در قسمت ترمینال باید پورت ها را initialize کرد که این کار در توابع با نام init در آنها انجام میشود:

```
void PORTS_init(void) {
   RCC->AHB1ENR |=  0x03;         /* enable GPIOB/A clock */

   /* PB5 for LCD R/S */
   /* PB6 for LCD R/W */
   /* PB7 for LCD EN */
   GPIOB->MODER &= ~0x0000FC00;    /* clear pin mode (00..00 11 11 11 00 00 00 00 00)*/
   GPIOB->MODER |=  0x00005400;    /* set pin output mode */
   GPIOB->BSRR = 0x00C00000;       /* turn off EN and R/W */

   /* PA4-PA11 for LCD D0-D7, respectively. */
   GPIOA->MODER &= ~0x00FFFF00;    /* clear pin mode */
   GPIOA->MODER |=  0x00555500;    /* set pin output mode */
}

void keypad_init(void) {
   RCC->AHB1ENR |=  0x14;               /* enable GPIOC clock */
   GPIOC->MODER &= ~0x0000FFFF;    /* clear pin mode to input */
   GPIOC->PUPDR =   0x00000055;    /* enable pull up resistors for column pins */
}

void LCD_init(void) {
   delayMs(30);          /* initialization sequence */
   LCD_command(0x30);
   delayMs(10);
   LCD_command(0x30);
   delayMs(1);
```

```
   LCD_command(0x30);


   LCD_command(0x38);     /* set 8-bit data, 2-line, 5x7 font */
   LCD_command(0x06);     /* move cursor right after each char */
   LCD_command(0x01);     /* clear screen, move cursor to home */
   LCD_command(0x0F);     /* turn on display, cursor blinking */
}


void UART2_init(void){
        RCC->APB1ENR |= 0x20000;  // Enable UART2 CLOCK
        RCC->AHB1ENR |= 0x01; // Enable GPIOA CLOCK


        GPIOA->MODER |= 0x000000A0;  // bits 7-4 = 1010 = 0xA --> Alternate Function for Pin PA2 &
PA3
        GPIOA->OSPEEDR |= 0x000000F0;  // bits 7-4 = 1111 = 0xF --> High Speed for PIN PA2 and PA3
        GPIOA->AFR[0] |= 0x07700;  // bits 15-8=01110111=0x77  --> AF7 Alternate function for USART2
at Pin PA2 & PA3


        USART2->BRR = 0x0683;   // Baud rate = 9600bps, CLK = 15MHz


        USART2->CR1 = 1<<13;  // Enable USART
        USART2->CR1 &= ~(1<<12);  // M =0; 8 bit word length


        USART2->CR1 |= (1<<2); // RE=1.. Enable the Receiver
        USART2->CR1 |= (1<<3);  // TE=1.. Enable Transmitter
}
```

در بخش CPU نیز توابع زیر را داریم:

* تابع UART2_init شبیه همان تابع در ترمینال است.

```
void LED_init(void){
//    enable PB0 for green LED
   RCC->AHB1ENR |=  0x02;              /* enable GPIOB clock/*
   GPIOB->MODER &= ~0x00000003;   /* clear pin mode/*
   GPIOB->MODER |=  0x00000001;   /* set pin output mode/*
}
```

*برای چک کردن درست کار کردن UART، در تابع main در terminal قطعه کد زیر را قرار دادم:

```
while(1) {
        key = keypad_getkey();
        if (key != 10){
                UART2_write(key);
                delayMs(500);
                key = UART2_read();
                delayMs(500);
                LCD_data(key);
                delayMs(500);
        }
}
```

در تابع main در CPU نیز قطعه کد زیر را:

```
while(1) {
        data = UART2_read();
        UART2_write(data);
}
```

و مشاهده شد مشکلی وجود ندارد.

روند برنامه نیز به این شکل است که در هر دو طرف یک شمارنده داریم که تعداد اعداد وارد شده را میشمارد (هرموقع # وارد میشود یک عدد جدید وارد شده – تا سقف ۳). در طرف CPU هم یک آرایه داریم که در آن اعداد را نگه میدارد (به این گونه که با وارد شدن هر digit آن را ضربدر ۱۰ و به علاوه مقدار قبلی میکند و این کار را تا وارد شدن # انجام میدهد و سپس عدد را در آرایه ذخیره میکند). با وارد شدن عدد چهارم نیز محاسبات گفته شده در سمت CPU انجام شده و به terminal فرستاده میشود. برای این کار از تابع UART2_write_number در سمت cpu استفاده میشود که در آن ارقام عدد را تک تک از سمت چپ جدا میکند و با تابع UART2_write کاراکتر بدست آمده را ارسال میکند.