

# Курс по STM32

## Лекция #5:

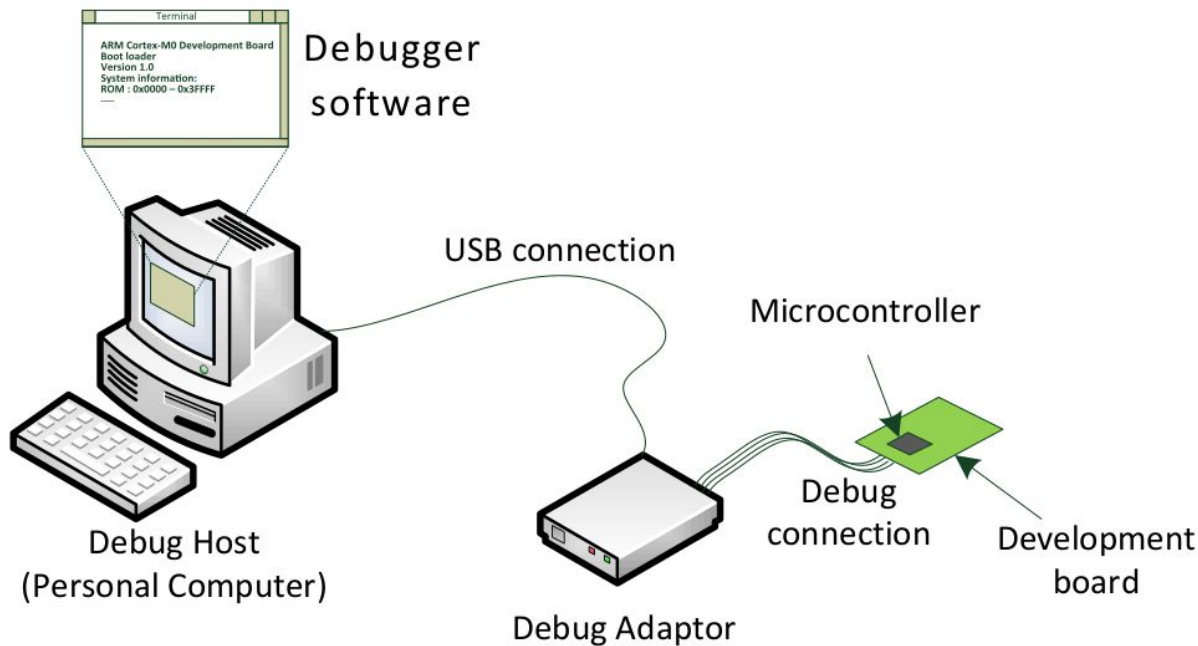
- Использование отладчика: красивый подводный камень.
- Схема реализации `timing_perfect_delay()`.
- Выдача ДЗ №3.
- Общая схема обработки исключений в Cortex-M0.
- Выдача требований к финальному проекту.

