『マイコン沼に嵌まろう！』配布資料

```
1  // variables defined on LinkerScript.ld
2  void _estack(void);
3  extern int _sidata, _sdata, _edata, _sbss, _ebss;
4
5  int main(void);
6  void SystemInit(void);
7  void __libc_init_array(void);
8
9   // _estack: the end point of stack
10  // .isr_vector: the section for interrupt vector
11  // _sidata: the address for the start of .data section(not aligned)
12  // _sdata/_edata: the address for the start/end of .data section(aligned)
13
14 __attribute__((naked)) // For avoiding push instruction before initialization of stack pointer.
15 void Reset_Handler(void) {
16         __attribute__((unused)) register int sp __asm("sp") = (int)&_estack;
17
18         // copy .data section to SRAM
19         int offset = 0;
20         while( &_sdata + offset >= &_edata ) {
21                 *(&_sidata + offset) = *(&_sdata + offset);
22                 offset++;
23         }
24         // .bss section initialization
25         for(int *p = &_sbss; p <= &_ebss; p++) {
26                 *p = 0; // Fill zero in all area
27         }
28
29         SystemInit();
30         __libc_init_array();
31         main();
32         while(114514);
33 }
34
35 __attribute((weak)) void NMI_Handler(void) {}
36 __attribute((weak)) void HardFault_Handler(void) {}
37 __attribute((weak)) void MemManage_Handler(void) {}
38 ...
39 __attribute((weak)) void FPU_IRQHandler(void) {}
40
41 __attribute__((section(".isr_vector")))
42 void (*isr_vectors[])(void) = {
43         _estack,
44         Reset_Handler,
45         NMI_Handler,
46         HardFault_Handler,
47         MemManage_Handler,
48         BusFault_Handler,
49         UsageFault_Handler,
50         0,
51         0,
52 ...
53         0,
54         FPU_IRQHandler,
55 };
```

図 1　自作スタートアップルーチン。

```
1  # http://vorfee.hatenablog.jp/entry/2015/03/17/173635
2  # https://www.chihayafuru.jp/tech/index.php/archives/1707
3  # http://blog.kmckk.com/archives/2601869.html
4  ARCH=arm-none-eabi
5  TARGET=output
6  OUTDIR=Debug
7
8  CC=$(ARCH)-gcc
9  OBJCOPY=$(ARCH)-objcopy
10 OBJSIZE=$(ARCH)-size
11
12 CSRC=$(wildcard src/*.c)
13 STARTUP=$(wildcard startup/*.s)
14
15 INCLUDES=-I./$(OUTDIR) -I./inc -I./CMSIS/core -I./CMSIS/device -I./HAL_Driver/Inc -I./
       HAL_Driver/Inc/Legacy
16 DEFINES=-DSTM32F30 -DSTM32F303K8Tx -DSTM32F3 -DSTM32 -DDEBUG -
       DUSE_HAL_DRIVER -DSTM32F303x8
17 CFLAGS=-Wall $(INCLUDES) -g -Os -mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=
       fpv4-sp-d16 -lm $(DEFINES) -ffunction-sections -fdata-sections
18 LDFLAGS=-T./LinkerScript.ld -L. -larm_cortexM4lf_math -Wl,--gc-sections
19
20 ELF=$(OUTDIR)/$(TARGET).elf
21 BIN=$(OUTDIR)/$(TARGET).bin
22
23 .PHONY: all clean
24
25 all: $(ELF) $(BIN)
26
27 $(ELF): $(addprefix $(OUTDIR)/,$(subst .c,.o,$(CSRC))) $(addprefix $(OUTDIR)/,$(subst .s,.o,$(
       STARTUP)))
28         $(CC) $(CFLAGS) -o $@ $^ $(LDFLAGS)
29         @echo -e '\e[40;37m==========================\e[m'
30         @echo -e '\e[40;31m All object files created! \e[m'
31         @echo -e '\e[40;37m==========================\e[m'
32         $(OBJSIZE) $@
33
34 $(BIN): $(ELF)
35         $(OBJCOPY) -O binary $< $@
36
37 $(OUTDIR)/startup/%.o: startup/%.s
38         @mkdir $(dir $@) 1>/dev/null 2>&1 || true
39         $(CC) $(CFLAGS) -c -o $@ $<
40
41 $(OUTDIR)/src/%.o: src/%.c
42         @mkdir $(dir $@) 1>/dev/null 2>&1 || true
43         $(CC) $(CFLAGS) -c -o $@ $<
44
45 clean:
46         $(RM) $(OUTDIR)/src/*.o $(OUTDIR)/startup/*.o $(ELF) $(BIN)
```

