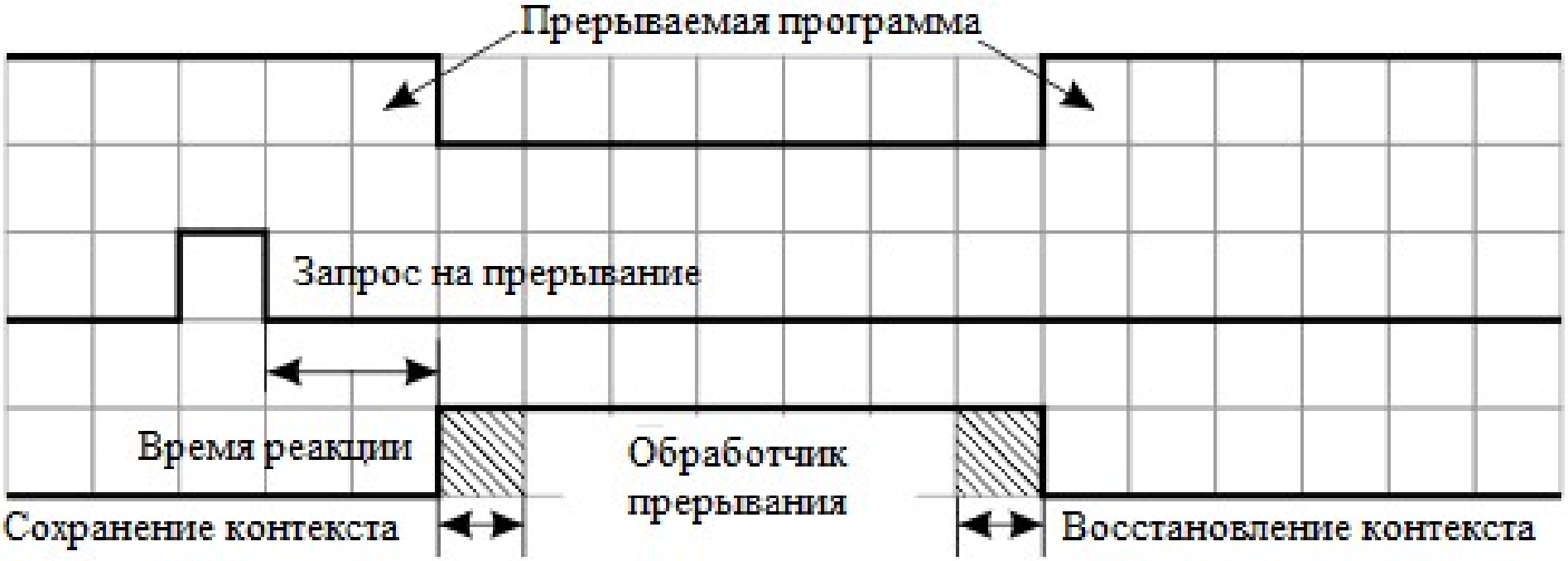
Микропроцессорные устройства обработки сигналов

Лекция L10
«Обработка прерываний»

