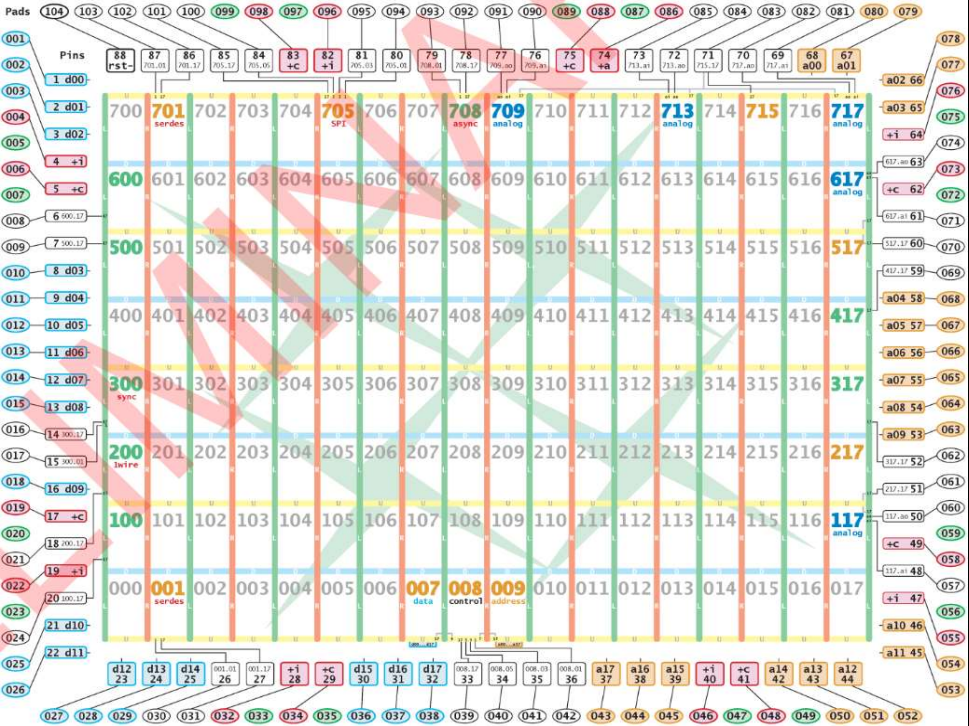
Reading and writing flash on GA144

James Bowman

July 22, 2017

GA144

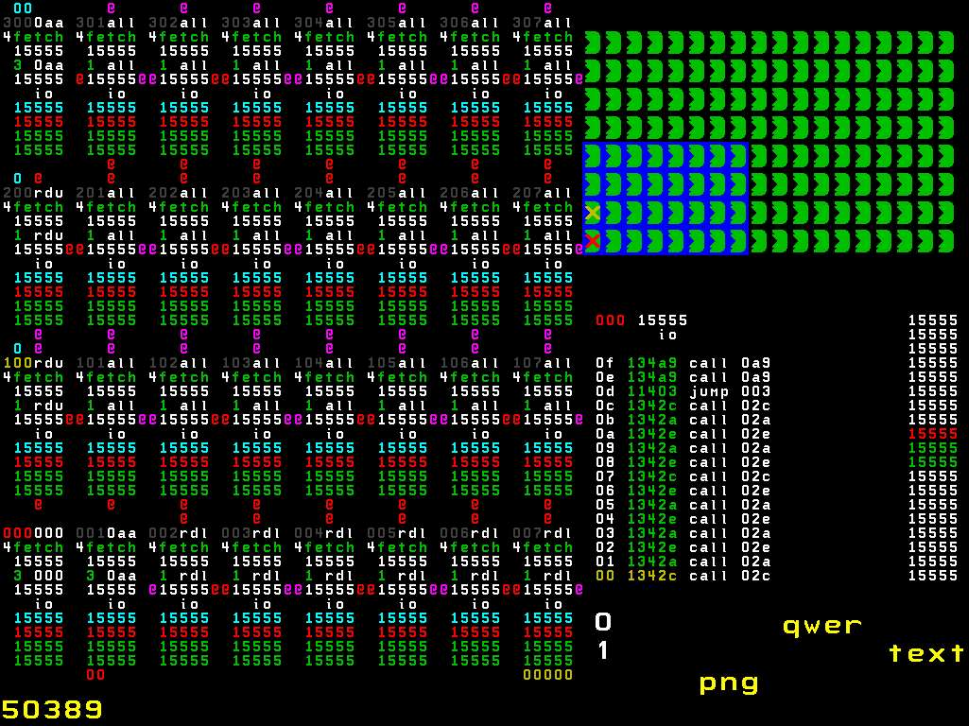
http://www.greenarraychips.com/


Pads
104 103 102 101 100 099 098 097 096 095 094 093 092 091 090 089 088 087 086 085 084 083 082 081 080 079
Pins
88 rst-
87 705.01
86 708.17
85 705.17
84 705.06
83 +c
82 +i
81 705.03
80 705.01
79 708.01
78 708.17
77 705.00
76 705.05
75 +c
74 +a
73 713.a1
72 713.a0
71 715.17
70 717.a0
69 717.a1
68 a00
67 a01
001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026
1 d00
2 d01
3 d02
4 +i
5 +c
6 600.17
7 500.17
8 d03
9 d04
10 d05
11 d06
12 d07
13 d08
14 300.17
15 300.01
16 d09
17 +c
18 200.17
19 +i
20 100.17
21 d10
22 d11
078 077 076 075 074 073 072 071 070 069 068 067 066 065 064 063 062 061 060 059 058 057 056 055 054 053
a02 66
a03 65
+i 64
617.a0 63
+c 62
617.a1 61
517.17 60
417.17 59
a04 58
a05 57
a06 56
a07 55
a08 54
a09 53
317.17 52
217.17 51
117.a0 50
+c 49
117.a1 48
+i 47
a10 46
a11 45
700 701 serdes 702 703 704 705 SPI 706 707 708 async 709 analog 710 711 712 713 analog 714 715 716 717 analog
600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 analog
500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517
400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417
300 sync 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317
200 1wire 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217
100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 analog
000 001 serdes 002 003 004 005 006 007 data 008 control 009 address 010 011 012 013 014 015 016 017
d12 23 d13 24 d14 25 001.01 26 001.17 27 +i 28 +c 29 d15 30 d16 31 d17 32 008.12 33 008.05 34 008.03 35 008.01 36 a17 37 a16 38 a15 39 +i 40 +c 41 a14 42 a13 43 a12 44
027 028 029 030 031 032 033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052

it's a kind of computer

For a taste of what it's like to program a node:

`https://mschuldt.github.io/www.colorforth.com/inst.htm`



This is the arrayForth simulator/IDE

Top right is a picture of the whole chip, 18x8

On the left is the state of the highlighted 8x4 region of CPUs

To the right is a memory dump of one node.

My alternative toolchain is at

https://github.com/jamesbowman/ga144tools/blob/master/src/flashwrite.ga

flash

The GA144 eval board uses a SST25WF080, an 8 Mbit flash

it’s a kind of memory

slow to write, slower to erase

For example, for 4K on SST25W:

