Talk on SVFIG Forth Day

Insight of esp32eForth_14

Sam Suan Chen

From FigTaiwan

November 18, 2017

9:15 ~ 9:40

Abstracts

. esp32eforth is for esp32WifiBoy.

. Derived from espForth_44 on esp8266.

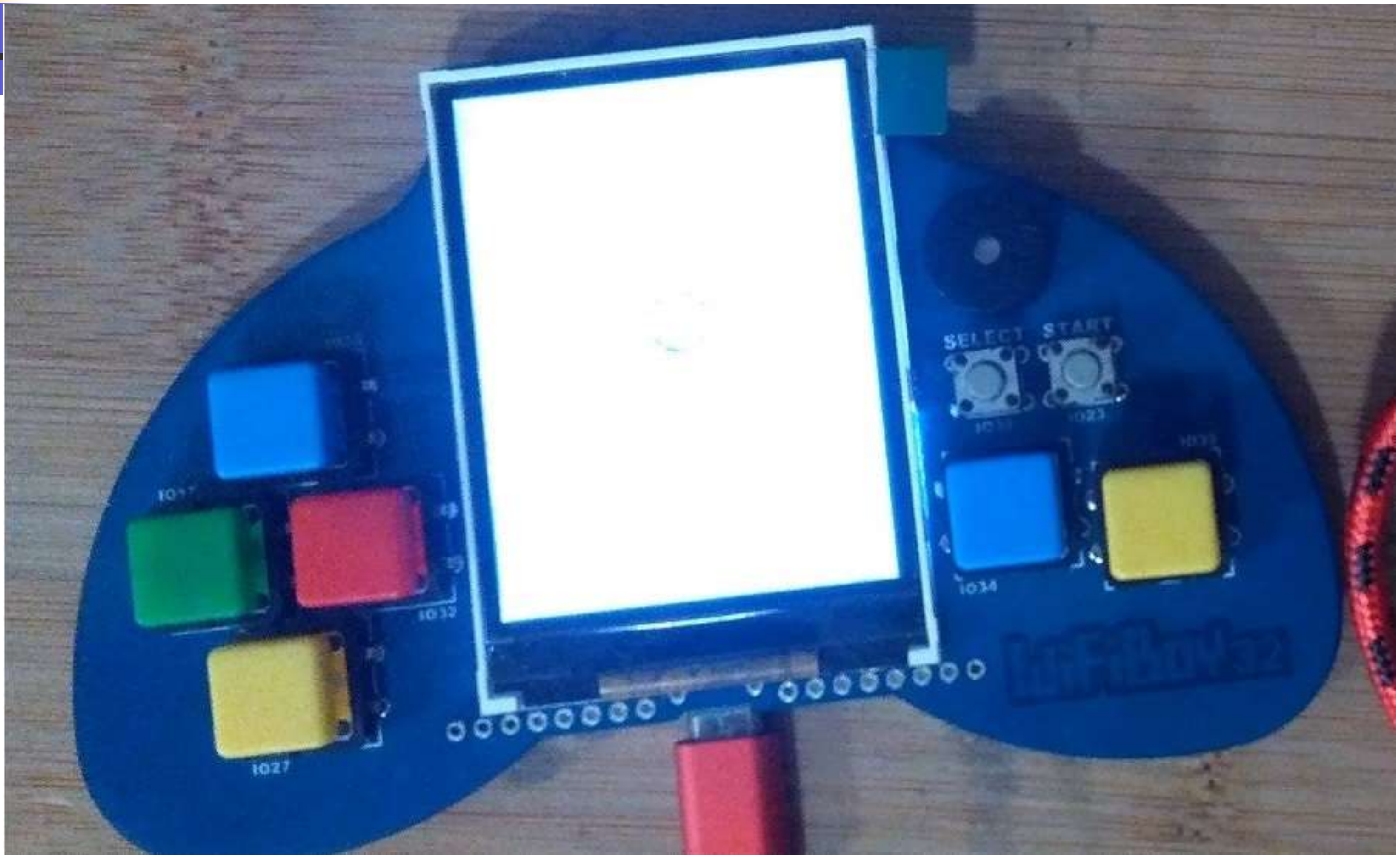
. 2-level token threaded, sequence of byte codes and sequence of forth word's CFAs.

. No WIFI yet. Only with Serial.

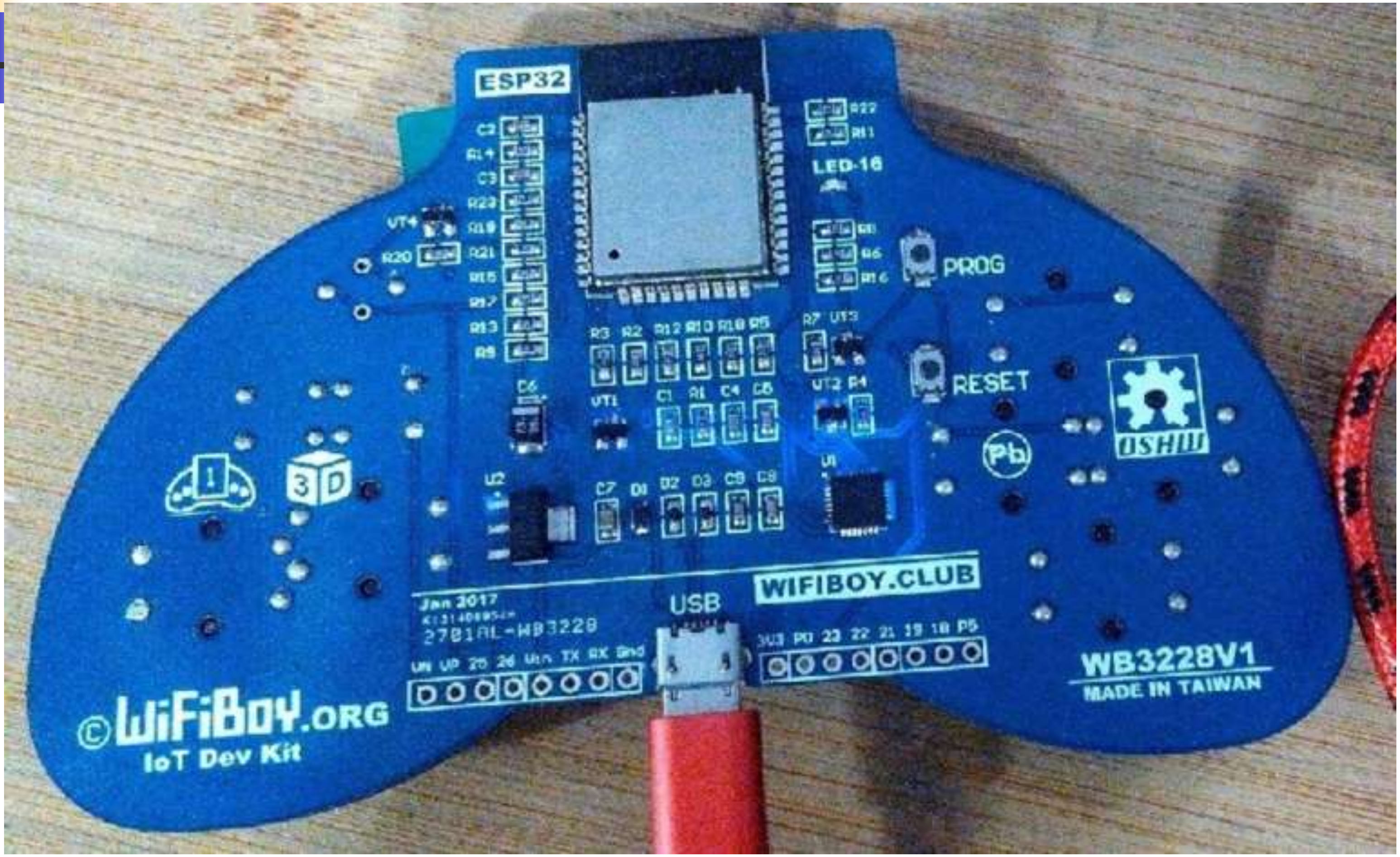
. Tracing mechanism and Debugger allow insight into esp32eforth.

. On boot, fully trace all words called by COLD, in all levels, then get into Debugger.

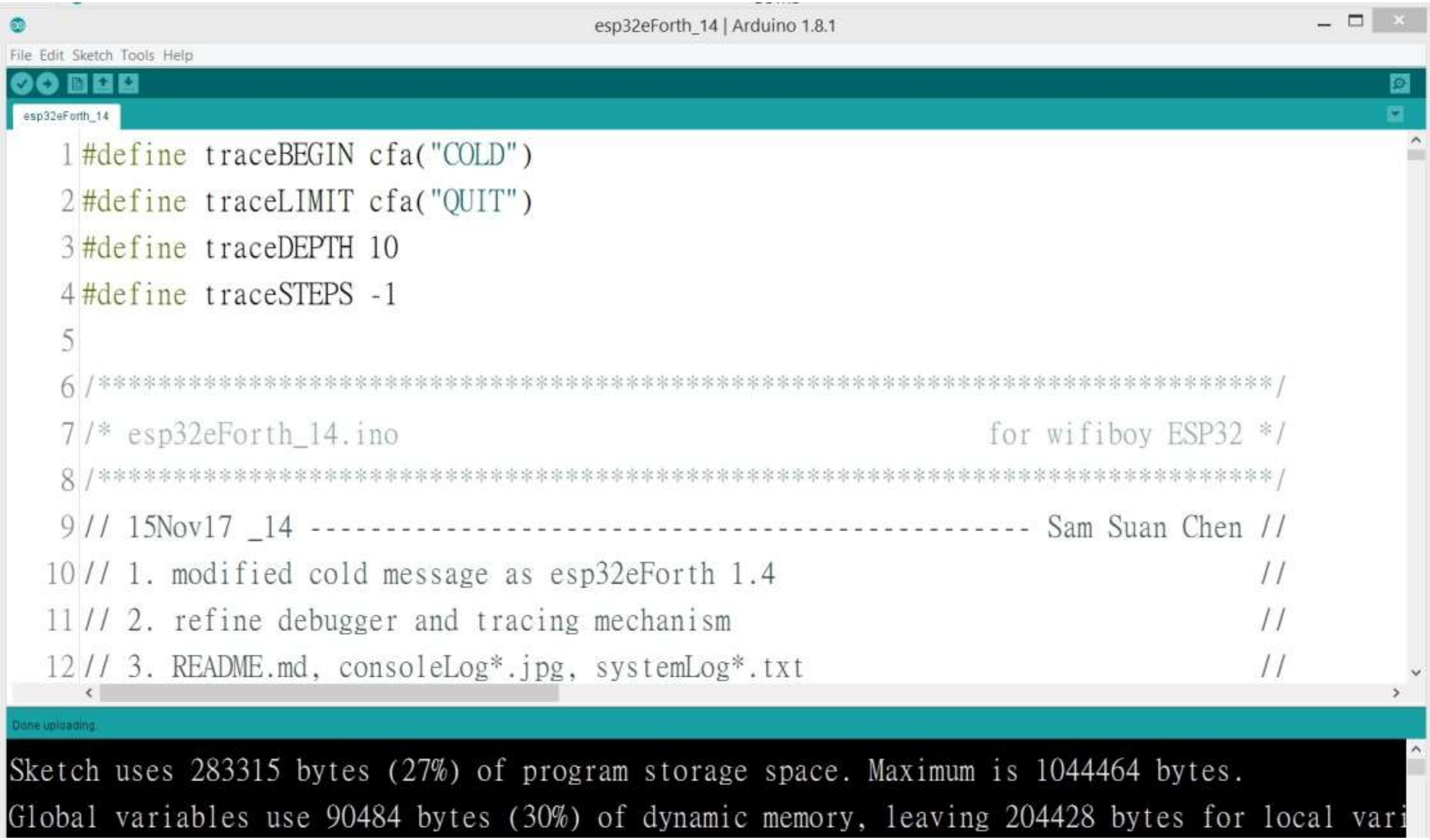
The WifiBoy esp32



Back side of WifiBoy esp32.



Setup tracing

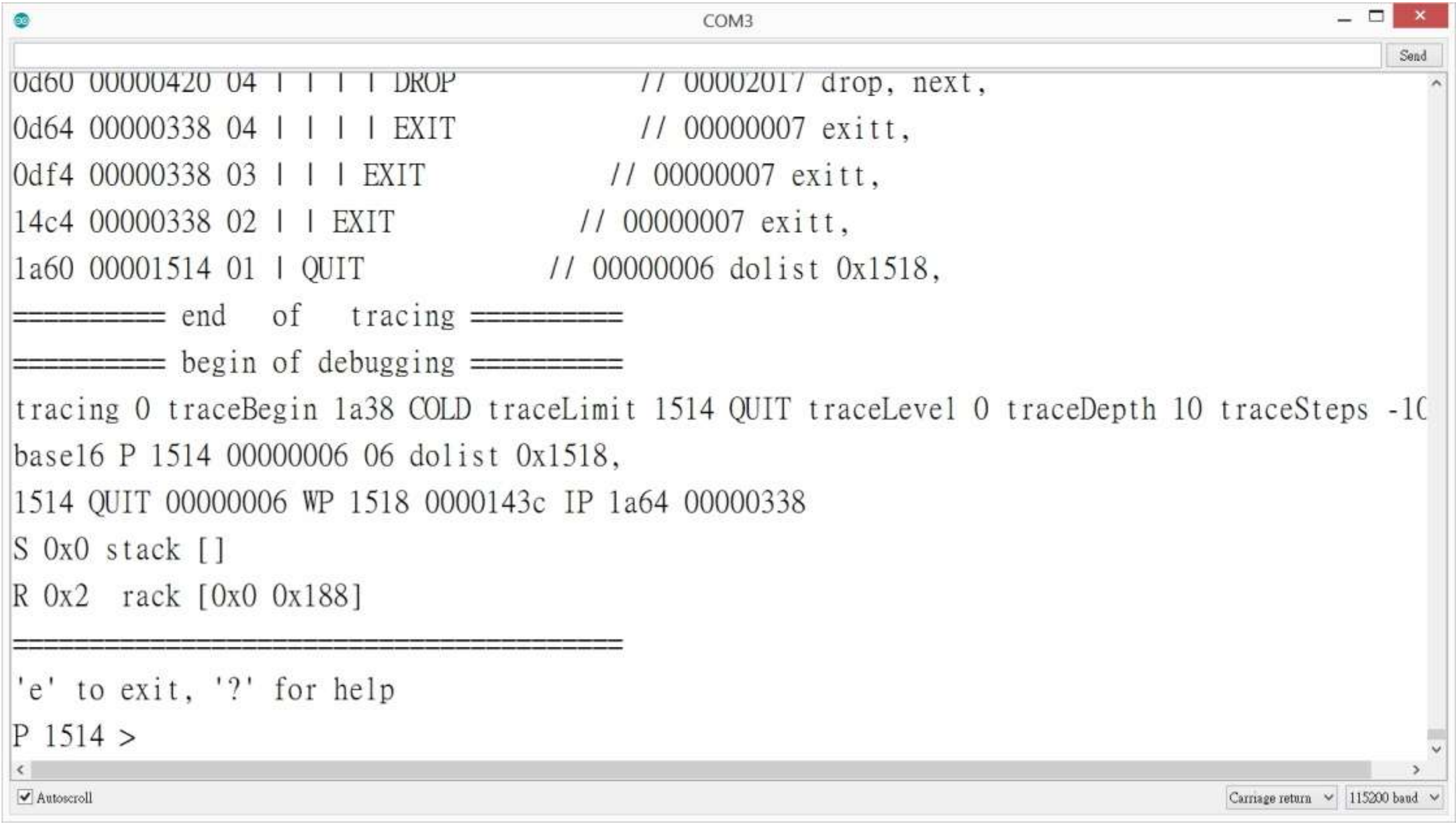


```
#define traceBEGIN cfa("COLD")
#define traceLIMIT cfa("QUIT")
#define traceDEPTH 10
#define traceSTEPS -1

/******************************************************************************/
/* esp32eForth_14.ino                                       for wifiboy ESP32 */
/******************************************************************************/
// 15Nov17 _14 ------------------------------------------------ Sam Suan Chen //
// 1. modified cold message as esp32eForth 1.4                               //
// 2. refine debugger and tracing mechanism                                  //
// 3. README.md, consoleLog*.jpg, systemLog*.txt                             //
```

Done uploading.

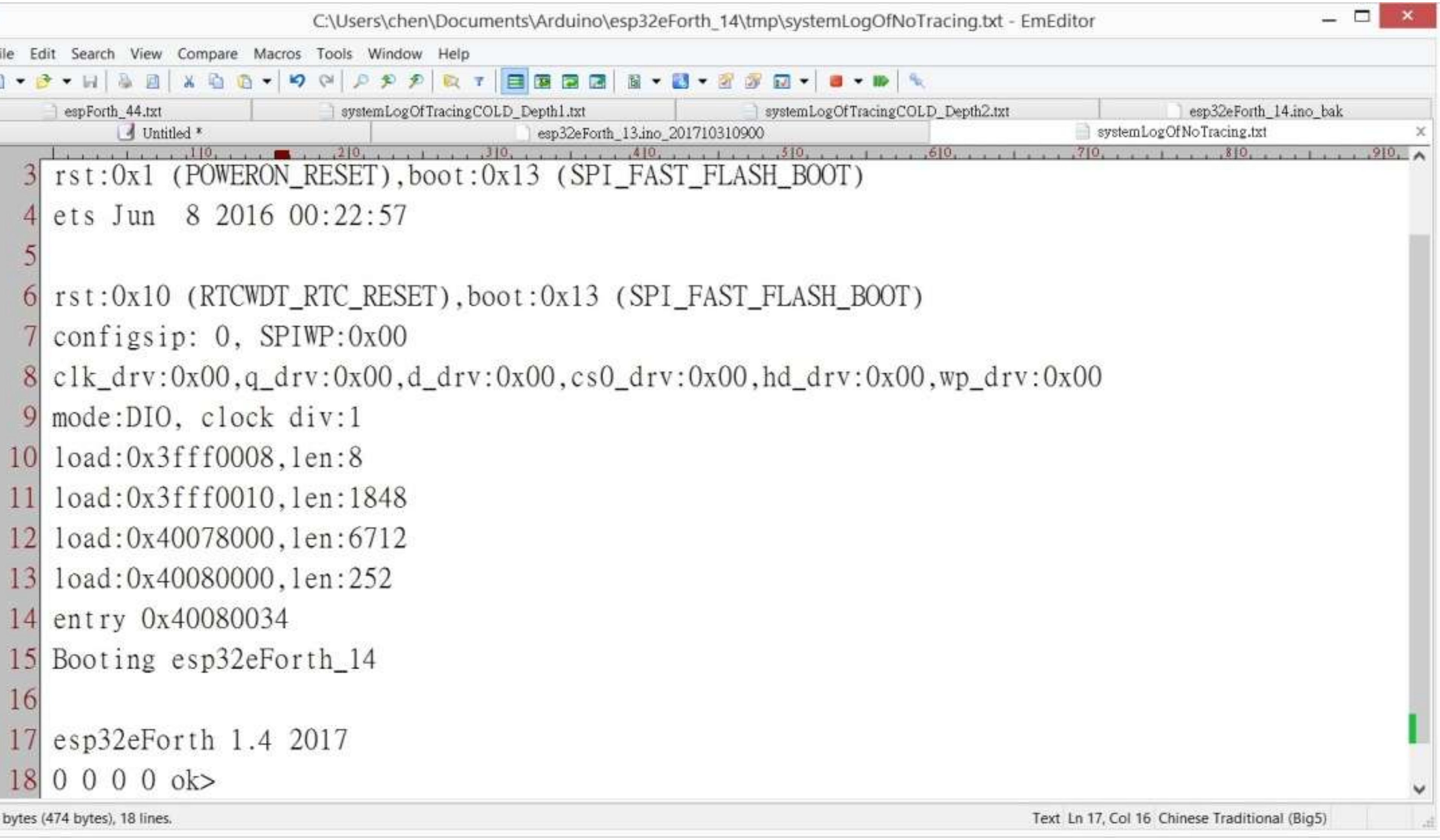
```
Sketch uses 283315 bytes (27%) of program storage space. Maximum is 1044464 bytes.
Global variables use 90484 bytes (30%) of dynamic memory, leaving 204428 bytes for local vari
```