http://vykhovanets.ru/course67/

Прерывание

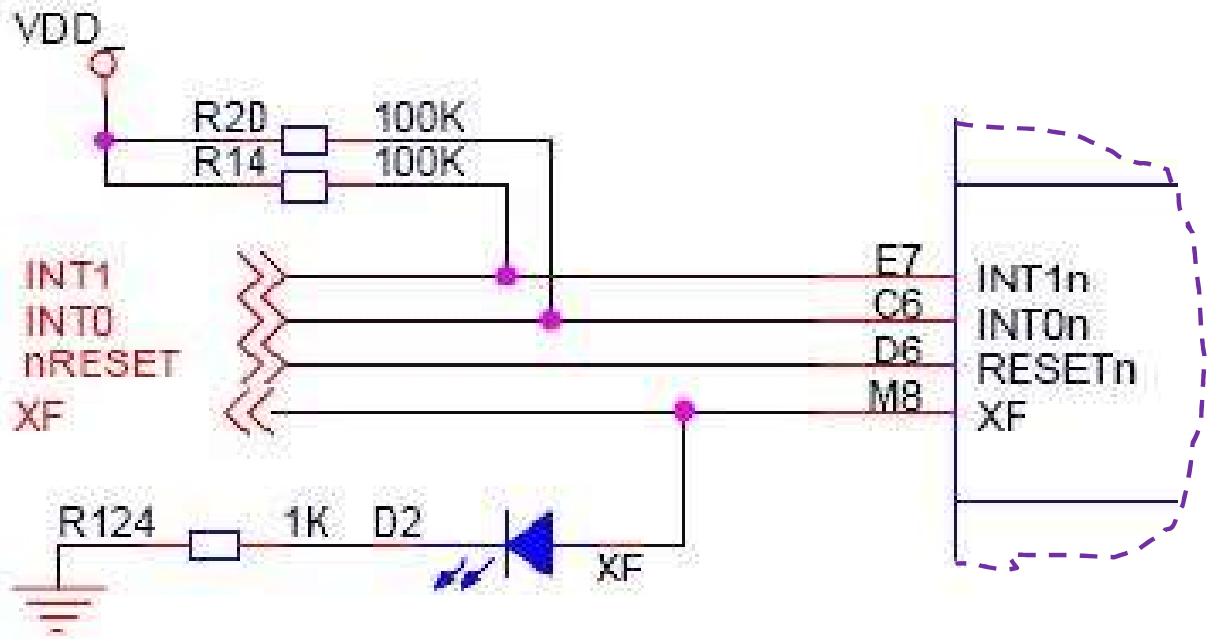
- **Прерывание (Interrupt)** – переключение микропроцессора на выполнение программ обработки асинхронно возникающих внутренних или внешних событий.

