CRACKING THE MITSUBISHI “KEYWORD”

A Technique to discover the password or “keyword” stored in Mitsubishi A series and FX series PLC’s

Written By Ian Sullivan

Application Software required:
Melsec GX Developer
Comlite 32 (Available free from http://www.rtcomm.com)

NOTE:
This technique is intended as a work around when you have been left with a password protected PLC and the original installer has gone bust!

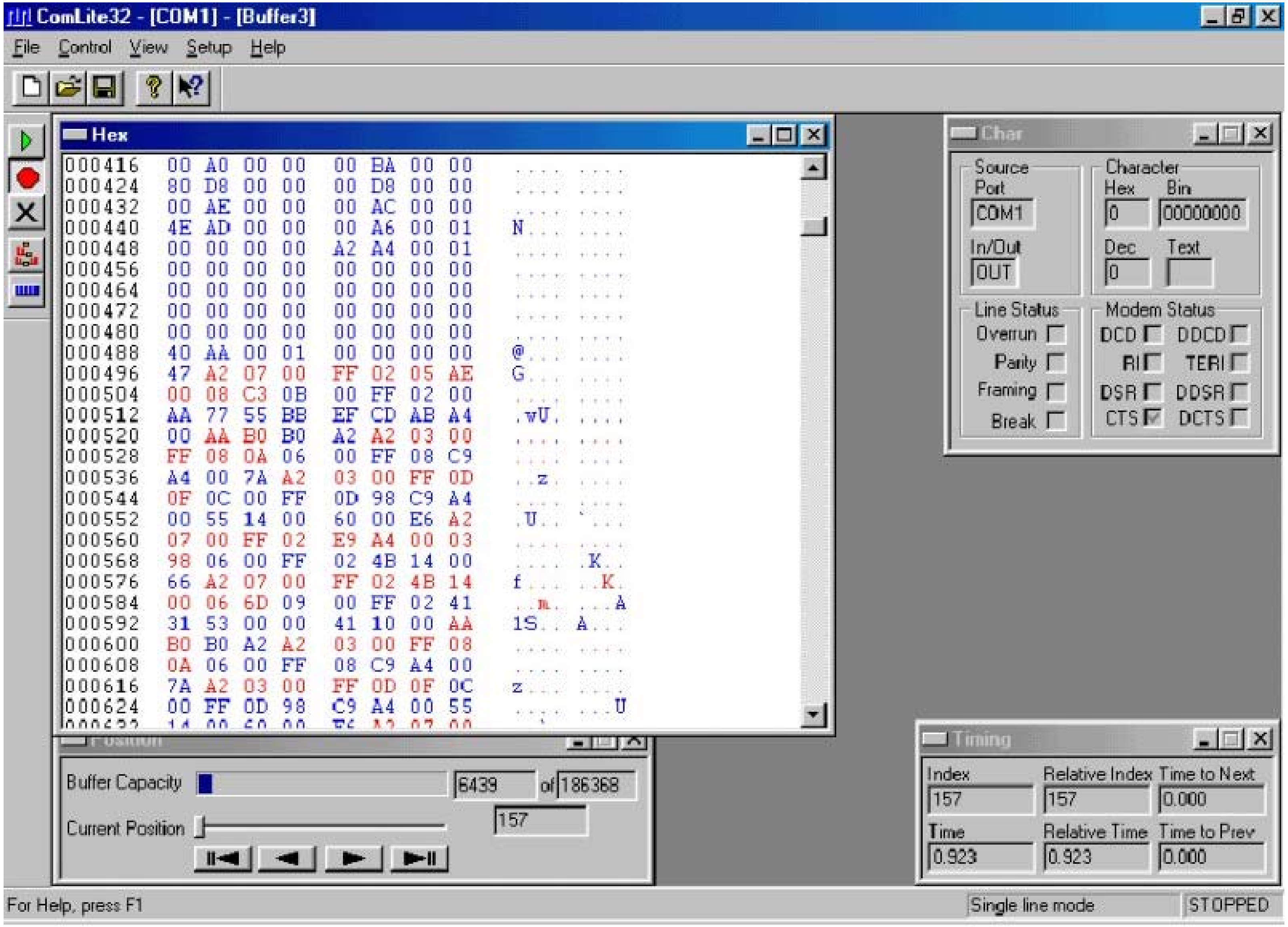
Introduction

The keyword within a Mitsubishi processor consists of a string of characters in the range 0-9 and A-F, in the case of the A-Series these are six characters long and in the FX they are 8 characters long. If a keyword has been set within the processor, it is required in order to read the program from the PLC to be able to monitor / modify the program. If you haven’t got the key, you can’t get in.
Mitsubishi Electrics UK technical support have been asked if it is possible to identify or get round the keyword, their answer is no, you must clear the PLC memory and start again. Not very good if you do not have the original code to begin with! I read an article on the forum on MrPLC.com (www.mrplc.com) where someone was asking the question on keywords and one user suggested the use of Comlite32 to discover what was going in and out of the com port. I can’t tell from the discussion threads whether or not it was successful so I had a go myself and documented the findings for the use of others.
(Note that ComLite32 does not work with NT/2000 – I used W98)

