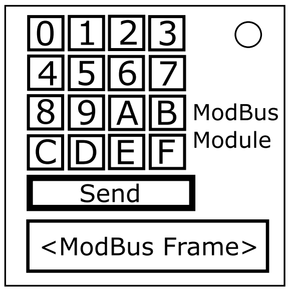
On the Subject of ModBus



"I got this, I'm a professional" was his last word.

The ModBus is a serial communications protocol originally published by Modicon in 1979, for use with it's PLCs (Programmable Logic Controllers).

A little bit of History is good, right?

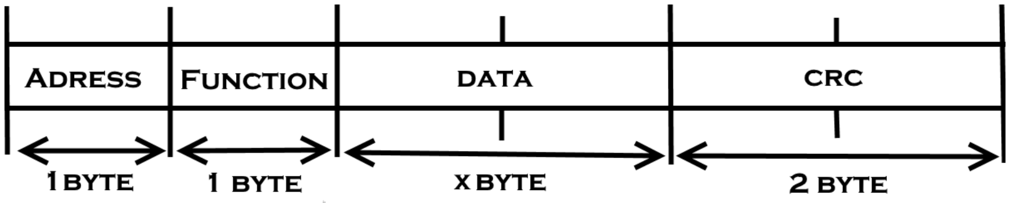
It's commonly used as means of connecting industrial electronic devices...and can be involved with making a bomb.
To defuse the module, it is simple. You only need to send the right frame.
Don't send anything before are done writing the frame.

You just have to follow the instructions. As simple as cooking a turkey!

ModBus Protocol

"READ THIS SECTION! please."

Like we said before, you need to follow a strict protocol. We are lucky, bomb-makers never use CRC (Cyclic Redundancy Check, "Error check").
Furthermore, we know they only use the RTU frame format described below



Like we said before, we don't need to calculate the CRC (lucky you).

16 bit words (2 bytes) are sent in "Big endian" (Most Signifiant Bit First).
i.e. : You have to send $(42)_{10} = (21)_{HEX}$, so you need to send "2" then "1".

Address

"Where do I have to enter the defuse code?"

To fill the address, you need to look at the first number (or letter) of the serial code on the bomb.
Convert it (they are all in ASCII) into a hexademal number (See Appendix ASCII Tab), this is the address (yay!).

Don't use decimal numbers!

Function

"Hmm... I guess it's an important section..."

Bomb-makers (or defusers, or someone else?) use only 2 functions:

1. Function 04, used to read a data word. (and Yes, we'll have to use it)
2. Function 06, used to write a data word.

The choice is simple, look at the last number (or letter) of the serial code.
...Convert it into a decimal number. (they are all in ASCII)
THEN calculate this number modulo 4(e.g. : 12%4 = 0, 15%4 = 3)

"Simple, right?"

After that, multiply it by 3, and calculate the result modulo 2
If the result is ...

1. ...0, use function 04.
2. ...1, use function 06.

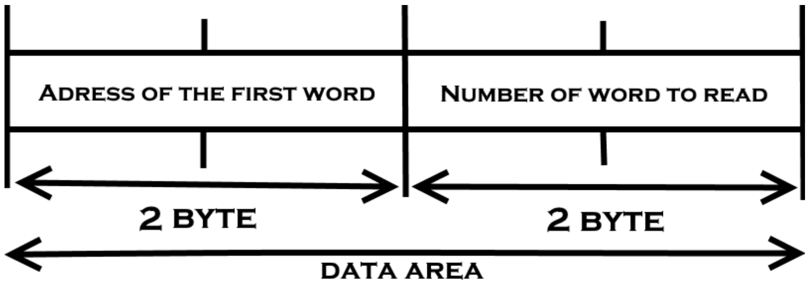
Function 04

Read data, defuse the module, be a hero. BOOM dream achieved.

Bomb-makers developed a new kind of protection. Instead of using the write function to defuse their module, You only need to read the right data on the module using function 04.

Don't care about it.

So, How is the data area composed? (Read the tab below)



You see, It's the last 4 bytes to find! Keep calm and defuse it!

Address of the word

First, to find the word address, multiply the third digit by the fourth digit.
Convert letters into decimal with appendix ASCII Tab.
Don't change numbers like above, take the number write (If it's write 5 take 5).
Next, multiply the result by 100.
Then, just take the least significant byte.
i.e. : $(45BA72C)_{HEX}$ -> $(A72C)_{HEX}$

The number of words to read

Hmmm, you might think, "I will read 210 words, that's nonsense!", and I will tell you, Yes.
BUT, it does not work like that, bomb-makers use this function sneakily, so they don't use it's real purpose.
Please find those last 2 bytes (even if they don't relate to anything).

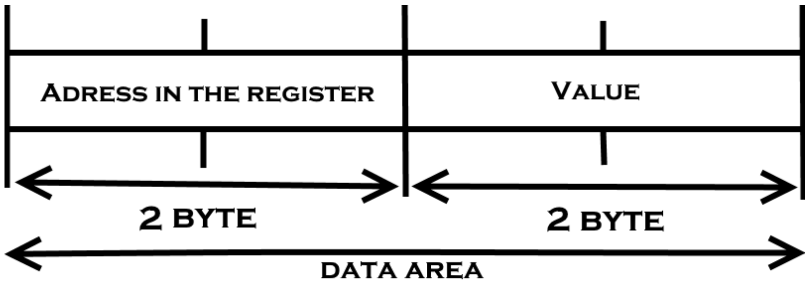
Take the address of the slave (decimal), multiply by the the number of function (4 or 6) then add the data number (decimal too).
Convert the result into Hexadecimal, simplify like just before. And you got it!

Send the data and you will be a hero (or not if you failed).

Function 06

Be careful, the keyboard is stronger than the sword!