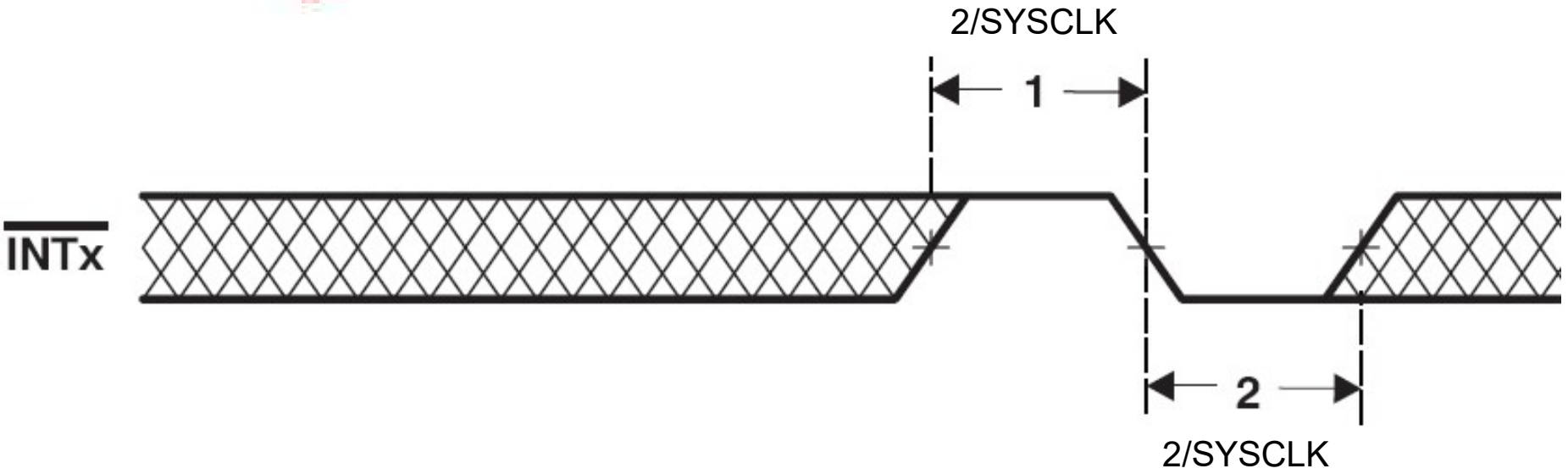


Источники прерываний

- **Источники прерываний**:
  – немаскируемые:
  а) внутреннее – от схем контроля NMI;
  б) сброс по входу RESET;
  – маскируемые:
  а) внешние – по входу INT0, INT1;
  б) аппаратурные – от модулей;
  в) программные – по командам INTR #k4, TRAP #k4.