04.03.2023 / 06.03.2023

# Использование отладчика: красивый подводный камень



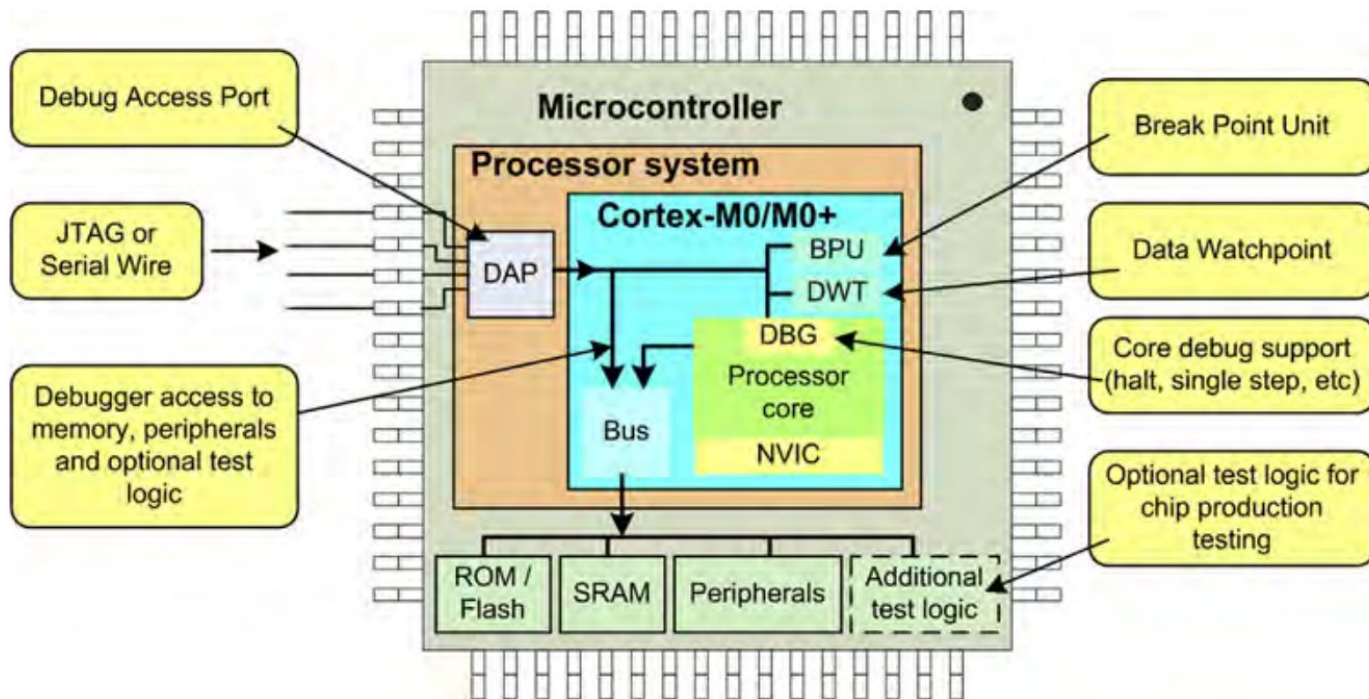
# Использование отладчика: общая схема



**Figure 13.1**

A classic microcontroller development environment.

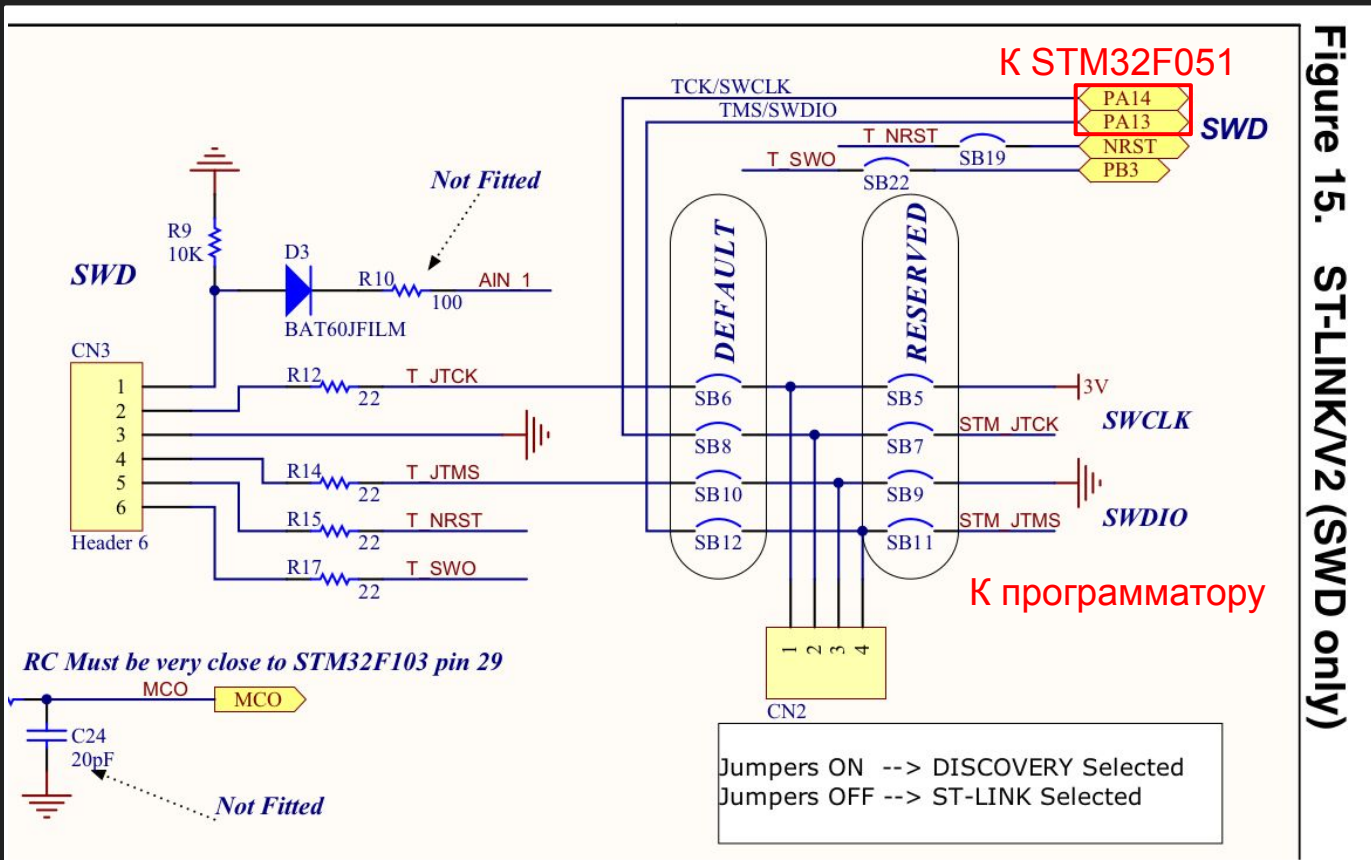
# Использование отладчика: подключение по SWD



