IV: Operators

This section lists each operator. It gives the name and the effect it has on the stack or robot state. The operands are as follows:

```
+           -           *           /           =
!           <           <           ABS         AND
ARCCOS      ARCSIN      ARCTAN      BEEP        CALL
CHS         COS         DEBUGGER    DIST        DROP
DROPALL     DUPLICATE   FLUSHINT    ICON0-9     IF
IFE         IFEG        IFG         INTOFF      INTON
JUMP        MAX         MIN         MOD         NOP
NOT         OR          PRINT       RETURN      ROLL
RTI         SETINT      SETPARAM    SIN         SND0-9
STORE       SQRT        SWAP        SYNC        TAN
VSTORE      VRECALL     XOR
```

A number of these commands have abbreviations: DUP is an acceptable substitute for DUPLICATE, STO for STORE, DEBUG for DEBUGGER, COSINE for COS, SINE for SIN, TANGENT for TAN, and EOR for XOR. RETURN and JUMP really perform the same function, as noted below; hence, in the debugger, both instructions are listed as JUMP. In the debugger, the special (RECALL) and (END) operators may appear; these are inserted by the Assembler and not used directly by the programmer.

+
Adds the top two numbers on the stack, removes them, and replaces them with the result.
Ex: “4 5 +” leaves 9 on the top of the stack.

-
Subtracts the top number from the second number on the stack, removes them, and replaces them with the result.
Ex: “9 3 -” leaves 6 on the top of the stack.

*
Multiplies the top two numbers on the stack, removes them, and replaces them with the result.
Ex: “2 4 *” leaves 8 on the top of the stack.

/
Divides the second number by the top number on the stack, removes them, and replaces them with the result.
Ex: “22 3 /” leaves 7 on the top of the stack. (7.3333 is truncated to 7)

=
Checks if the top two numbers on the stack are equal, then removes them. If they are equal, it places a 1 on the stack, otherwise it pushes a 0.
Ex: “2 2 =” leaves a 1 on the stack.

!
Checks if the top two numbers on the stack are not equal, then removes them. If they are not equal, it places a 1 on the stack, otherwise it pushes a 0. (The symbol ! comes from the logical NOT command in the language C.)
Ex: “5 5 !” leaves a 0 on the stack.

>
Checks if the second number on the stack exceeds the top number, then removes them. If the second number is greater, it places a 1 on the stack, otherwise it pushes a 0.
Ex: “5 4 >” leaves a 1 on the stack.

<
Checks if the second number on the stack is less than the top number, then removes them. If the second number is less, it places a 1 on the stack, otherwise it pushes a 0.
Ex: “7 3 <” leaves a 0 on the stack.

ABS
Absolute Value. Removes the top argument and returns its absolute value.
EX: “-8 ABS” leaves 8 on the stack.

AND
Checks if the top two numbers on the stack are both not zero, then removes them. If they are both not zero, it places a 1 on the stack, otherwise it pushes a 0.
EX: “2 3 AND” leaves a 1 on the top of the stack.

ARCCOS
Inverse Cosine. Computes the principal inverse cosine of the top number divided by the second number. Returns the result in degrees in the range of 0 to 180 (just like the C library routine). Note that you may have to convert the result to the appropriate coordinate system before using it to set your AIM register or such.
EX: “1000 707 ARCCOS” leaves 45 on the stack.

ARCSIN
Inverse Sine. Computes the principal inverse sine of the top number divided by the second number. Returns the result in degrees in the range of -90 to 90 (just like the C library routine). Note that you may have to convert the result to the appropriate coordinate system before using it to set your AIM register or such.
EX: “1000 707 ARCSIN” leaves 45 on the stack.

ARCTAN
Inverse Tangent. Computes the inverse tangent of the ratio of the top two numbers. The y value must be the top operand; the x value must be the second operand. ARCTAN removes the top two operands and returns the arctangent of y/x. The result is in degrees between 0 and 359, with 0 degrees pointing up, just as with AIM angles. Also remember that the positive x values are on the right side of the arena, while positive y values are on the bottom of the arena.
EX: “-5 0 ARCTAN” leaves 270 on the stack.

BEEP
Beeps once. Most useful in debugging a robot.
EX: “BEEP” leaves nothing on the stack.

CALL
Jumps to the instruction number specified by the top element of the stack, removes the top element, places the return address (the instruction number previously being executed) on the top of the stack, and resumes execution at the new instruction. Very similar to JUMP, but leaves the return address on the stack.

CHS
CHange Sign. Multiplies the top operand on the stack by -1, removes it, and returns the result on the stack.
EX: “3 CHS” leaves -3 on the stack.