Setting The Keyword

A1S Processor

I had a distinct advantage over some users, whereby I did not have a protected PLC to crack, I had an unprotected one which I could set any keyword in it so I knew what I was looking for. On the A1s processor, using GX Developer, I set the keyword to ABCDEF, then closed the file. I started ComLite32 to monitor com1 in single line mode. I then did a "read from PLC" into a blank project. When "Param & Prog" is selected, switch to ComLite and start logging. Switch back to GX Developer and hit the execute button. A dialog then appears asking for the keyword, at this point type in any keyword (e.g. 123456), the dialog will appear again (because the keywords don't match). At this point, switch back to ComLite and see what you've got. It will appear something like this:



```
000416  00 A0 00 00  00 BA 00 00   ....  ....
000424  80 D8 00 00  00 D8 00 00   ....  ....
000432  00 AE 00 00  00 AC 00 00   ....  ....
000440  4E AD 00 00  00 A6 00 01   N...  ....
000448  00 00 00 00  A2 A4 00 01   ....  ....
000456  00 00 00 00  00 00 00 00   ....  ....
000464  00 00 00 00  00 00 00 00   ....  ....
000472  00 00 00 00  00 00 00 00   ....  ....
000480  00 00 00 00  00 00 00 00   ....  ....
000488  40 AA 00 01  00 00 00 00   @...  ....
000496  47 A2 07 00  FF 02 05 AE   G...  ....
000504  00 08 C3 0B  00 FF 02 00   ....  ....
000512  AA 77 55 BB  EF CD AB A4   .wU.  ....
000520  00 AA B0 B0  A2 A2 03 00   ....  ....
000528  FF 08 0A 06  00 FF 08 C9   ....  ....
000536  A4 00 7A A2  03 00 FF 0D   ..z.  ....
000544  0F 0C 00 FF  0D 98 C9 A4   ....  ....
000552  00 55 14 00  60 00 E6 A2   .U..  `...
000560  07 00 FF 02  E9 A4 00 03   ....  ....
000568  98 06 00 FF  02 4B 14 00   ....  .K..
000576  66 A2 07 00  FF 02 4B 14   f...  ..K.
000584  00 06 6D 09  00 FF 02 41   ..m.  ...A
000592  31 53 00 00  41 10 00 AA   1S..  A...
000600  B0 B0 A2 A2  03 00 FF 08   ....  ....
000608  0A 06 00 FF  08 C9 A4 00   ....  ....
000616  7A A2 03 00  FF 0D 0F 0C   z...  ....
000624  00 FF 0D 98  C9 A4 00 55   ....  ...U
```

Source
Port: COM1
In/Out: OUT

Character
Hex: 0
Bin: 00000000
Dec: 0
Text:

Line Status
Overrun □
Parity □
Framing □
Break □

Modem Status
DCD □ DDCD □
RI □ TERI □
DSR □ DDSR □
CTS ☑ DCTS □

Buffer Capacity 6439 of 186368
Current Position 157

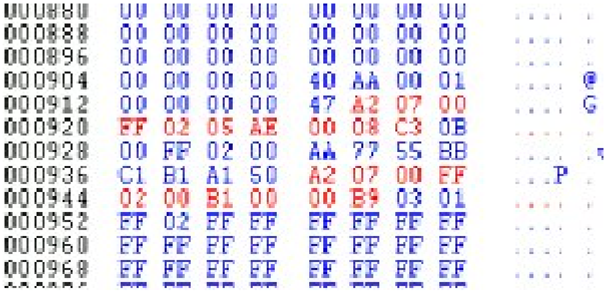
Index: 157
Relative Index: 157
Time to Next: 0.000
Time: 0.923
Relative Time: 0.923
Time to Prev: 0.000

For Help, press F1 | Single line mode | STOPPED

The red data is what your PC is sending, Blue data is sent from the PLC.
It looks like the PC sends a command to the PLC asking for the keyword, the PLC then sends it back and GX Developer compares the two, if they

match, it allows you to continue. The red A2 07 00 FF 02 05 AE 00 08 C3 looks like the request for the keyword. What the PLC sends back is 0B 00 FF 02 00 AA 77 55 BB EF CD AB A4 00. Looks meaningless doesn’t it? Until you know that the keyword that I set was ABCDEF. So, if you ignore the last two characters (A4) and work backwards in packets of two we get AB CD EF.

