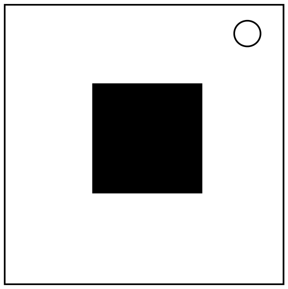
On the Subject of Not Colored Cube



the cylinder??7?7?77?777777 wait, that's not right...dotdot..dotdot.dot.dot..

See Appendix CLC-DLC from Colored Cube for identifying Colored Cube variants.

This module contains a Colored Cube, which can be either (R)ed, (G)reen, (B)lue, (Y)ellow, (M)agenta, (C)yan, (W)hite or Blac(K). The entire cube is selectable.

Throughout this module, you will need to keep track of a variable **C**. This variable can be of 4 different variable types given below. Initially, **C** starts out Undefined. **Keep in mind, that when C is defined as a variable of some type, its type cannot be changed afterwards.**

- Number - an integer number in range 0-9999. If at any point **C** goes out of this range, take **C** modulo 10000. Operations include: addition, subtraction, multiplication, division, modulo (dividing by 0 and taking a number modulo 0 both return 0; when dividing, always immediately drop decimals) and reversing the number.
- String - a string containing exclusively letters. The string can be modified by prepending/inserting/appending characters or strings to it (appending "CUBE" to "COLORED" yields "COLORED**CUBE**"; prepending "NOT" into "COLORED" yields "**NOT**COLORED"; inserting "U" to "COLORED" at position 5 yields "COLO**U**RED"), reversing it, or shifing it to the left/right by a certain amount ("ABC" shifted to the left twice yields "CAB"; "XYZ" shifted to the right twice yields "YZX"). Additionally, you are able to take one of the string's substrings for certain operations or conditions.
- Boolean - a truth value which can be either **TRUE** or **FALSE**. This variable can be modified by applying one of the logical operators to it (refer to the last table on Page 6 for the truth table for each of the operators). This variable can also be used/set as the answer of a **TRUE**/**FALSE** question in certain conditions.
- Undefined - the only operation that can be done with this variable is defining it as a variable of one of the types above.

Follow the rules for each of the 10 stages of this module (with the number in the top left corner of the cube's top face indicating the stage number) on the next pages.

Note: in the rules: "S#" refers to the Serial Number; unless stated otherwise, alphabetic positions are A1Z26, so A = 1, B = 2... Z = 26; **keep in mind that you may need to use the cube color from one of the previous stages - remember to keep track of all of the previous stages' cube colors.**

Stage 1

- If the cube is...
    - Red or Cyan – define **C** as a Number equal to the last digit of the S#. Press the cube when the last digit of the timer is equal to **C**.
    - Yellow – define **C** as a String – the concatenation of S#'s letters (left-to-right). Press the cube when the seconds digits of the timer form a number that is divisible by [alphabetic position of the last letter in **C**, plus 1].
    - White or Black – define **C** as a Boolean that is equal to the answer to the question "Is the first digit of the S# even?". Press the cube when the answer to the question "Does the last digit of the timer share parity with the number of battery holders?" matches **C**.
    - None of the above – press the cube when the last digit of the timer is prime.

Stage 2

- If **C** is...
    - A Number – multiply it by 3 and then add 1. Press the cube when the seconds digits of the timer form the number [**C** % 60].
    - A String – append [batteries + battery holders], taken modulo 26 and converted into a letter A0Z25 (0 = A, 1 = B... 25 = Z) to **C**. Press the cube when the seconds digits of the timer, when taken modulo 26 and converted into a letter A0Z25, are present in **C**.
    - A Boolean – if the cube is Black, Blue, Green or Magenta, apply the NOT operator to **C**. Press the cube when the answewr to the question "Is the last digit of the timer composite?" matches **C**.
    - Undefined – if there is an empty port plate, define **C** as a **TRUE** Boolean. Otherwise, if there are 5 or more ports, define **C** as a Number equal to [ports * port plates]. Press the cube at any time.