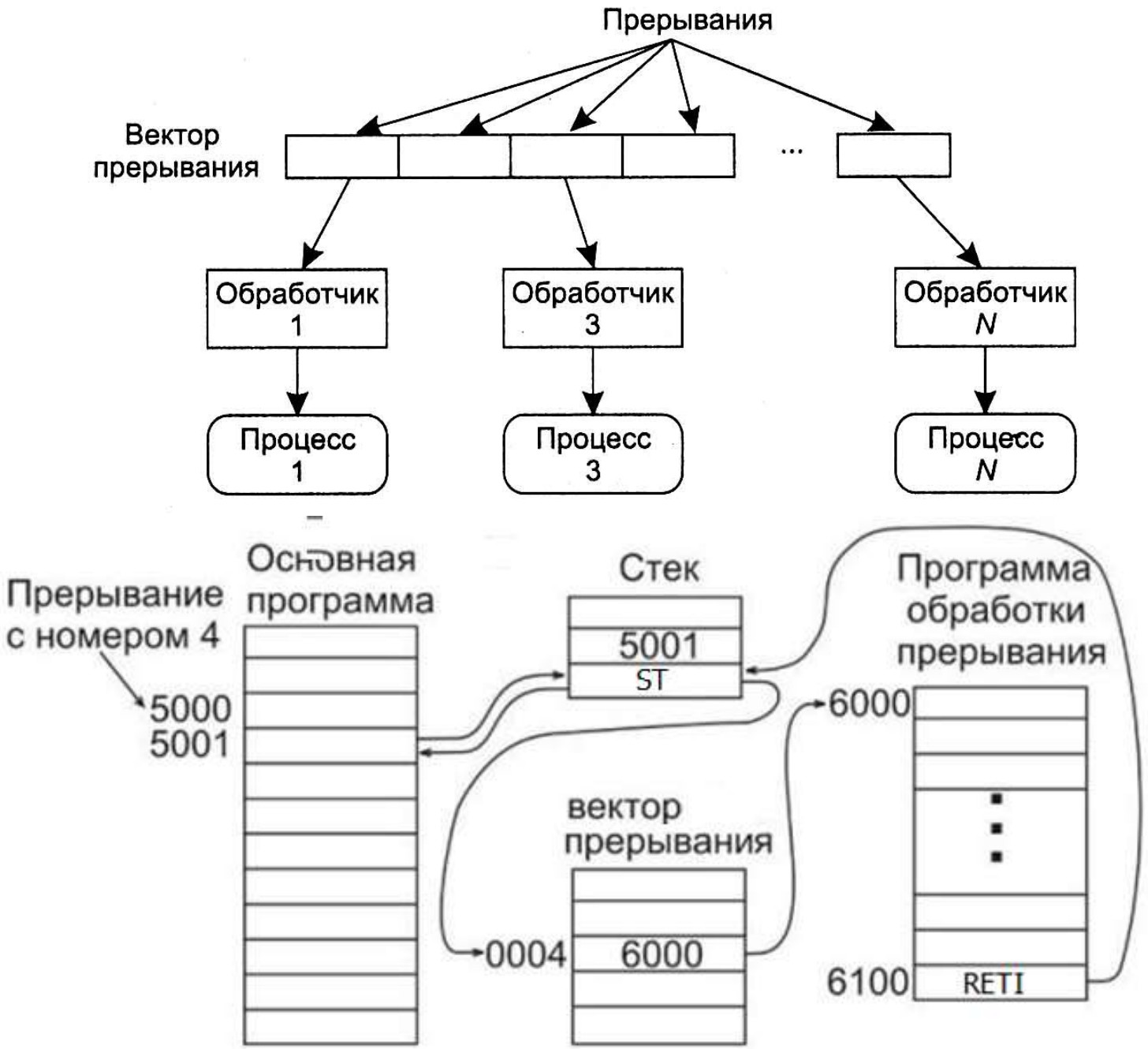
Внешние прерывания



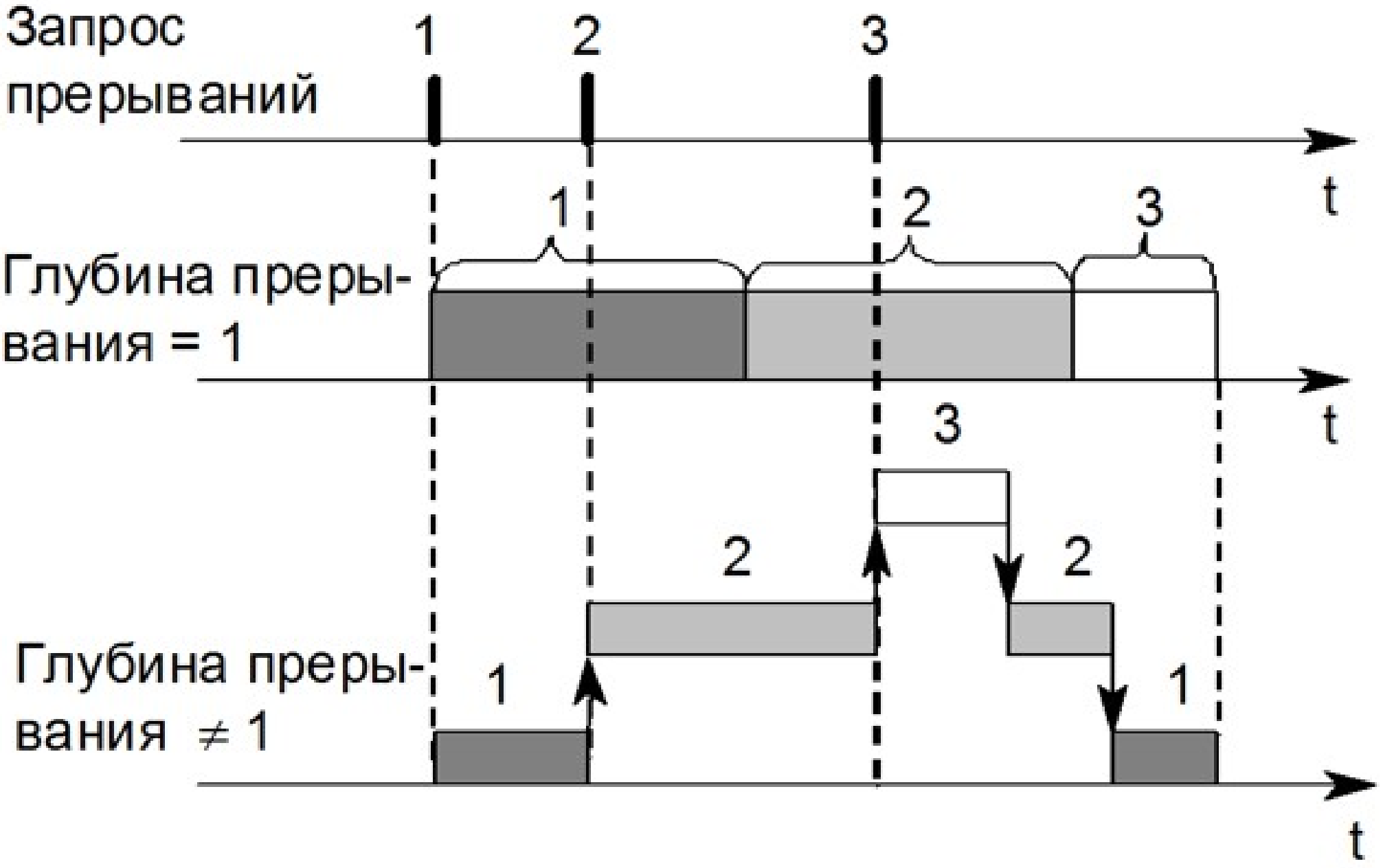


Последовательность прерывания

Обработка прерываний



Вложенные прерывания

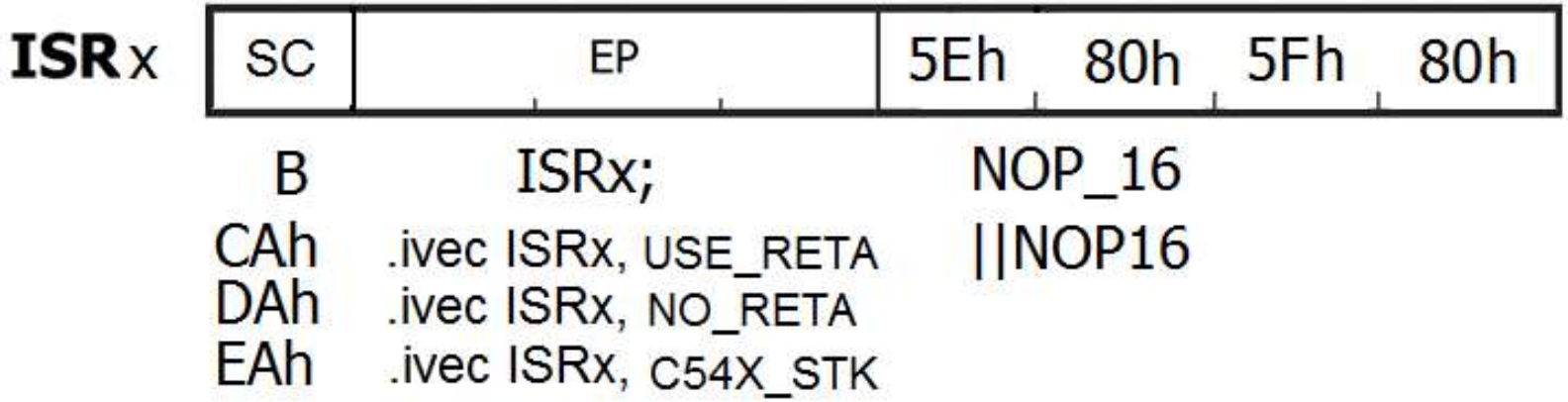


Вектор прерываний

Вектор	**IVPD** | 00h	Байт
00	ISR0	+00h
01	ISR1	+08h
…		…
15	ISR15	+78h
16		+80h
…		…
23		+B8h
24	ISR24	+C0h
…		…
31	ISR31	+F8h

Вектор	**IVPH** | 00h	Байт
00		+00h
01		+08h
…		…
15		+78h
16	ISR16	+80h
…		…
23	ISR23	+B8h
24		+C0h
…		…
31		+F8h

Процедура прерываний



ISR – **I**nterrupt **S**ervice **R**outine (процедура обработки прерывания).

SC – **S**tack **C**onfiguration (байт конфигурации стека, игнорируется для всех векторов, кроме RESET): CAh – быстрый вызов-возврат и два независимых 16-разрядных стека; DAh – медленный вызов-возврат и два независимых 16-разрядных стека; EAh – медленный вызов-возврат и единый 32-разрядный стек.