I thought that this must be too simple, so I used a different CPU and a different keyword. This time I used “A1B1C1” as the keyword. Did the same routine as above and this time I got:



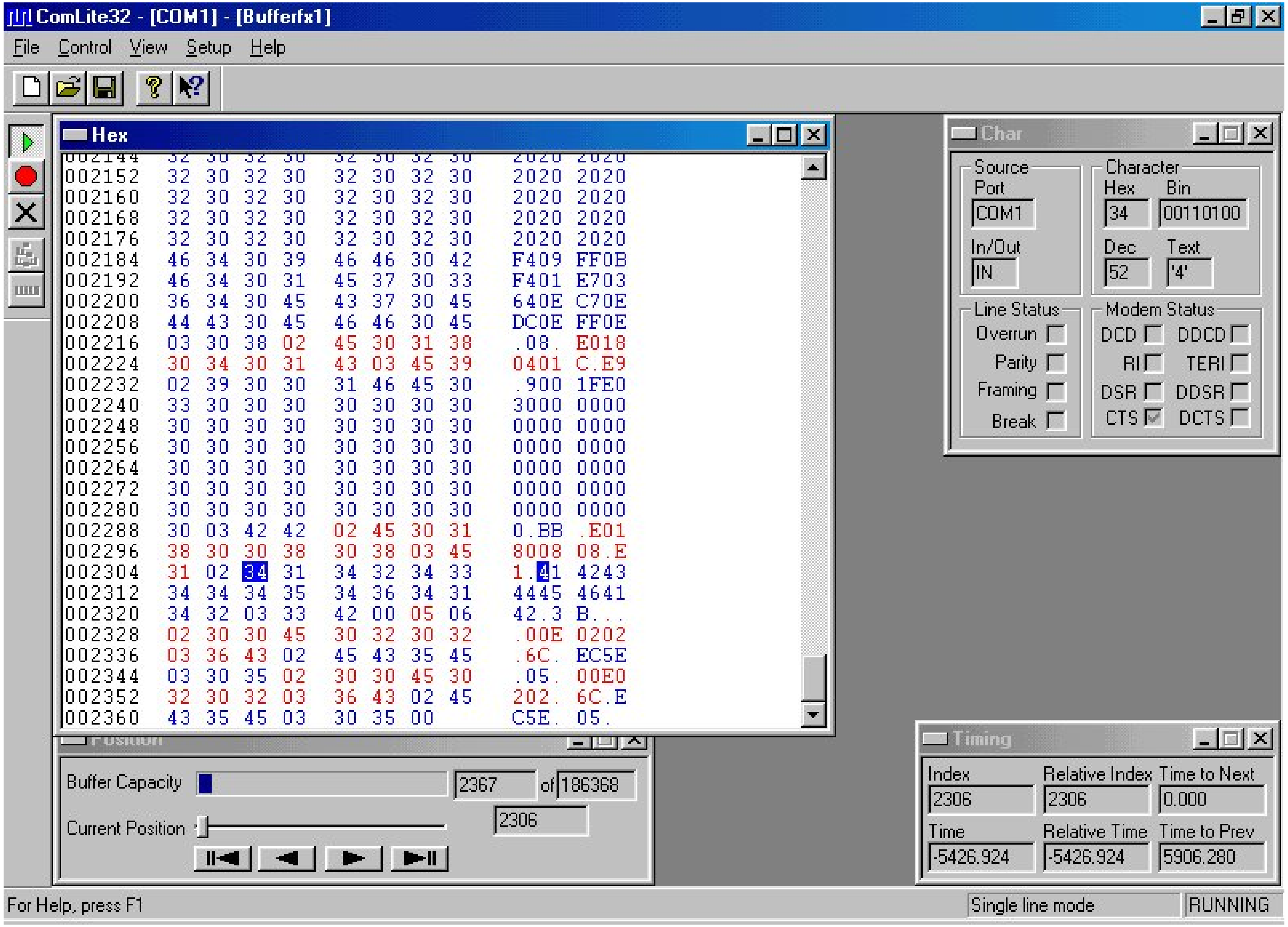
```
000880  00 00 00 00  00 00 00 00   ....  .
000888  00 00 00 00  00 00 00 00   ....  .
000896  00 00 00 00  00 00 00 00   ....  .
000904  00 00 00 00  40 AA 00 01   ....  @
000912  00 00 00 00  47 A2 07 00   ....  G
000920  FF 02 05 AE  00 08 C3 0B   ....  .
000928  00 FF 02 00  AA 77 55 BB   ....  .
000936  C1 B1 A1 50  A2 07 00 FF   ...P  .
000944  02 00 B1 00  00 B9 03 01   ....  .
000952  FF 02 FF FF  FF FF FF FF   ....  .
000960  FF FF FF FF  FF FF FF FF   ....  .
000968  FF FF FF FF  FF FF FF FF   ....  .
```