### Figure 13.3

Debug interface connection inside the processor.

# Использование отладчика: подключение по SWD



# Использование отладчика: пины PA13 и PA14

Table 13. Pin definitions (continued)

Pin number						Pin name (function upon reset)	Pin type	I/O structure	Notes	Pin functions	
LQFP64	UFBGA64	LQFP48/UFQFPN48	WLCSP36	LQFP32	UFQFPN32					Alternate functions	Additional functions
45	B8	33	A1	22	22	PA12	I/O	FT	-	USART1_RTS, TIM1_ETR, COMP2_OUT, TSC_G4_IO4, EVENTOUT	-
46	A8	34	B1	23	23	PA13 (SWDIO)	I/O	FT	(6)	IR_OUT, SWDIO	-
47	D6	35	-	-	-	PF6	I/O	FT	-	I2C2_SCL	-
48	E6	36	-	-	-	PF7	I/O	FT	-	I2C2_SDA	-
49	A7	37	B2	24	24	PA14 (SWCLK)	I/O	FT	(6)	USART2_TX, SWCLK	-

# Использование отладчика: пины PA13 и PA14

**Table 24. GPIO register map and reset values**

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
0x00	GPIOA_MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]	
	Reset value	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	GPIOx_MODER (where x = B..F)	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

# Использование отладчика: выстрел в колено

Корректный код

```
// Configure mode register:  
*GPIOA_MODER |= 0x1555554U;
```

Некорректный код

```
// Configure mode register:  
*GPIOA_MODER = 0x1555554U;
```

```
~/path/to/stm32f051_rewind/labs/02_gpio>  
> make hardware // Запуск утилиты для связи с отладчиком  
st-util -p 1234  
...  
[!] send_recv send request failed: LIBUSB_ERROR_BUSY  
[!] send_recv STLINK_DEBUG_RUNCORE  
[!] send_recv send request failed: LIBUSB_ERROR_BUSY  
[!] send_recv STLINK_JTAG_WRITEDEBUG_32BIT
```

Также возможны проблемы с прошивкой платы!



**ДЗ №3: timing-perfect delay**



# Имеющийся delay: константное кол-во операций


Проблемы решения:

- delay неточный, и нагло притворяется очень точным :(
- Нельзя задавать время ожидания во время исполнения.

```
#define CPU_FREQUENCY 48000000U // CPU frequency: 48 MHz
#define ONE_MILLISECOND CPU_FREQUENCY/1000U

void totally_accurate_quantum_femtosecond_precise_super_delay_3000_1000ms()
{
    for (uint32_t i = 0; i < 1000U * ONE_MILLISECOND; ++i)
    {
        // Insert NOP for power consumption:
        __asm__ volatile("nop");
    }
}
```