Stage 3

- If the cube is Red, Green or Blue and **C** is...
    - A Number – reverse it, then multiply it by its digital root.
    - A String – reverse it, then, if **C**'s length is odd, prepend "ODD" to it, and finally set **C** to be equal to its first half.
    - A Boolean – XNOR it with the answer to the question "Is there an even number of regular modules on this bomb?".
    - Undefined – define it as a **FALSE** Boolean, and then follow the rule for **C** being a Boolean above.
- Take the product of the following edgework values (if any of them are equal to 0, substitute 1 for them instead): batteries, battery holders, indicators, ports, port plates, sum of S#'s digits. If **C** is a non-zero Number, multiply the result by **C**. Press the cube when the seconds digits of the timer form the number that is equal to the result of the previous calculation taken modulo 60.

Stage 4

- If the cube is White and **C** is defined:
    - If **C** is…
        - A Number – set **C** to be equal to the concatenation of S#'s digits (left-to-right; set **C** to be equal to 0 if there aren't any digits in the S#).
        - A String – set **C** to be the name of the module (regular/needy) on the bomb that comes first alphabetically, without any spaces (exclude all modules that do not consist of only letters A-Z and spaces).
        - A Boolean – apply the NOT operator to **C**.
    - After following one of the rules above, press the cube at any time.
- Otherwise, if **C** is…
    - A Number – press the cube when the tens digit of the timer is equal to **C**'s first digit modulo 6.
    - A Boolean – press the cube when the answer to the question "Is the tens digit of the timer composite?" matches **C**.
    - None of the above – press the cube when both of the seconds digits of the timer match.

Stage 5

- If the last digit of the S# is even and **C** is…
    - A Number – subtract it from 10000 and set **C** to be the result.
    - A String – shift it to the right by [last digit of the S# + 1], then shift it to the left by [first digit of the S# + 1].
    - Undefined and there is a prime digit in the S# – define **C** as the String "CUBE".
- Press the cube when the last digit of timer matches the…
    - If **C** is a Number – parity of **C**.
    - If **C** is a String – parity of the sum of **C**'s first and last letters' alphabetic positions.
    - If **C** is a Boolean – parity of S#'s first digit if **C** is **TRUE** and parity of S#'s last digit if **C** is **FALSE**.
    - If **C** is Undefined – parity of 3525854.

Stages 6-10

- If C is defined, modify it as follows:
    - If C is a Number – use the table below to obtain the modification operation, and if needed, use the other table to obtain the modifier number (M) for this press' operation (which will be some edgework value).

Prev. stage's cube color →	**Red**	**Green**	**Blue**	**Yellow**
Operation →	Add [2 * M] to C	Multiply C by M	Divide C by [2 * M]	Multiply C by 2, then divide C by M
Prev. stage's cube color →	**Magenta**	**Cyan**	**White**	**Black**
Operation →	Subtract M from C, then divide C by 3	Subtract [C % M] from C, then divide C by M	Reverse C	Do not modify C

Stage num. → This stage's cube color ↓	**6**	**7**	**8**	**9**	**10**
Red	Battery holders	Stereo RCA ports	Last digit of the S#	DVI-D ports	Sum of S#'s digits
Green	First digit of the S#	Letters in the S#	PS/2 ports	Last digit of the S#	Consonants in the S#
Blue	Letters in the S#	Serial ports	D Batteries	Digits in the S#	Indicators
Yellow	Batteries	Sum of S#'s digits	Port plates	Battery holders	AA Batteries
Magenta	Parallel ports	AA Batteries	Last digit of the S#	Batteries	Vowels in the S#
Cyan	Consonants in the S#	Digits in the S#	Battery holders	S#'s 3rd character	D Batteries
White	Lit indicators	Ports	First digit of the S#	Stereo RCA ports	Indicators
Black	Ports	Vowels in the S#	RJ-45 ports	Port plates	Unlit indicators