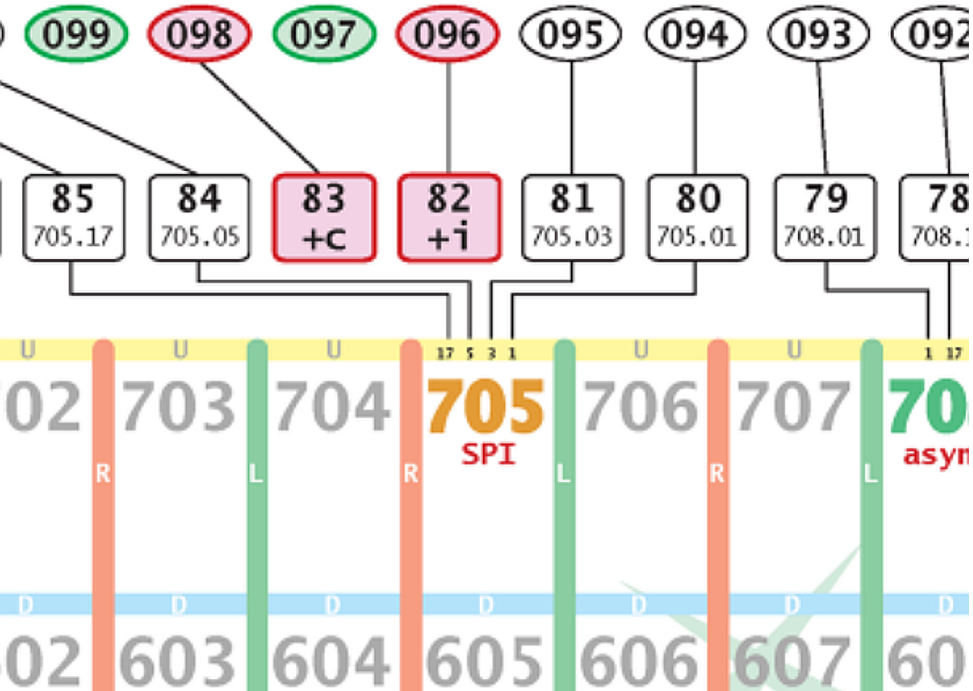
write .1 ms $(4096 \times 25\mu s)$

erase 35 ms

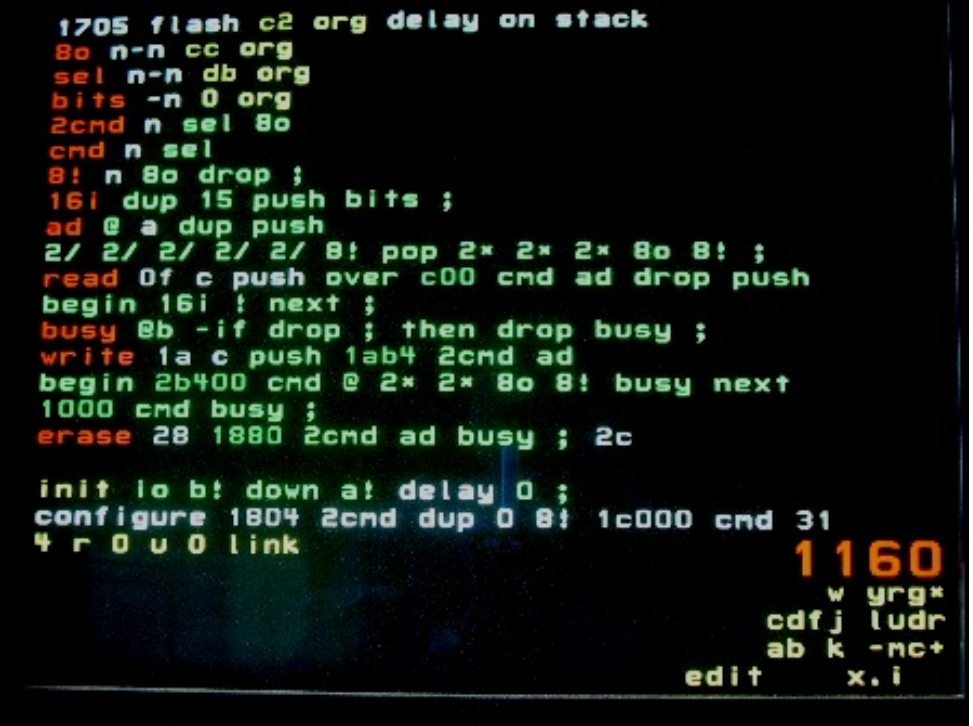
On MX25L:

