Terms

|  |  |
| --- | --- |
| **Term** | **Notes** |
| Integer constant | Represents an integer constant |
| Identifier | Value of an identifier (may be function calls). Identifiers begin with A-Z or underscore and are followed by A-Z, 0-9 or underscore. |
| “(character)” | ASCII code of character – must be one character only, except “” is code zero (e.g. end of ASCIIZ string) |
| -<term> | Negation |
| ~<term> | One’s complement (e.g. invert all bits) |
| !<term> | 32 bit read of given address, which must be legal and on a 4 byte boundary. One can write demo!4 or !(demo+4), they are the same. |
| ?<term> | 8 bit read of given address, must be legal. |
| (<expression>) | Expression in parenthesis. |
| LEN(<expression>) | Returns length of string at the expression. |
| ABS(<expression>) | Returns absolute value of expression |
| RND(<expression>) | Returns random integer from 1 - <expression>, values < 0 return a random 32 bit integer |
| SGN(<expression>) | Return the sign of the expression (-1, 0 or 1) |
| MEM(<expression>) | Return the base address of allocated memory. *(Normally the global variables start at 0, Allocated Memory follows them, and the stack winds down from the top of memory)*. |
| PADX(<expression>) | Return the position of the X joystick (-1,0,1) |
| PADY(<expression>) | Return the position of the Y joystick (-1,0,1) |
| FIRE(<expression>) | Return the state of A(0) B(1) as -1 or 0, other values return 0. |

Note that the functions (LEN etc.) are all system functions and compile as <expression code> $$name e.g. reading fire button 0 is “0 $$FIRE”

Expression Levels

|  |  |  |
| --- | --- | --- |
| **Level** | **Operators** | **Notes** |
| 0 | & | ^ | Bitwise operations, AND OR NOT |
| 1 | < <= > >= == <> | Comparison operators, all return -1 or 0. |
| 2 | + - | Additive operators |
| 3 | \* / % | Multiplicative operators |
| 4 | ? ! | Memory indirection operators, byte and word. |