-
    - If **C** is a String – apply the following modifications:
        1. Add the last digit of the S# to the stage number, then, if the first digit of the S# is odd, subtract 5. Convert this number into a letter A1Z26 (1 = A, 2 = B… 26 = Z). If the stage number is even, **append** the obtained letter to **C**, otherwise, if the stage number is odd, **prepend** the obtained letter to **C**.
        2. Apply Atbash Cipher to the letter obtained from the previous step (subtract the letter's alphabetic position from 27 and convert the result back into a letter A1Z26). Take [(the stage number % the length of **C**) + 1], insert the newly obtained letter into **C** at that position.
        3. Use the table below with the color of the cube to obtain a word, then use the other table with your stage number and [the current length of **C** % 10] to figure out what letter/substring you should take from the word. If the stage number is even, **prepend** the obtained letter to **C**, otherwise, if the stage number is odd, **append** the obtained letter to **C**.

Cube color →	**Red**	**Green**	**Blue**	**Yellow**	**Magenta**	**Cyan**	**White**	**Black**
Word →	REBOOT	TOUCHE	WATERS	GOLDEN	PARSED	BREEZY	BRIGHT	VOIDED

Stage num. → **[C's length % 10] ↓**	**6**	**7**	**8**	**9**	**10**
0	1	2-6	5-6	4	1-6
1	3-4	6	4-5	1-6	4
2	1-6	3-4	5	1-2	2
3	1-3	5	6	1-6	2
4	2-6	1-5	1-6	4-6	4
5	2-4	4-5	3	1-6	1-5
6	1-6	2-3	1-3	3	4-6
7	2	2-5	1-6	3	5-6
8	1-4	4	1-6	1	4-5
9	5	1-6	2	2-4	3-5

# – the letter at that position;
#1-#2 – a substring beginning at #1 and ending at #2

-
  - If **C** is a Boolean – use the second table from the section of this stage for **C** being a Number to obtain a number, and then use the table below to obtain a condition. If the obtained number meets the condition obtained from the table below, the boolean value (B) for this stage is **TRUE**, otherwise **B** is **FALSE**. Use the color of the cube in the other table to obtain a logical operator. Apply the obtained logical operator to **C** and **B** (unless the obtained logical operator is NOT, in which case **B** is not important) and set **C** to be the result.

Stage num. →	**6**	**7**	**8**	**9**	**10**
Condition. →	**The number is even.**	**The number is composite.**	**The number is odd.**	**The number is divisible by 5.**	**The number is prime.**

Cube color →	**Red**	**Green**	**Blue**	**Yellow**	**Magenta**	**Cyan**	**White**	**Black**
Operator →	OR	NAND	XOR	XOR	NOR	XNOR	AND	NOT

C	B	NOT	OR	AND	XOR	NOR	NAND	XNOR
F	F	T	F	F	F	T	T	T
F	T	T	T	F	T	F	T	F
T	F	F	T	F	T	F	T	F
T	T	F	T	T	F	F	F	T

- Determine when to press the cube – if **C** is…
  - A Number – if the module is in…
    - Stage 6 – press the cube when the last digit of the timer is equal to the digital root of **C**.
    - Stage 7 – press the cube when the sum of the two seconds digits of the timer is equal to [(**C** % 9) + 3].
    - Stage 8 – press the cube when the difference between the two seconds digits of the timer is equal to [**C** % 5].
    - Stage 9 – press the cube when both of the seconds digits of the timer match the parity of **C**.
    - Stage 10 – press the cube when the last digit of the timer is equal to the last digit of **C** **and** when the tens digit of the timer is equal to [the first digit of **C** % 6].
  - A String – take the difference between [the sum of **C**’s first and second letters’ alphabetic positions] and [the sum of **C**’s last and second-to-last letters’ alphabetic positions]. Press the cube when the seconds digits of the timer form the number that is equal to this difference.

-
  - A Boolean – press the cube when the answer to the question “Does the last digit of the timer meet the condition used to obtain the boolean value for this stage?” matches **C**.
  - Undefined – subtract 5 from your current stage number, then fully follow that stage’s rules from one of the previous pages of this manual (this may cause **C** to get defined – after defining **C**, keep following the stage’s rules all the way until the cube press).