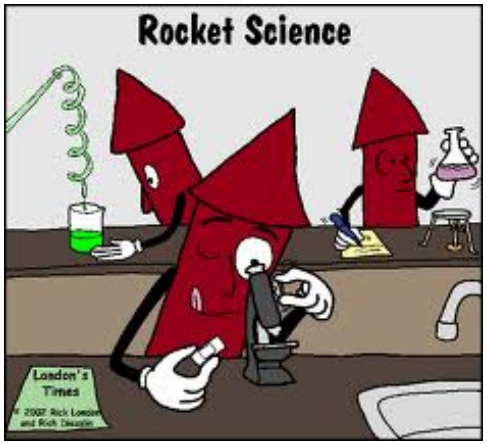
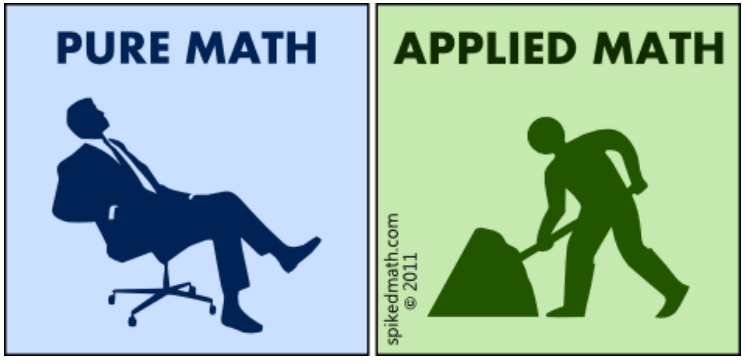
HP 20b → RPN Calculator