write 22.4 ms

erase 60 ms

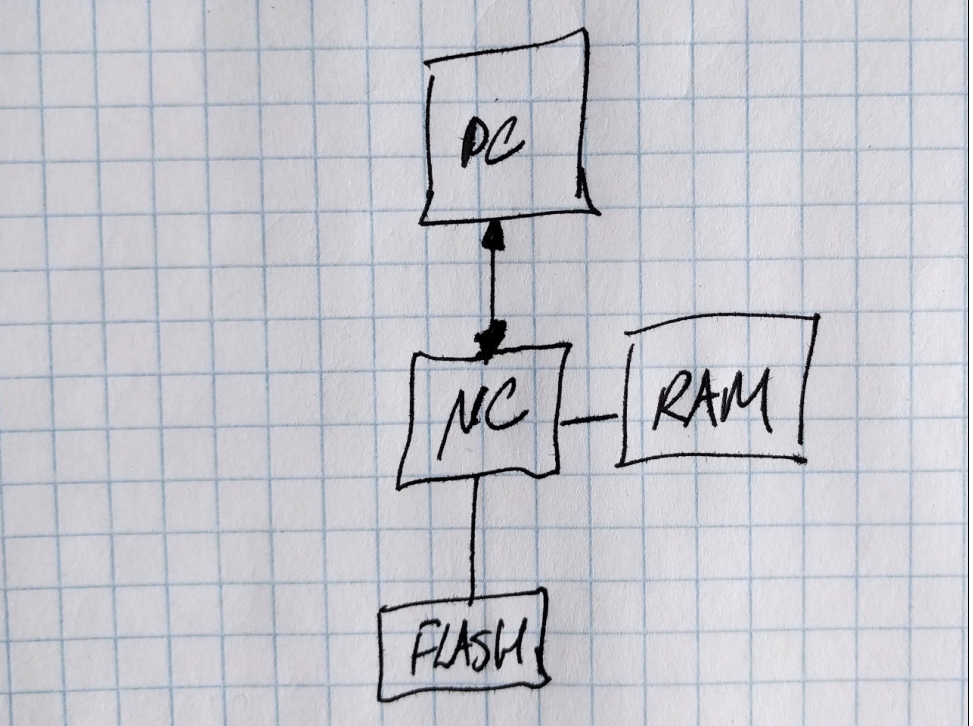


The source for node 705's ROM is in block 1428 at `http://excamera.com/files/cf.html`



Chuck's code is at:

https://mschuldt.github.io/www.colorforth.com/flash.htm

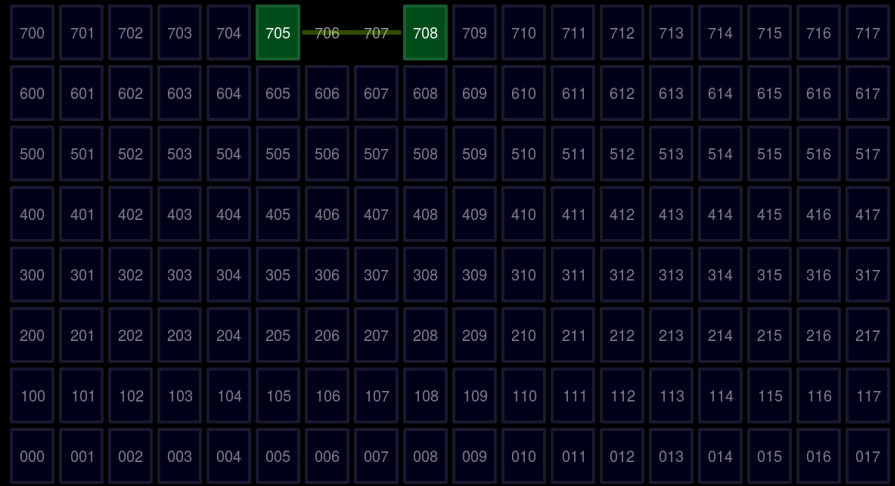


A traditional way of managing flash contents.

- The **PC** has the original flash image and runs a driver program
- the microcontroller μc writes blocks into **RAM** then copies them to **flash**

A bootloader is one example of this.
For GA144, there's no microcontroller or RAM. So must do something different.