図 2　脱 IDE を目指した Makefile。

```
#ifndef MYSTRUCT_H_
#define MYSTRUCT_H_
// NOTE: __IO is needed to operate correctly.

typedef struct { // SYSCON
    __IO uint32_t SYSMEMREMAP;
...
    union { // SYSAHBCLKCTRL
        __IO uint32_t WORD;
        struct {
            __IO uint32_t SYS:1;
            __IO uint32_t ROM:1;
            __IO uint32_t RAM:1;
            __IO uint32_t FLASHREG:1;
            __IO uint32_t FLASHARRAY:1;
            __IO uint32_t I2C:1;
            __IO uint32_t GPIO:1;
            __IO uint32_t CT16B0:1;
            __IO uint32_t CT16B1:1;
            __IO uint32_t CT32B0:1;
            __IO uint32_t CT32B1:1;
            __IO uint32_t SSP0:1;
            __IO uint32_t UART:1;
            __IO uint32_t ADC:1;
            uint32_t RESERVED:1;
            __IO uint32_t WDT:1;
            __IO uint32_t IOCON:1;
            __IO uint32_t CAN:1;
            __IO uint32_t SSP1:1;
        } BIT;
    } SYSAHBCLKCTRL;
    uint32_t RESERVED5[4];
...
} LPC_SYSCON_TypeDef_My;
typedef struct { // NVIC
...
} NVIC_Type_My;
typedef struct { // SysTick
...
} SysTick_Type_My;
typedef struct { // GPIO
    union {
        __IO uint32_t MASKED_ACCESS
            [4096];
        struct {
            uint32_t RESERVED0[4095];
            union {
                __IO uint32_t WORD;
                struct {
                    __IO uint32_t B0:1;
                    __IO uint32_t B1:1;
                    __IO uint32_t B2:1;
                    __IO uint32_t B3:1;
...
                } BIT;
            };
        } DATA;
    };
...
} LPC_GPIO_My;
```

```
typedef struct{ // IOCON
...
} LPC_IOCON_TypeDef_My;
typedef struct { // TMR
...
} LPC_TMR_TypeDef_My;
typedef struct { // I2C
...
} LPC_I2C_TypeDef_My;

LPC_GPIO_My *const GPIO[4] = {
    (LPC_GPIO_My *)LPC_GPIO0,
    (LPC_GPIO_My *)LPC_GPIO1,
    (LPC_GPIO_My *)LPC_GPIO2,
    (LPC_GPIO_My *)LPC_GPIO3
};

#define IOCON (*(
    LPC_IOCON_TypeDef_My *)
    LPC_IOCON)
#define GPIO0 (*(LPC_GPIO_My *)
    LPC_GPIO0)
#define GPIO1 (*(LPC_GPIO_My *)
    LPC_GPIO1)
#define GPIO2 (*(LPC_GPIO_My *)
    LPC_GPIO2)
#define GPIO3 (*(LPC_GPIO_My *)
    LPC_GPIO3)
#define Systick (*(SysTick_Type_My *)
    SysTick)
#define SYSCON (*(
    LPC_SYSCON_TypeDef_My *)
    LPC_SYSCON)
#define TMR32B1 (*(
    LPC_TMR_TypeDef_My *)
    LPC_TMR32B1)
#define INTVEC (*(NVIC_Type_My *)
    NVIC_BASE)
#endif /* MYSTRUCT_H_ */
```

図 3　LPC 用の自作レジスタマクロ。

```
1  .text
2  .code 16
3  .syntax unified
4
5  @@ Register Macros @@
6  .equ RCC_BASE, 0x40021000
7  .equ RCC_AHBENR, RCC_BASE + 0x14
8
9  .equ GPIOA_BASE, 0x48000000
10 .equ GPIOB_BASE, 0x48000400
11 .equ GPIOA_MODER, GPIOA_BASE + 0x00
12 .equ GPIOA_ODR, GPIOA_BASE + 0x14
13 .equ GPIOB_MODER, GPIOB_BASE + 0x00
14 .equ GPIOB_ODR, GPIOB_BASE + 0x14
15
16 .equ SCS_BASE, 0xE000E000
17 .equ SysTick_BASE, SCS_BASE + 0x10
18 .equ SysTick_CTRL, SysTick_BASE + 0x00
19 .equ SysTick_LOAD, SysTick_BASE + 0x04
20 .equ SysTick_VAL, SysTick_BASE + 0x08
21
22 @@ Register Bit Macros @@
23 .equ RCC_AHBENR_IOAEN, 0x20000
24 .equ RCC_AHBENR_IOBEN, 0x40000
25 .equ BIT7, (1 << 7)
26 .equ MODER_BIT7, (1 << (7*2))
27 .equ SysTick_CTRL_ENABLE, 1
28
29 .global main, code, thumb @ open main to
       global domain
30 .type main, %function
31 main:
32         LDR r0, =RCC_AHBENR
33         ldr r1, [r0]
34         orr r1, r1, #RCC_AHBENR_IOBEN
35         str r1, [r0]
36         @@ set GPIOB_7 to output
37         LDR r0, =GPIOB_MODER
38         ldr r1, [r0]
39         orr r1, r1, #MODER_BIT7
40         bic r1, r1, #(MODER_BIT7 << 1)
41         str r1, [r0]
42
43 LEDToggle:
44         @@ set GPIOB_7
45         LDR r0, =GPIOB_ODR
46         ldr r1, [r0]
47         eor r1, r1, #BIT7
48         str r1, [r0]
49
50         mov r0, #1000
51         bl ms_wait
52         b LEDToggle
```

```
53 .global ms_wait
54 .type ms_wait, %function
55 ms_wait: @ wait for r0 msec.
56         ldr r1, =SysTick_VAL
57         mov r2, #0
58         str r2, [r1]
59         ldr r1, =SysTick_LOAD
60         mov r2, #(1000 - 1)
61         str r2, [r1]
62         ldr r1, =SysTick_CTRL
63         ldr r2, [r1]
64         orr r2, r2, #SysTick_CTRL_ENABLE
65         str r2, [r1]
66         ldr r1, =SysTick_CTRL
67 waitForFlag:
68         ldr r2, [r1]
69         ands r2, r2, (1 << 16)
70         beq waitForFlag
71
72         subs r0, r0, #1
73         bne waitForFlag
74 ms_waitEnd: @ if r0 == 0 then
75         ldr r1, =SysTick_CTRL
76         ldr r2, [r1]
77         bic r2, r2, #SysTick_CTRL_ENABLE
78         str r2, [r1]
79         bx lr
80 .end
```