By Jayne (Jay) Shim &
SonYon (Sonny) Song

Accounting


09:40:04
NASDAQ

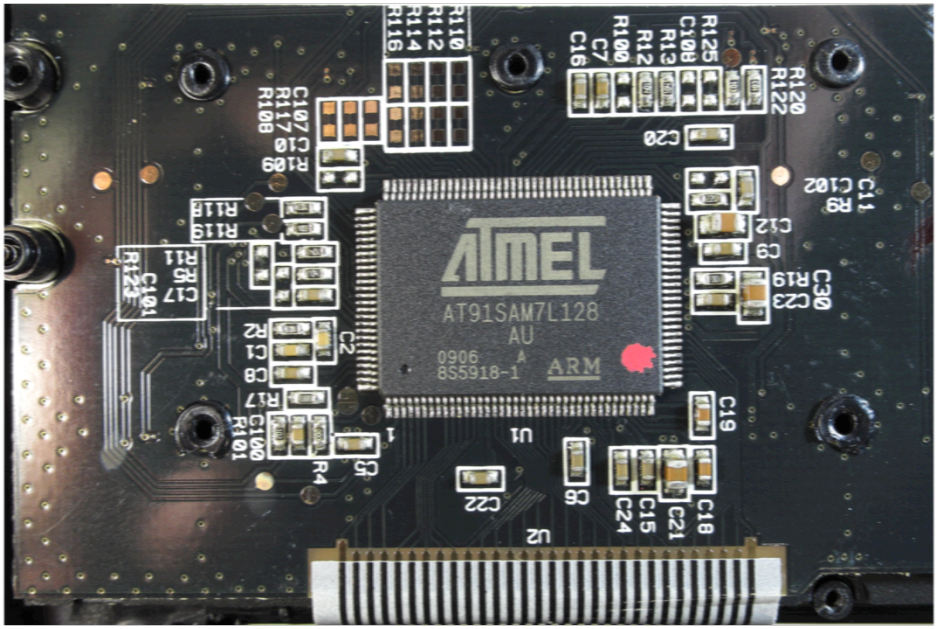

Rocket Science
London's Times

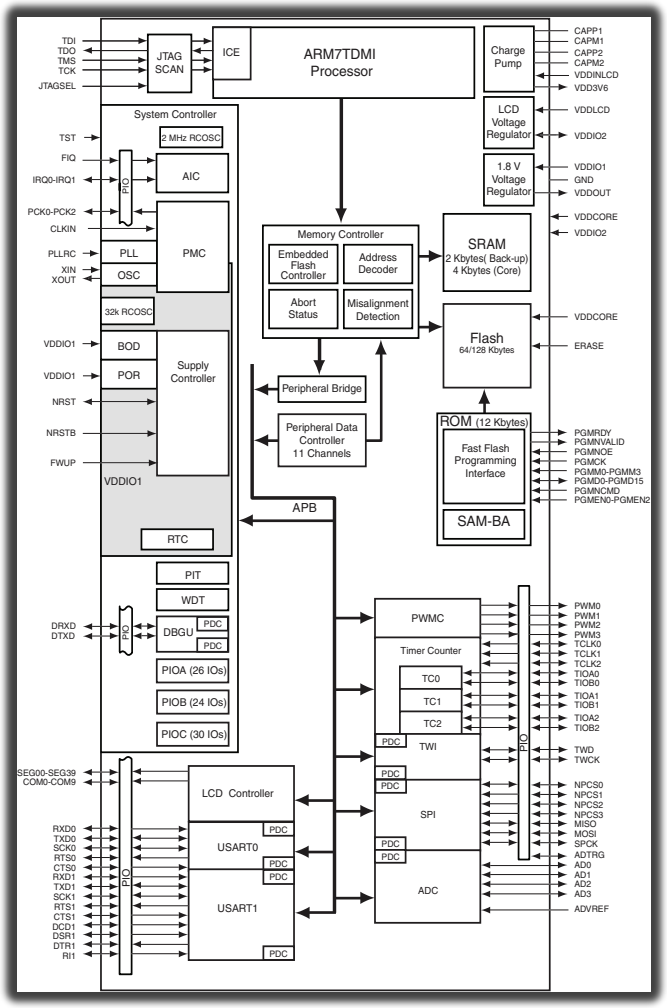

PURE MATH
APPLIED MATH
spikedmath.com
© 2011


HP 20b Business Consultant
N
I/YR
PV
PMT
FV
Amort
CshFl
IRR
NPV
Bond
%
RCL
INPUT
Memory
Mode
Reset
7
8
9
4
5
6
1
2
3
ON/CE
Off
0
AMS
BND

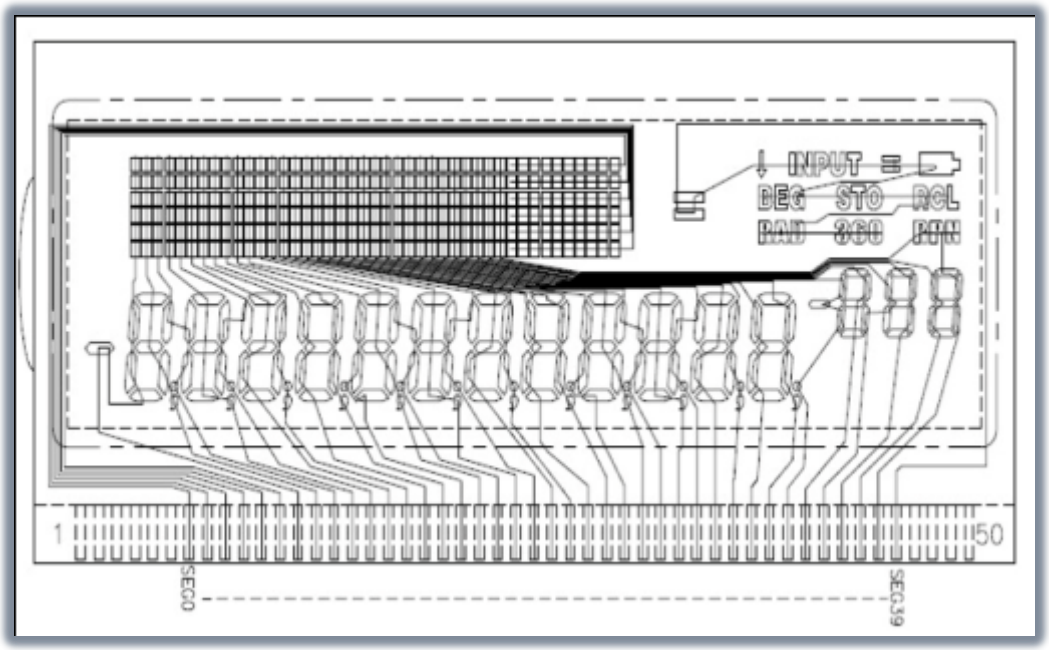
← ?