You previously determine the slave address and the function "write".
But now you need to find what you need to write, and where (in the slave memory)?
See below for how the data area is composed when you want to write something.



I believe in you! You got this!

Address in the register

I hope you reviewed your binary operation because you will use it! (It's not a joke).
First step, take the 4th number (or letter, they are all ASCII), convert it with the ASCII table (in binary this time).
*When you've got the number, fill the rest of the 16 bits with 0 to convert it into the same number in 16 bits. example : 101 = 0000 0101 (for 8 bits).
You have to invert this number (NOT Operation), truth table below:

A	NOT A
1	0
0	1

You just have to convert the 16 bits binary number into an decimal number.
Simplify by cut the higher value. i.e. : 265486 -> 5486
Annnnndddd, you have the address in the register.

Value

The last part is the easiest! To find the value to enter at this adress, take the first digit of the serial number (in decimal) and subtract the number of function.
Then multiply this result by the 5th digit of the serial number (or letter, again in decimal) and simplify like above.

If you there, and you think it's finised, and yes, it's finished!
Now, enter all the parameters in the right order, and click on the send button.
YOU DID IT MY BOY !

Appendix ASCII table

Dec	Bin	Hex	Char	Dec	Bin	Hex	Char	Dec	Bin	Hex	Char	Dec	Bin	Hex	Char
0	0000 0000	00	[NUL]	32	0010 0000	20	space	64	0100 0000	40	@	96	0110 0000	60	`
1	0000 0001	01	[SOH]	33	0010 0001	21	!	65	0100 0001	41	A	97	0110 0001	61	a
2	0000 0010	02	[STX]	34	0010 0010	22	"	66	0100 0010	42	B	98	0110 0010	62	b
3	0000 0011	03	[ETX]	35	0010 0011	23	#	67	0100 0011	43	C	99	0110 0011	63	c
4	0000 0100	04	[EOT]	36	0010 0100	24	$	68	0100 0100	44	D	100	0110 0100	64	d
5	0000 0101	05	[ENQ]	37	0010 0101	25	%	69	0100 0101	45	E	101	0110 0101	65	e
6	0000 0110	06	[ACK]	38	0010 0110	26	&	70	0100 0110	46	F	102	0110 0110	66	f
7	0000 0111	07	[BEL]	39	0010 0111	27	'	71	0100 0111	47	G	103	0110 0111	67	g
8	0000 1000	08	[BS]	40	0010 1000	28	(	72	0100 1000	48	H	104	0110 1000	68	h
9	0000 1001	09	[TAB]	41	0010 1001	29	)	73	0100 1001	49	I	105	0110 1001	69	i
10	0000 1010	0A	[LF]	42	0010 1010	2A	*	74	0100 1010	4A	J	106	0110 1010	6A	j
11	0000 1011	0B	[VT]	43	0010 1011	2B	+	75	0100 1011	4B	K	107	0110 1011	6B	k
12	0000 1100	0C	[FF]	44	0010 1100	2C	,	76	0100 1100	4C	L	108	0110 1100	6C	l
13	0000 1101	0D	[CR]	45	0010 1101	2D	-	77	0100 1101	4D	M	109	0110 1101	6D	m
14	0000 1110	0E	[SO]	46	0010 1110	2E	.	78	0100 1110	4E	N	110	0110 1110	6E	n
15	0000 1111	0F	[SI]	47	0010 1111	2F	/	79	0100 1111	4F	O	111	0110 1111	6F	o
16	0001 0000	10	[DLE]	48	0011 0000	30	0	80	0101 0000	50	P	112	0111 0000	70	p
17	0001 0001	11	[DC1]	49	0011 0001	31	1	81	0101 0001	51	Q	113	0111 0001	71	q
18	0001 0010	12	[DC2]	50	0011 0010	32	2	82	0101 0010	52	R	114	0111 0010	72	r
19	0001 0011	13	[DC3]	51	0011 0011	33	3	83	0101 0011	53	S	115	0111 0011	73	s
20	0001 0100	14	[DC4]	52	0011 0100	34	4	84	0101 0100	54	T	116	0111 0100	74	t
21	0001 0101	15	[NAK]	53	0011 0101	35	5	85	0101 0101	55	U	117	0111 0101	75	u
22	0001 0110	16	[SYN]	54	0011 0110	36	6	86	0101 0110	56	V	118	0111 0110	76	v
23	0001 0111	17	[ETB]	55	0011 0111	37	7	87	0101 0111	57	W	119	0111 0111	77	w
24	0001 1000	18	[CAN]	56	0011 1000	38	8	88	0101 1000	58	X	120	0111 1000	78	x
25	0001 1001	19	[EM]	57	0011 1001	39	9	89	0101 1001	59	Y	121	0111 1001	79	y
26	0001 1010	1A	[SUB]	58	0011 1010	3A	:	90	0101 1010	5A	Z	122	0111 1010	7A	z
27	0001 1011	1B	[ESC]	59	0011 1011	3B	;	91	0101 1011	5B	[	123	0111 1011	7B	{
28	0001 1100	1C	[FS]	60	0011 1100	3C	<	92	0101 1100	5C	\	124	0111 1100	7C	|
29	0001 1101	1D	[GS]	61	0011 1101	3D	=	93	0101 1101	5D	]	125	0111 1101	7D	}
30	0001 1110	1E	[RS]	62	0011 1110	3E	>	94	0101 1110	5E	^	126	0111 1110	7E	~
31	0001 1111	1F	[US]	63	0011 1111	3F	?	95	0101 1111	5F	_	127	0111 1111	7F	[DEL]

When you need to convert a letter (or an number in some cases), you need to find the letter (or number) into the Char column

Bonus : You got Dec/Bin/Hex/Char conversion