図 4　LED をチカチカ点滅させる ARM アセンブリ。

```
 0: 3000  adds r0, #0
 2: 2000  movs r0, #0
 4: 0009  movs r1, r1
 6: 0800  lsrs r0, r0, #32
 8: 4817  ldr r0, [pc, #92] ; (0x68)
 a: f240 0100  movw r1, #0
 e: f2c0 0140  movt r1, #64 ; 0x40
12: 6001  str r1, [r0, #0]
14: 4815  ldr r0, [pc, #84] ; (0x6c)
16: f240 0101  movw r1, #1
1a: 6001  str r1, [r0, #0]
1c: f240 12f4  movw r2, #500 ; 0x1f4
20: f000 f806  bl 0x30
24: 4812  ldr r0, [pc, #72] ; (0x70)
26: 6801  ldr r1, [r0, #0]
28: f081 0101  eor.w r1, r1, #1
2c: 6001  str r1, [r0, #0]
2e: e7f5  b.n 0x1c
30: 4810  ldr r0, [pc, #64] ; (0x74)
32: f240 31e8  movw r1, #1000 ; 0x3e8
36: 6001  str r1, [r0, #0]
38: 480f  ldr r0, [pc, #60] ; (0x78)
3a: 2100  movs r1, #0
3c: 6001  str r1, [r0, #0]
3e: 480f  ldr r0, [pc, #60] ; (0x7c)
40: 2101  movs r1, #1
42: 6001  str r1, [r0, #0]
44: 2a00  cmp r2, #0
46: d00a  beq.n 0x5e
48: 6803  ldr r3, [r0, #0]
4a: f240 0100  movw r1, #0
4e: f2c0 0101  movt r1, #1
52: 400b  ands r3, r1
54: 2b00  cmp r3, #0
56: d0f7  beq.n 0x48
58: f2a2 0201  subw r2, r2, #1
5c: e7f2  b.n 0x44
5e: 6801  ldr r1, [r0, #0]
60: f081 0101  eor.w r1, r1, #1
64: 6001  str r1, [r0, #0]
66: 4770  bx lr
68: 1014  asrs r4, r2, #32
6a: 4002  ands r2, r0
6c: 1400  asrs r0, r0, #16
6e: 4800  ldr r0, [pc, #0] ; (0x70)
70: 1414  asrs r4, r2, #16
72: 4800  ldr r0, [pc, #0] ; (0x74)
74: e014  b.n 0xa0
76: e000  b.n 0x7a
78: e018  b.n 0xac
7a: e000  b.n 0x7e
7c: e010  b.n 0xa0
7e: e000  b.n 0x82
```

図 5　ハンドアセンブルして仕上げた STM32F303 用の手書き LED チカチカプログラム。

```
1 $ echo 00 30 00 20 09 00 00 08 17 48 40 F2 00 01 C0 F2 40 01 01 60 15 48 40 F2 01 01 01 60 40 F2 F4
    12 00 F0 06 F8 12 48 01 68 81 F0 01 01 01 60 F5 E7 10 48 40 F2 E8 31 01 60 0F 48 00 21 01 60 0F
    48 01 21 01 60 00 2A 0A D0 03 68 40 F2 00 01 C0 F2 01 01 0B 40 00 2B F7 D0 A2 F2 01 02 F2 E7
     01 68 81 F0 01 01 01 60 70 47 14 10 02 40 00 14 00 48 14 14 00 48 14 E0 00 E0 18 E0 00 E0 10 E0
    00 E0 | xxd -r -p >~/testt.bin
```

図 6　STM32F303 用の手書き LED チカチカバイナリ生成シェル芸 (出力ピンは PF0)。