After tracing, get into debugger



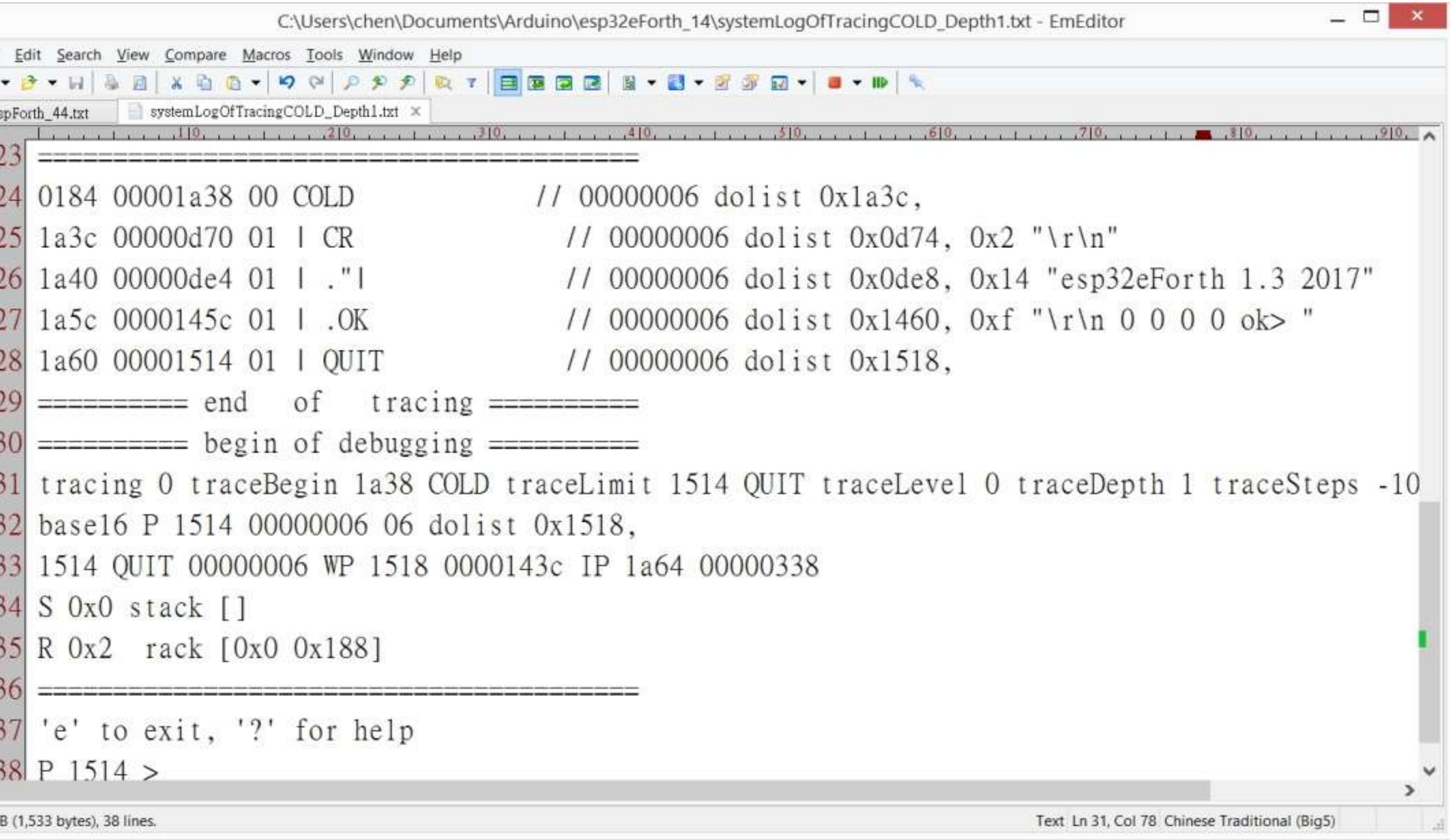
```
0d60 00000420 04 | | | | DROP           // 00002017 drop, next,
0d64 00000338 04 | | | | EXIT           // 00000007 exitt,
0df4 00000338 03 | | | EXIT           // 00000007 exitt,
14c4 00000338 02 | | EXIT           // 00000007 exitt,
1a60 00001514 01 | QUIT           // 00000006 dolist 0x1518,
========== end  of  tracing ==========
========== begin of debugging ==========
tracing 0 traceBegin 1a38 COLD traceLimit 1514 QUIT traceLevel 0 traceDepth 10 traceSteps -1
base16 P 1514 00000006 06 dolist 0x1518,
1514 QUIT 00000006 WP 1518 0000143c IP 1a64 00000338
S 0x0 stack []
R 0x2  rack [0x0 0x188]
========================================
'e' to exit, '?' for help
P 1514 >
```

Without tracing


C:\Users\chen\Documents\Arduino\esp32eForth_14\tmp\systemLogOfNoTracing.txt - EmEditor
ile Edit Search View Compare Macros Tools Window Help
espForth_44.txt
systemLogOfTracingCOLD_Depth1.txt
systemLogOfTracingCOLD_Depth2.txt
esp32eForth_14.ino_bak
Untitled *
esp32eForth_13.ino_201710310900
systemLogOfNoTracing.txt
bytes (474 bytes), 18 lines.
Text Ln 17, Col 16 Chinese Traditional (Big5)

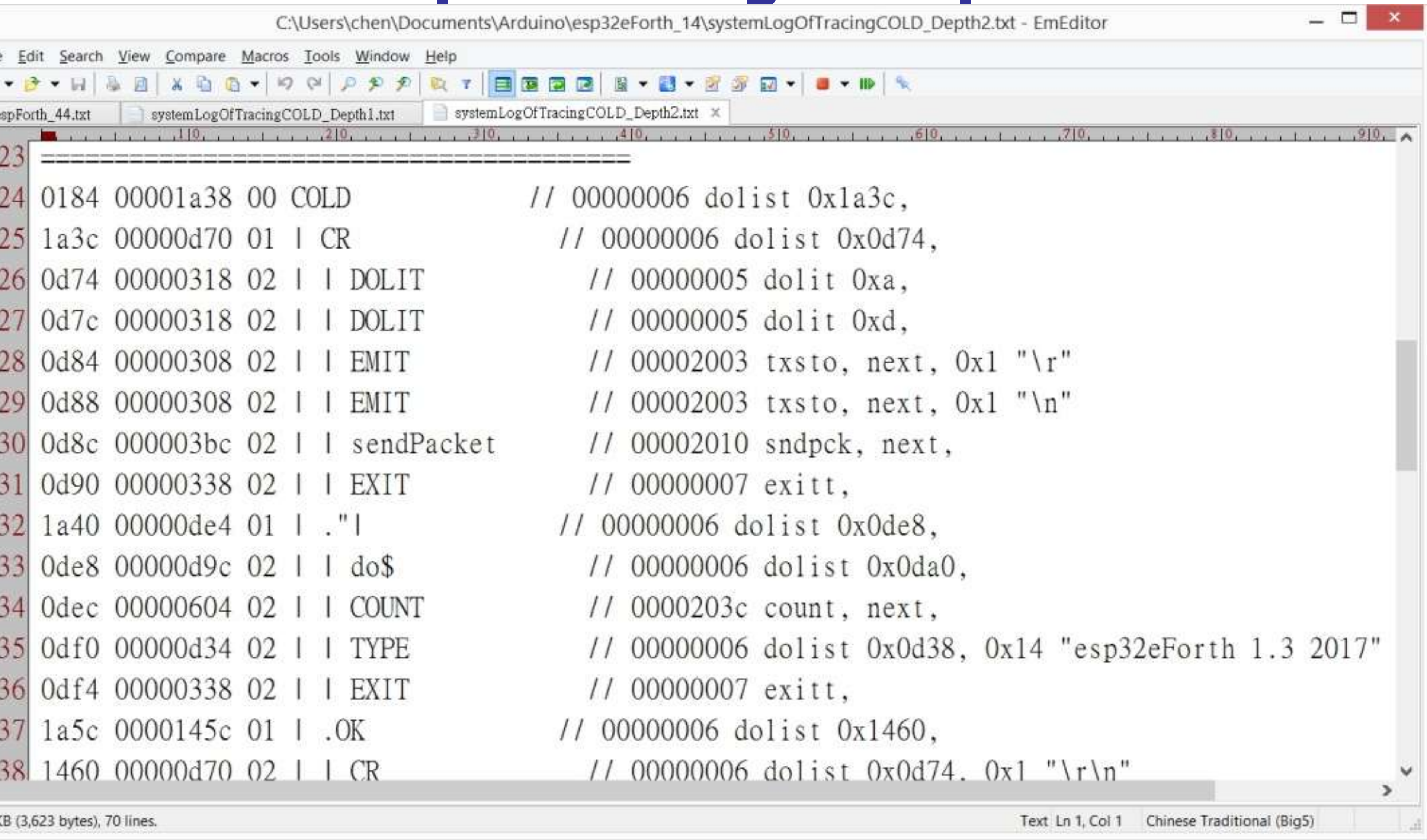
```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
ets Jun  8 2016 00:22:57

rst:0x10 (RTCWDT_RTC_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0x00
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0008,len:8
load:0x3fff0010,len:1848
load:0x40078000,len:6712
load:0x40080000,len:252
entry 0x40080034
Booting esp32eForth_14

esp32eForth 1.4 2017
0 0 0 0 ok>
```

Shallow tracing


C:\Users\chen\Documents\Arduino\esp32eForth_14\systemLogOfTracingCOLD_Depth1.txt - EmEditor
Edit
Search
View
Compare
Macros
Tools
Window
Help
spForth_44.txt
systemLogOfTracingCOLD_Depth1.txt
B (1,533 bytes), 38 lines.
Text Ln 31, Col 78 Chinese Traditional (Big5)

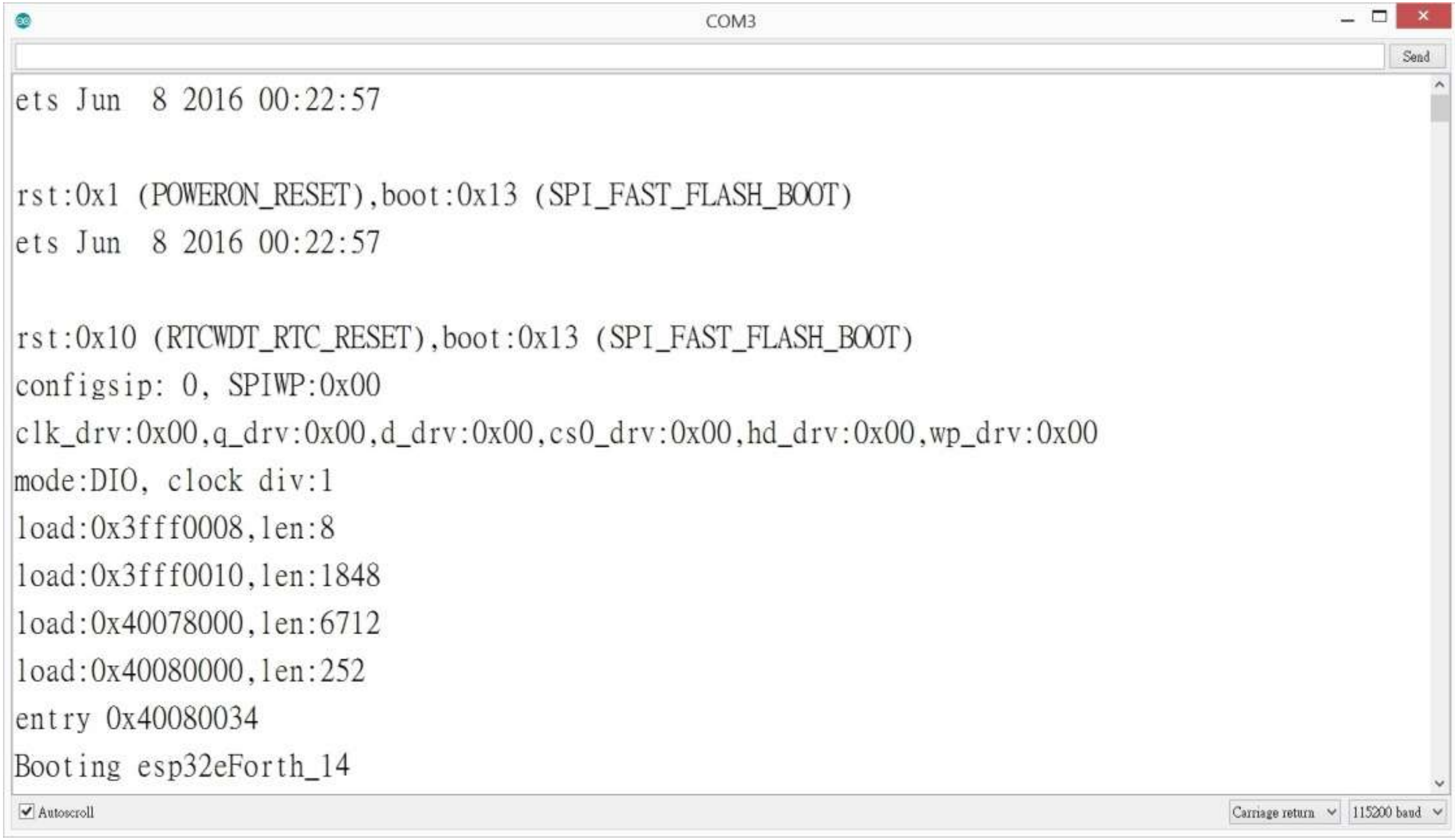
```
========================================
0184 00001a38 00 COLD                 // 00000006 dolist 0x1a3c,
1a3c 00000d70 01 | CR                   // 00000006 dolist 0x0d74, 0x2 "\r\n"
1a40 00000de4 01 | ."|                  // 00000006 dolist 0x0de8, 0x14 "esp32eForth 1.3 2017"
1a5c 0000145c 01 | .OK                  // 00000006 dolist 0x1460, 0xf "\r\n 0 0 0 0 ok> "
1a60 00001514 01 | QUIT                 // 00000006 dolist 0x1518,
========== end   of   tracing ==========
========== begin of debugging ==========
tracing 0 traceBegin 1a38 COLD traceLimit 1514 QUIT traceLevel 0 traceDepth 1 traceSteps -10
base16 P 1514 00000006 06 dolist 0x1518,
1514 QUIT 00000006 WP 1518 0000143c IP 1a64 00000338
S 0x0 stack []
R 0x2  rack [0x0 0x188]
========================================
'e' to exit, '?' for help
P 1514 >
```

Deeper tracing in depth 2


C:\Users\chen\Documents\Arduino\esp32eForth_14\systemLogOfTracingCOLD_Depth2.txt - EmEditor
Edit
Search
View
Compare
Macros
Tools
Window
Help
spForth_44.txt
systemLogOfTracingCOLD_Depth1.txt
systemLogOfTracingCOLD_Depth2.txt
KB (3,623 bytes), 70 lines.
Text
Ln 1, Col 1
Chinese Traditional (Big5)

```
========================================
0184 00001a38 00 COLD            // 00000006 dolist 0x1a3c,
1a3c 00000d70 01 | CR              // 00000006 dolist 0x0d74,
0d74 00000318 02 | | DOLIT           // 00000005 dolit 0xa,
0d7c 00000318 02 | | DOLIT           // 00000005 dolit 0xd,
0d84 00000308 02 | | EMIT            // 00002003 txsto, next, 0x1 "\r"
0d88 00000308 02 | | EMIT            // 00002003 txsto, next, 0x1 "\n"
0d8c 000003bc 02 | | sendPacket      // 00002010 sndpck, next,
0d90 00000338 02 | | EXIT            // 00000007 exitt,
1a40 00000de4 01 | ."|             // 00000006 dolist 0x0de8,
0de8 00000d9c 02 | | do$             // 00000006 dolist 0x0da0,
0dec 00000604 02 | | COUNT           // 0000203c count, next,
0df0 00000d34 02 | | TYPE            // 00000006 dolist 0x0d38, 0x14 "esp32eForth 1.3 2017"
0df4 00000338 02 | | EXIT            // 00000007 exitt,
1a5c 0000145c 01 | .OK             // 00000006 dolist 0x1460,
1460 00000d70 02 | | CR              // 00000006 dolist 0x0d74, 0x1 "\r\n"
```

Tracing begins right after booting



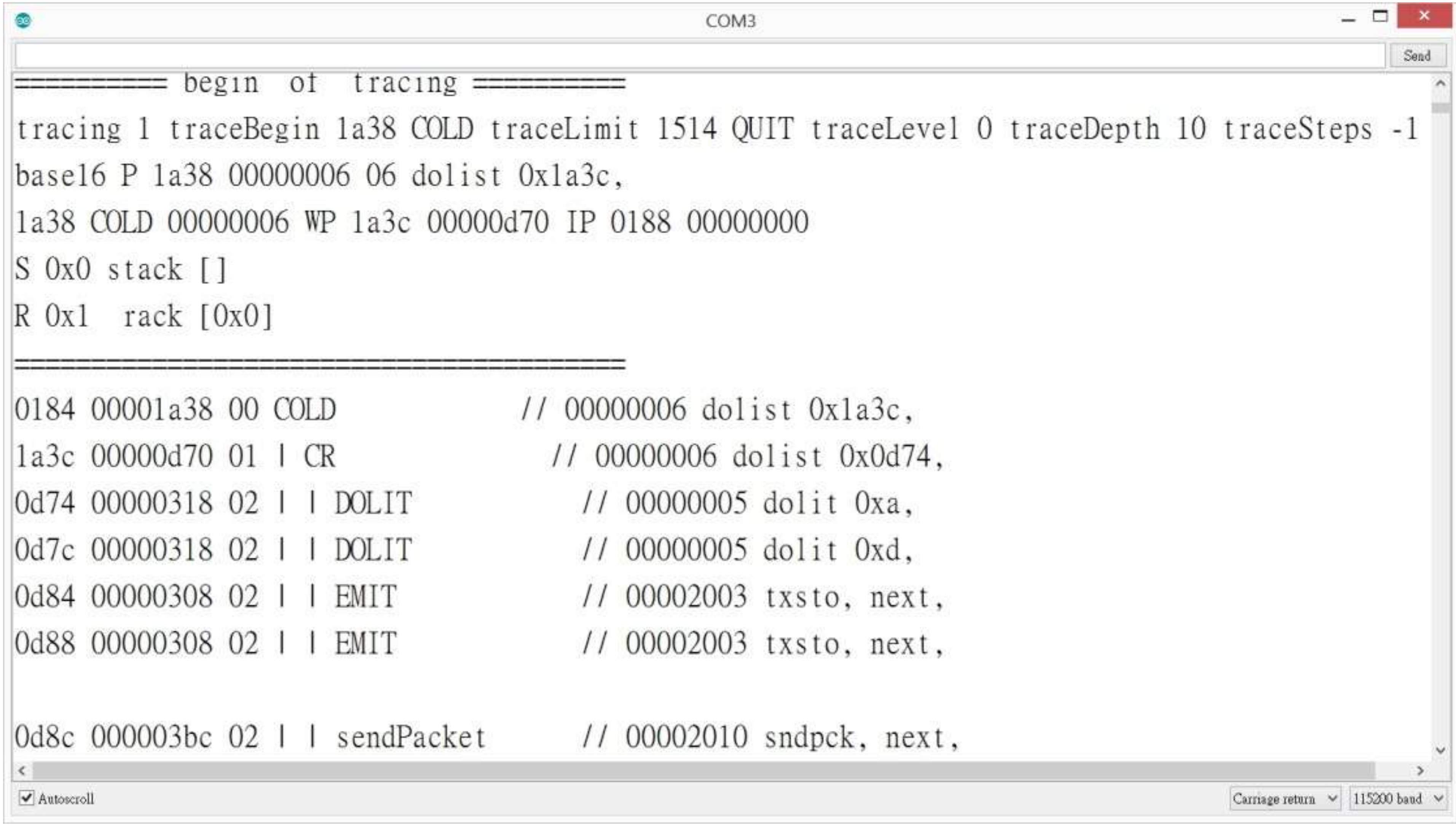
```
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
ets Jun  8 2016 00:22:57

rst:0x10 (RTCWDT_RTC_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0x00
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0008,len:8
load:0x3fff0010,len:1848
load:0x40078000,len:6712
load:0x40080000,len:252
entry 0x40080034
Booting esp32eForth_14
```

☑ Autoscroll

Carriage return

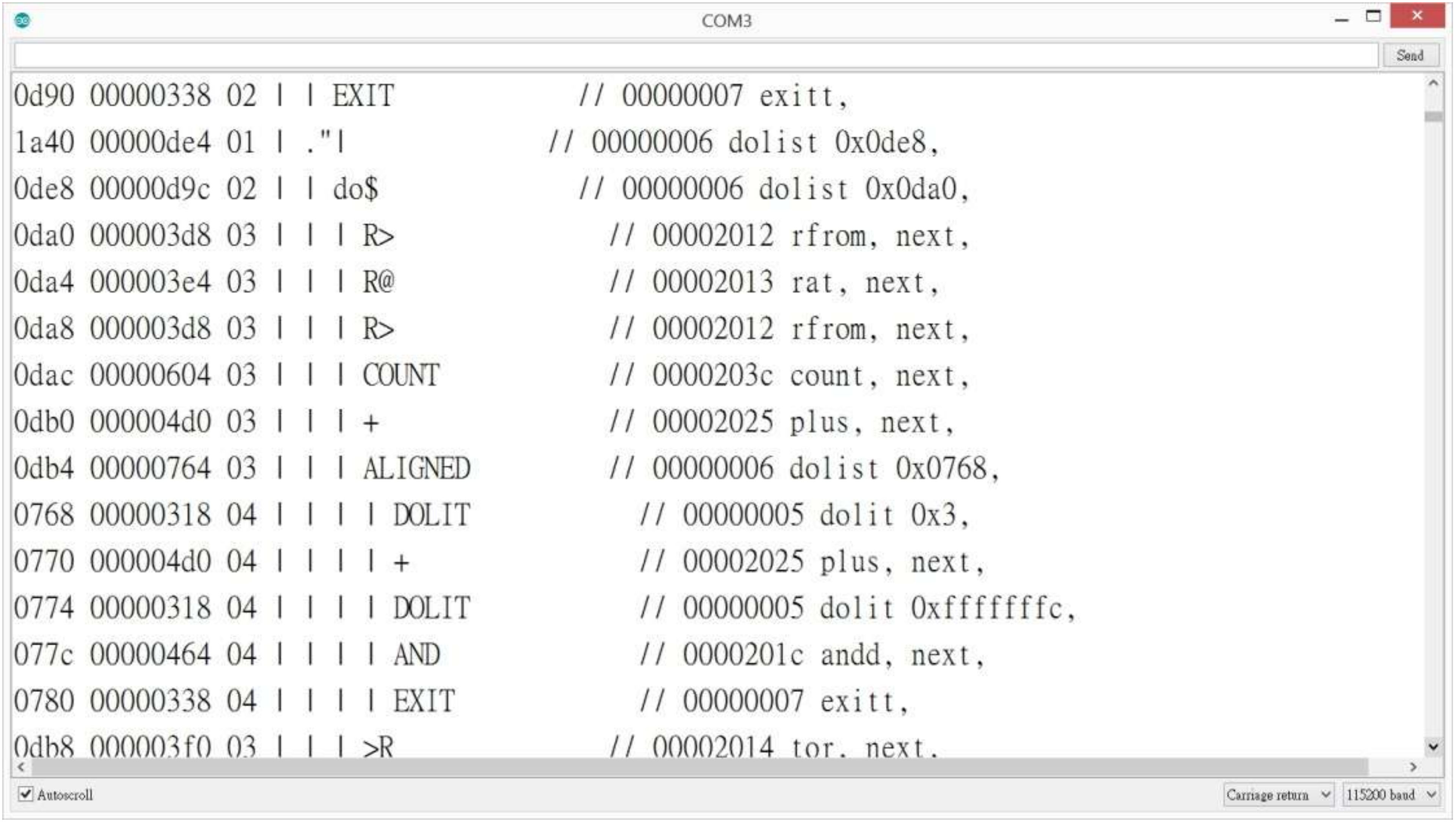
115200 baud

Tracing begins at the CFA of COLD



```
========== begin  of  tracing ==========
tracing 1 traceBegin 1a38 COLD traceLimit 1514 QUIT traceLevel 0 traceDepth 10 traceSteps -1
base16 P 1a38 00000006 06 dolist 0x1a3c,
1a38 COLD 00000006 WP 1a3c 00000d70 IP 0188 00000000
S 0x0 stack []
R 0x1  rack [0x0]
========================================
0184 00001a38 00 COLD            // 00000006 dolist 0x1a3c,
1a3c 00000d70 01 | CR              // 00000006 dolist 0x0d74,
0d74 00000318 02 | | DOLIT           // 00000005 dolit 0xa,
0d7c 00000318 02 | | DOLIT           // 00000005 dolit 0xd,
0d84 00000308 02 | | EMIT            // 00002003 txsto, next,
0d88 00000308 02 | | EMIT            // 00002003 txsto, next,

0d8c 000003bc 02 | | sendPacket      // 00002010 sndpck, next,
```