COS or COSINE
Cosine function. The top argument should be the hypotenuse and the second argument should be the angle. COS removes the top two arguments and returns the hypotenuse times the cosine of the angle, truncated to an integer value. The angle must be between 0 and 359 or the result is undefined. Also, note that while 0 degrees is pointing straight up on the turret, the cosine of 90 degrees is still 0 (e.g. COS uses the standard trig coordinate system).
EX: “30 100 COS” leaves 86 (=100*cos(30) on the stack.

DEBUG/DEBUGGER
A hook for debugging. If the robot is selected for debugging (i.e. has the bug icon beside it in the Arena display), tthis command acts as a sync instruction, pausing until the end of the chronon (because the debugger cannot turn on in the middle of a chronon). Execution pauses immediately after the debugger instruction and one can use the debugger facilities. If the robot is not selected for debugging, nothing happens and no time is used (so that one can run accurate timing on a robot without having to remove DEBUG statements).
EX: "DEBUGGER" pauses until the end of chronon and enters the debugger if debugger is on

DIST
Calculate the distance from the origin to some point (x,y) = Sqrt(x*x+y*y). The top two elements of the stack are x and y; they are removed and replaced with the distance.
EX: “10 10 DIST” leaves 14 on the stack.

DROP
Drops the top element from the stack.
EX: “5 DROP” leaves nothing on the stack.

DROPALL
This command drops everything off of the stack. It is useful for interrupt routines that may not return to where they were called, thus leaving an unknown amount of junk on the stack. It is also useful for sloppy programmers who let junk accumulate on the stack and run into stack overflows after lots of chronons.
EX: “87 42 99 -32 DROPALL” leaves absolutely nothing on the stack.

DUPLICATE or DUP (either token is allowed)
Duplicates the number on the top of the stack.
EX: “5 DUP” leaves 5 on the top of the stack and 5 in the second position.

FLUSHINT
Removes all pending interrupts from the interrupt queue.
EX: “FLUSHINT” does nothing to the stack.

ICON0-9
Sets the robot's icon. ICON0 is special; it causes the robot to display icon #0 if shields are off, but icon #1 if shields are on. Other ICONx commands display icon #x. Takes zero chronons to execute. The various icon commands are used for icon animation.
EX: “ICON3” does nothing to the stack but makes the robot display icon #3.

IF
Checks the second operand on the stack. If it is not zero then it leaves the return address on the stack and jumps to the label specified on the top of the stack. In any case, it removes the second operand and the destination label from the stack.
EX: “1 MySub IF” jumps to the subroutine MySub and leaves the return address on the stack.

IFE
Stands for IF-THEN-ELSE. Checks the third operand on the stack. If it is not zero than it leaves the return address on the stack and jumps to the label specified in the second position on the stack. If it is zero then it leaves the return address on the stack and jumps to the label specified on the top of the stack. In any case it removes the first, second, and third elements from the stack.
EX: “0 SubA SubB IFE” jumps to the subroutine SubB and leaves the return address on the stack.

IFG
Stands for IF-GOTO. Checks the second operand on the stack. If it is not zero then it jumps to the label specified on the top of the stack wiXaving a return address. In any case, it removes the second operand and the destination label from the stack.
EX: “RANGE TARGETING IFG” jumps to the label TARGETING if a hostile robot is in the line of fire and leaves nothing on the stack.

IFEG
Stands for IF-THEN-ELSE-GOTO. Checks the third operand on the stack. If it is not zero than it jumps to the label specified in the second position on the stack. If it is zero then it jumps to the label specified on the top of the stack. Unlike IFE, it leaves no return address. In any case it removes the first, second, and third elements from the stack.
EX: “0 SubA SubB IFEG” jumps to the subroutine SubB, leaving nothing on the stack

INTOFF
Turns interrupts off until next INTON command.
EX: “INTOFF” does nothing to the stack.

INTON
Turns interrupts on. Must be used near start of any robot that uses interrupts.
EX: “INTON” does nothing to the stack.

JUMP
Jumps to the instruction number specified by the top element of the stack, removes the top element, and resumes execution at the new instruction. This is the same operation as RETURN.

MAX
Leaves only the greater of the top two numbers on the stack.
EX: “4 8 MAX” leaves 8 on the stack.

MIN
Leaves only the lesser of the top two numbers on the stack.
EX: “4 8 MIN” leaves 4 on the stack.

MOD
Performs a modulus operation (remainder of integer division) on the top two elements of the stack. Removes them and returns the result on the stack. Note that this follows the same rules as the modulus operation in C; for example, -69 mod 360 = -69.
EX: “10 3 MOD” leaves 10 mod 3 = 10-3*Trunc(10/3) = 1 on the stack.

NOP
No OPeration. Does nothing whatsoever, except take up time and space. May be used when some timing loop is necessary.
EX: “NOP” leaves nothing on the stack.