# Имеющийся delay: константное кол-во операций



```
dbg.totally_accurate_quantum_femtosecond_precise_super_delay_3000_1000ms ();
; var uint32_t i @ fp+0x4
; var int16_t var_0h @ sp+0x0
0x000000f8      push    {r7, lr}    ; void totally_accurate_quantum_femtosecor
0x000000fa      sub     sp, 8
0x000000fc      add     r7, var_0h
0x000000fe      movs    r3, 0
0x00000100      str     r3, [r7, 4]
0x00000102      b       0x10c
0x00000104      mov     r8, r8
0x00000106      ldr     r3, [r7, 4]
0x00000108      adds    r3, 1
0x0000010a      str     r3, [r7, 4]
0x0000010c      ldr     r3, [r7, 4]
0x0000010e      ldr     r2, [0x00000120] ; 288
0x00000110      cmp     r3, r2
0x00000112      bls     0x104
0x00000114      mov     r8, r8
0x00000116      mov     r8, r8
0x00000118      mov     sp, r7
0x0000011a      add     sp, 8
0x0000011c      pop     {r7, pc}
0x0000011e      mov     r8, r8
0x00000120      strh    r6, [r3, 0x34]
0x00000122      movs    r3, r1
```

# Имеющийся delay: константное кол-во операций

```
[0x000000f8]
;-- totally_accurate_quantum_femtosecond_precise_super_delay_3000_1000ms:
dbg.totally_accurate_quantum_femtosecond_precise_super_delay_3000_1000ms ();
; var uint32_t i @ fp+0x4
; var int16_t var_0h @ sp+0x0
push    {r7, lr}                ; void totally_accurate_quantum_femtosecond_precise_super_dela...
sub     sp, 8
add     r7, var_0h
movs    r3, 0
str     r3, [r7, 4]
b       0x10c
```

Препамбула

Условие  
цикла

```
[0x0000010c]
ldr     r3, [r7, 4]
ldr     r2, [0x00000120]        ; 288
cmp     r3, r2
bls     0x104
```

Выход из  
функции

```
[0x00000114]
mov     r8, r8
mov     r8, r8
mov     sp, r7
add     sp, 8
pop     {r7, pc}
```

Тело цикла

```
[0x00000104]
mov     r8, r8
ldr     r3, [r7, 4]
adds    r3, 1
str     r3, [r7, 4]
```

# Имеющийся delay: преамбула

```
; var uint32_t i      @ fp+0x4
```

```
0x00f8    push    {r7, lr}      ; Сохранение регистров
```

```
0x00fa    sub     sp, #8        ; Аллокация места на стеке
```

```
0x00fc    add     r7, sp, #0     ; r7(fp) - адрес стек фрейма
```

```
0x00fe    movs    r3, #0        ; Зануление счётчика цикла
```

```
0x0100    str     r3, [r7, #4] ;
```

```
0x0102    b       0x010c        ; Переход к циклу
```

```
; Вид стека:
```

```
0x20001FFC [return_addr] < здесь лежит бывший lr
```

```
0x20001FF8 [ prev_r7   ] < здесь лежит бывший r7(fp)
```

```
0x20001FF4 [    i      ] < [r7, #4]
```

```
0x20001FF0 [ padding  ] < сюда указывает текущий r7(fp) и sp
```

# Имеющийся delay: цикл

```
0x0104    nop                ; Тот самый наш nop
0x0106    ldr      r3, [r7, 4] ; r3 = i;
0x0108    adds    r3, #1      ; r3 += 1;
0x010a    str      r3, [r7, #4] ; i = r3
0x010c    ldr      r3, [r7, #4] ; r3 = i
0x010e    ldr      r2, [0x0120] ; r2 = *0x0120
0x0110    cmp      r3, r2      ; if (r3 < r2)
0x0112    bls      0x0104      ; goto 0x0104
...
0x0120    <instruction>        ; Кол-во итераций цикла
```

## Имеющийся delay: выход из цикла

```
0x0114    nop                ; (mov r8, r8)
0x0116    nop                ; (mov r8, r8)
0x0118    mov     sp, r7     ;
0x011a    add     sp, #8     ;
0x011c    pop     {r7, pc}
```

; Вид стека:

```
0x20001FFC [return_addr] >> ляжет в pc
0x20001FF8 [ prev_r7   ] >> ляжет в r7(fp)
0x20001FF4 [    i      ]
0x20001FF0 [ padding  ] < sp до “add sp, #8”
```

# Требования к ДЗ №3

- [ ] Реализовать функцию `timing_perfect_delay(uint32_t millis)`
  - [ ] Код функции написан на языке ассемблера.
  - [ ] Доказать, что время исполнения цикла точное.
- [+] Могут быть полезны:
  - [!] Техническая документация на процессор ([docs/cortex\\_m0\\_trm.pdf](#)).
  - [!] [Соглашение о вызове функций в ARM](#).
  - [!] [Задание адреса относительно регистра PC](#).



