# Writing Modbus Passwords from Babel Buster SPX, BB2-6010, or Modbus Gateways In General

---

**How to write a Modbus password using Babel Buster SPX (or BB2-6010)**

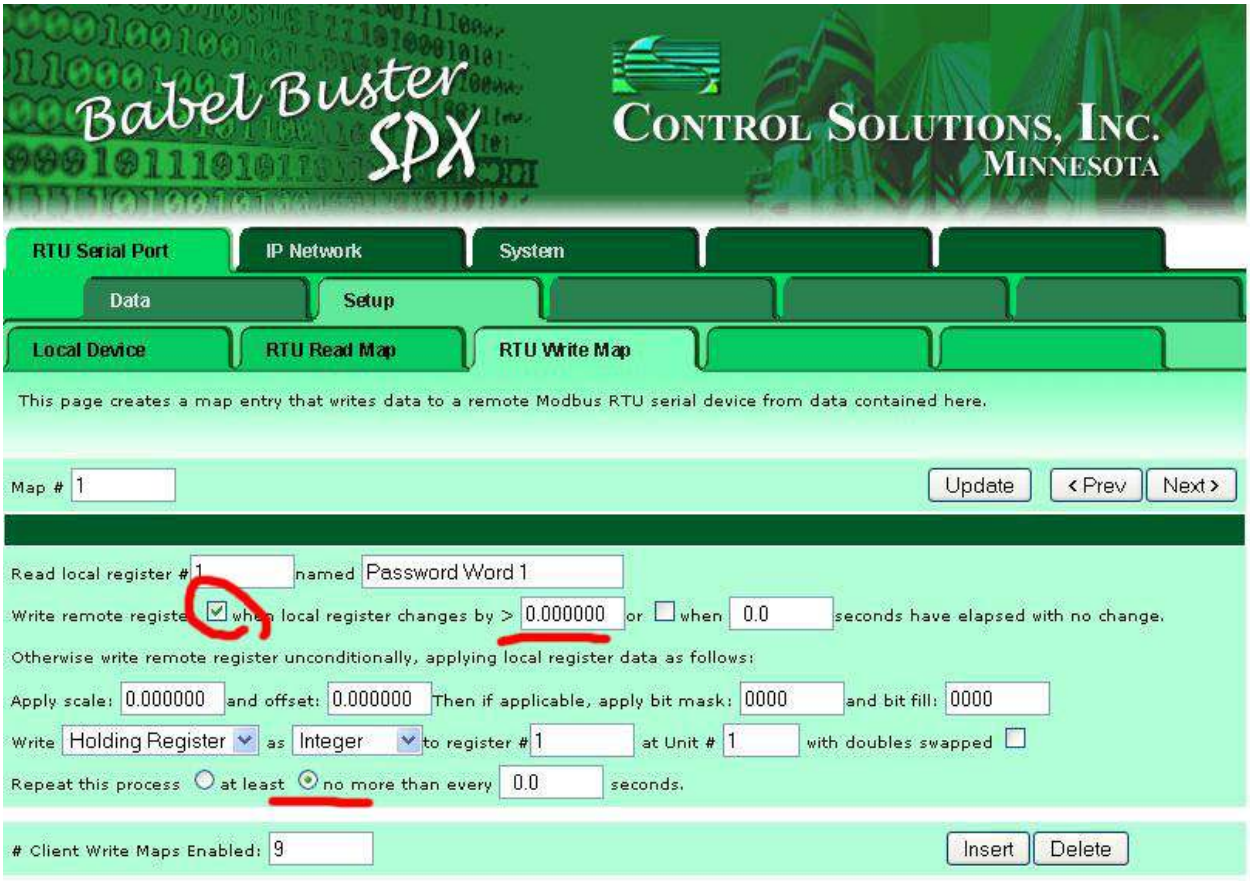
There is sometimes a requirement to write a ‘password’ or some sort of unlock code to a Modbus device. This unlocking usually requires writing a series of registers in one single request, generally with function code 16. The Babel Buster SPX includes the options necessary to accomplish this. While the SPX is illustrated here, the examples are applicable the Babel Buster BB2-6010 and most Control Solutions products with Modbus gateway capability.

First, determine the number of registers you will need to write. If the Modbus device talks about a "16-character” password, that means 8 registers with 2 characters per register. In the example that follows, we are assuming a 16-character password. This means we need to define 8 consecutive write maps. We will illustrate RTU, but the same principle applies to TCP.

You can enter almost everything you will need from the map list page illustrated above. Once you have created your series of registers, proceed to modify as illustrated in the following screen shot.

Note: It is important that consecutive registers be defined in consecutive write maps, and that the next write map following the password is NOT in consecutive order, otherwise more than the password will be sent in that single request.