reading



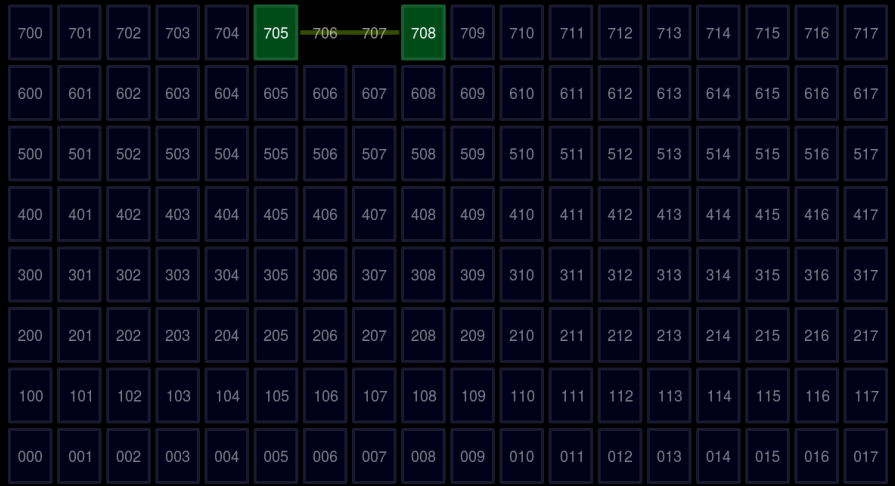
Node 705 reads the flash, sends the data east. Node 708 is a UART transmitter.

From the PC's point of view, it loads the program and the GA144 dumps the flash contents.

Code is at:

`https://github.com/jamesbowman/ga144tools/blob/master/src/flashread.ga`

writing

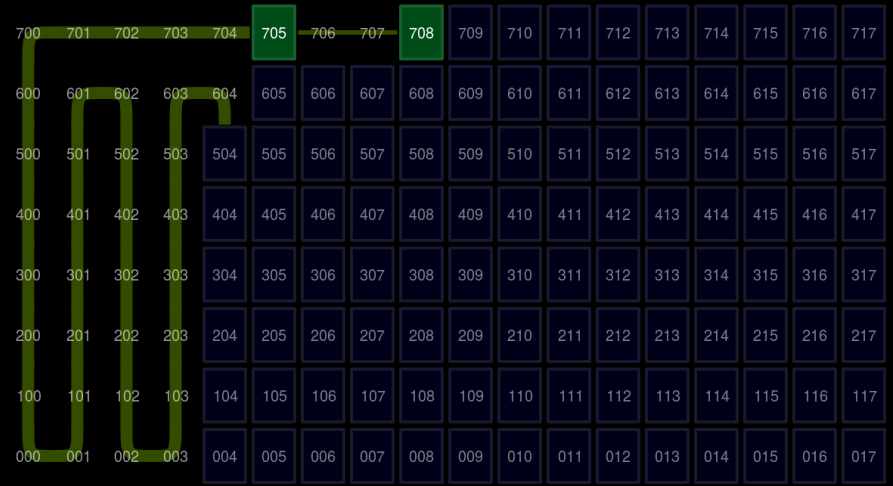


This won't work. The async node needs to wait for each flash operation.

need 4K of storage

But nobody has really figured out how to make an effective large RAM yet.

RECITE nodes write out their contents, then become wire nodes.

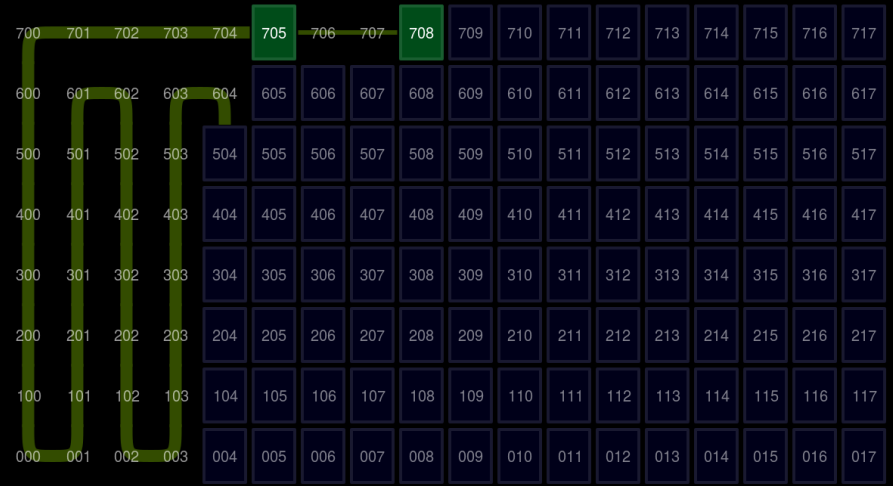


Code is at

`https://github.com/jamesbowman/ga144tools/blob/master/src/flashwrite.ga`

```
\ Send contents to DST then carry SRC to DST
    @p @p @p @p
      SRC
      DST
      3
      60
    push a! b!
: main
    @+ !b unext
    a!
: wire
    @ !b jump wire
```

Each RECITE node uses 3 words for program. The remaining 61 words of program are available for data.
This is why the constants 3 and 60 (61-1) appear here.
So each node is 122 bytes, and 4K requires 33 nodes.