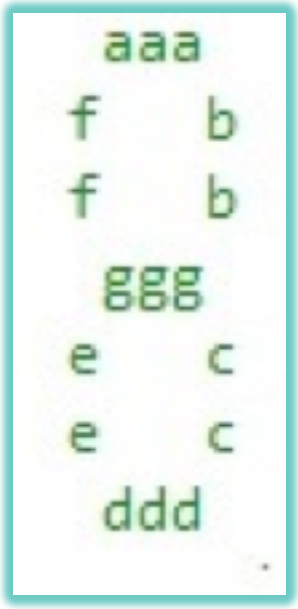
THE PROCESSOR



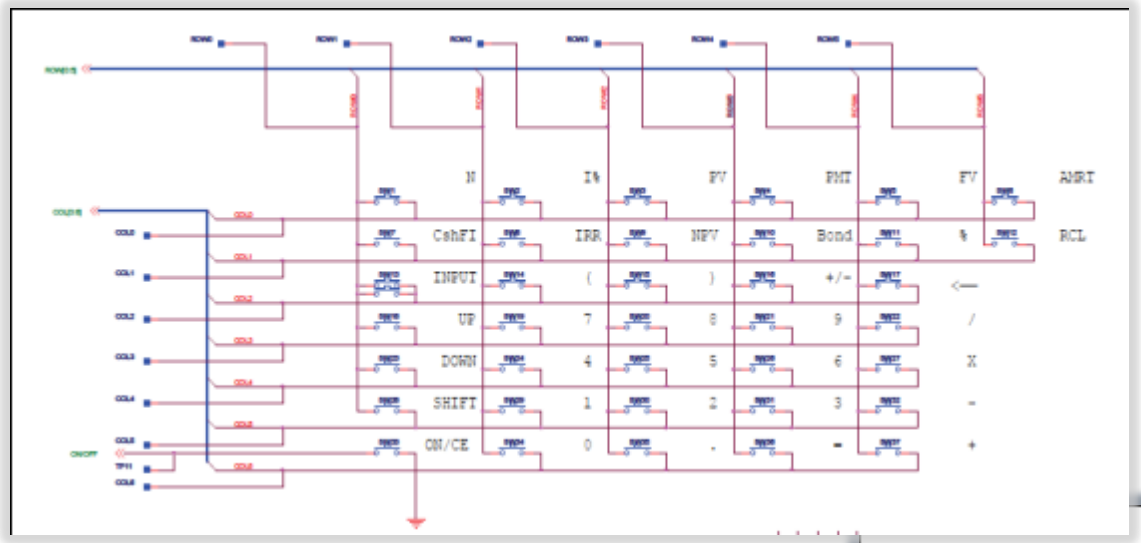


Liquid Crystal Display (LCD)



