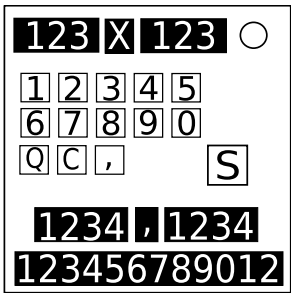
# On the Subject of Fixing Malfunctions




*Turning it off and on again didn't help....this time.*

Follow the original manual for instructions on how to operate the module.

There are 3 cases you will need to differentiate between in order to solve Malfunctions: Either 1 is added/subtracted to the query results, the query responses, or the function number itself. There will only be one case active and it will never change.

Query the 3 functions, in order, into the module. If the response can be found "staircasing" either up or down the table, it's the third case. "Staircasing" up means X = -1, and the starting function is the function from the first query response. "Staircasing" down means X = +1.

If instead, the responses could all be found on the same row, BUT are all either 1 greater or 1 fewer than the ones shown in the table, then it's the second case, and X = +1 or -1, respectively, with the original function being the row in the table.

If neither of these are true, then more queries are needed. Next, input the column queries like before, but take away 1 from each query (i.e. 2450,114 instead of 2451, 115). Do the same for each query, plus 1. If one of the trio of results from these sets exactly matches a row in your table, that function is the target function. If it came from the set where you subtracted 1 from each query, then X = +1. If it came from the set where you added 1 to each query, then X = -1. If both trios of results matches a row, an additional query will be needed to figure out which function is correct.

| | 2451, 115 | 6313, 201 | 1550, 45 |
|---|---|---|---|
| 0 | 274176 | 1249380 | 64974 |
| 1 | 36 | 82 | 20 |
| 2 | 196 | 2687 | 700 |
| 3 | 7 | 7 | 7 |
| 4 | 1 | 1 | 4 |
| 5 | 6 | 6 | 3 |
| 6 | 9 | 9 | 8 |
| 7 | 84 | 39 | 99 |
| 8 | 281865 | 1268913 | 69750 |

| | | | |
|---|---|---|---|
| 9 | 10 | 10 | 11 |
| 10 | 5 | 6 | 5 |
| 11 | 126 | 28 | 125 |
| 12 | 2455 | 6313 | 1555 |
| 13 | 6680865 | 2945913 | 8069750 |
| 14 | 0 | 2 | 2 |
| 15 | 4 | 5 | 4 |
| 16 | 79 | 119 | 25 |
| 17 | 2 | 2 | 2 |
| 18 | 25666 | 65142 | 15955 |
| 19 | 2566 | 6514 | 1595 |
| 20 | * | * | * |
| 21 | 5 | 3 | 0 |
| 22 | 6 | 5 | 6 |
| 23 | 4140 | 16482 | 900 |
| 24 | 15676 | 46714 | 3575 |
| 25 | 11 | 11 | 9 |
| 26 | 2336 | 6112 | 1505 |
| 27 | 12 | 27 | 35 |
| 28 | 10500 | 10500 | 9000 |
| 29 | 3 | 4 | 3 |
| 30 | 6007516 | 39854170 | 2402545 |
| 31 | 350 | 901 | 258 |
| 32 | * | * | * |
| 33 | 8 | 8 | 7 |
| 34 | 116 | 214 | 45 |
| 35 | 21 | 31 | 34 |

Once you determine if your letter condition is true or false, apply the offset ABOVE your letter if X = -1, and BELOW your letter if X = +1. (i.e. if X= +1, and your module has the letter S, you'd use S's rule but apply T's offset.)

Once you have found the starting function and X, apply the rule corresponding to the letter on the bomb based on the following table:

| Condition | True | False |
|---|---|---|
| A: KBU or M in SN? | +6 | −4 |
| B: Battery, indicator, or port count = 2? | +2 | −3 |
| C: First character in SN a digit? | +5 | −4 |
| D: Lit BOB indicator? | +8 | −8 |
| E: Unlit BOB indicator? | +6 | −2 |
| F: First character in SN a letter? | +6 | −5 |
| G: Parallel port but no Serial port? | +1 | −5 |
| H: At least one empty port plate? | +1 | −3 |
| I: No batteries? | +1 | +5 |
| J: Vowel in SN? | +5 | −3 |
| K: Indicators > 3? | +4 | −1 |
| L: Battery count even? | +6 | +7 |
| M: Ports > indicators? | +3 | −7 |
| N: More lit than unlit indicators? | +3 | −5 |
| O: Indicators > batteries? | +6 | −1 |
| P: Indicator count even? | +2 | −3 |
| Q: ERI or S in SN? | +1 | −3 |
| R: Exactly 3 letters in SN? | +3 | −2 |
| S: Batteries > ports? | +2 | +4 |
| T: Batteries > 4? | +4 | +1 |
| U: Lit and unlit indicator count equal? | +2 | −2 |
| V: JQX or Z in SN? | +7 | +1 |
| W: At least three ports? | +3 | −5 |
| X: No indicators? | +3 | −3 |
| Y: 4+ SN digits? | +4 | −1 |
| Z: No ports? | +5 | −1 |

Use your new function as the final function, and the two numbers surrounding the letter on the module as a and b to solve the module, not applying X or its rule in any way.

#0: abs(a minus 3) times abs(b minus 3)
#1: Larger modulo smaller
#2: 10,000 modulo Larger
#3: 7
#4: (Larger divided by smaller) modulo 10
#5: Triple the number of odd numbers
#6: 10 minus (abs(digits in a minus digits in b))
#7: (sum of digits in a) times (sum of digits in b)
#8: a times b
#9: a plus b, modulo 12
#10: Highest digit
#11: ((a modulo 10) cubed) plus ((b modulo 10) cubed)
#12: Lunar Addition
#13: abs(10,000 minus a) times abs(1,000 minus b)
#14: (Larger modulo smaller) modulo 8
#15: Number of different digits
#16: Smaller minus (Larger modulo Smaller)
#17: Number of different odd digits
#18: ((a + b) times 10) plus (abs(a - b) modulo 10)
#19: a plus b
#20: (a times b) divided by (ports + 2)
#21: (a times b) modulo 10
#22: Number of different digits missing
#23: (Larger modulo smaller) times smaller
#24: a + (b squared)
#25: 11 minus twice the number of non-two-digit variables
#26: abs(a minus b)
#27: (a times b) modulo 73
#28: Digits in a and b times 1,500
#29: 3 plus the number of numbers over 2,500
#30: (a squared) + b
#31: Larger divided by (digits in both a and b)
#32: a + b + concatenated serial number digits + 1
#33: 8 minus (Number of numbers below 100)
#34: (a modulo 50) + b
#35: Larger divided by smaller

* Additional queries may be needed to be assured of the correction function and X, and to potentially differeniate the two edgework-related functions (#20, and #32).