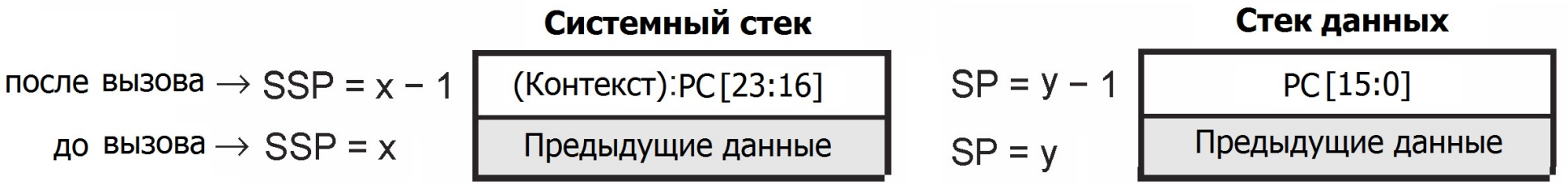
EP – **E**ntry **P**oint (точка входа, или 24-разрядный адрес процедуры).

ST1_55

15	14	13	12	11	10	9	8
BRAF	CPL	XF	HM	INTM	**M40†**	SATD	SXMD
R/W−0	R/W−0	R/W−1	R/W−0	R/W−1	R/W−0	R/W−0	R/W−1

Конфигурация стеков

Медленный вызов и возврат NO_RETA (ISR DAXX:XXXXh)



а) вызов процедуры и возврат из процедуры

б) вызов прерывания и возврат из прерывания

Быстрый вызов и возврат USE_RETA (ISR CAXX:XXXXh)

CALL: [PC, Контекст повторения]→[RETA, CFCT] :: [RETA, CFCT]→[*-SP, *-SSP]

RET: [RETA, CFCT]→[PC, Контекст повторения] :: [*SP-, *SSP-]→[RETA, CFCT]

Регистры прерываний

IER0 (IFR0, DBIER0)

15	14	13	12	11	10	9	8
INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8
R/W–0	R/W–0	R/W–0	R/W–0	R/W–0	R/W–0	R/W–0	R/W–0

7	6	5	4	3	2	1	0
INT7	INT6	INT5	INT4	INT3	INT2	Резерв	
R/W–0	R/W–0	R/W–0	R/W–0	R/W–0	R/W–0	R–0	

IER1 (IFR1, DBIER1)

15	11	10	9	8
Резерв		RTOSINT	DLOGINT	BERRINT
R–0		R/W–0	R/W–0	R/W–0

7	6	5	4	3	2	1	0
INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
R/W–0	R/W–0	R/W–0	R/W–0	R/W–0	R/W–0	R/W–0	R/W–0

INT2-INT23 – **I**nterrupt **2-23** (флаги прерываний 2-23).
RTOSINTF – **R**eal **T**ime **O**peration **S**ystem **Int**errupt **F**lag (флаг прерывания операционной системы реального времени).
DLOGINTF – **D**ata log **Int**errupt **F**lag (флаг прерывания протоколирования данных).
BERRINTF – **B**us **Err**or **Int**errupt **F**lag (флаги прерывания по ошибке обмена по шинам и магистралям).
DBIERx – **D**ebug **I**nterrupt **E**nable **R**egister (регистры аппаратурной отладки, флаги указывают, какие прерывания вызывают переход в отладочный режим)