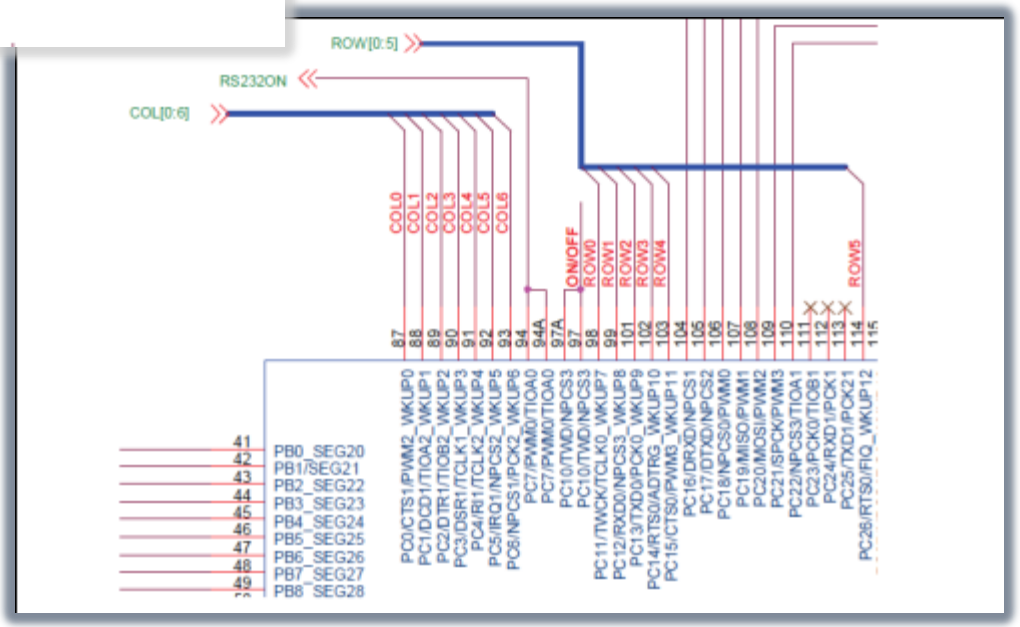


The Keyboard

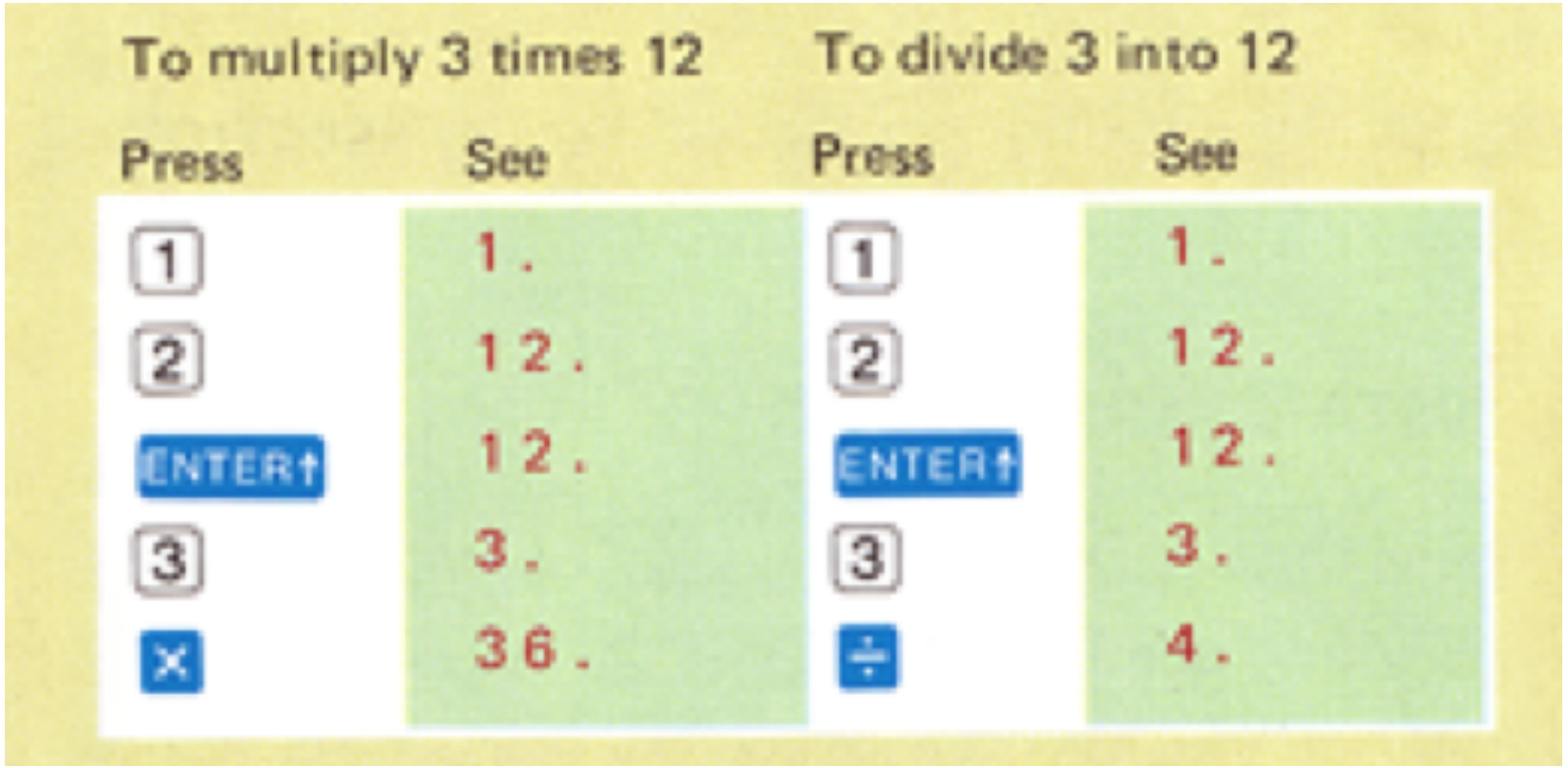


← Keyboard Matrix

Keyboard connected
to processor →




USER
GUIDE



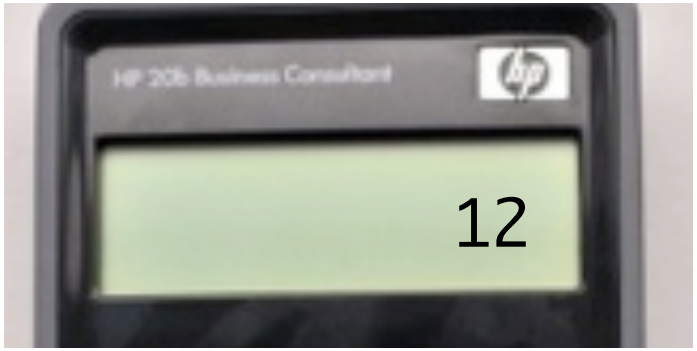
12 x 3 = 36

12 ÷ 3 = 4

ERROR

12 x 3 = 36

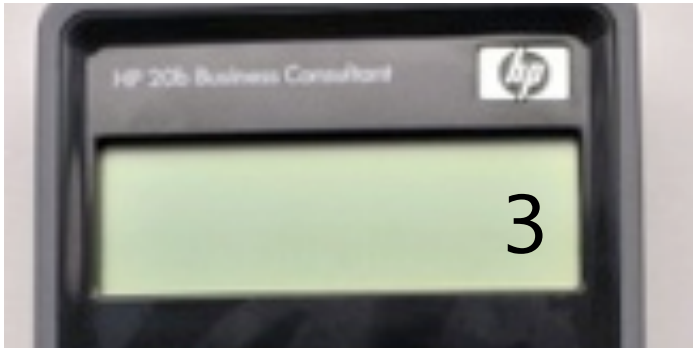
1. ENTER “1”
2. ENTER “2”
3. INPUT⇧



12				

12 x 3 = 36