# Требования к финальному проекту



# Требования к финальному проекту

[1] Необходимо составить требования к устройству и согласовать их с ментором.

Устройство должно:

[А] Либо решать проблему.

[В] Либо быть объектом искусства и вызывать чувства.

[С] Либо быть сугубо образовательным (балл ниже).

Требования должны:

[ ] Давать общее описание устройства, принципов и сценариев его работы.

[ ] Давать примерный список используемых компонентов.

# Требования к финальному проекту

[2] Необходимо создать устройство.

[ ] Допускается работа в командах  
при указании зон ответственности участников.

[ ] По вопросам оборудования можно писать  
ответственному за РТ-комнату!

[3] Необходимо сдать устройство.

В работе оцениваются:

[ ] Адекватность требований.

[ ] Соответствие устройства требованиям и сценариям.

[ ] Техническая сложность устройства.

[ ] Качество программного кода.

# Лучшие проекты: глинтвейноварилка



# Лучшие проекты: Drumkit Glove



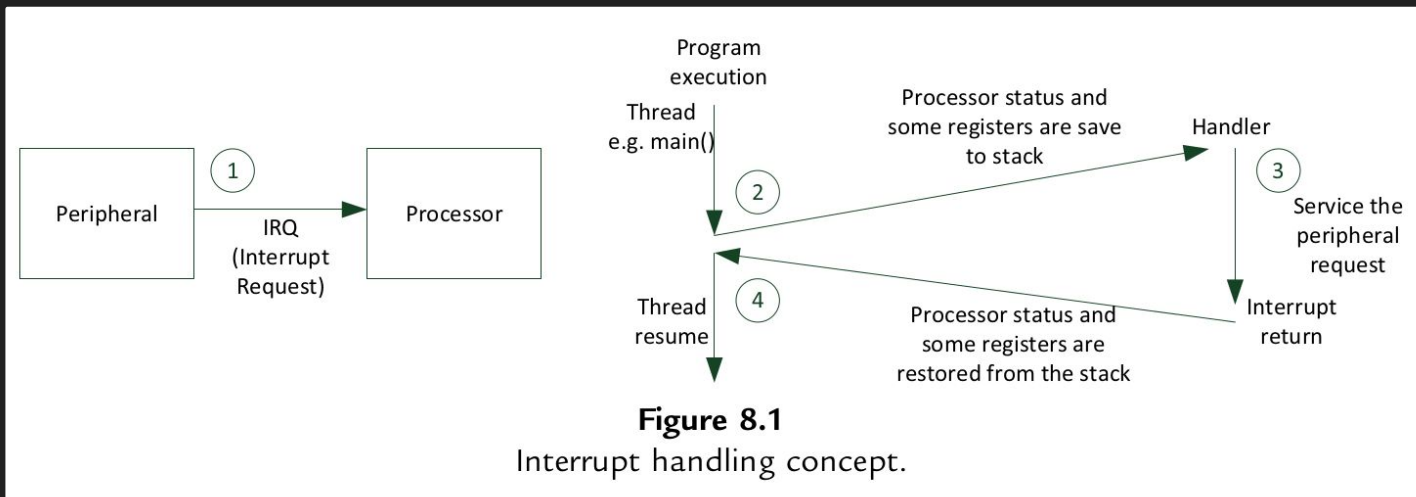
# Исключения в Cortex-M0

(пока только общая схема)



# Что такое исключение?

- 1) Периферия генерирует запрос на прерывание – оповещает процессор.
- 2) Процессор сохраняет текущее состояние.
- 3) Процессор определяет адрес обработчика прерывания и исполняет его.
- 4) Процессор возвращается в исходное состояние.



# Типы потока управления: постоянный опрос

Постоянный опрос (polling):

А) А нажата ли кнопка?

=> Изменить состояние кнопки!

Б) Пора ли выводить данные на семисегментник?

=> Вывести данные на пины!

В) А пора ли мигнуть диодом?