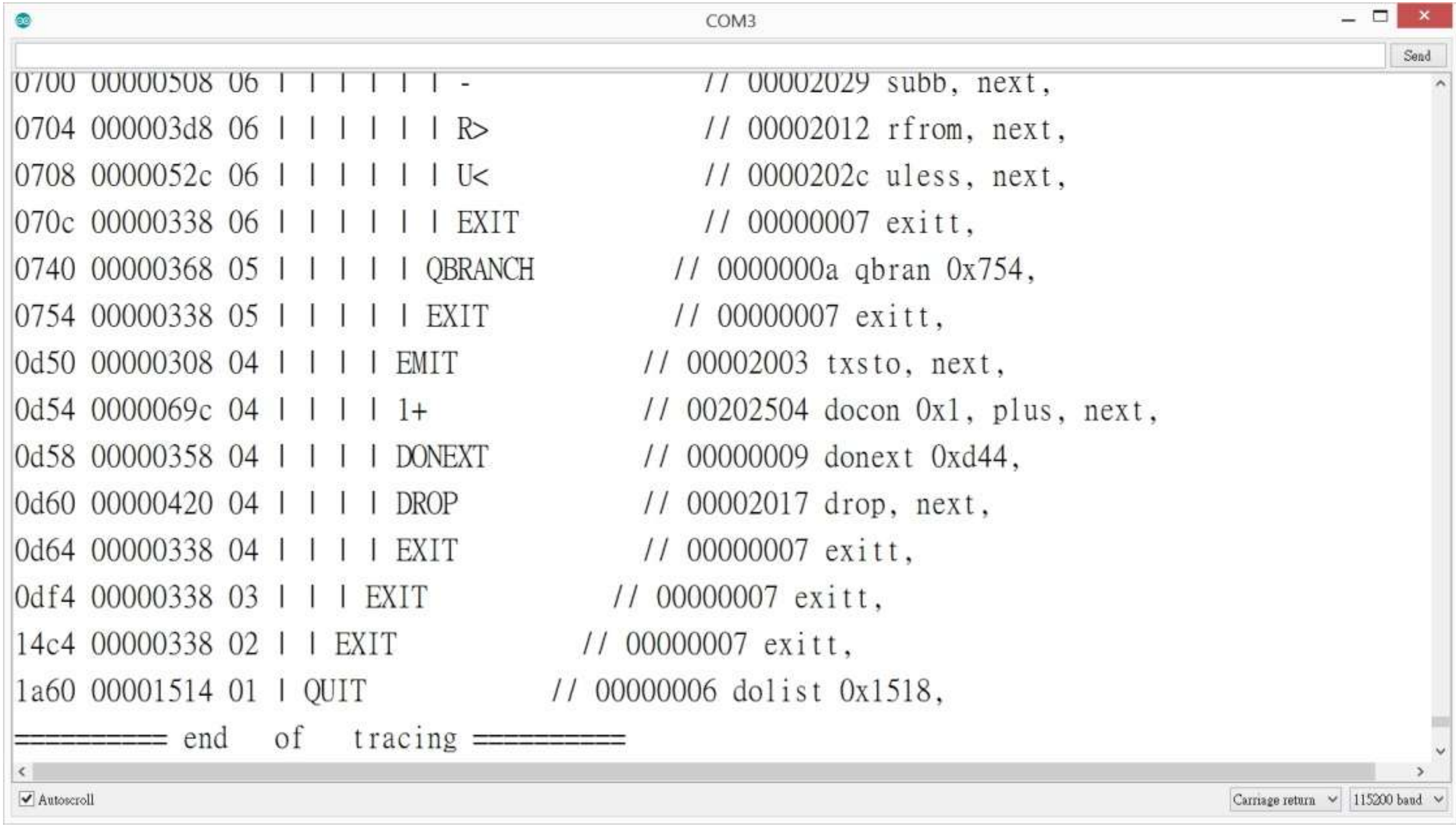
Tracing get to all levels of words called



```
0d90 00000338 02 | | EXIT              // 00000007 exitt,
1a40 00000de4 01 | ."|              // 00000006 dolist 0x0de8,
0de8 00000d9c 02 | | do$              // 00000006 dolist 0x0da0,
0da0 000003d8 03 | | | R>              // 00002012 rfrom, next,
0da4 000003e4 03 | | | R@              // 00002013 rat, next,
0da8 000003d8 03 | | | R>              // 00002012 rfrom, next,
0dac 00000604 03 | | | COUNT           // 0000203c count, next,
0db0 000004d0 03 | | | +               // 00002025 plus, next,
0db4 00000764 03 | | | ALIGNED         // 00000006 dolist 0x0768,
0768 00000318 04 | | | | DOLIT           // 00000005 dolit 0x3,
0770 000004d0 04 | | | | +               // 00002025 plus, next,
0774 00000318 04 | | | | DOLIT           // 00000005 dolit 0xfffffffc,
077c 00000464 04 | | | | AND             // 0000201c andd, next,
0780 00000338 04 | | | | EXIT            // 00000007 exitt,
0db8 000003f0 03 | | | >R              // 00002014 tor, next,
```

☑ Autoscroll   Carriage return   115200 baud

Tracing ends at the CFA of QUIT



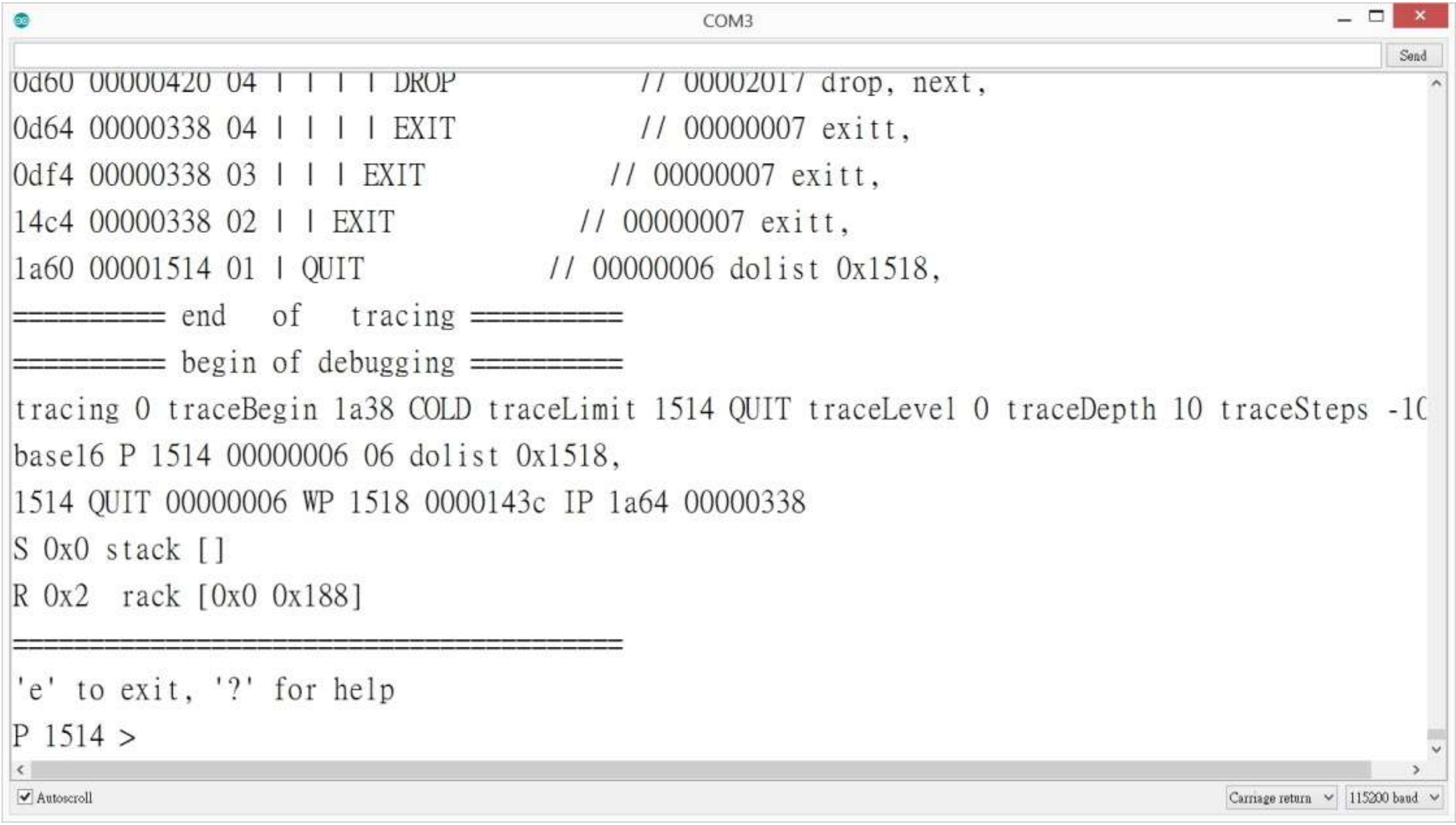
```
0700 00000508 06 | | | | | | -               // 00002029 subb, next,
0704 000003d8 06 | | | | | | R>              // 00002012 rfrom, next,
0708 0000052c 06 | | | | | | U<              // 0000202c uless, next,
070c 00000338 06 | | | | | | EXIT            // 00000007 exitt,
0740 00000368 05 | | | | | QBRANCH         // 0000000a qbran 0x754,
0754 00000338 05 | | | | | EXIT            // 00000007 exitt,
0d50 00000308 04 | | | | EMIT            // 00002003 txsto, next,
0d54 0000069c 04 | | | | 1+              // 00202504 docon 0x1, plus, next,
0d58 00000358 04 | | | | DONEXT          // 00000009 donext 0xd44,
0d60 00000420 04 | | | | DROP            // 00002017 drop, next,
0d64 00000338 04 | | | | EXIT            // 00000007 exitt,
0df4 00000338 03 | | | EXIT            // 00000007 exitt,
14c4 00000338 02 | | EXIT            // 00000007 exitt,
1a60 00001514 01 | QUIT            // 00000006 dolist 0x1518,
========== end  of  tracing ==========
```

☑ Autoscroll

Carriage return

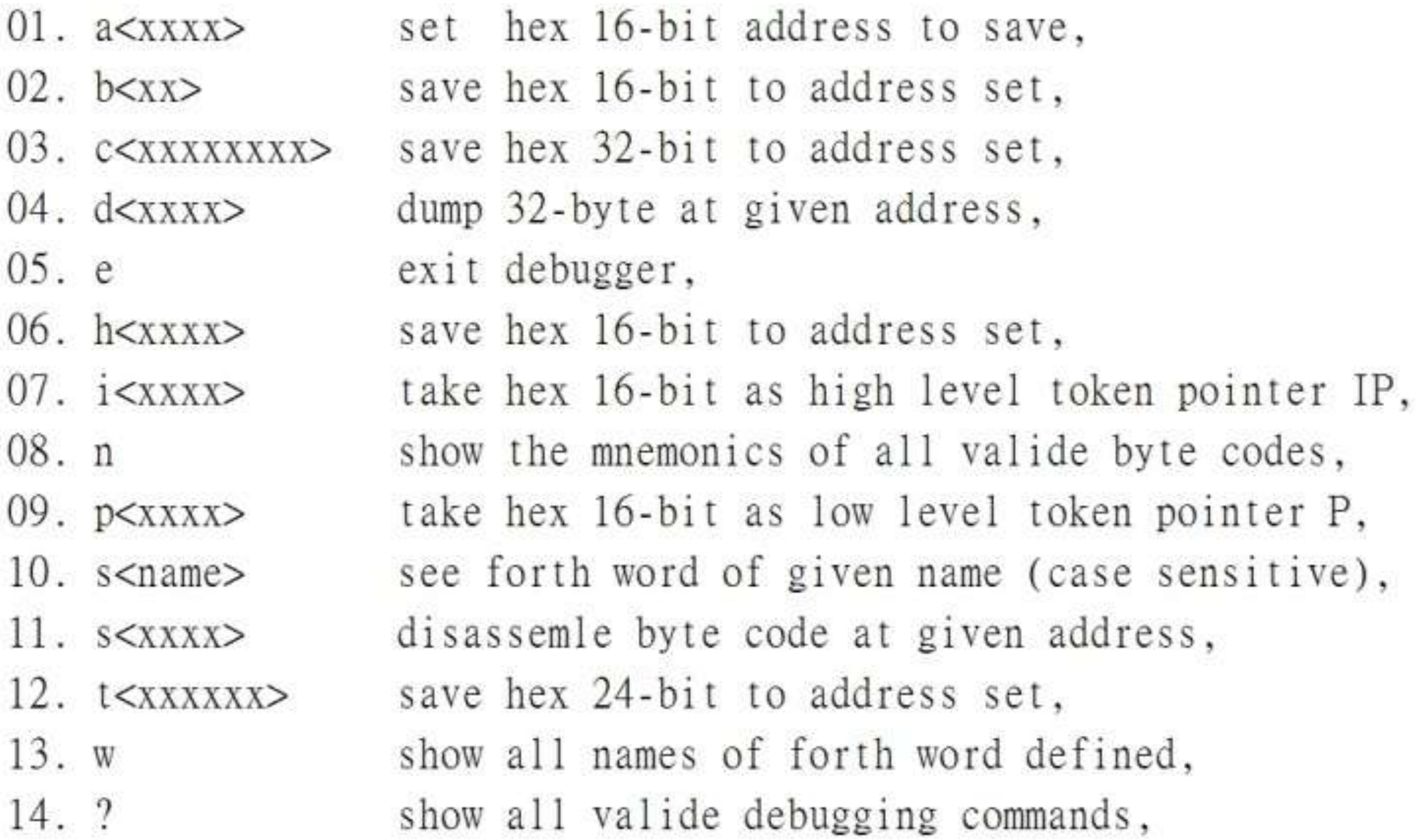
115200 baud

After tracing, get into debugger



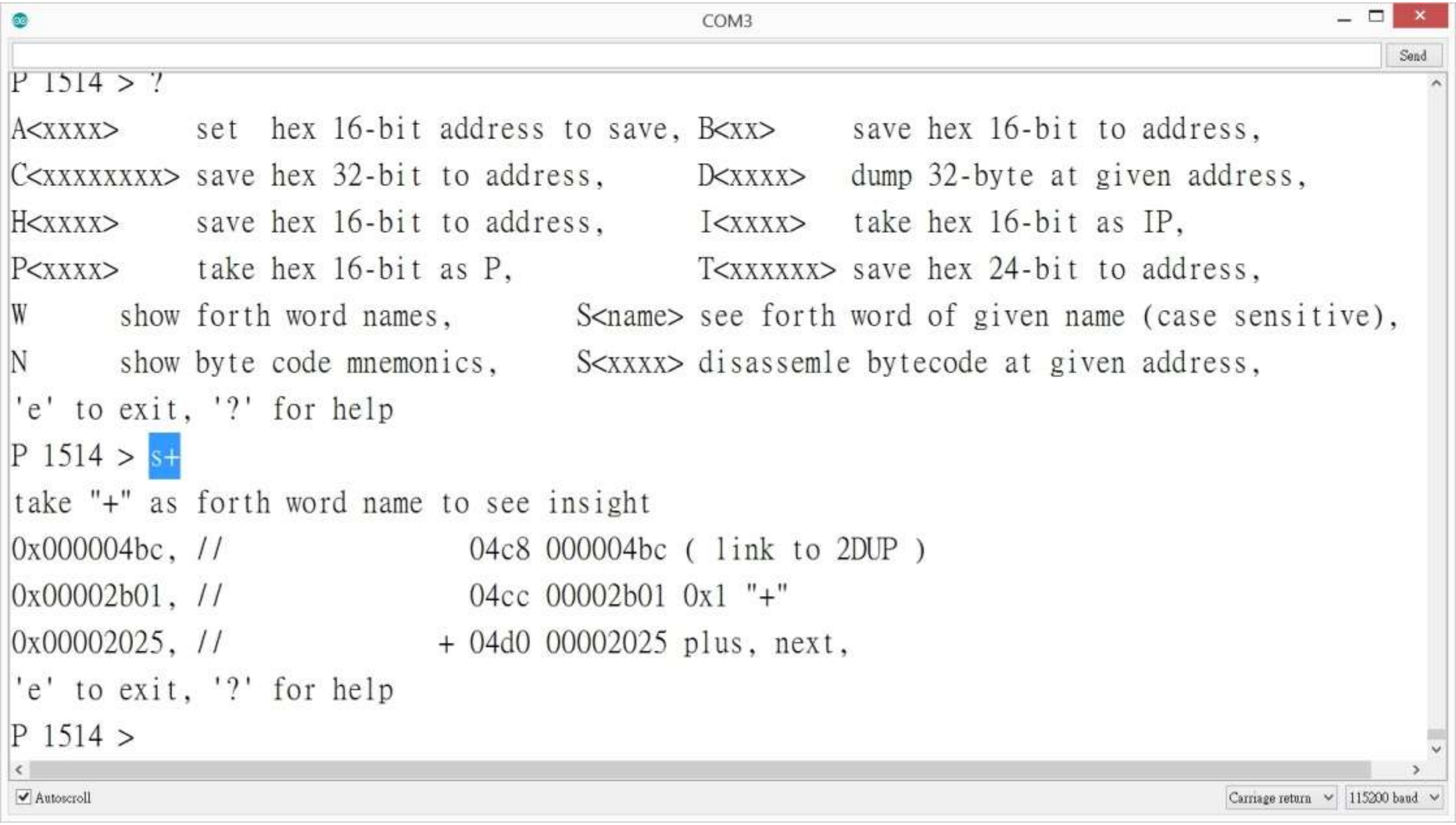
```
0d60 00000420 04 | | | | DROP               // 00002017 drop, next,
0d64 00000338 04 | | | | EXIT               // 00000007 exitt,
0df4 00000338 03 | | | EXIT               // 00000007 exitt,
14c4 00000338 02 | | EXIT               // 00000007 exitt,
1a60 00001514 01 | QUIT               // 00000006 dolist 0x1518,
========== end   of   tracing ==========
========== begin of debugging ==========
tracing 0 traceBegin 1a38 COLD traceLimit 1514 QUIT traceLevel 0 traceDepth 10 traceSteps -1(
base16 P 1514 00000006 06 dolist 0x1518,
1514 QUIT 00000006 WP 1518 0000143c IP 1a64 00000338
S 0x0 stack []
R 0x2  rack [0x0 0x188]
========================================
'e' to exit, '?' for help
P 1514 >
```