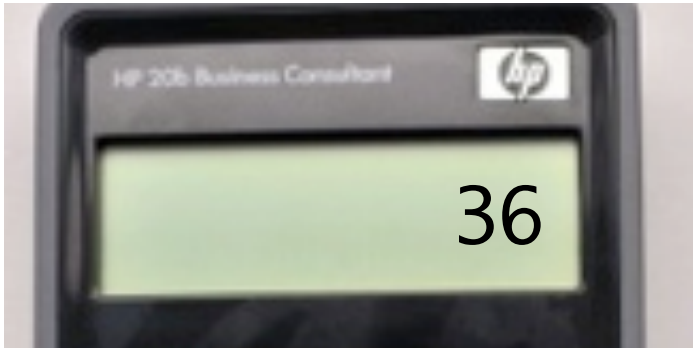
4. Enter “3”



12	3			

12 x 3 = 36

5. MULT “x”



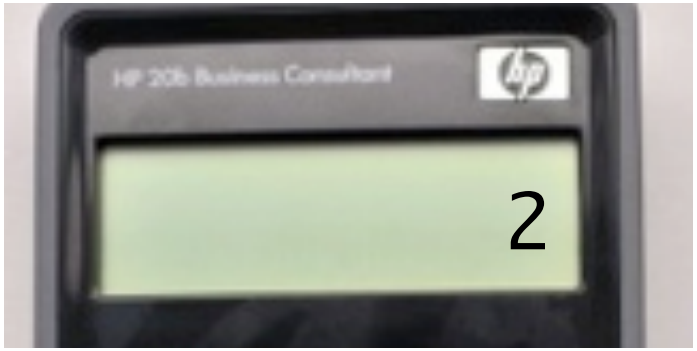
36				

What about...

a complex expression?!

$((2 + 3) \times 5) \times 6$

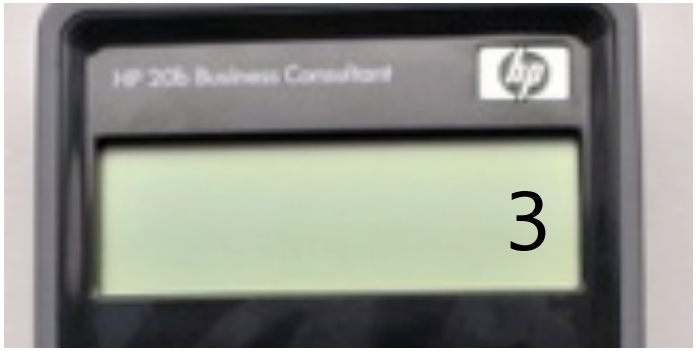
1. Enter “2”
2. INPUT⇧



2				

$((2 + 3) \times 5) \times 6$

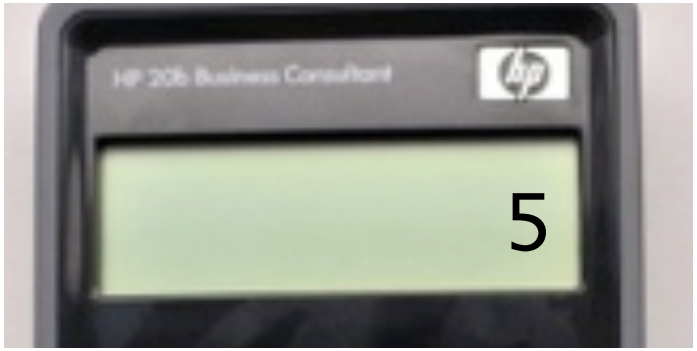
3. Enter “3”