Same command to get the data from the PLC (A2 07 00 FF 02 05 AE 00 08 C3) and the data back was 0B 00 FF 02 00 AA 77 55 BB C1 B1 A1 A4. Working backwards ignoring the last packet we have A1 B1 C1

FX Series PLC’s

The keyword structure in the FXCPU is somewhat different to that of the A-Series, you now have 8 characters instead of 6. Though I did think that it would work the same way. Not quite. The FX CPU that I used was an FX2N128MR, with this model being newer and more advanced than the A1s may explain the differences, however, the A-Series technique may work on older FX’s or F1/F2 processors. I haven’t got one to try it on but would welcome any feedback.

I approached the FX the same as the A-Series using the keyword “ABCDEFAB”, with the ComLite logger running I could not see the pattern ABCDEFAB or AB EF CD AB anywhere in the data. The pattern that I did see was quite interesting though


ComLite32 - [COM1] - [Bufferfx1]
File Control View Setup Help
Hex
Char
Position
Timing

```
002144  32 30 32 30  32 30 32 30  2020 2020
002152  32 30 32 30  32 30 32 30  2020 2020
002160  32 30 32 30  32 30 32 30  2020 2020
002168  32 30 32 30  32 30 32 30  2020 2020
002176  32 30 32 30  32 30 32 30  2020 2020
002184  46 34 30 39  46 46 30 42  F409 FF0B
002192  46 34 30 31  45 37 30 33  F401 E703
002200  36 34 30 45  43 37 30 45  640E C70E
002208  44 43 30 45  46 46 30 45  DC0E FF0E
002216  03 30 38 02  45 30 31 38  .08. E018
002224  30 34 30 31  43 03 45 39  0401 C.E9
002232  02 39 30 30  31 46 45 30  .900 1FE0
002240  33 30 30 30  30 30 30 30  3000 0000
002248  30 30 30 30  30 30 30 30  0000 0000
002256  30 30 30 30  30 30 30 30  0000 0000
002264  30 30 30 30  30 30 30 30  0000 0000
002272  30 30 30 30  30 30 30 30  0000 0000
002280  30 30 30 30  30 30 30 30  0000 0000
002288  30 03 42 42  02 45 30 31  0.BB .E01
002296  38 30 30 38  30 38 03 45  8008 08.E
002304  31 02 34 31  34 32 34 33  1.41 4243
002312  34 34 34 35  34 36 34 31  4445 4641
002320  34 32 03 33  42 00 05 06  42.3 B...
002328  02 30 30 45  30 32 30 32  .00E 0202
002336  03 36 43 02  45 43 35 45  .6C. EC5E
002344  03 30 35 02  30 30 45 30  .05. 00E0
002352  32 30 32 03  36 43 02 45  202. 6C.E
002360  43 35 45 03  30 35 00     C5E. 05.
```

Buffer Capacity 2367 of 186368

Current Position 2306

Source

Port COM1

In/Out IN

Character

Hex 34 Bin 00110100

Dec 52 Text '4'