Debugging commands



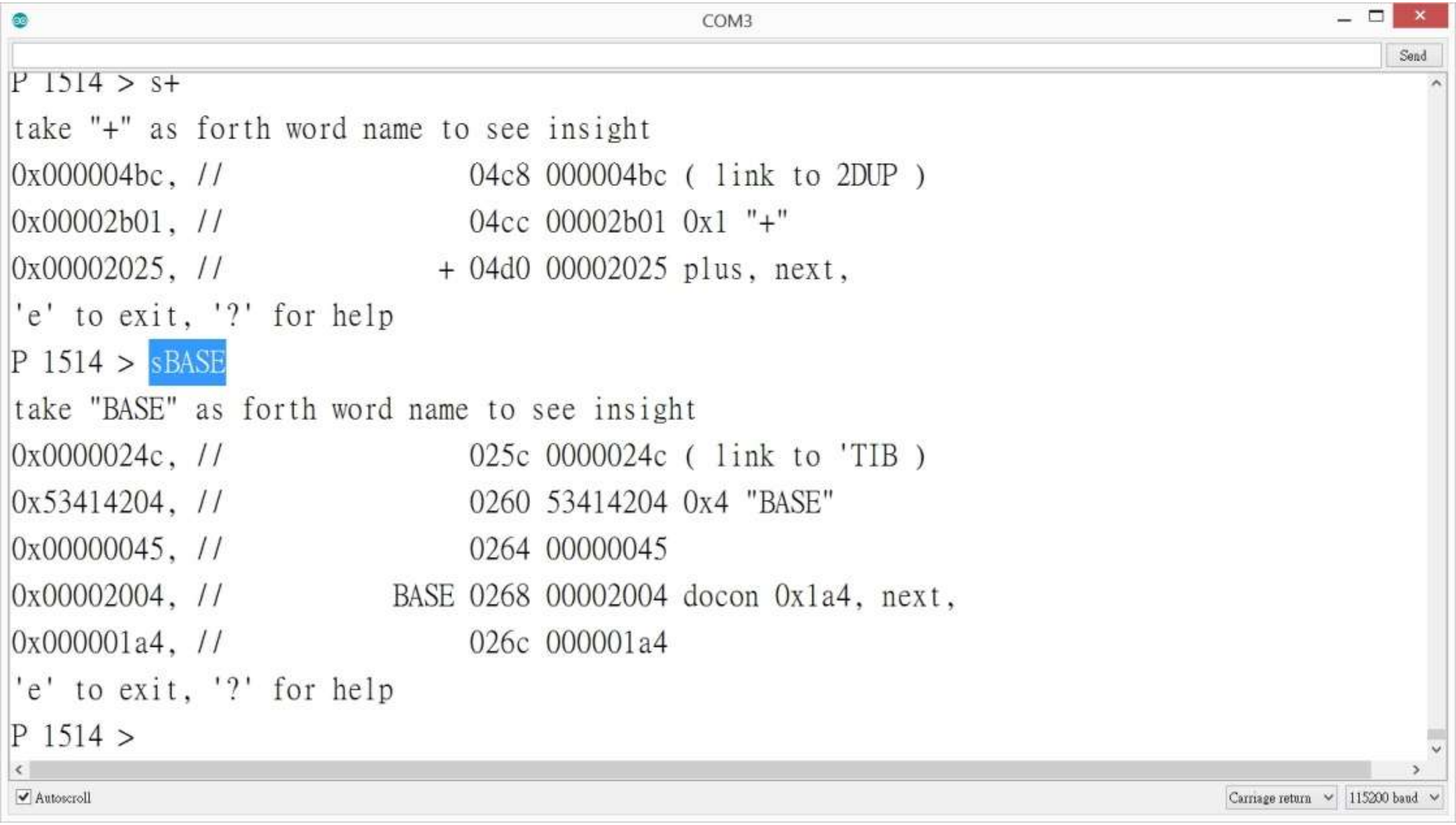
```
01. a<xxxx>      set  hex 16-bit address to save,
02. b<xx>        save hex 16-bit to address set,
03. c<xxxxxxxx>  save hex 32-bit to address set,
04. d<xxxx>      dump 32-byte at given address,
05. e            exit debugger,
06. h<xxxx>      save hex 16-bit to address set,
07. i<xxxx>      take hex 16-bit as high level token pointer IP,
08. n            show the mnemonics of all valide byte codes,
09. p<xxxx>      take hex 16-bit as low level token pointer P,
10. s<name>      see forth word of given name (case sensitive),
11. s<xxxx>      disassemle byte code at given address,
12. t<xxxxxx>    save hex 24-bit to address set,
13. w            show all names of forth word defined,
14. ?            show all valide debugging commands,
```

See a word, e.g. + (the plus). Type “s+”.



```
P 1514 > ?
A<xxxx>     set  hex 16-bit address to save, B<xx>     save hex 16-bit to address,
C<xxxxxxxx> save hex 32-bit to address,       D<xxxx>   dump 32-byte at given address,
H<xxxx>     save hex 16-bit to address,       I<xxxx>   take hex 16-bit as IP,
P<xxxx>     take hex 16-bit as P,             T<xxxxxx> save hex 24-bit to address,
W       show forth word names,        S<name> see forth word of given name (case sensitive),
N       show byte code mnemonics,     S<xxxx> disassemle bytecode at given address,
'e' to exit, '?' for help
P 1514 > s+
take "+" as forth word name to see insight
0x000004bc, //                  04c8 000004bc ( link to 2DUP )
0x00002b01, //                  04cc 00002b01 0x1 "+"
0x00002025, //                + 04d0 00002025 plus, next,
'e' to exit, '?' for help
P 1514 >
```

See system variable word, e.g. BASE



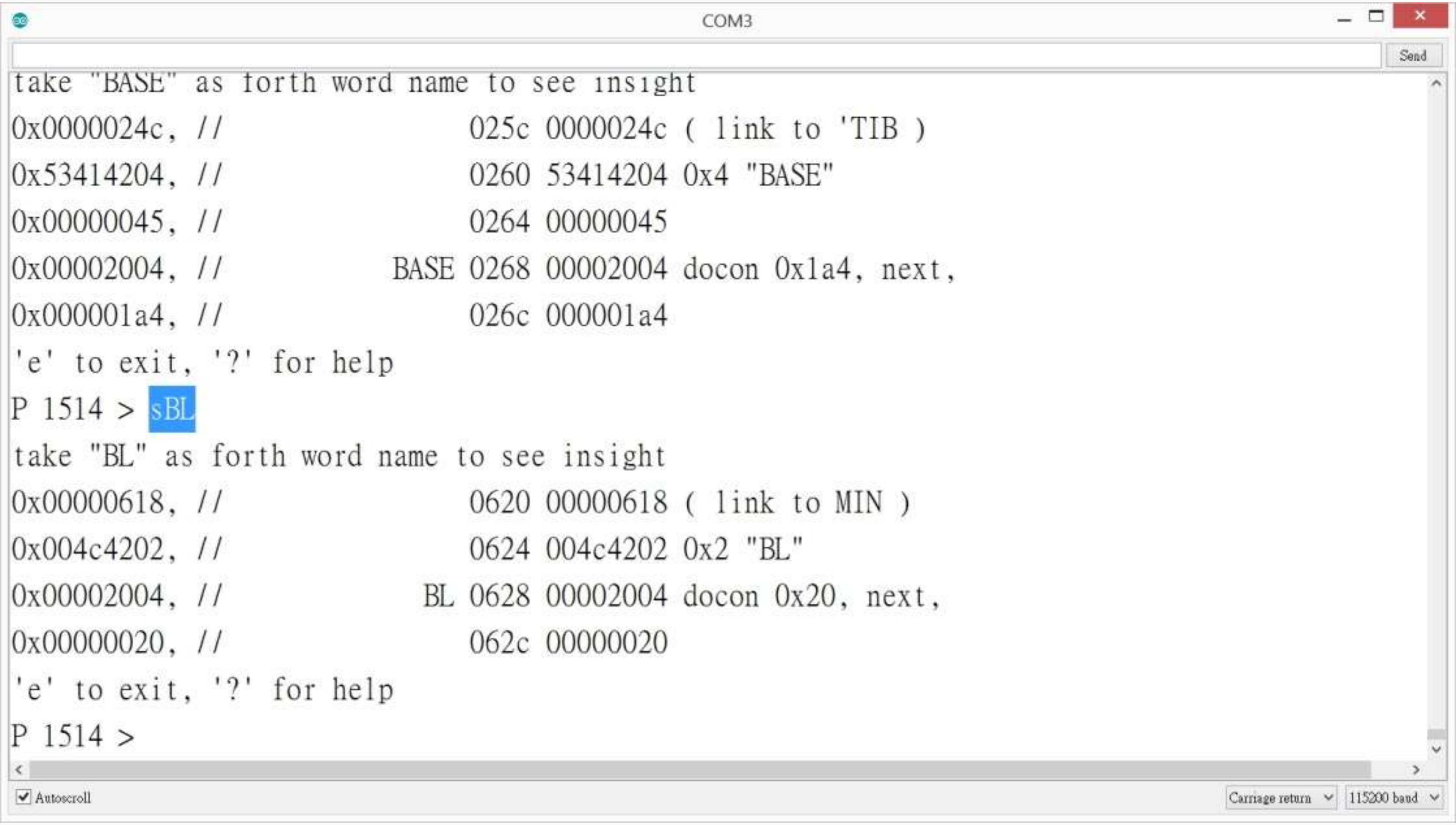
```
P 1514 > s+
take "+" as forth word name to see insight
0x000004bc, //                   04c8 000004bc ( link to 2DUP )
0x00002b01, //                   04cc 00002b01 0x1 "+"
0x00002025, //                 + 04d0 00002025 plus, next,
'e' to exit, '?' for help
P 1514 > sBASE
take "BASE" as forth word name to see insight
0x0000024c, //                   025c 0000024c ( link to 'TIB )
0x53414204, //                   0260 53414204 0x4 "BASE"
0x00000045, //                   0264 00000045
0x00002004, //              BASE 0268 00002004 docon 0x1a4, next,
0x000001a4, //                   026c 000001a4
'e' to exit, '?' for help
P 1514 >
```

☑ Autoscroll

Carriage return

115200 baud

See constant word, e.g. BL



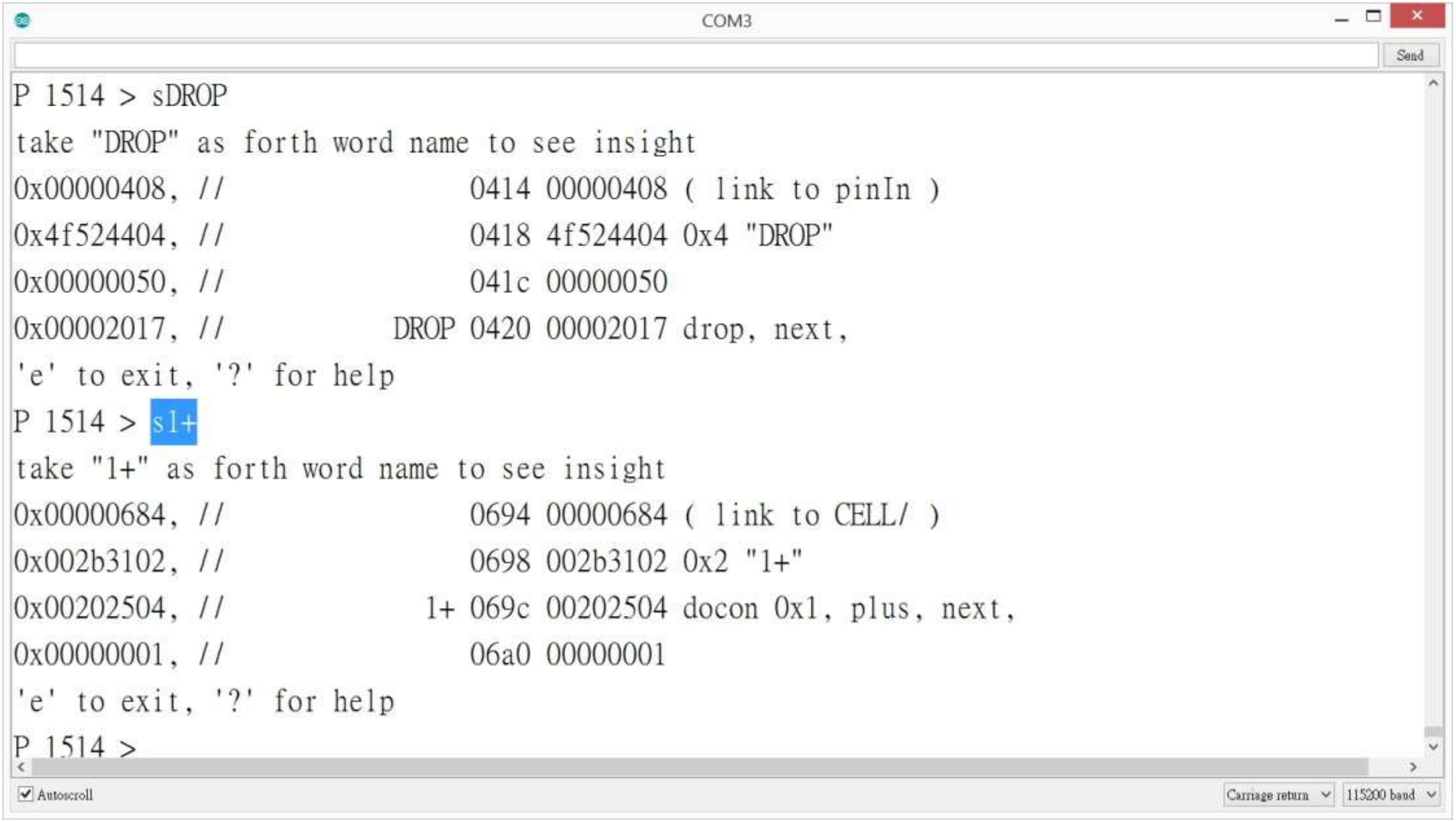
```
take "BASE" as forth word name to see insight
0x0000024c, //                  025c 0000024c ( link to 'TIB )
0x53414204, //                  0260 53414204 0x4 "BASE"
0x00000045, //                  0264 00000045
0x00002004, //             BASE 0268 00002004 docon 0x1a4, next,
0x000001a4, //                  026c 000001a4
'e' to exit, '?' for help
P 1514 > sBL
take "BL" as forth word name to see insight
0x00000618, //                  0620 00000618 ( link to MIN )
0x004c4202, //                  0624 004c4202 0x2 "BL"
0x00002004, //               BL 0628 00002004 docon 0x20, next,
0x00000020, //                  062c 00000020
'e' to exit, '?' for help
P 1514 >
```

☑ Autoscroll

Carriage return

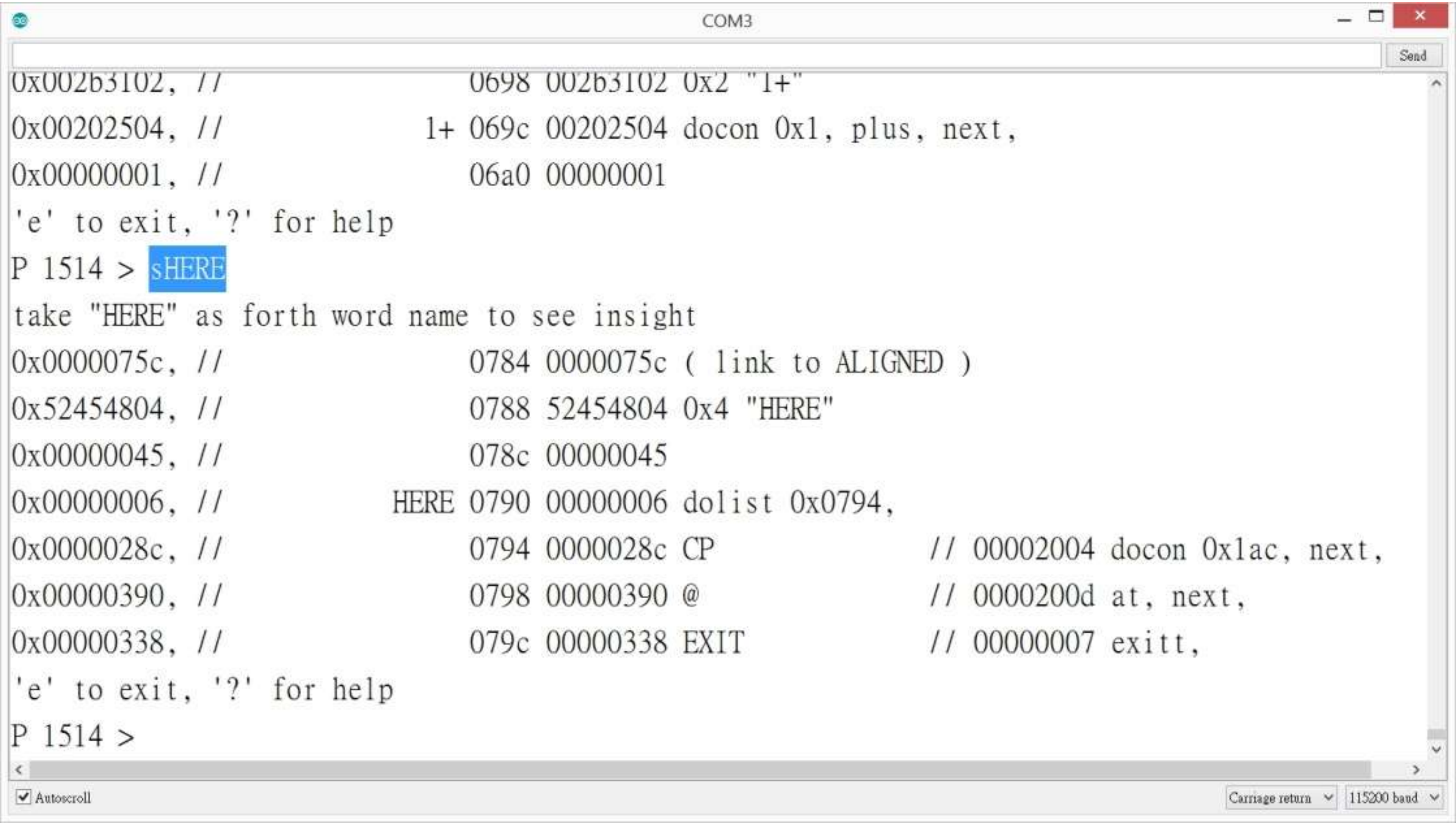
115200 baud

See constant calcution word, e.g. 1+



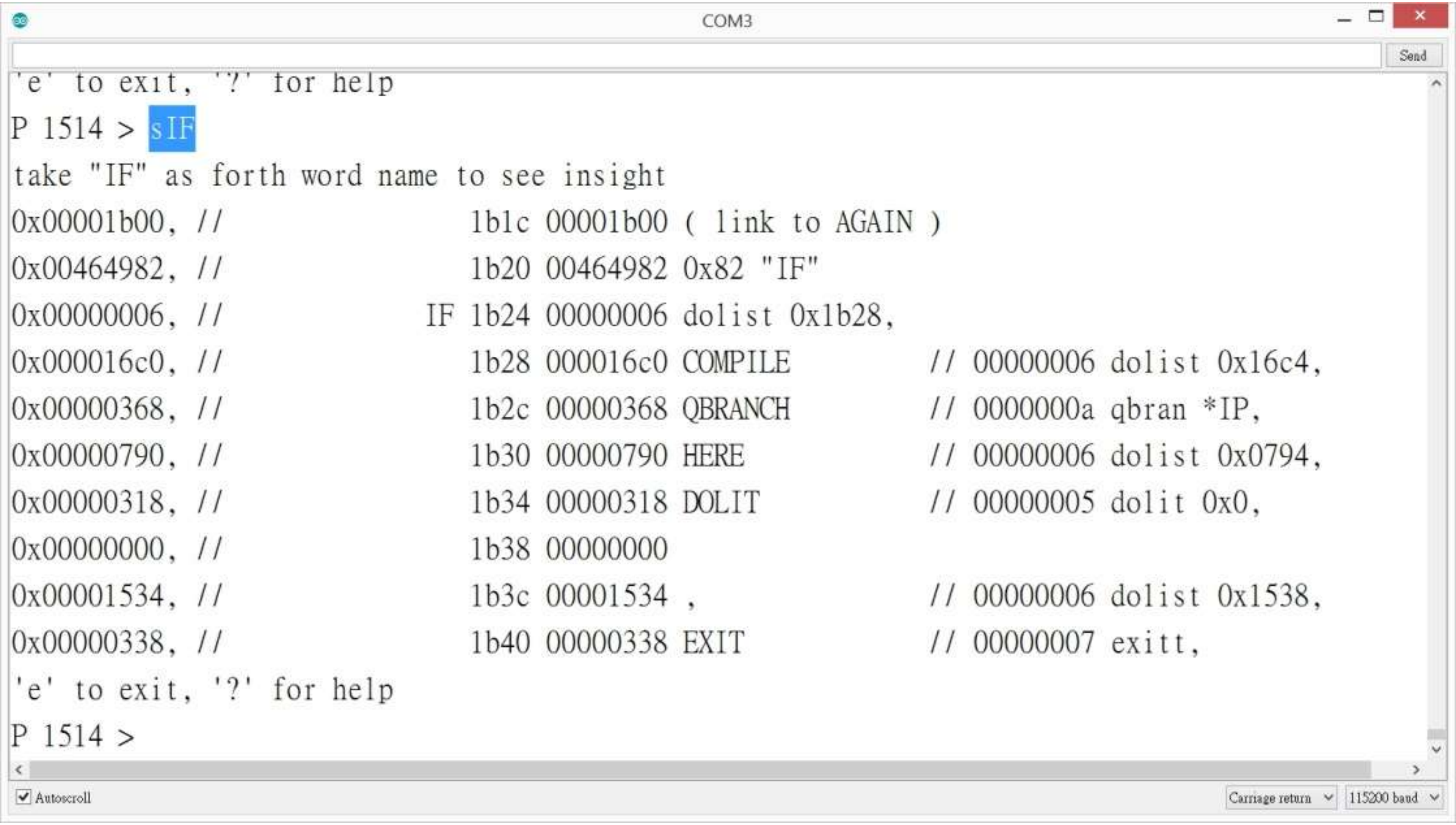
```
P 1514 > sDROP
take "DROP" as forth word name to see insight
0x00000408, //                0414 00000408 ( link to pinIn )
0x4f524404, //                0418 4f524404 0x4 "DROP"
0x00000050, //                041c 00000050
0x00002017, //           DROP 0420 00002017 drop, next,
'e' to exit, '?' for help
P 1514 > s1+
take "1+" as forth word name to see insight
0x00000684, //                0694 00000684 ( link to CELL/ )
0x002b3102, //                0698 002b3102 0x2 "1+"
0x00202504, //             1+ 069c 00202504 docon 0x1, plus, next,
0x00000001, //                06a0 00000001
'e' to exit, '?' for help
P 1514 >
```