2	3			

$((2 + 3) \times 5) \times 6$

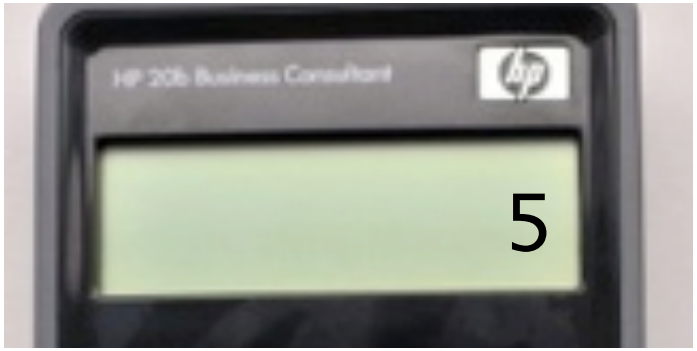
4. ADD “+”



5				

$((2 + 3) \times 5) \times 6$

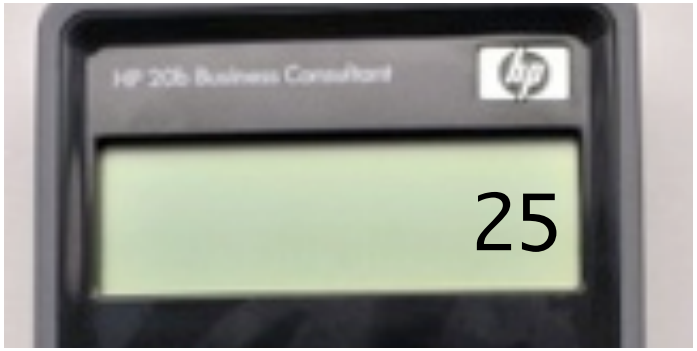
5. Enter “5”



5	5			

$((2 + 3) \times 5) \times 6$

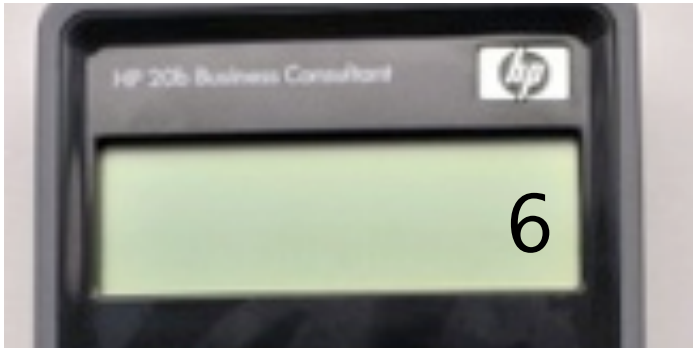
6. MULT “×”



25				

$$((2 + 3) \times 5) \times 6$$

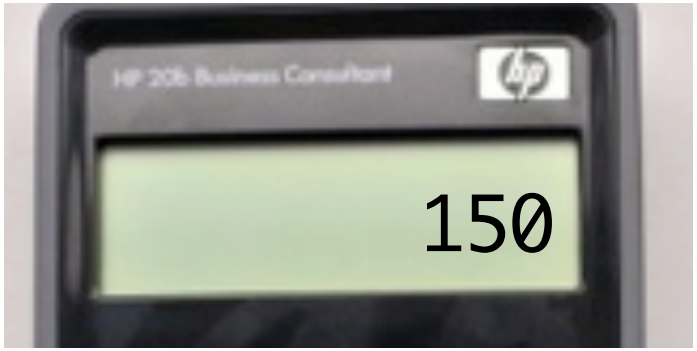
7. Enter “6”