Line Status

Overrun □

Parity □

Framing □

Break □

Modem Status

DCD □ DDCD □

RI □ TERI □

DSR □ DDSR □

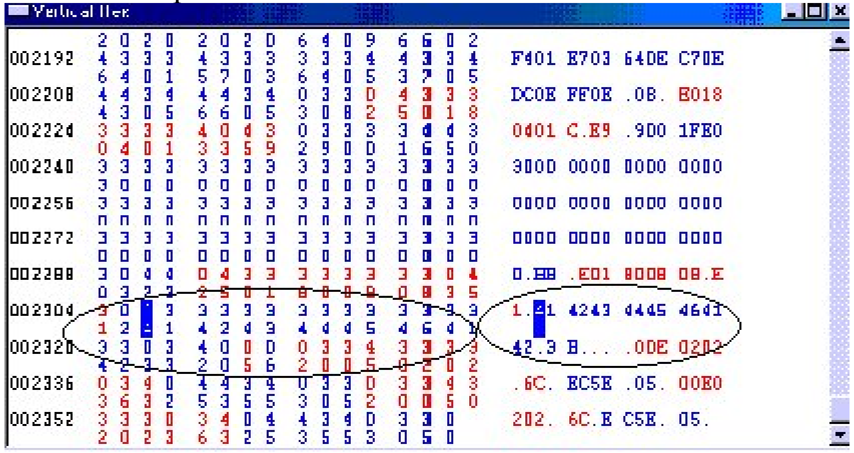
CTS ☑ DCTS □

Index	Relative Index	Time to Next
2306	2306	0.000
Time	**Relative Time**	**Time to Prev**
-5426.924	-5426.924	5906.280

For Help, press F1 | Single line mode | RUNNING

I kept seeing a pattern of 34’s, i.e. 34, 31, 34, 32, 34, 33, 34, 34, 34, 35, 34, 36, 34, 31, 34, 32.
We know in the code that each number represents and ASCII character, these numbers were then translated from ASCII and the result was 4 1 4 2 4 3 4 4 4 5 4 6 4 1 4 2. Group them into twos and you get 41 42 43 44 45 46 41 42
What is the character equivalent if these values are in ASCII??
They magically appear to be A B C D E F !!

The request for data from PC to PLC works in a similar way: 02 45 30 31 38 30 30 38 30 38 03 45 31.
The next block of data is the key sent back from the PLC, this time it’s in the correct order. Ignoring the first two characters (the 02) the next sixteen blocks (I’m calling a 34 one block) represent your keyword. All you have to do is take the first two blocks (34 31) covert each one into an ASCII character (4 1) put the two together (41) and then convert that hex number into an ASCII character (A), do this for the next 7 blocks of two that you can see and that should equal the keyword.
It looked simpler to me when I viewed it as “Vertical Hex” in ComLite – I could see straight away how they had split the ASCII number up



```
        2 0 2 0  2 0 2 0  6 4 0 9  6 6 0 2
002192  4 3 3 3  4 3 3 3  3 3 3 4  4 3 3 4   F401 E703 640E C70E
        6 4 0 1  5 7 0 3  6 4 0 5  3 7 0 5
002208  4 4 3 4  4 4 3 4  0 3 3 0  4 3 3 3   DC0E FF0E .08. E018
        4 3 0 5  6 6 0 5  3 0 8 2  5 0 1 8
002224  3 3 3 3  4 0 4 3  0 3 3 3  3 4 4 3   0401 C.E9 .900 1FE0
        0 4 0 1  3 3 5 9  2 9 0 0  1 6 5 0
002240  3 3 3 3  3 3 3 3  3 3 3 3  3 3 3 3   3000 0000 0000 0000
        3 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0
002256  3 3 3 3  3 3 3 3  3 3 3 3  3 3 3 3   0000 0000 0000 0000
        0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0
002272  3 3 3 3  3 3 3 3  3 3 3 3  3 3 3 3   0000 0000 0000 0000
        0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0
002288  3 0 4 4  0 4 3 3  3 3 3 3  3 3 0 4   0.BB .E01 8008 08.E
        0 3 2 2  2 5 0 1  8 0 0 8  0 8 3 5
002304  3 0 3 3  3 3 3 3  3 3 3 3  3 3 3 3   1.41 4243 4445 4641
        1 2 4 1  4 2 4 3  4 4 4 5  4 6 4 1
002320  3 3 0 3  4 0 0 0  0 3 3 4  3 3 3 3   42.3 B... .00E 0202
        4 2 3 3  2 0 5 6  2 0 0 5  0 2 0 2
002336  0 3 4 0  4 4 3 4  0 3 3 0  3 3 4 3   .6C. EC5E .05. 00E0
        3 6 3 2  5 3 5 5  3 0 5 2  0 0 5 0
002352  3 3 3 0  3 4 0 4  4 3 4 0  3 3 0     202. 6C.E C5E. 05.
        2 0 2 3  6 3 2 5  3 5 5 3  0 5 0
```

Now, looking back, I can see it in the normal hex view though, but there are always more than one way of solving a problem.

What's Next?? Rockwell Automation - SLC500 Series password protection - DONE!! Click here to find out how

Ideas & Comments are welcome at navillusi@hotmail.com