.
ON?	(--- flag)	Return true if the given pin is on.
OFF?	(--- flag)	Return true if the given pin is off.
?ON	(--- flag)	Return true if the given pin is on. Same as ON?
?OFF	(--- flag)	Return true if the given pin is off. Same as OFF?
GETBIT	(--- 8b)	Return the bit value of the given pin.
PWM <name>	(16b addr ---)	Define a PWM input pin <name> using bit mask 16b at addr.
PWMIN	(---)	Select PWMIN methods for compilation.

31. Analog-to-Digital Converter

Word	Stack Effect	Description
ADC7 ADC6 ADC5 ADC4 ADC3 ADC2 ADC1 ADC0	(---)	Select the given pin for the following I/O operation.
ANALOGIN	(--- +n)	Perform an A/D conversion on the selected pin, and return the result +n. The result is in the range 0-7FF8. (The 12-bit A/D result is left-shifted 3 places.) 7FF8 corresponds to an input of Vref. 0 corresponds to an input of 0 volts.
ADC-INPUT <name>	(n addr ---)	Define an analog input pin <name> for channel n at addr.
ADCS	(---)	Select ADCS methods for compilation.