See colon word, e.g. HERE



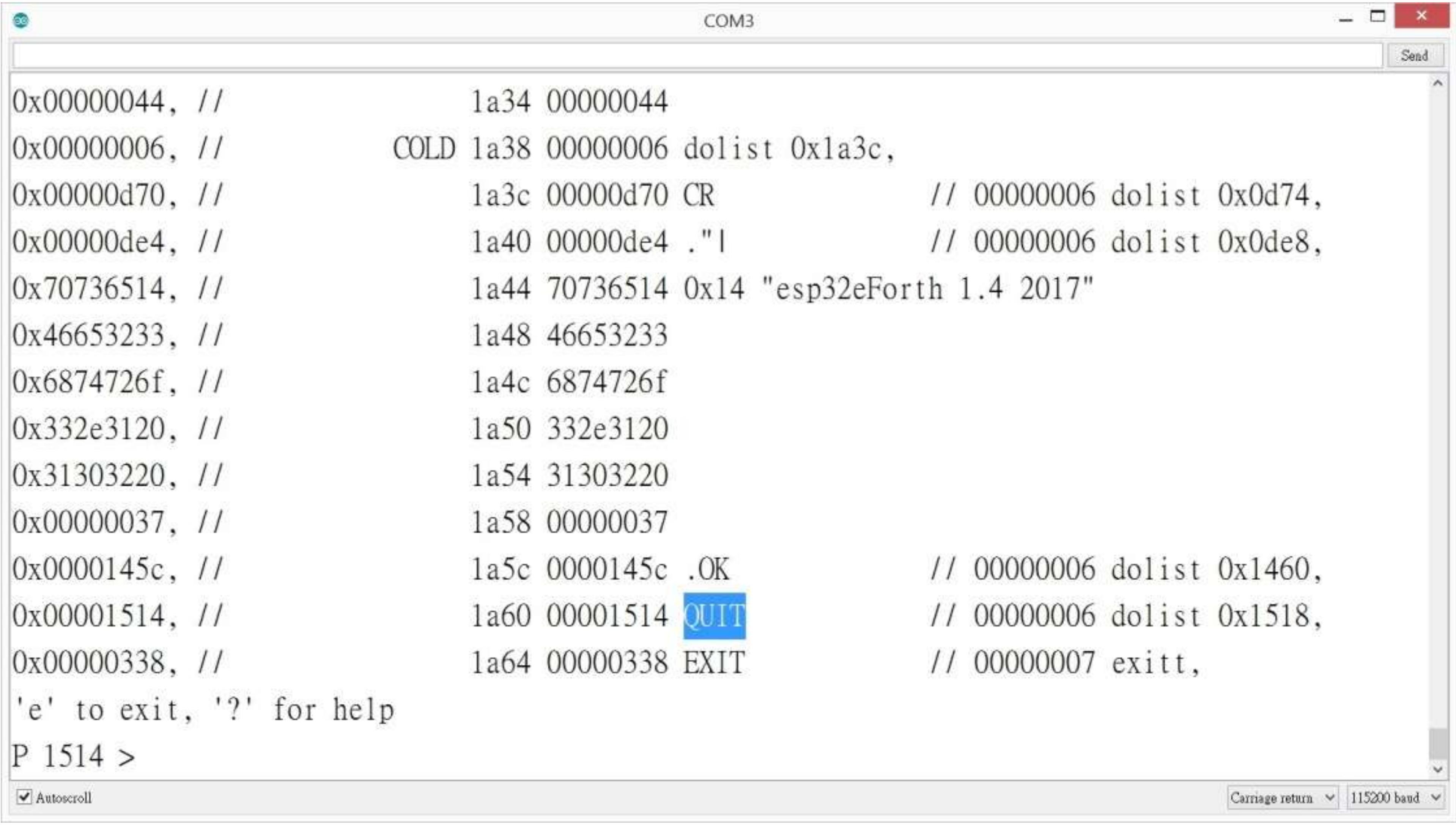
```
0x002b3102, //              0698 002b3102 0x2 "1+"
0x00202504, //           1+ 069c 00202504 docon 0x1, plus, next,
0x00000001, //              06a0 00000001
'e' to exit, '?' for help
P 1514 > sHERE
take "HERE" as forth word name to see insight
0x0000075c, //              0784 0000075c ( link to ALIGNED )
0x52454804, //              0788 52454804 0x4 "HERE"
0x00000045, //              078c 00000045
0x00000006, //         HERE 0790 00000006 dolist 0x0794,
0x0000028c, //              0794 0000028c CP            // 00002004 docon 0x1ac, next,
0x00000390, //              0798 00000390 @             // 0000200d at, next,
0x00000338, //              079c 00000338 EXIT          // 00000007 exitt,
'e' to exit, '?' for help
P 1514 >
```

See immediate word, e.g. IF



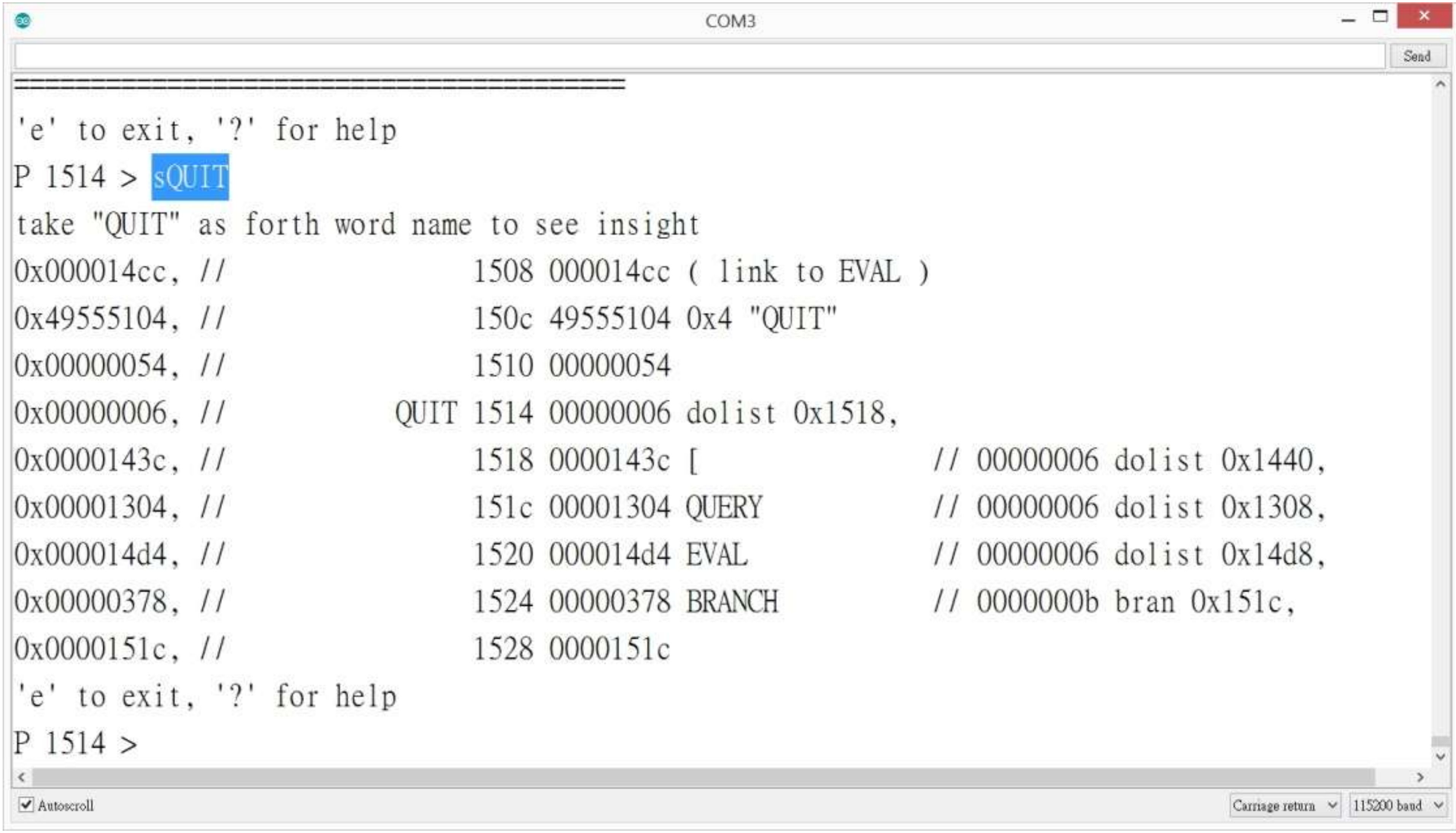
```
'e' to exit, '?' for help
P 1514 > sIF
take "IF" as forth word name to see insight
0x00001b00, //           1b1c 00001b00 ( link to AGAIN )
0x00464982, //           1b20 00464982 0x82 "IF"
0x00000006, //        IF 1b24 00000006 dolist 0x1b28,
0x000016c0, //           1b28 000016c0 COMPILE      // 00000006 dolist 0x16c4,
0x00000368, //           1b2c 00000368 QBRANCH      // 0000000a qbran *IP,
0x00000790, //           1b30 00000790 HERE         // 00000006 dolist 0x0794,
0x00000318, //           1b34 00000318 DOLIT        // 00000005 dolit 0x0,
0x00000000, //           1b38 00000000
0x00001534, //           1b3c 00001534 ,            // 00000006 dolist 0x1538,
0x00000338, //           1b40 00000338 EXIT         // 00000007 exitt,
'e' to exit, '?' for help
P 1514 >
```

The word COLD



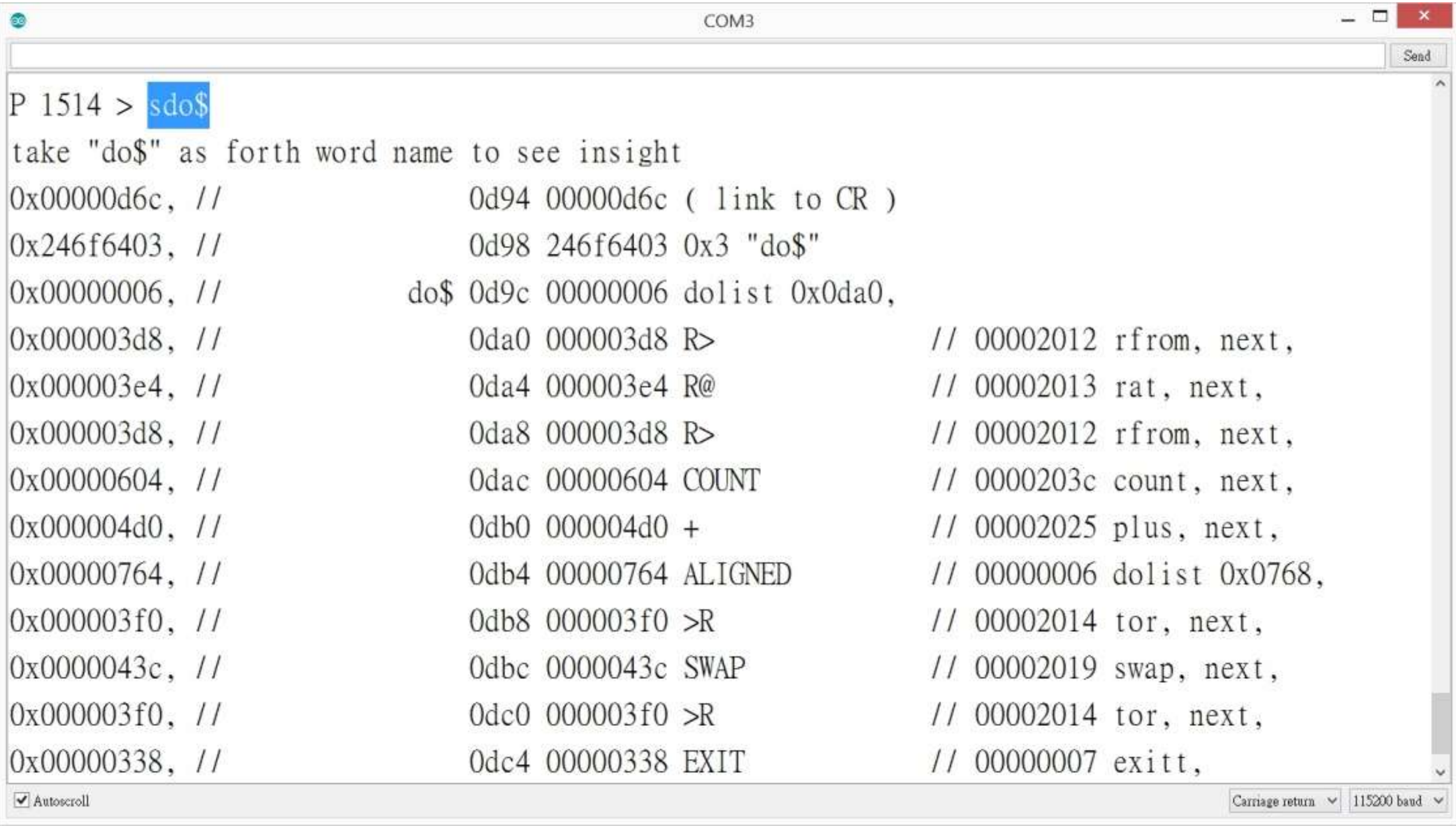
```
0x00000044, //           1a34 00000044
0x00000006, //      COLD 1a38 00000006 dolist 0x1a3c,
0x00000d70, //           1a3c 00000d70 CR         // 00000006 dolist 0x0d74,
0x00000de4, //           1a40 00000de4 ."|        // 00000006 dolist 0x0de8,
0x70736514, //           1a44 70736514 0x14 "esp32eForth 1.4 2017"
0x46653233, //           1a48 46653233
0x6874726f, //           1a4c 6874726f
0x332e3120, //           1a50 332e3120
0x31303220, //           1a54 31303220
0x00000037, //           1a58 00000037
0x0000145c, //           1a5c 0000145c .OK        // 00000006 dolist 0x1460,
0x00001514, //           1a60 00001514 QUIT       // 00000006 dolist 0x1518,
0x00000338, //           1a64 00000338 EXIT       // 00000007 exitt,
'e' to exit, '?' for help
P 1514 >
```

See the word QUIT



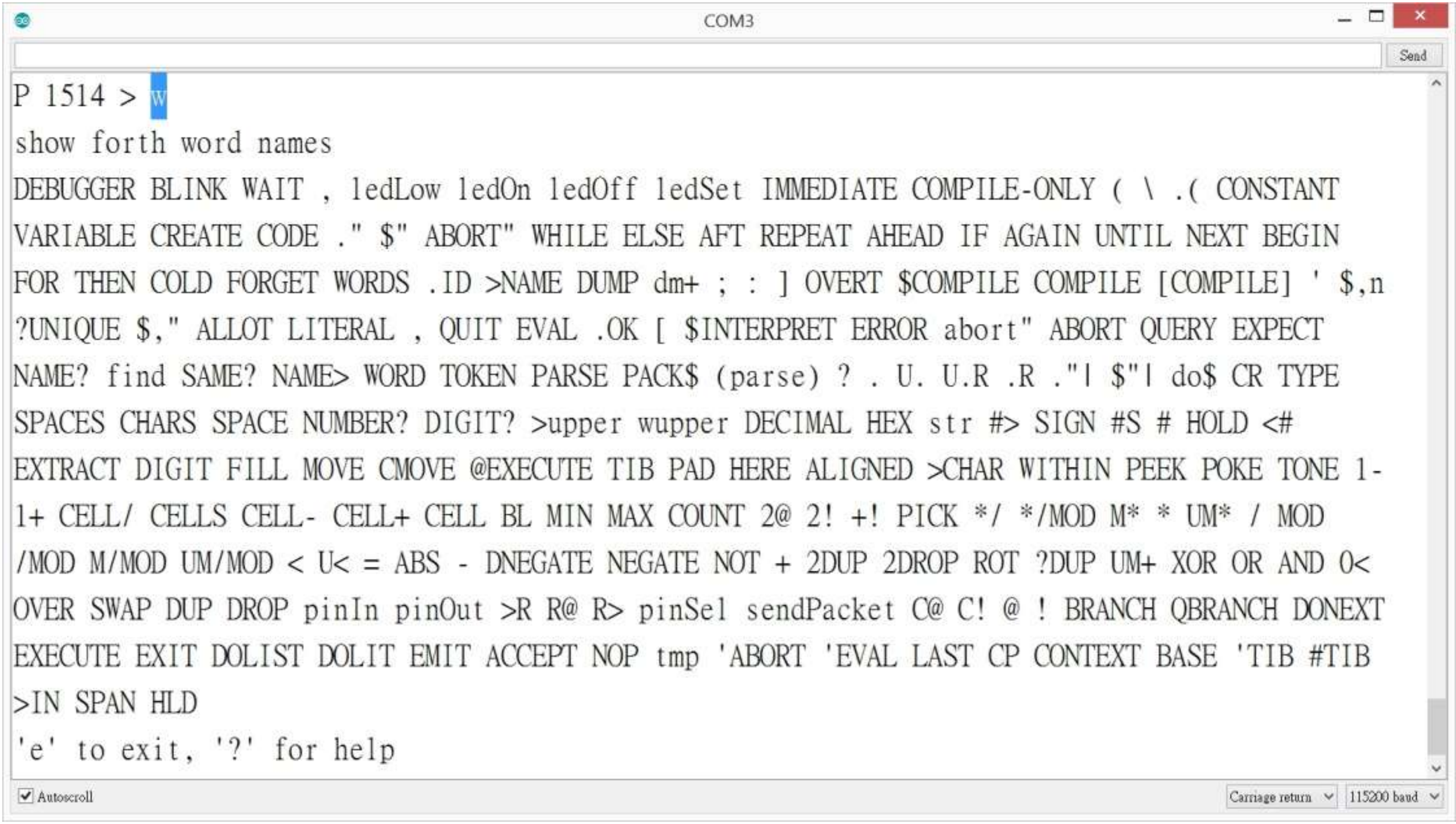
```
==========================================
'e' to exit, '?' for help
P 1514 > sQUIT
take "QUIT" as forth word name to see insight
0x000014cc, //             1508 000014cc ( link to EVAL )
0x49555104, //             150c 49555104 0x4 "QUIT"
0x00000054, //             1510 00000054
0x00000006, //        QUIT 1514 00000006 dolist 0x1518,
0x0000143c, //             1518 0000143c [          // 00000006 dolist 0x1440,
0x00001304, //             151c 00001304 QUERY      // 00000006 dolist 0x1308,
0x000014d4, //             1520 000014d4 EVAL       // 00000006 dolist 0x14d8,
0x00000378, //             1524 00000378 BRANCH     // 0000000b bran 0x151c,
0x0000151c, //             1528 0000151c
'e' to exit, '?' for help
P 1514 >
```

See the word do$ (lowercase)



```
P 1514 > sdo$
take "do$" as forth word name to see insight
0x00000d6c, //                0d94 00000d6c ( link to CR )
0x246f6403, //                0d98 246f6403 0x3 "do$"
0x00000006, //           do$ 0d9c 00000006 dolist 0x0da0,
0x000003d8, //                0da0 000003d8 R>        // 00002012 rfrom, next,
0x000003e4, //                0da4 000003e4 R@        // 00002013 rat, next,
0x000003d8, //                0da8 000003d8 R>        // 00002012 rfrom, next,
0x00000604, //                0dac 00000604 COUNT     // 0000203c count, next,
0x000004d0, //                0db0 000004d0 +         // 00002025 plus, next,
0x00000764, //                0db4 00000764 ALIGNED   // 00000006 dolist 0x0768,
0x000003f0, //                0db8 000003f0 >R        // 00002014 tor, next,
0x0000043c, //                0dbc 0000043c SWAP      // 00002019 swap, next,
0x000003f0, //                0dc0 000003f0 >R        // 00002014 tor, next,
0x00000338, //                0dc4 00000338 EXIT      // 00000007 exitt,
```

Show all words



```
P 1514 > w
show forth word names
DEBUGGER BLINK WAIT , ledLow ledOn ledOff ledSet IMMEDIATE COMPILE-ONLY ( \ .( CONSTANT
VARIABLE CREATE CODE ." $" ABORT" WHILE ELSE AFT REPEAT AHEAD IF AGAIN UNTIL NEXT BEGIN
FOR THEN COLD FORGET WORDS .ID >NAME DUMP dm+ ; : ] OVERT $COMPILE COMPILE [COMPILE] ' $,n
?UNIQUE $," ALLOT LITERAL , QUIT EVAL .OK [ $INTERPRET ERROR abort" ABORT QUERY EXPECT
NAME? find SAME? NAME> WORD TOKEN PARSE PACK$ (parse) ? . U. U.R .R ."| $"| do$ CR TYPE
SPACES CHARS SPACE NUMBER? DIGIT? >upper wupper DECIMAL HEX str #> SIGN #S # HOLD <#
EXTRACT DIGIT FILL MOVE CMOVE @EXECUTE TIB PAD HERE ALIGNED >CHAR WITHIN PEEK POKE TONE 1-
1+ CELL/ CELLS CELL- CELL+ CELL BL MIN MAX COUNT 2@ 2! +! PICK */ */MOD M* * UM* / MOD
/MOD M/MOD UM/MOD < U< = ABS - DNEGATE NEGATE NOT + 2DUP 2DROP ROT ?DUP UM+ XOR OR AND 0<
OVER SWAP DUP DROP pinIn pinOut >R R@ R> pinSel sendPacket C@ C! @ ! BRANCH QBRANCH DONEXT
EXECUTE EXIT DOLIST DOLIT EMIT ACCEPT NOP tmp 'ABORT 'EVAL LAST CP CONTEXT BASE 'TIB #TIB
>IN SPAN HLD
'e' to exit, '?' for help
```

New Words (all defined in byte codes)

, (comma), IMMEDIATE and COMPILE-ONLY replaced in byte codes.