25	6			

((2 + 3) × 5) × 6

8. MULT “×”

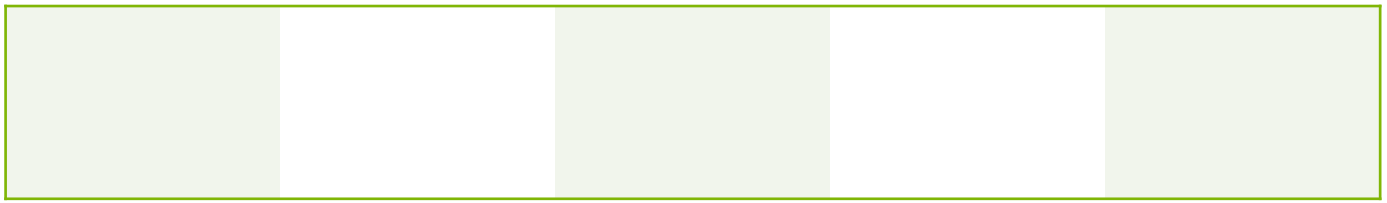


150				

ERROR: EMPTY STACK

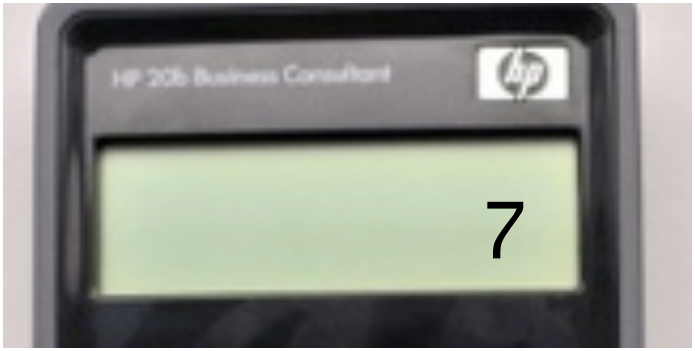
1. MULT “x”

Empty Stack

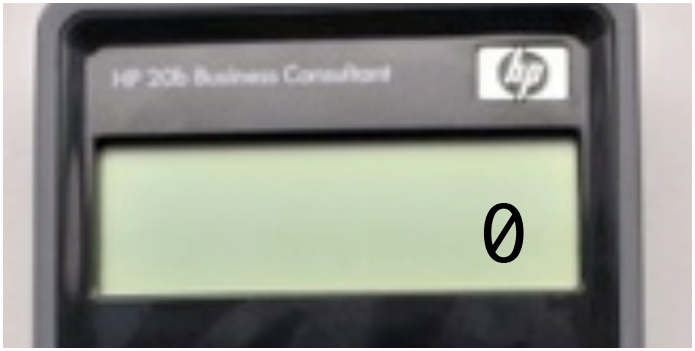


OTHER USEFUL KEYS

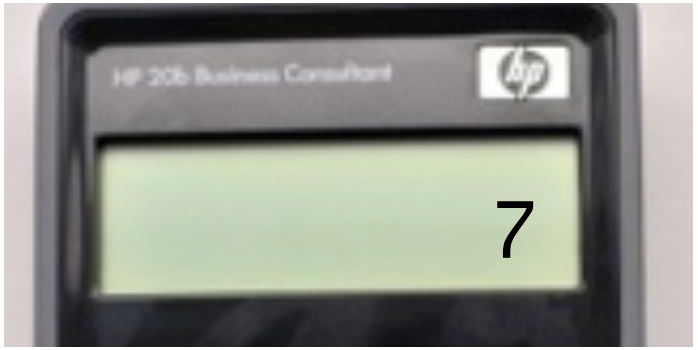
Backspace (🡐)



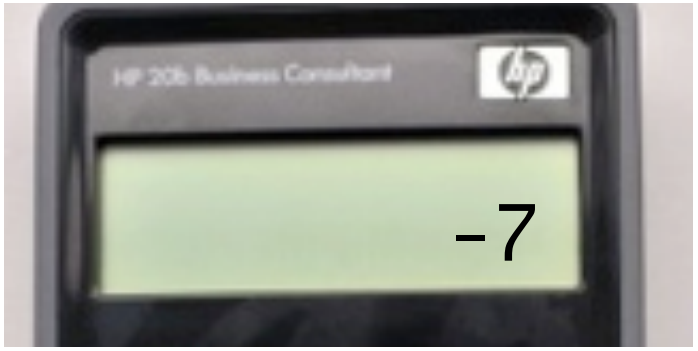
Press Backspace “🡐”



Sign (+/–)



Press Sign “+/–”



Getting Started: Hello World

```
#include "AT91SAM7L128.h“
#include "lcd.h"

int main()
{
    int number = 999 ;
    int digit;
    int lcdIndex = 11; /
    lcd_init();
    if (number > 999999999 || number < -999999999) {
      lcd_put_char7('O', 0);
      lcd_put_char7('V', 1);
      lcd_put_char7('E', 2);
      lcd_put_char7('R', 3);
      lcd_put_char7('F', 4);
      lcd_put_char7('L', 5);
      lcd_put_char7('O', 6);
      lcd_put_char7('W', 7);
     }
```