Nodes 604 through 704 hold 4K of data.

Node 705 erases the flash, then reads from its WEST port to get each word, and writes it to flash.

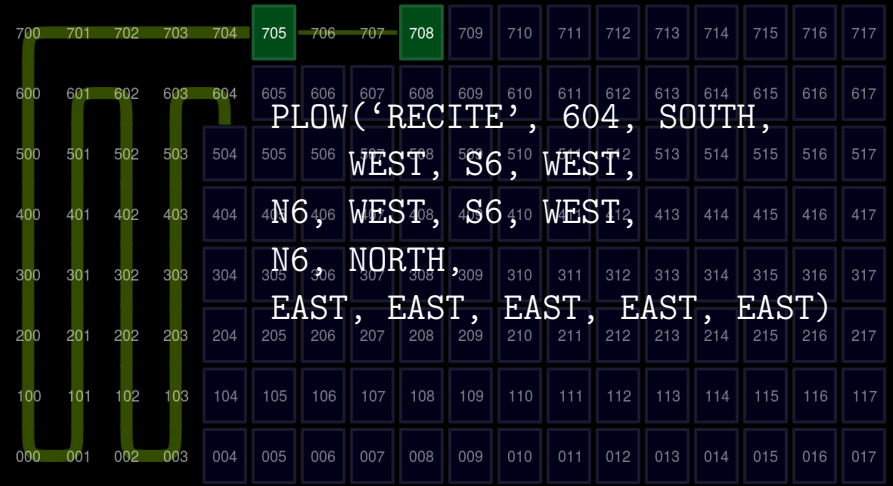
After it completes it sends a byte to node 708, which outputs it.

```
P=/dev/ttyUSB0

./flash.py $P write war_and_peace.txt

./flash.py $P read new.txt 3359550
```

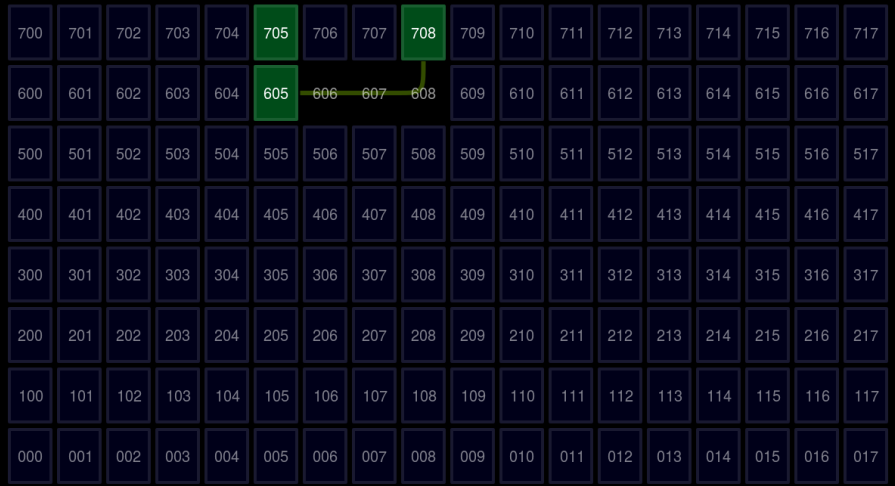
The flash utility is now included in ga144tools

Write speed is about 18K/s


PLOW


PLOW(‘RECITE’, 604, SOUTH,
WEST, S6, WEST,
N6, WEST, S6, WEST,
N6, NORTH,
EAST, EAST, EAST, EAST, EAST)

Next


700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717
600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617
500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517
400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417
300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317
200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217
100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
000 001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017