NOT
Logical Not. Checks top operand, removes it. Returns 1 if it was 0, 0 otherwise.
EX: “4 NOT” leaves 0 on the stack.

OR
Checks if either of the top two numbers on the stack is not zero, then removes them. If either is not zero, it places a 1 on the stack, otherwise it pushes a 0.
EX: “0 4 OR” leaves a 1 on the top of the stack.

PRINT
Displays the top element of the stack in a dialog box. Takes zero chronons to execute. Intended for debugging purposes only.
EX: “5 PRINT” leaves 5 on the stack and displays a dialog box reading “5”

RETURN
Jumps to the instruction number specified by the top element of the stack, removes the top element, and resumes execution at the new instruction. The same operator, usually written with the name RETURN, returns after a subroutine call made by IF or CALL by jumping to the return address that the IF or CALL left on the top of the stack.

ROLL
Rolls the second element of the stack back the number of places specified by the top operand, then removes the top operand.
EX: “1 2 3 4 5 2 ROLL” rolls 5 back 2 places, leaving 1 2 5 3 4 on the stack.

RTI
Returns from interrupt. First enables interrupts like INTON, then pops return address from top of stack and jumps there like RETURN.
EX: “RTI” removes return address from stack.

SETINT

Sets an interrupt procedure to execute when interrupt occurs and interrupts are enabled. The top of the stack must have the name of the register corresponding to the interrupt (choices are COLLISION, WALL, DAMAGE, TOP, BOTTOM, LEFT, RIGHT, RADAR, RANGE, and SIGNAL). The second item on the stack must be the address of a routine that handles the interrupt.
EX: "panicsub RADAR' SETINT" leaves nothing on the stack.

SETPARAM
Sets a parameter controlling when interrupts occur. The top of the stack must have the name of the register corresponding to the interrupt. The second item on the stack is the value to which the parameter will be set. Note that COLLISION and WALL ignore their parameters. The remaining default parameters are 150 for DAMAGE, 20 for TOP, 280 for BOTTOM, 20 for LEFT, 280 for RIGHT, 600 for RADAR and RANGE, and 0 for SIGNAL. SETPARAM may also be used to set information for the PROBE and HISTORY registers.
EX: "100 RANGE' SETPARAM" leaves nothing on the stack.

SIN or SINE
Sine function. The top argument should be the hypotenuse and the second argument should be the angle. Sine removes the top two arguments and returns the hypotenuse times the sine of the angle, truncated to an integer value. The angle must be between 0 and 359 or the result is undefined. Also, note that while 0 degrees is pointing straight up on the turret, the sine of 0 degrees is 0 (e.g. sin uses the standard trig coordinate system).
EX: "30 100 SIN" leaves 50 (=100*sin(30) on the stack.

SND0-9
Plays a sound defined in the Recording Studio. SNDx plays sound #x defined in the Recording Studio. Takes zero chronons to execute. The various other SND commands are used to enliven a robot with sound effects and to waste ungodly amounts of disk space.
EX: "SND2" does nothing to the stack but plays sound #2.

STO or STORE (either token is allowed)
Stores the second number on the stack in the variable specified at the top of the stack, and removes both the number and variable reference. The operand on the top of the stack must be a quoted variable or an error is reported. Also see the section on variables below, as values cannot be stored in some variables, such as RANGE.
EX: "20 aim' store" stores 20 in the variable AIM and leaves nothing on the stack.

SQRT
Square root function. Removes the top argument and returns its square root, truncated to an integer. If the argument is less than zero, an error results and the robot self-destructs.
EX: "10 SQRT" leaves 3 on the stack.

SWAP
Swaps the top and second elements on the stack.
EX: "1 2 SWAP" leaves 1 at the top of the stack and 2 in the second position.

SYNC
Ceases execution of code until the end of chronon. The next instruction will be executed at the next chronon. Has no effect on the stack.

TAN or TANGENT
Like SIN and COS, but computes the tangent function. If the result is greater than 19999 it is clipped to 19999; if the result is less than -19999, it is clipped to -19999.

VSTORE
Store a element in a vector (1 dimensional array). The top element specifies the element of the vector (0 to 100) while the second element is the number to store. Both arguments are removed from the stack.
EX: "42 1 VSTORE" stores 42 in the first element of the vector.

VRECALL
Returns the nth element of the vector. The top of the stack holds the element to reference; it is removed and replaced by the contents of the vector. Uninitialized vector elements default to zero. If the element number is less than 0 or greater than 100, zero is also returned.

EX: “1 VRECALL” leaves 42 on the stack assuming the previous example was last executed.

XOR or EOR (either token is allowed)
Exclusive OR. Checks if the top or second item on the stack, but not both, are nonzero, then removes them. If so, it places a 1 on the stack; otherwise it pushes a 0.
EX: “1 1 XOR” leaves a 0 on the top of the stack.