=> Мигнуть диодом!

Лишние действия на опрос.

Тяжело измерять время.

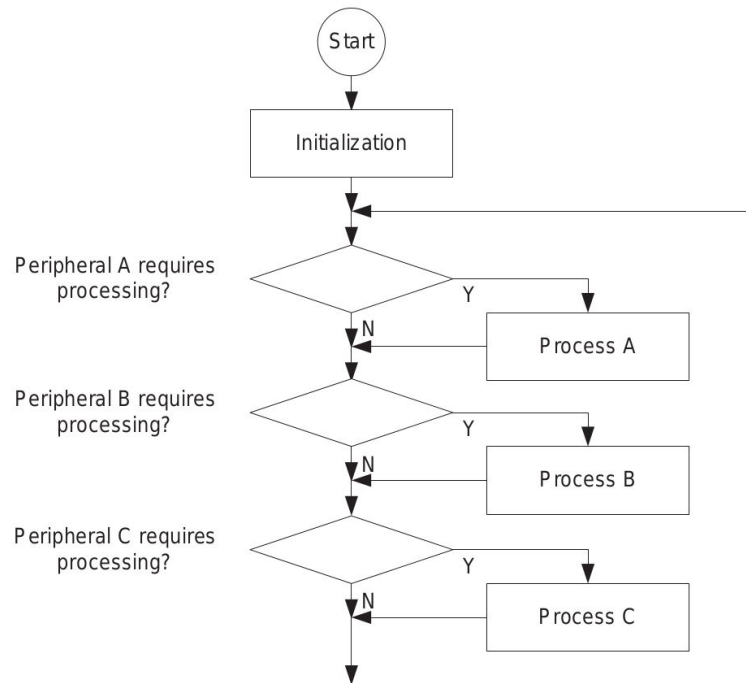


Figure 3.3

Polling method for simple application processing.



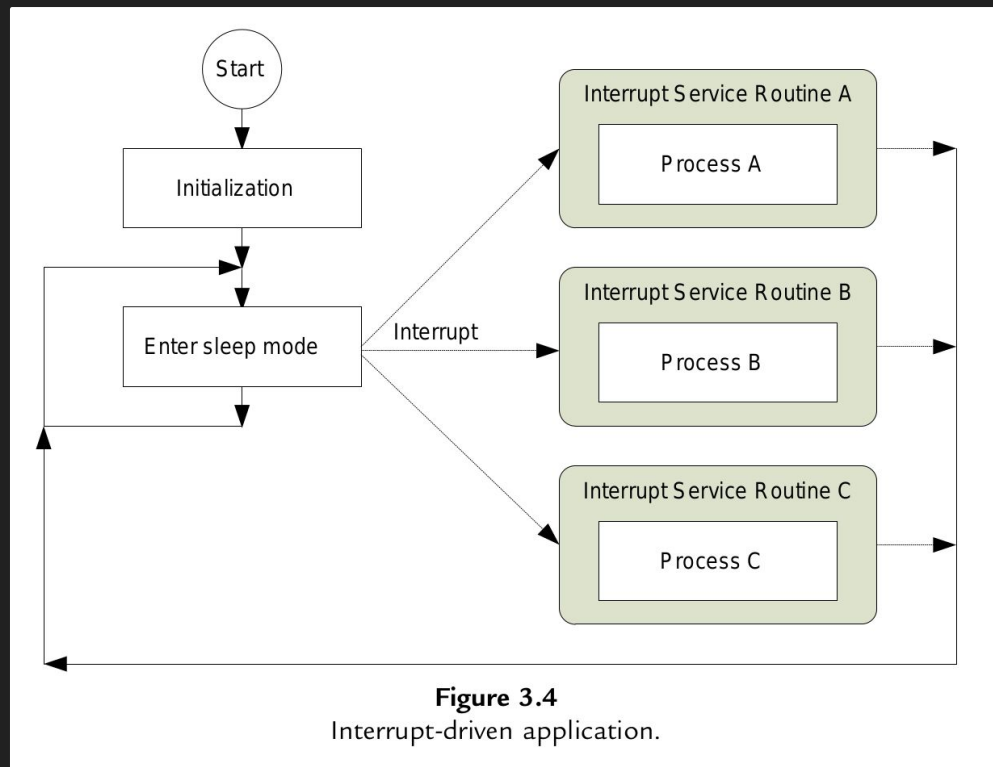
# Типы потока управления: по прерываниям

Исполнение по прерываниям  
(interrupt driven):

- Ниже энергопотребление
- Приоритеты прерываний
- Внешние/внутренние прерывания

Режим энергосбережения:

- Инструкции WFI/WFE.
- Более глубокий сон.