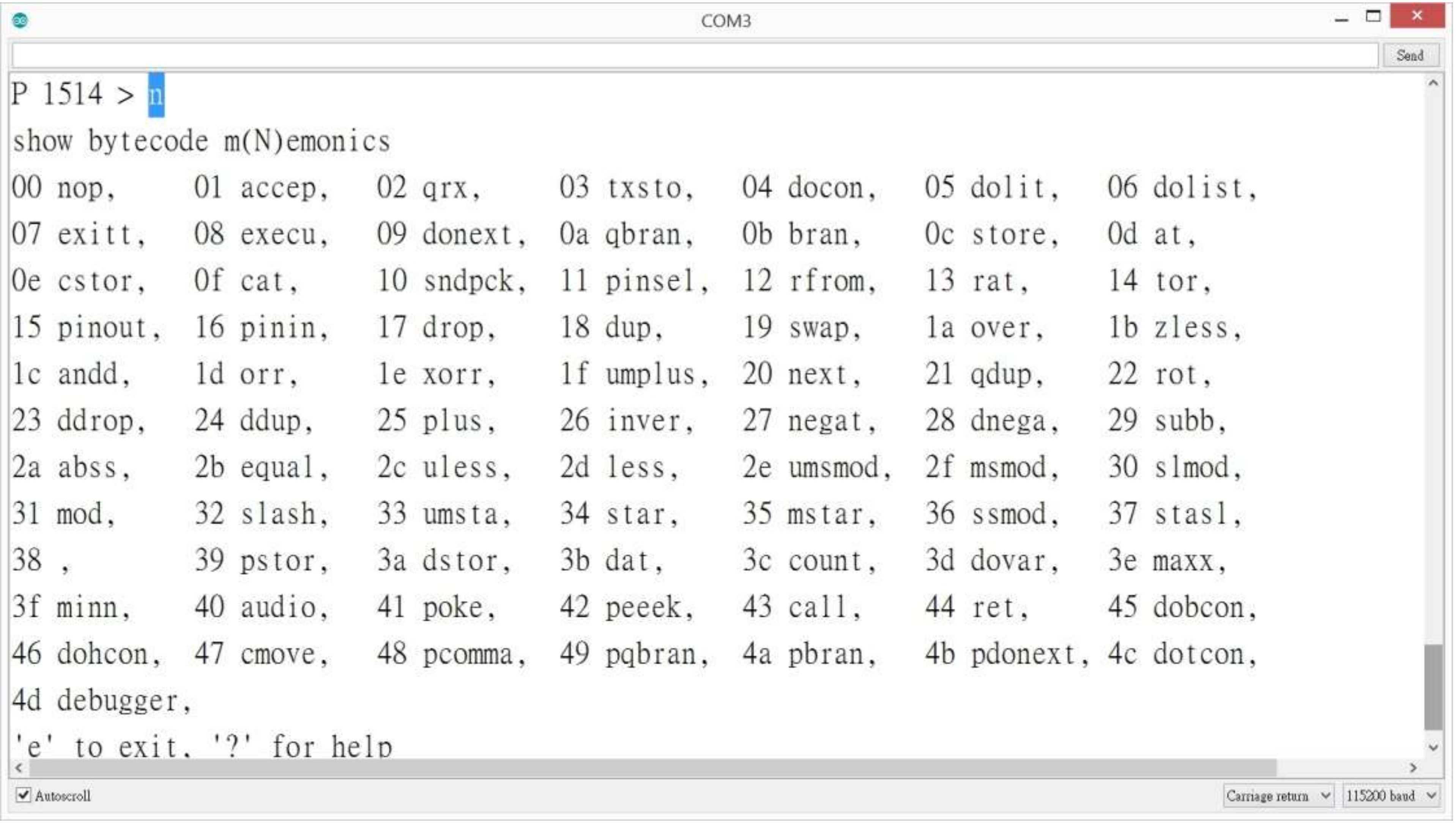
ledLow turns on led, via two subroutine calls to push values onto data stack.

ledset, ledOff, ledOn truns on/off led.

9 WAIT runs empty FOR-NEXT Loop 10 times.
CMOVE replaced by calling C function strncpy(dest,sour,nchr) directly.

DEBUGGER gets into debugger.

Byte Code Mnemonics of 11 new



```
P 1514 > n
show bytecode m(N)emonics
00 nop,     01 accep,   02 qrx,     03 txsto,   04 docon,   05 dolit,   06 dolist,
07 exitt,   08 execu,   09 donext,  0a qbran,   0b bran,    0c store,   0d at,
0e cstor,   0f cat,     10 sndpck,  11 pinsel,  12 rfrom,   13 rat,     14 tor,
15 pinout,  16 pinin,   17 drop,    18 dup,     19 swap,    1a over,    1b zless,
1c andd,    1d orr,     1e xorr,    1f umplus,  20 next,    21 qdup,    22 rot,
23 ddrop,   24 ddup,    25 plus,    26 inver,   27 negat,   28 dnega,   29 subb,
2a abss,    2b equal,   2c uless,   2d less,    2e umsmod,  2f msmod,   30 slmod,
31 mod,     32 slash,   33 umsta,   34 star,    35 mstar,   36 ssmod,   37 stasl,
38 ,        39 pstor,   3a dstor,   3b dat,     3c count,   3d dovar,   3e maxx,
3f minn,    40 audio,   41 poke,    42 peeek,   43 call,    44 ret,     45 dobcon,
46 dohcon,  47 cmove,   48 pcomma,  49 pqbran,  4a pbran,   4b pdonext, 4c dotcon,
4d debugger,
'e' to exit, '?' for help
```

New byte codes tried

01. 0x43 call,   0x44 ret,

for subroutine call in byte code programing.

02. 0x45 dobcon, 0x46 dohcon, 0x4c dotcon,

for 8-bit, 16-bit and 24-bit small integers.

03. 0x47 cmove, 0x48 comma,

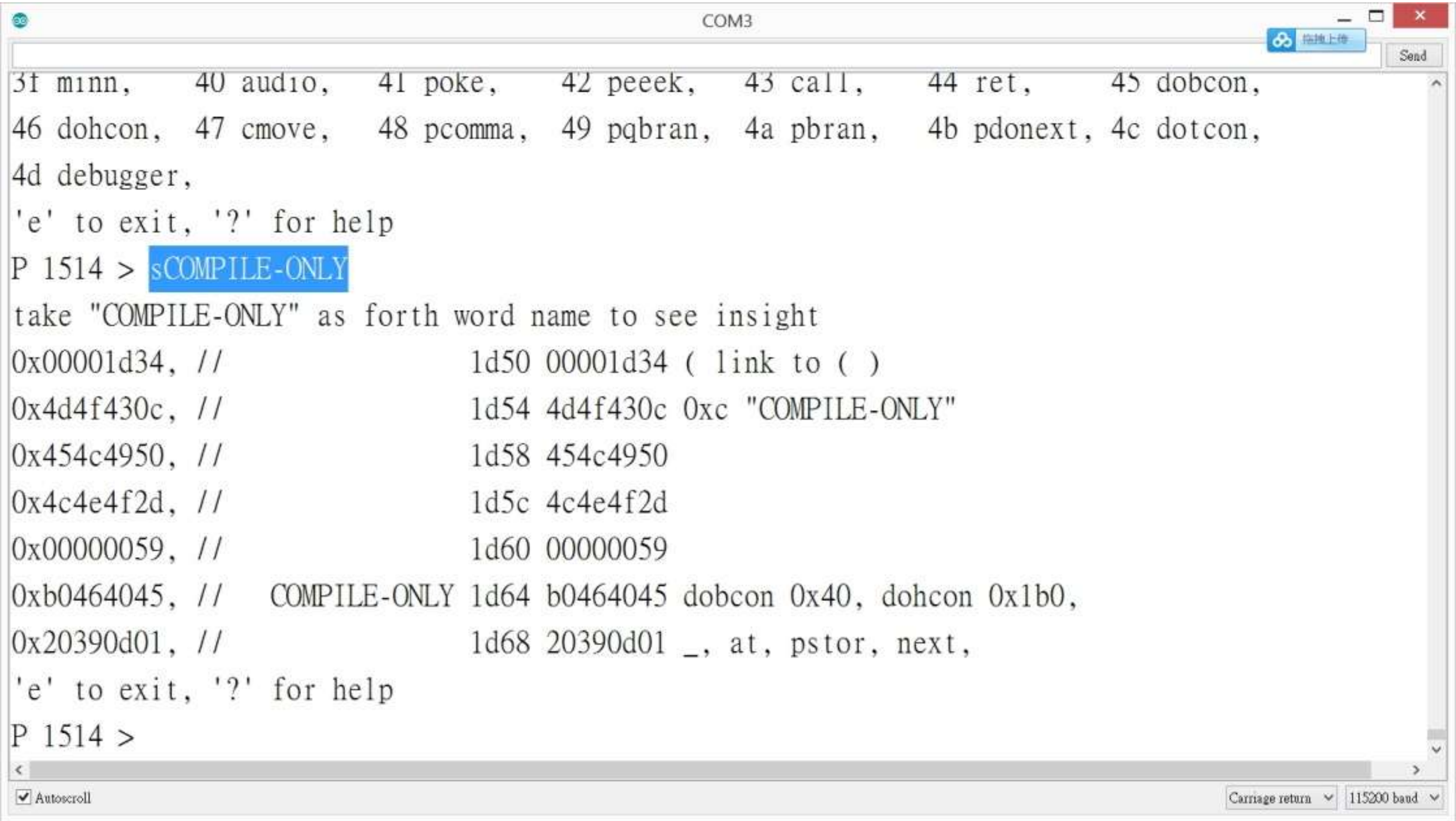
try to optimize.

04. 0x49 pqbran, 0x4a pbran, 0x4b pdonext,

for branching and looping in byte code programing.

05. 0x4d debugger, get into debugger.

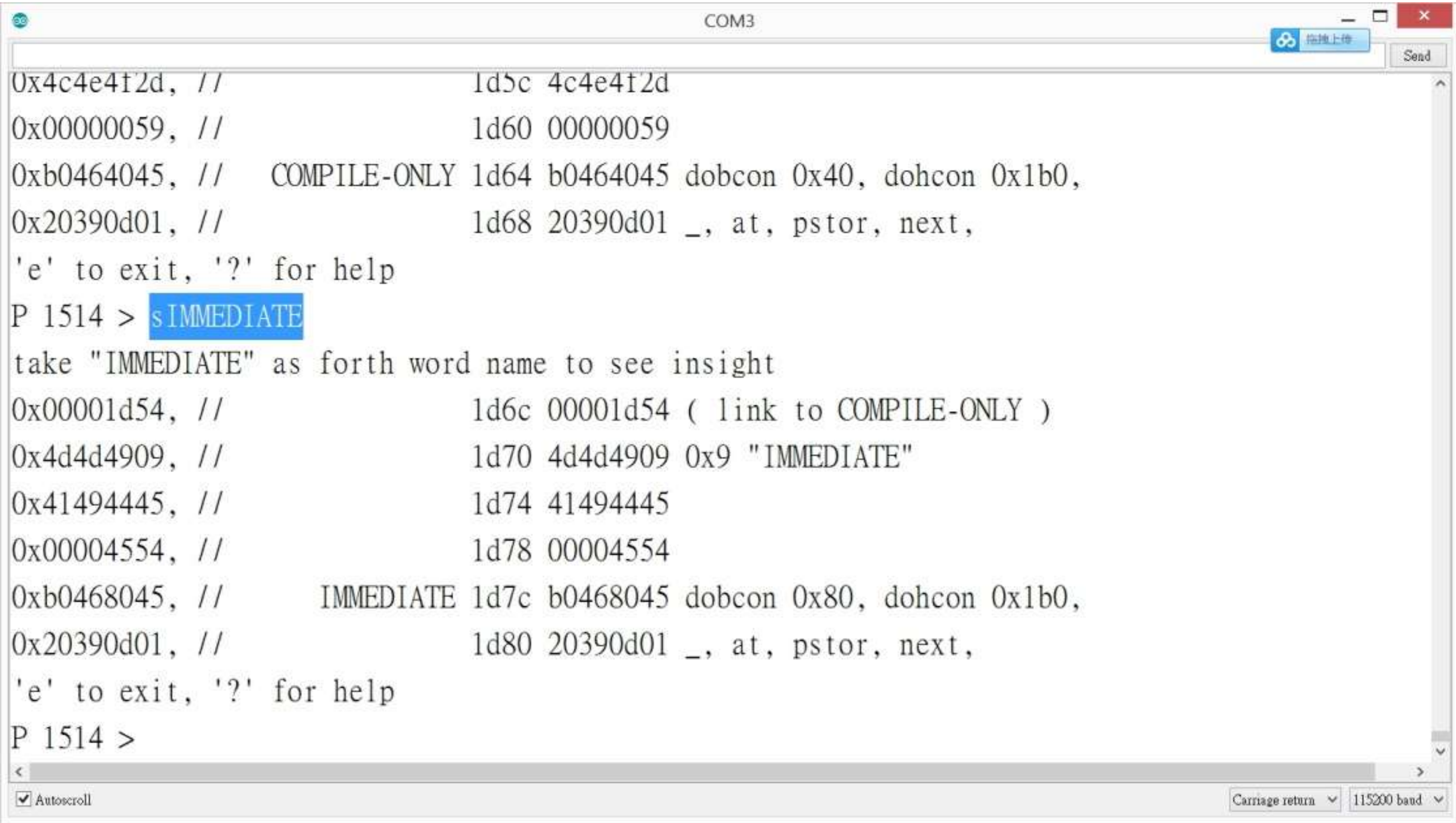
The word COMPILIE-ONLY



```
3f minn,    40 audio,   41 poke,    42 peeek,   43 call,    44 ret,     45 dobcon,
46 dohcon,  47 cmove,   48 pcomma,  49 pqbran,  4a pbran,   4b pdonext, 4c dotcon,
4d debugger,
'e' to exit, '?' for help
P 1514 > sCOMPILE-ONLY
take "COMPILE-ONLY" as forth word name to see insight
0x00001d34, //              1d50 00001d34 ( link to ( )
0x4d4f430c, //              1d54 4d4f430c 0xc "COMPILE-ONLY"
0x454c4950, //              1d58 454c4950
0x4c4e4f2d, //              1d5c 4c4e4f2d
0x00000059, //              1d60 00000059
0xb0464045, //  COMPILE-ONLY 1d64 b0464045 dobcon 0x40, dohcon 0x1b0,
0x20390d01, //              1d68 20390d01 _, at, pstor, next,
'e' to exit, '?' for help
P 1514 >
```

new byte codes used: 0x45 dobcon, and 0x46 dohcon,

The word IMMEDIATE



```
0x4c4e4f2d, //                1d5c 4c4e4f2d
0x00000059, //                1d60 00000059
0xb0464045, //   COMPILE-ONLY 1d64 b0464045 dobcon 0x40, dohcon 0x1b0,
0x20390d01, //                1d68 20390d01 _, at, pstor, next,
'e' to exit, '?' for help
P 1514 > sIMMEDIATE
take "IMMEDIATE" as forth word name to see insight
0x00001d54, //                1d6c 00001d54 ( link to COMPILE-ONLY )
0x4d4d4909, //                1d70 4d4d4909 0x9 "IMMEDIATE"
0x41494445, //                1d74 41494445
0x00004554, //                1d78 00004554
0xb0468045, //      IMMEDIATE 1d7c b0468045 dobcon 0x80, dohcon 0x1b0,
0x20390d01, //                1d80 20390d01 _, at, pstor, next,
'e' to exit, '?' for help
P 1514 >
```