The SPX (and all Control Solutions gateways) will attempt to send multiple registers in a single write request when they are found to be consecutive in the list of write maps.For purposes of multiple registers per write request, ‘consecutive’ means contiguous in the list of write maps, but also means the same device or slave address, and consecutively incrementing register numbers.




The goal is to get a series of 8 registers written in a single request, but only once, and only upon update of a single Modbus register. Getting the SPX to write only once, and only on demand, requires setting each of the 8 write maps as illustrated above. Pay attention to the check boxes – only the box illustrated as checked (ticked) should be checked, and all others should be left off. Also be sure to select the correct radio button after "Repeat this process”. It should say "no more than every 0.0 seconds”.

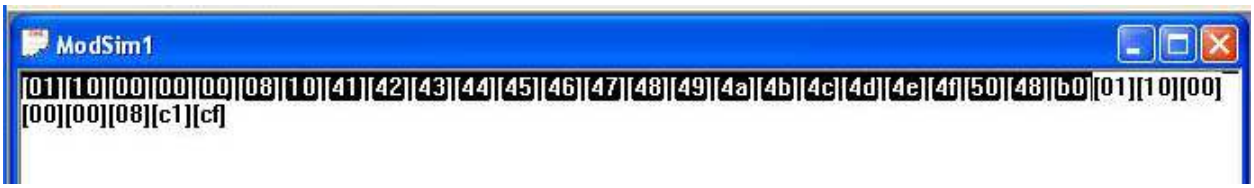
The combination of "changed by 0.0” and "no more than every 0.0 seconds” is a special case that tells the SPX (or BB2-6010) to only write to Modbus when an update is received via Modbus or

SNMP, and write to Modbus regardless of whether Modbus or SNMP actually changed any values.

You will need to write the correct password value to at least the first register in the series of registers that create the password. But the remaining registers can be predefined using the 'Constant' Action Rules as illustrated below:


The screen shot above shows a 16-character password set up as default values for a series of registers (registers 1-8 in this case). These values form the string 'ABCDEFGHIJKLMNOP'.

The result of writing the value 16706 to register 1 (either via Modbus TCP or SNMP), with the write maps defined as first illustrated above, will be writing a series of 8 registers to Modbus using function code 16, illustrated in the following traffic capture on the RTU network:




The bytes 41, 42, 43, etc, are the hexadecimal values for A, B, C, etc. Since two characters will be sent per single Modbus register, you must calculate a value the corresponds to two ASCII characters (assuming the password has been defined as an ASCII string – otherwise use whatever code you are instructed to use by the Modbus device manufacturer).

The letters AB are hexadecimal values 41 and 42. The first character will be in the high order byte. This means the concatenated value will be hex 4142 (or represented often as 0x4142 or 4142H). Now convert this to decimal using a hex to decimal calculator (or use your PC's calculator, enter in hex, and switch to decimal). The decimal number is 16706.

You can also calculate character codes in decimal. The letter A is decimal 65, and B is 66. To calculate the 16-bit value to write via the Modbus register, you would multiply the first character times 256, then add the second character as follows:

(65 * 256) + 66 = 16706

You can find ASCII code charts by simply doing a web search for "ASCII", or go to http://en.wikipedia.org/wiki/ASCII where you will find information including the following chart:

| Binary | Oct | Dec | Hex | Glyph |
|---|---|---|---|---|
| 010 0000 | 040 | 32 | 20 | SP |
| 010 0001 | 041 | 33 | 21 | ! |
| 010 0010 | 042 | 34 | 22 | " |
| 010 0011 | 043 | 35 | 23 | # |
| 010 0100 | 044 | 36 | 24 | $ |
| 010 0101 | 045 | 37 | 25 | % |
| 010 0110 | 046 | 38 | 26 | & |
| 010 0111 | 047 | 39 | 27 | ' |
| 010 1000 | 050 | 40 | 28 | ( |
| 010 1001 | 051 | 41 | 29 | ) |
| 010 1010 | 052 | 42 | 2A | * |
| 010 1011 | 053 | 43 | 2B | + |
| 010 1100 | 054 | 44 | 2C | , |
| 010 1101 | 055 | 45 | 2D | - |
| 010 1110 | 056 | 46 | 2E | . |
| 010 1111 | 057 | 47 | 2F | / |
| 011 0000 | 060 | 48 | 30 | 0 |
| 011 0001 | 061 | 49 | 31 | 1 |
| 011 0010 | 062 | 50 | 32 | 2 |
| 011 0011 | 063 | 51 | 33 | 3 |
| 011 0100 | 064 | 52 | 34 | 4 |
| 011 0101 | 065 | 53 | 35 | 5 |
| 011 0110 | 066 | 54 | 36 | 6 |
| 011 0111 | 067 | 55 | 37 | 7 |
| 011 1000 | 070 | 56 | 38 | 8 |
| 011 1001 | 071 | 57 | 39 | 9 |
| 011 1010 | 072 | 58 | 3A | : |
| 011 1011 | 073 | 59 | 3B | ; |
| 011 1100 | 074 | 60 | 3C | < |
| 011 1101 | 075 | 61 | 3D | = |
| 011 1110 | 076 | 62 | 3E | > |
| 011 1111 | 077 | 63 | 3F | ? |