# Типы потока управления: смешанный тип

Смешанный поток управления:

Прерывание:

```
need_X = true;
```

Цикл обработки:

```
if (need_X) <сделать X>;
```

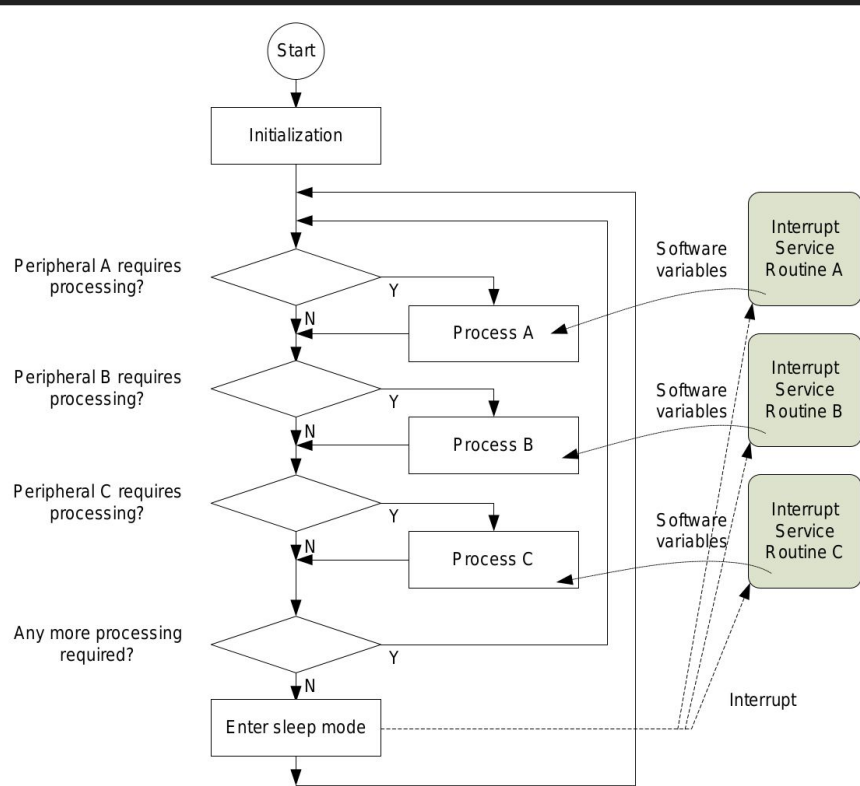
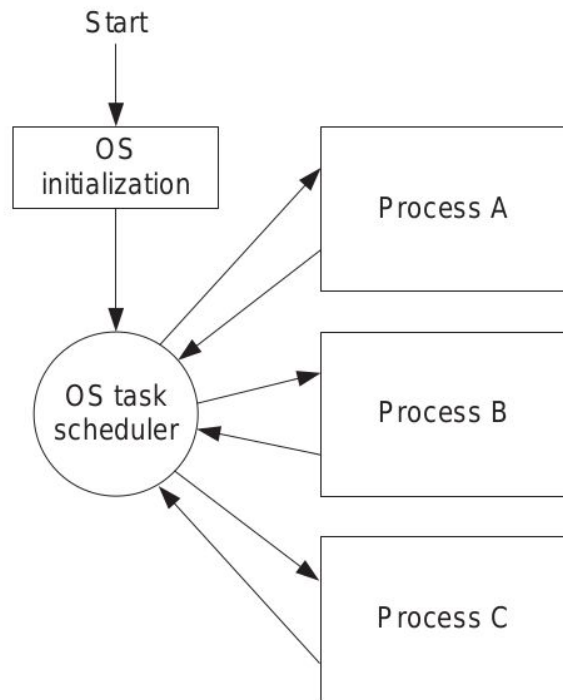


Figure 3.5

Combination of polling and interrupt-driven application.

# Типы потока управления: операционная система



**Figure 3.7**

Using an real-time operating system to handle multiple concurrent application processes.

**Спасибо за внимание!**