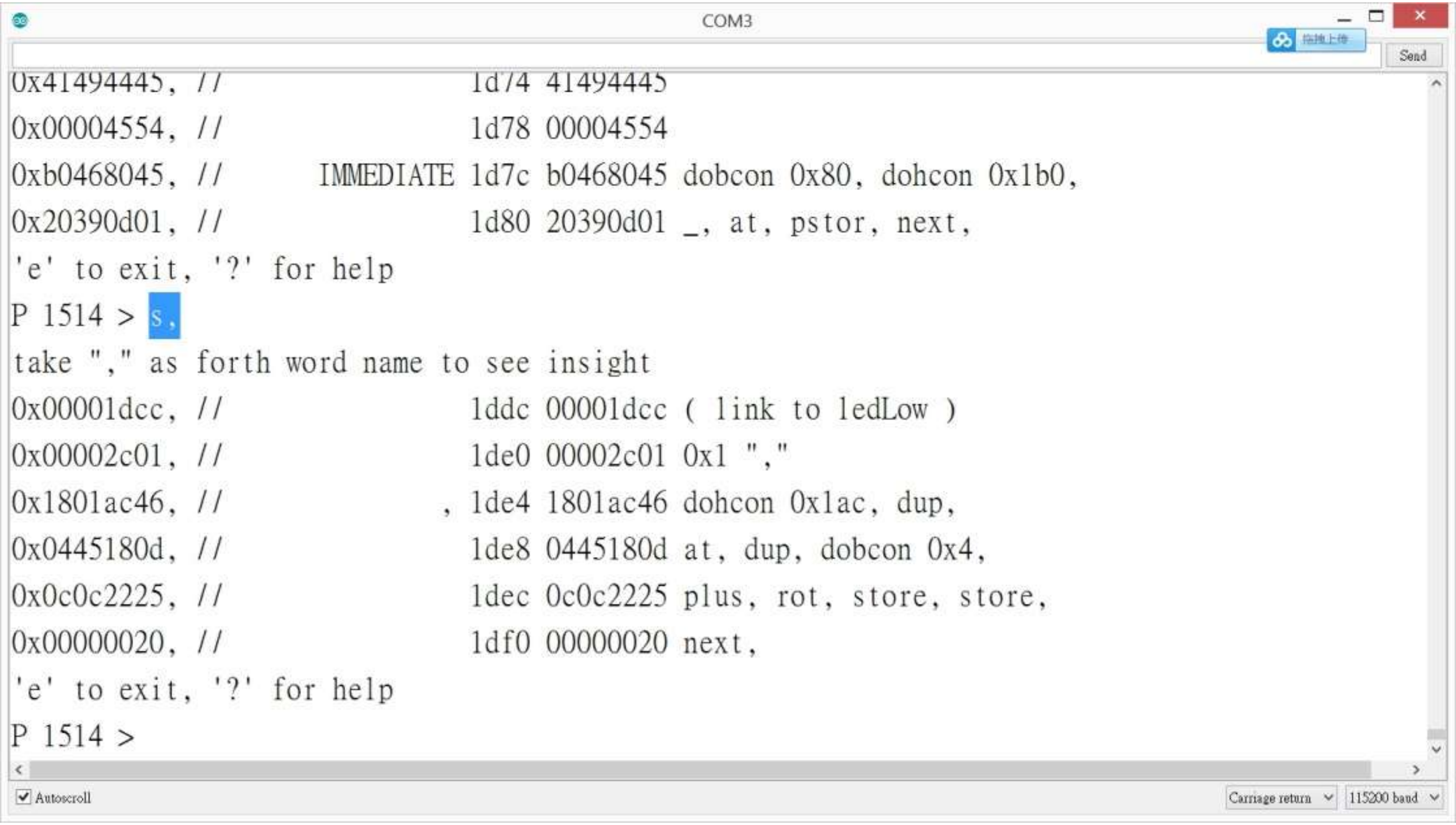
☑ Autoscroll

Carriage return

115200 baud

new byte codes used: 0x45 dobcon, and 0x46 dohcon,

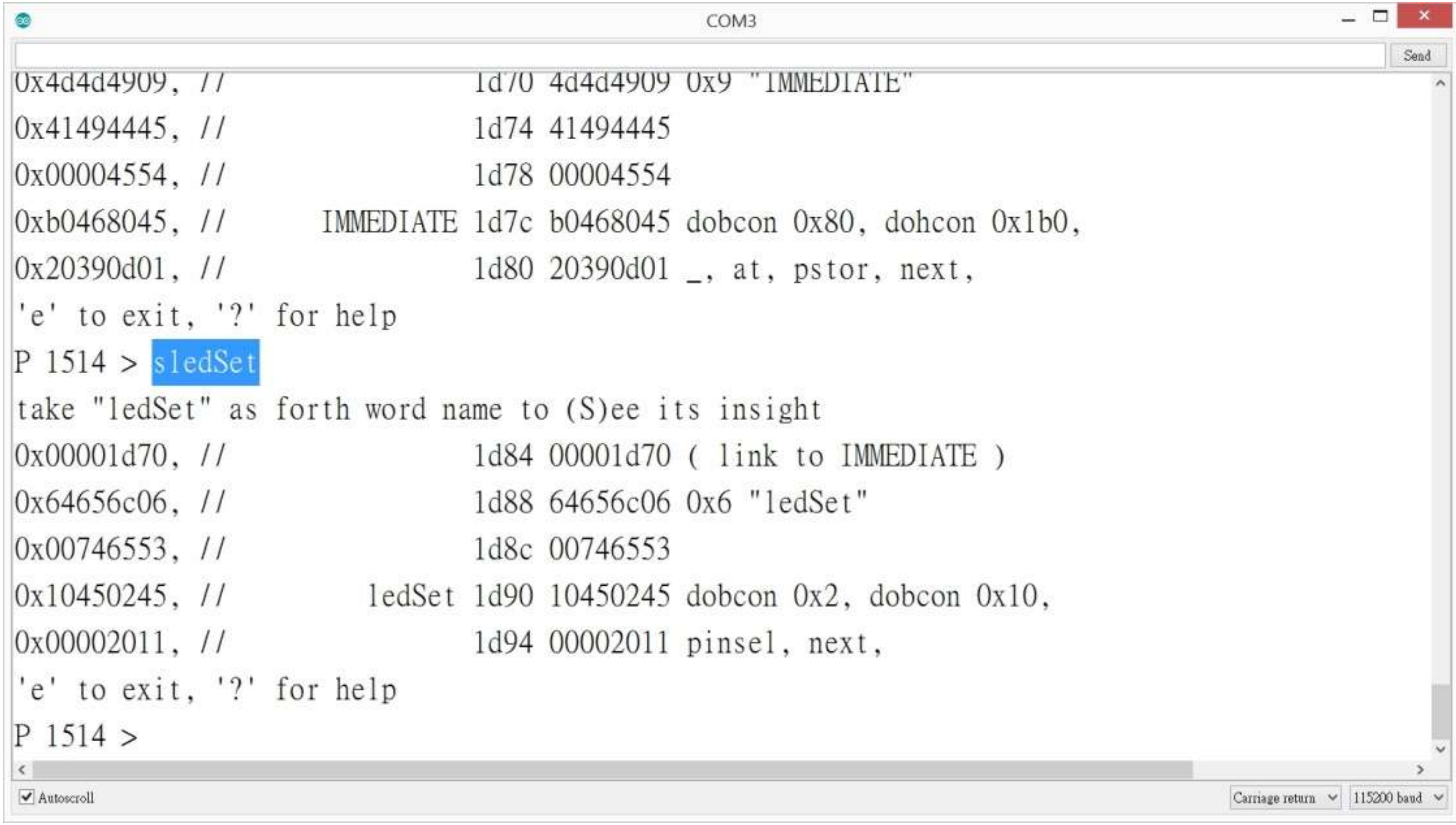
The word , (the comma)



```
0x41494445, //                  1d74 41494445
0x00004554, //                  1d78 00004554
0xb0468045, //        IMMEDIATE 1d7c b0468045 dobcon 0x80, dohcon 0x1b0,
0x20390d01, //                  1d80 20390d01 _, at, pstor, next,
'e' to exit, '?' for help
P 1514 > s ,
take "," as forth word name to see insight
0x00001dcc, //                  1ddc 00001dcc ( link to ledLow )
0x00002c01, //                  1de0 00002c01 0x1 ","
0x1801ac46, //                , 1de4 1801ac46 dohcon 0x1ac, dup,
0x0445180d, //                  1de8 0445180d at, dup, dobcon 0x4,
0x0c0c2225, //                  1dec 0c0c2225 plus, rot, store, store,
0x00000020, //                  1df0 00000020 next,
'e' to exit, '?' for help
P 1514 >
```

new byte codes used: 0x45 dobcon, and 0x46 dohcon,

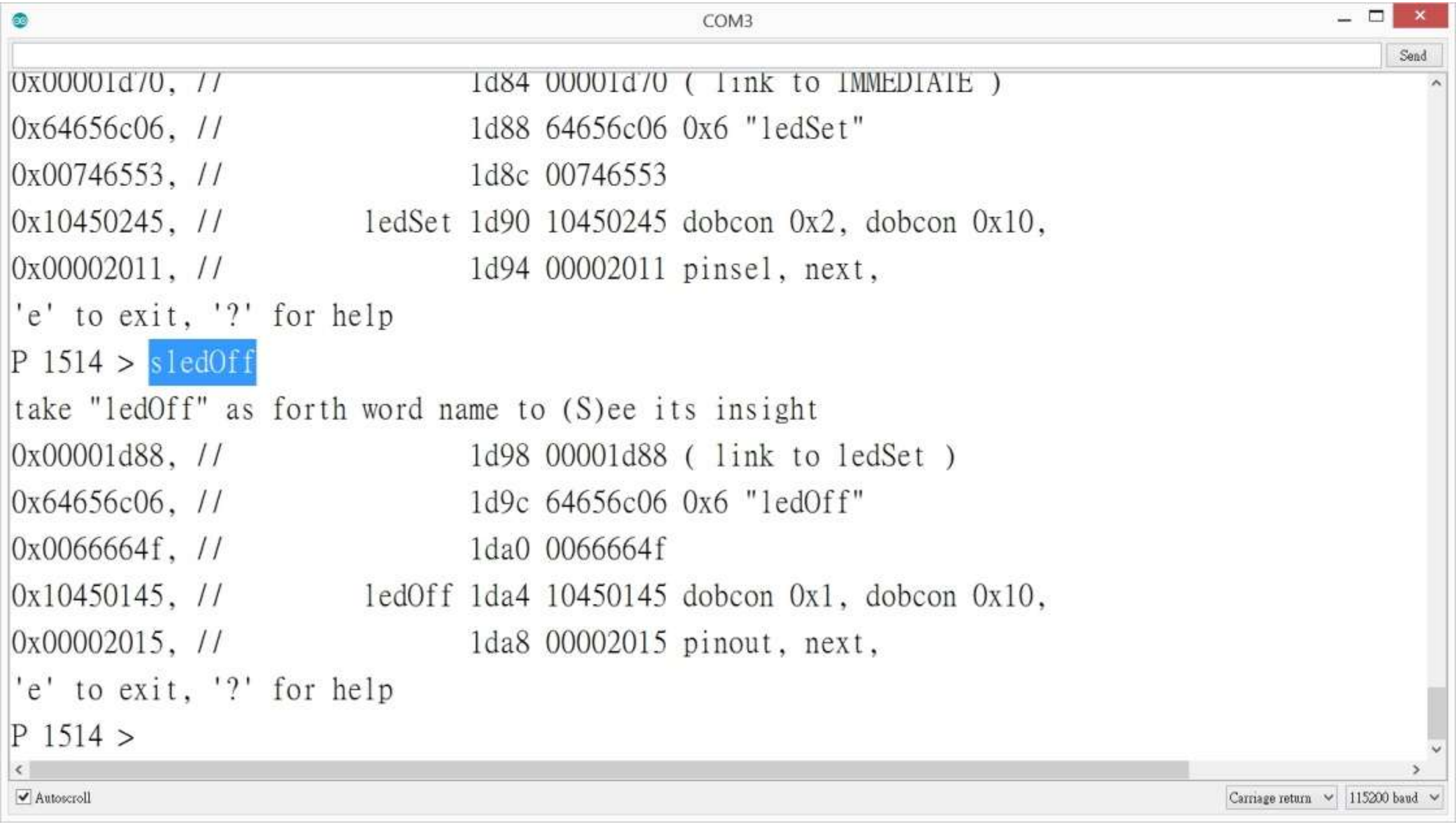
The word ledSet, setup led pin



```
0x4d4d4909, //                  1d70 4d4d4909 0x9 "IMMEDIATE"
0x41494445, //                  1d74 41494445
0x00004554, //                  1d78 00004554
0xb0468045, //        IMMEDIATE 1d7c b0468045 dobcon 0x80, dohcon 0x1b0,
0x20390d01, //                  1d80 20390d01 _, at, pstor, next,
'e' to exit, '?' for help
P 1514 > sledSet
take "ledSet" as forth word name to (S)ee its insight
0x00001d70, //                  1d84 00001d70 ( link to IMMEDIATE )
0x64656c06, //                  1d88 64656c06 0x6 "ledSet"
0x00746553, //                  1d8c 00746553
0x10450245, //           ledSet 1d90 10450245 dobcon 0x2, dobcon 0x10,
0x00002011, //                  1d94 00002011 pinsel, next,
'e' to exit, '?' for help
P 1514 >
```

new byte code used: 0x45 dobcon,

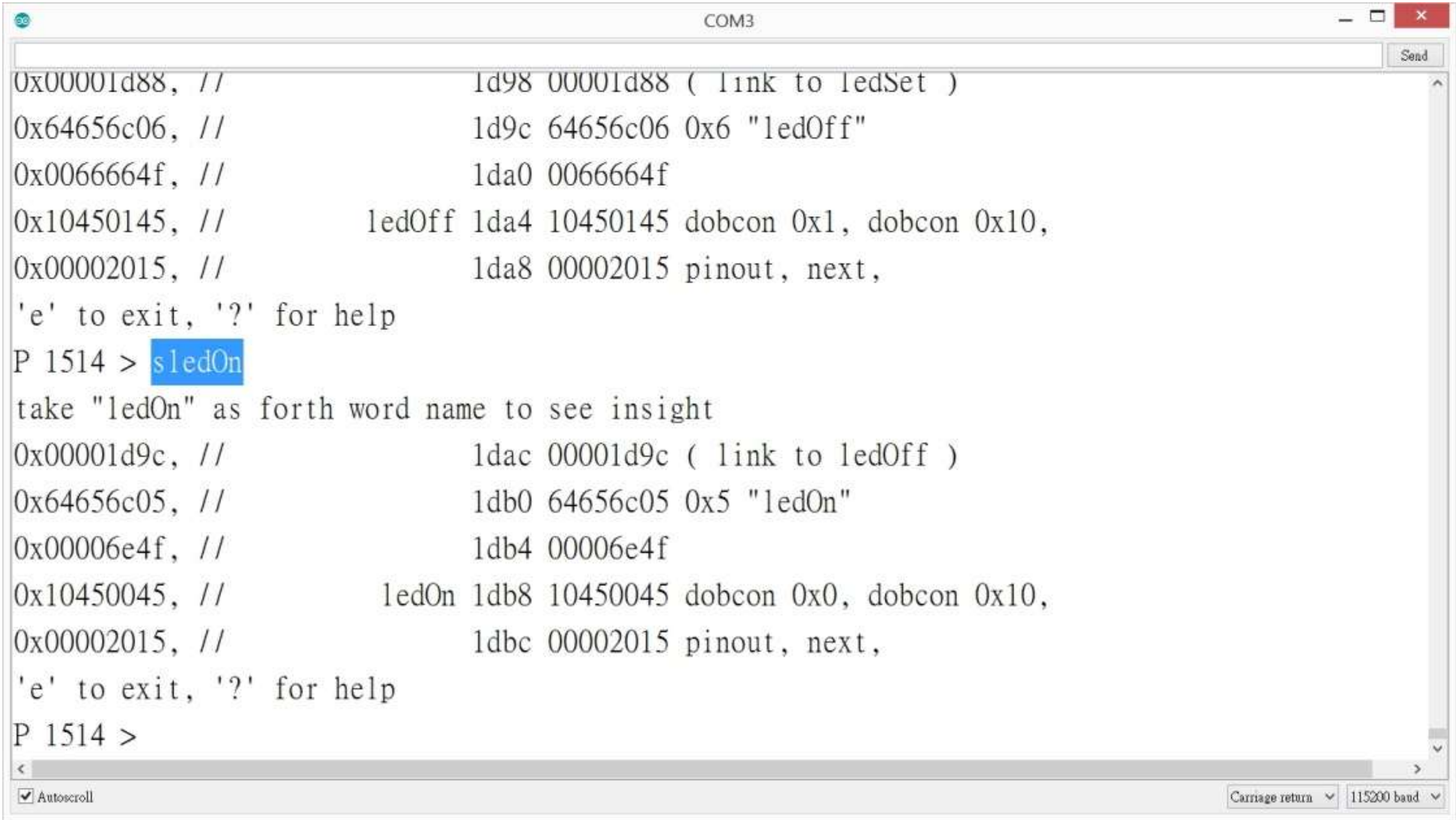
The word ledOff, turn off led



```
0x00001d70, //                  1d84 00001d70 ( link to IMMEDIATE )
0x64656c06, //                  1d88 64656c06 0x6 "ledSet"
0x00746553, //                  1d8c 00746553
0x10450245, //           ledSet 1d90 10450245 dobcon 0x2, dobcon 0x10,
0x00002011, //                  1d94 00002011 pinsel, next,
'e' to exit, '?' for help
P 1514 > sledOff
take "ledOff" as forth word name to (S)ee its insight
0x00001d88, //                  1d98 00001d88 ( link to ledSet )
0x64656c06, //                  1d9c 64656c06 0x6 "ledOff"
0x0066664f, //                  1da0 0066664f
0x10450145, //           ledOff 1da4 10450145 dobcon 0x1, dobcon 0x10,
0x00002015, //                  1da8 00002015 pinout, next,
'e' to exit, '?' for help
P 1514 >
```

new byte code used: 0x45 dobcon,

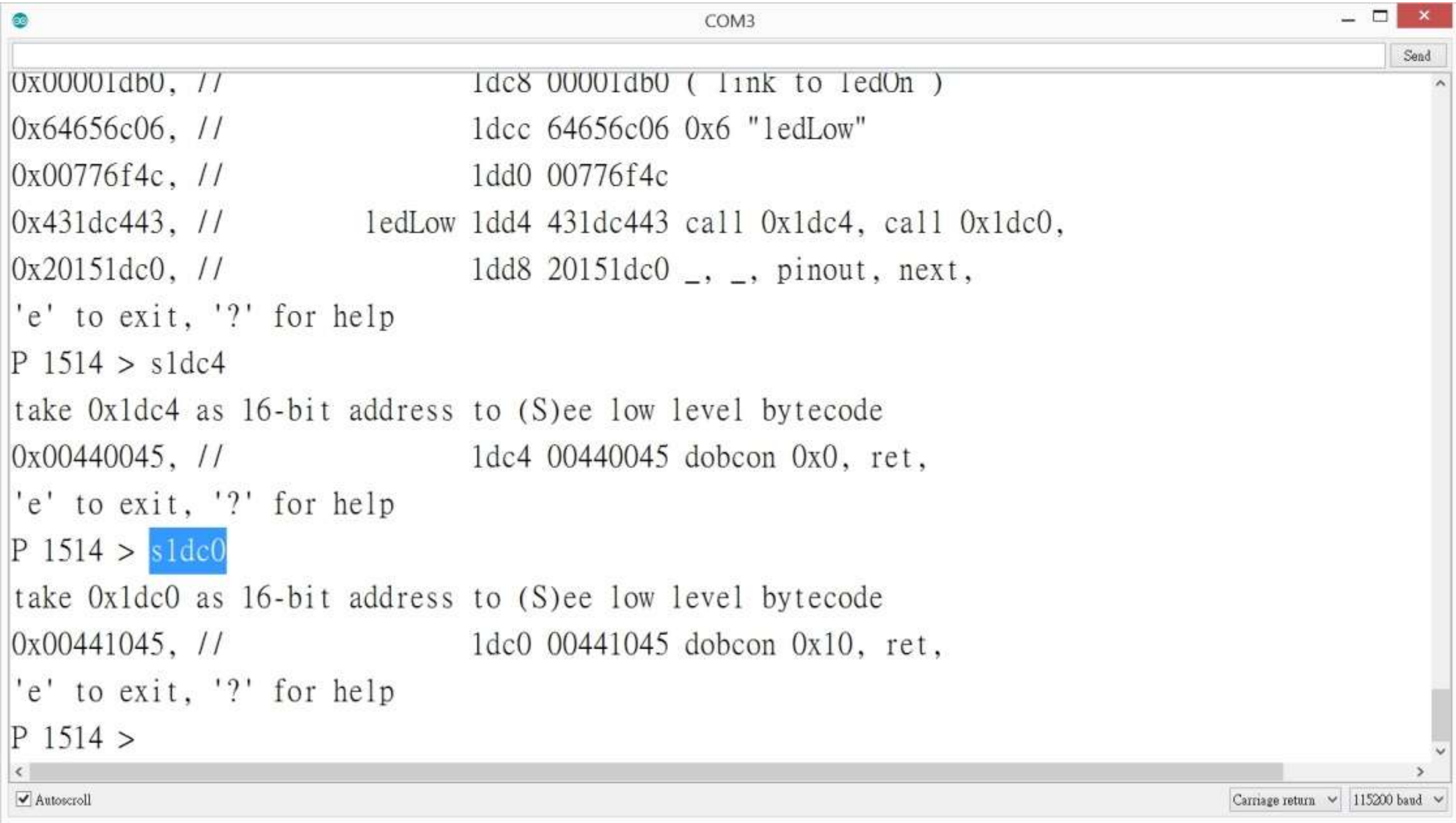
The word ledOn, turn on led



```
0x00001d88, //                    1d98 00001d88 ( link to ledSet )
0x64656c06, //                    1d9c 64656c06 0x6 "ledOff"
0x0066664f, //                    1da0 0066664f
0x10450145, //             ledOff 1da4 10450145 dobcon 0x1, dobcon 0x10,
0x00002015, //                    1da8 00002015 pinout, next,
'e' to exit, '?' for help
P 1514 > sledOn
take "ledOn" as forth word name to see insight
0x00001d9c, //                    1dac 00001d9c ( link to ledOff )
0x64656c05, //                    1db0 64656c05 0x5 "ledOn"
0x00006e4f, //                    1db4 00006e4f
0x10450045, //              ledOn 1db8 10450045 dobcon 0x0, dobcon 0x10,
0x00002015, //                    1dbc 00002015 pinout, next,
'e' to exit, '?' for help
P 1514 >
```

new byte code used: 0x45 dobcon,

Yet another word ledLow, turn on led

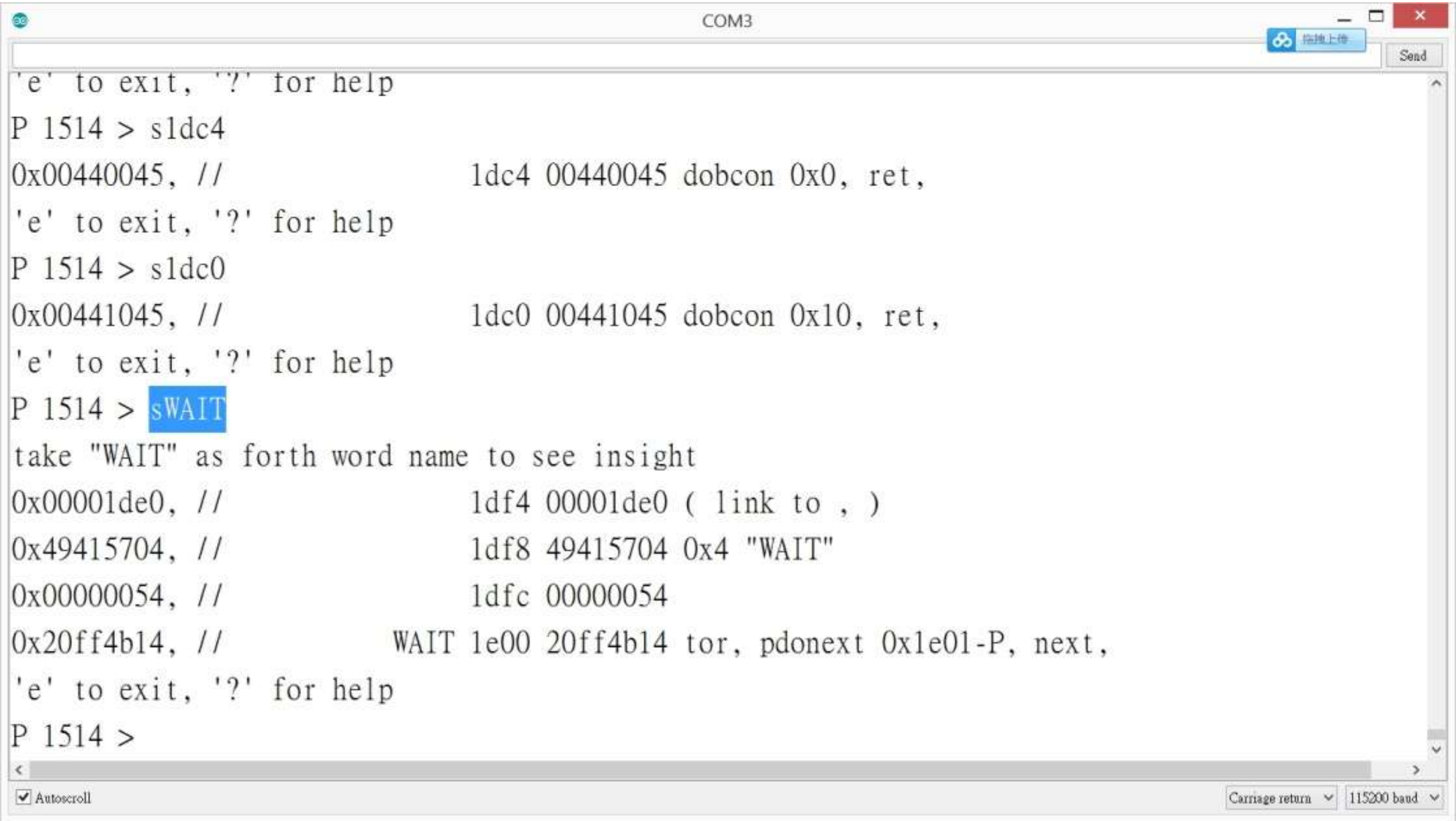


```
0x00001db0, //                1dc8 00001db0 ( link to ledOn )
0x64656c06, //                1dcc 64656c06 0x6 "ledLow"
0x00776f4c, //                1dd0 00776f4c
0x431dc443, //         ledLow 1dd4 431dc443 call 0x1dc4, call 0x1dc0,
0x20151dc0, //                1dd8 20151dc0 _, _, pinout, next,
'e' to exit, '?' for help
P 1514 > s1dc4
take 0x1dc4 as 16-bit address to (S)ee low level bytecode
0x00440045, //                1dc4 00440045 dobcon 0x0, ret,
'e' to exit, '?' for help
P 1514 > s1dc0
take 0x1dc0 as 16-bit address to (S)ee low level bytecode
0x00441045, //                1dc0 00441045 dobcon 0x10, ret,
'e' to exit, '?' for help
P 1514 >
```