| Binary | Oct | Dec | Hex | Glyph |
|---|---|---|---|---|
| 100 0000 | 100 | 64 | 40 | @ |
| 100 0001 | 101 | 65 | 41 | A |
| 100 0010 | 102 | 66 | 42 | B |
| 100 0011 | 103 | 67 | 43 | C |
| 100 0100 | 104 | 68 | 44 | D |
| 100 0101 | 105 | 69 | 45 | E |
| 100 0110 | 106 | 70 | 46 | F |
| 100 0111 | 107 | 71 | 47 | G |
| 100 1000 | 110 | 72 | 48 | H |
| 100 1001 | 111 | 73 | 49 | I |
| 100 1010 | 112 | 74 | 4A | J |
| 100 1011 | 113 | 75 | 4B | K |
| 100 1100 | 114 | 76 | 4C | L |
| 100 1101 | 115 | 77 | 4D | M |
| 100 1110 | 116 | 78 | 4E | N |
| 100 1111 | 117 | 79 | 4F | O |
| 101 0000 | 120 | 80 | 50 | P |
| 101 0001 | 121 | 81 | 51 | Q |
| 101 0010 | 122 | 82 | 52 | R |
| 101 0011 | 123 | 83 | 53 | S |
| 101 0100 | 124 | 84 | 54 | T |
| 101 0101 | 125 | 85 | 55 | U |
| 101 0110 | 126 | 86 | 56 | V |
| 101 0111 | 127 | 87 | 57 | W |
| 101 1000 | 130 | 88 | 58 | X |
| 101 1001 | 131 | 89 | 59 | Y |
| 101 1010 | 132 | 90 | 5A | Z |
| 101 1011 | 133 | 91 | 5B | [ |
| 101 1100 | 134 | 92 | 5C | \ |
| 101 1101 | 135 | 93 | 5D | ] |
| 101 1110 | 136 | 94 | 5E | ^ |
| 101 1111 | 137 | 95 | 5F | _ |

| Binary | Oct | Dec | Hex | Glyph |
|---|---|---|---|---|
| 110 0000 | 140 | 96 | 60 | ` |
| 110 0001 | 141 | 97 | 61 | a |
| 110 0010 | 142 | 98 | 62 | b |
| 110 0011 | 143 | 99 | 63 | c |
| 110 0100 | 144 | 100 | 64 | d |
| 110 0101 | 145 | 101 | 65 | e |
| 110 0110 | 146 | 102 | 66 | f |
| 110 0111 | 147 | 103 | 67 | g |
| 110 1000 | 150 | 104 | 68 | h |
| 110 1001 | 151 | 105 | 69 | i |
| 110 1010 | 152 | 106 | 6A | j |
| 110 1011 | 153 | 107 | 6B | k |
| 110 1100 | 154 | 108 | 6C | l |
| 110 1101 | 155 | 109 | 6D | m |
| 110 1110 | 156 | 110 | 6E | n |
| 110 1111 | 157 | 111 | 6F | o |
| 111 0000 | 160 | 112 | 70 | p |
| 111 0001 | 161 | 113 | 71 | q |
| 111 0010 | 162 | 114 | 72 | r |
| 111 0011 | 163 | 115 | 73 | s |
| 111 0100 | 164 | 116 | 74 | t |
| 111 0101 | 165 | 117 | 75 | u |
| 111 0110 | 166 | 118 | 76 | v |
| 111 0111 | 167 | 119 | 77 | w |
| 111 1000 | 170 | 120 | 78 | x |
| 111 1001 | 171 | 121 | 79 | y |
| 111 1010 | 172 | 122 | 7A | z |
| 111 1011 | 173 | 123 | 7B | { |
| 111 1100 | 174 | 124 | 7C | \| |
| 111 1101 | 175 | 125 | 7D | } |
| 111 1110 | 176 | 126 | 7E | ~ |

Article ID: 1
Created On: Thu, Dec 6, 2012 at 10:58 PM
Last Updated On: Wed, May 20, 2015 at 2:06 PM