Getting Started: Hello World

```
else {
        if (number > 0) {
            while (number > 0) {
                digit = number % 10;
                number = number / 10;
                lcd_put_char7(digit+48, lcdIndex);
                lcdIndex--;
             }
    } else if (number == 0) {
        lcd_put_char7(48, lcdIndex);
    } else if (number <0) {
        while (number < 0) {
            digit = number % 10;
            digit = digit*-1;
            number = number / 10;
            lcd_put_char7(digit+48, lcdIndex);
            lcdIndex--;
        }
        if (number == 0) {
            lcd_put_char7(45, 0);
        }
    }
 }
  return 0;
}
```

Listening to the Keyboard

```
const char keyboard_layout [7][6] = {{'N',
'I', 'P', 'M', 'F', 'A'},

{'C', 'R', 'N', 'B', '%', 'L'},

{'T', '(', ')', 'G', '<', '_'},

{'U', '7', '8', '9', '/', '_'},

{'D', '4', '5', '6', '*', '_'},

{'S', '1', '2', '3', '-', '_'},

{'O', '0', '.', '=', '+', '_'}};
```

Listening to the Keyboard

```
char keyboard_key()
{
  int column;
  int row;
  char button;

  for (column = 0; column <= COLUMN; column++) {
      keyboard_column_low(column);
      for (row = 0; row <= ROW; row++) {
          if(!keyboard_row_read(row)) {
              keyboard_column_high(column);
              button = keyboard_layout[column][row];
              return button;
          }
      }
      keyboard_column_high(column);
  }
    return 0;
}
```

Listening to the Keyboard

```
int main() {

    for (;;) {
       char key = keyboard_key();
       if (key != 0) {
           for(j = 0; j < 14; j++) {
               lcd_put_char7(' ',j);
           }
           lcd_put_char7(key,11);
       } else {
           lcd_print7("no key pressed");
       }
    }
  return 0;

}
```

Entering and Displaying Numbers

```
void keyboard_get_entry(struct entry *result) {
    int input = 0;
    int counter = 0;
    int sign = 1;

    for (;;) {
        int c = keyboard_key();
        if (c <= '9' && c >= '0') {
            input = input * 10 + (c - '0');
            counter++;
        } else if (c == '~') {
            sign *= -1;
        } else if (c == '\b') {
            input /= 10;
            if(input == 0)
                lcd_put_char7(' ', 11);
        } else if (c != -1){
            result->operation = c;
            break;
        }
```

Entering and Displaying Numbers

```
        while (c != -1) {
            c = keyboard_key();
        }
        if (input != 0)
            lcd_print_int_neg(sign == -1, input);
    }

    if (counter == 0) {
        result->number = INT_MAX;
    } else {
        result->number = sign*input;
    }

}
```

Entering and Displaying Numbers

```
int main()
{
  struct entry entry;
  *AT91C_WDTC_WDMR = AT91C_WDTC_WDDIS;
  lcd_init();
  keyboard_init();
  lcd_print7("PRESS");

    while (1) {
        keyboard_get_entry(&entry);
        if (entry.operation != 0 && entry.number!
=INT_MAX) {
            lcd_print_int(entry.number);
            lcd_put_char7(entry.operation, 0);
            entry.operation = 0;
        }
    }
  return 0;
}
```

An RPN Calculator

```
void calculate(char operation, int index, int *stack) {
    if (index < 0) {
        lcd_print7("ERROR");
    } else {
        int n1 = stack[index];
        int n2 = stack[index+1];
        switch (operation) {
            case '+': stack[index] = n1 + n2; break;
            case '-': stack[index] = n1 - n2; break;
            case '*': stack[index] = n1 * n2; break;
            case '/': stack[index] = n1 / n2; break;
        }
        stack[index+1] = 0;
        lcd_print_int (stack[index]);
    }
}
```

An RPN Calculator

```
int main(){
    int stack [50];
    int index = 0;
    while (1) {
        keyboard_get_entry(&entry);
        if (entry.operation != 0 && entry.number!=INT_MAX) {
            stack[index] = entry.number;
            lcd_print_int(entry.number);
            if (entry.operation == '\r') {
                index++;
            } else if (entry.operation == '+' || entry.operation == '-' ||
                       entry.operation == '*' || entry.operation == '/') {
                index--;
                calculate(entry.operation, index, stack);
                index++;
            }
        } else if (entry.operation == '+' || entry.operation == '-' ||
                   entry.operation == '*' || entry.operation == '/') {
            if (index >= 2) {
                index = index - 2;
                calculate(entry.operation, index, stack);
                index++;
            } else {
                lcd_print7("ERROR");
             }
         }
    }
}
```