☑ Autoscroll   Carriage return   115200 baud

new byte codes used: 0x43 call, 0x44 ret, and 0x45 dobcon,

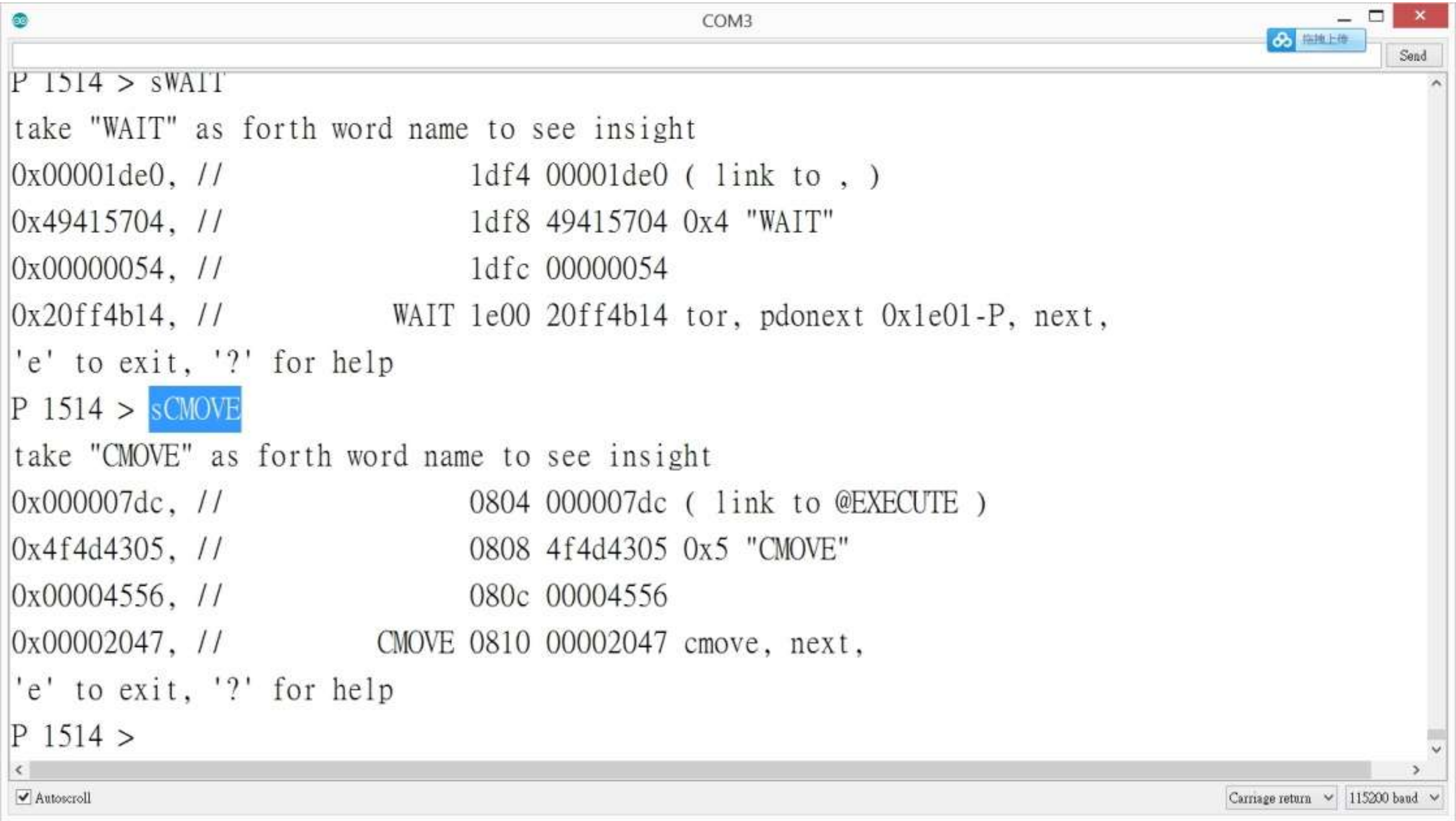
The word WAIT, run empty FOR-NEXT



```
'e' to exit, '?' for help
P 1514 > sldc4
0x00440045, //              1dc4 00440045 dobcon 0x0, ret,
'e' to exit, '?' for help
P 1514 > sldc0
0x00441045, //              1dc0 00441045 dobcon 0x10, ret,
'e' to exit, '?' for help
P 1514 > sWAIT
take "WAIT" as forth word name to see insight
0x00001de0, //              1df4 00001de0 ( link to , )
0x49415704, //              1df8 49415704 0x4 "WAIT"
0x00000054, //              1dfc 00000054
0x20ff4b14, //         WAIT 1e00 20ff4b14 tor, pdonext 0x1e01-P, next,
'e' to exit, '?' for help
P 1514 >
```

new byte code used: 0x4b pdonext,

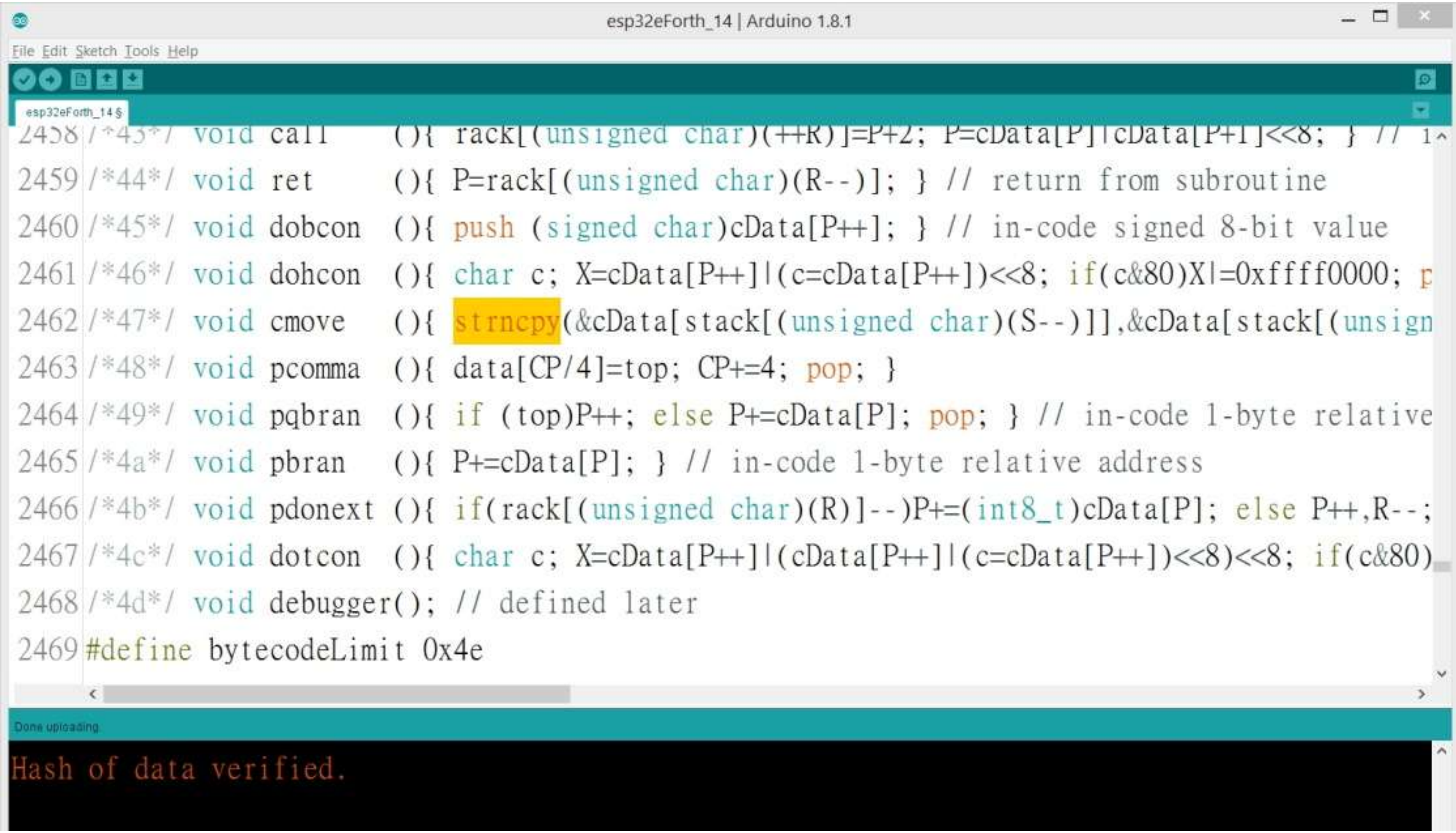
The word CMOVE



```
P 1514 > sWAIT
take "WAIT" as forth word name to see insight
0x00001de0, //                1df4 00001de0 ( link to , )
0x49415704, //                1df8 49415704 0x4 "WAIT"
0x00000054, //                1dfc 00000054
0x20ff4b14, //           WAIT 1e00 20ff4b14 tor, pdonext 0x1e01-P, next,
'e' to exit, '?' for help
P 1514 > sCMOVE
take "CMOVE" as forth word name to see insight
0x000007dc, //                0804 000007dc ( link to @EXECUTE )
0x4f4d4305, //                0808 4f4d4305 0x5 "CMOVE"
0x00004556, //                080c 00004556
0x00002047, //          CMOVE 0810 00002047 cmove, next,
'e' to exit, '?' for help
P 1514 >
```

new byte code used: 0x47 cmove,

Byte code 0x47 runs C function strncpy()

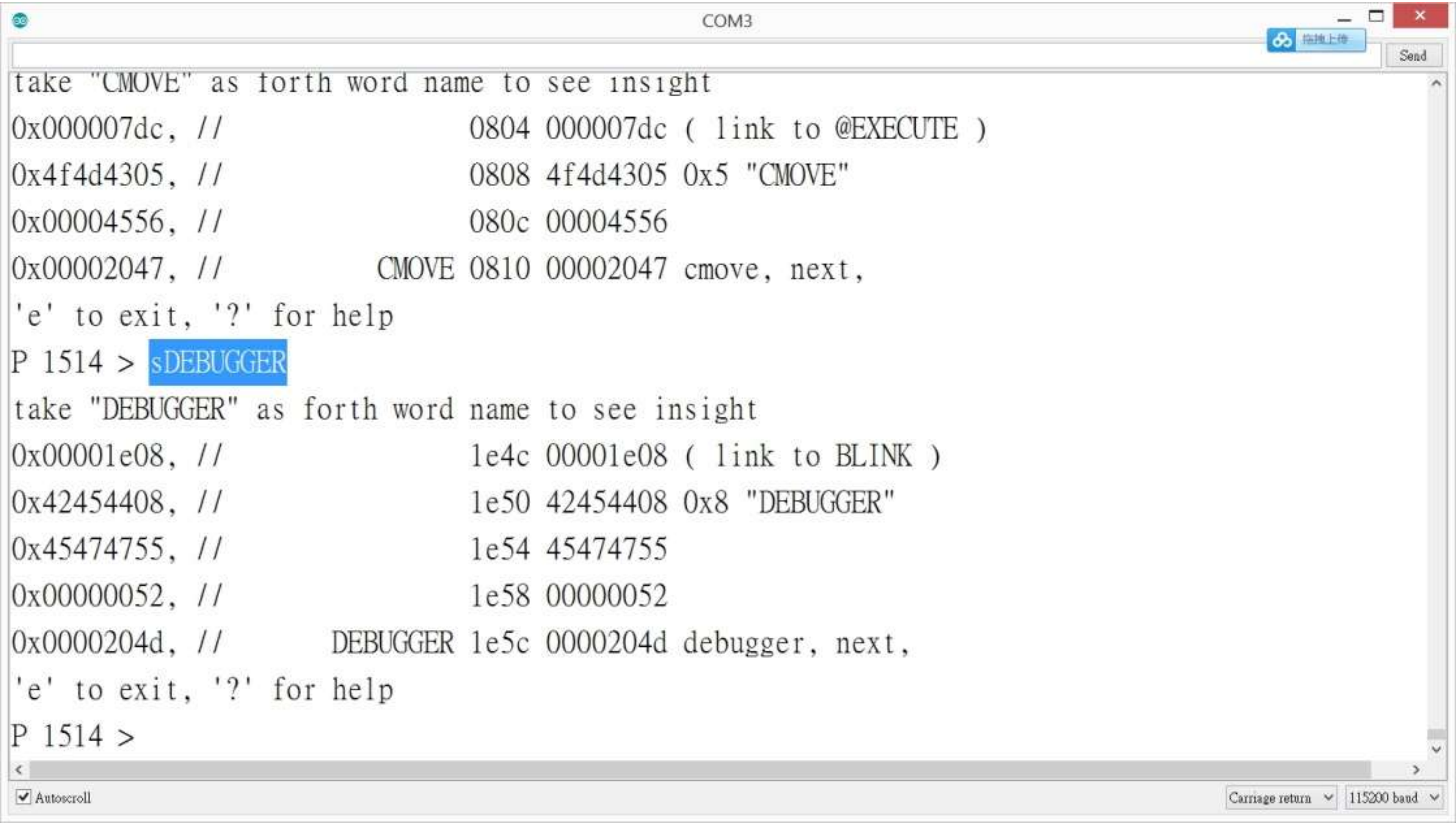


```
/*43*/ void call    (){ rack[(unsigned char)(++R)]=P+2; P=cData[P]|cData[P+1]<<8; } // 1
/*44*/ void ret     (){ P=rack[(unsigned char)(R--)]; } // return from subroutine
/*45*/ void dobcon  (){ push (signed char)cData[P++]; } // in-code signed 8-bit value
/*46*/ void dohcon  (){ char c; X=cData[P++]|(c=cData[P++])<<8; if(c&80)X|=0xffff0000; p
/*47*/ void cmove   (){ strncpy(&cData[stack[(unsigned char)(S--)]],&cData[stack[(unsign
/*48*/ void pcomma  (){ data[CP/4]=top; CP+=4; pop; }
/*49*/ void pqbran  (){ if (top)P++; else P+=cData[P]; pop; } // in-code 1-byte relative
/*4a*/ void pbran   (){ P+=cData[P]; } // in-code 1-byte relative address
/*4b*/ void pdonext (){ if(rack[(unsigned char)(R)]--)P+=(int8_t)cData[P]; else P++,R--;
/*4c*/ void dotcon  (){ char c; X=cData[P++]|(cData[P++]|(c=cData[P++])<<8)<<8; if(c&80)
/*4d*/ void debugger(); // defined later
#define bytecodeLimit 0x4e
```

Done uploading.

```
Hash of data verified.
```

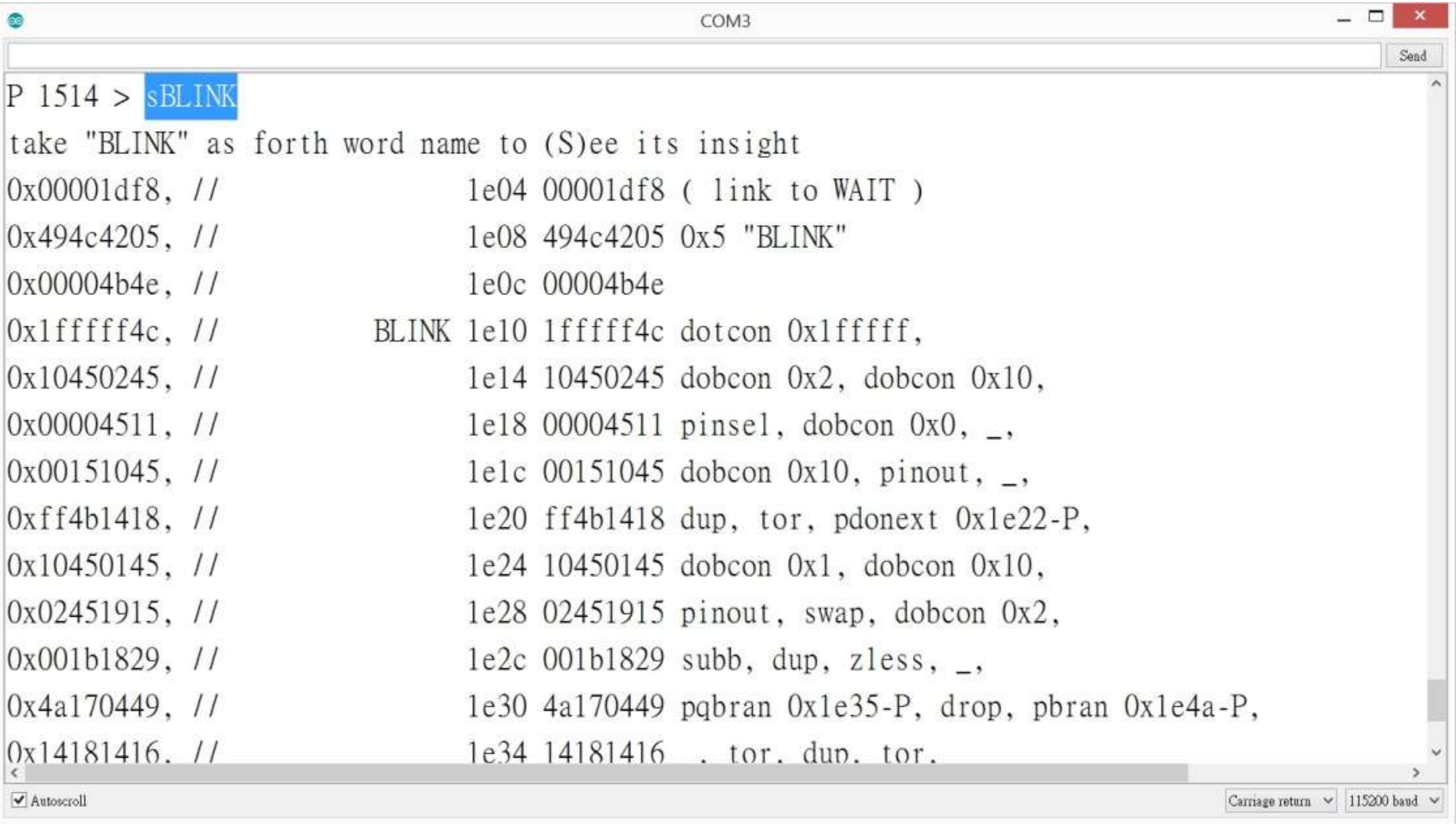
The word DEBUGGER



```
take "CMOVE" as forth word name to see insight
0x000007dc, //                 0804 000007dc ( link to @EXECUTE )
0x4f4d4305, //                 0808 4f4d4305 0x5 "CMOVE"
0x00004556, //                 080c 00004556
0x00002047, //           CMOVE 0810 00002047 cmove, next,
'e' to exit, '?' for help
P 1514 > sDEBUGGER
take "DEBUGGER" as forth word name to see insight
0x00001e08, //                 1e4c 00001e08 ( link to BLINK )
0x42454408, //                 1e50 42454408 0x8 "DEBUGGER"
0x45474755, //                 1e54 45474755
0x00000052, //                 1e58 00000052
0x0000204d, //        DEBUGGER 1e5c 0000204d debugger, next,
'e' to exit, '?' for help
P 1514 >
```

new byte code used: 0x4d debugger,

The word BLINK, blink led n times



```
P 1514 > sBLINK
take "BLINK" as forth word name to (S)ee its insight
0x00001df8, //               1e04 00001df8 ( link to WAIT )
0x494c4205, //               1e08 494c4205 0x5 "BLINK"
0x00004b4e, //               1e0c 00004b4e
0x1ffff4c, //          BLINK 1e10 1ffff4c dotcon 0x1ffff,
0x10450245, //               1e14 10450245 dobcon 0x2, dobcon 0x10,
0x00004511, //               1e18 00004511 pinsel, dobcon 0x0, _,
0x00151045, //               1e1c 00151045 dobcon 0x10, pinout, _,
0xff4b1418, //               1e20 ff4b1418 dup, tor, pdonext 0x1e22-P,
0x10450145, //               1e24 10450145 dobcon 0x1, dobcon 0x10,
0x02451915, //               1e28 02451915 pinout, swap, dobcon 0x2,
0x001b1829, //               1e2c 001b1829 subb, dup, zless, _,
0x4a170449, //               1e30 4a170449 pqbran 0x1e35-P, drop, pbran 0x1e4a-P,
0x14181416, //               1e34 14181416 . tor, dup, tor,
```

☑ Autoscroll   Carriage return   115200 baud

45 dobcon, 49 pqbran, 4a pbran, 4b pdonext, 4c dotcon,

Not only work Hard but Smart

Use tools, even design tools, is important.

Via tracing, debugging, and testing, I’ve learned:

1. Redefine colon words in byte codes.

2. Define subroutines in byte codes.

3. Redefine byte codes to call C function.

4. The words defined in byte codes could not directly call other words.

5. Not indirectly calling to CFAs, but directly calling to subroutine at CFAs instead, could improve system even more.

Insight of esp32eForth_14

Questions?

Insight of esp32eForth_14

Thank you.