Источники прерываний IVPD

Вектор	Адрес	Приор.	Обозначение и источник прерывания
ISR00	IVPD:00h	1	RESET, по сбросу и инициализации
ISR01	IVPD:08h	3	NMI, от схем контроля (внутреннее немаскируемое)
ISR02	IVPD:10h	5	INT0, внешнее прерывание по входу INT0
ISR03	IVPD:18h	7	INT1, внешнее прерывание по входу INT1
ISR04	IVPD:20h	8	TINT, агрегированное прерывание от таймеров
ISR05	IVPD:28h	9	PROG0, от контроллера I2S0 и MMC/SD0 по передаче
ISR06	IVPD:30h	11	UART, от приемо-передатчика UART
ISR07	IVPD:38h	12	PROG1, от контроллера I2S0 и MMC/SD0 по приему
ISR08	IVPD:40h	13	DMA, от канала прямого доступа к памяти
ISR09	IVPD:48h	15	PROG2, от контроллера I2S1 и MMC/SD1 по передаче
ISR10	IVPD:50h	16	FFT, от аппаратного ускорителя БПФ
ISR11	IVPD:58h	17	PROG3, от контроллера I2S1 и MMC/SD1 по приему
ISR12	IVPD:60h	19	LCD, от контроллера дисплея
ISR13	IVPD:68h	20	SAR, агрегированное от АЦП
ISR14	IVPD:70h	23	XTM2, от контроллера I2S2 по передаче
ISR15	IVPD:78h	24	RCV2, от контроллера I2S2 по приему

Источники прерываний IVPH

Вектор	Адрес	Приор.	Обозначение и источник прерывания
ISR16	IVPH:80h	6	XMT3, от контроллера I2S3 по передаче
ISR17	IVPH:88h	10	RCV3, от контроллера I2S3 по приему
ISR18	IVPH:90h	14	RTC, от часов реального времени
ISR19	IVPH:98h	18	SPI, от контроллера SPI
ISR20	IVPH:A0h	21	USB, от контроллера USB
ISR21	IVPH:A8h	22	GPIO, от портов ввода-вывода общего назначения
ISR22	IVPH:B0h	23	EMIF, от интерфейса внешней памяти при ошибках
ISR23	IVPH:B8h	26	I2C, от контроллера I2C
ISR24	IVPD:C0h	4	BERR, при ошибке обмена по шинам и магистралям
ISR25	IVPD:C8h	27	DLOG, протоколирование данных (программное)
ISR26	IVPD:D0h	28	RTOS, вызов операционной системы (программное)
ISR27	IVPD:D8h	–	RTDRCV, от эмулятора по приему
ISR28	IVPD:E0h	–	RTDXMT, от эмулятора по передаче
ISR29	IVPD:E8h	2	EMUINT, от монитора эмулятора
ISR30	IVPD:F0h	–	SINT30, пользовательское (программное)
ISR31	IVPD:F8h	–	SINT31, пользовательское (программное)

Секция vectors

```
            .sect   "vectors"
            .ref    _c_int00, _nmi_isr, _sarISR, _int0_isr, _int1_isr, _tim_isr
            .ref    _i2s0_mmc0_tx_isr, _uart_isr, _i2s0_mmc0_rx_isr, _dma_isr
            .ref    _i2s1_mmc1_tx_isr, _coprocfft_isr, …, _sint31_isr
            .def    _VECSTART, _no_isr
_VECSTART:                                  ; USE_RETA, NO_RETA, C54X_STK
RST:        .ivec   _c_int00, USE_RETA      ; 00 Reset / Software Interrupt #0
NMI:        .ivec   _nmi_isr                ; 01 Nonmaskable Interrupt
INT0:       .ivec   _int0_isr               ; 02 External User Interrupt #0
INT1:       .ivec   _int1_isr               ; 03 External User Interrupt #1
TINT:       .ivec   _tim_isr                ; 04 Timer #0 / Software Interrupt #4
PROG0:      .ivec   _i2s0_mmc0_tx_isr       ; 05 Programmable 0 Interrupt
UART:       .ivec   _uart_isr               ; 06 IIS #1 Receive Interrupt
PROG1:      .ivec   _i2s0_mmc0_rx_isr       ; 07 Programmable 1 Interrupt
DMA:        .ivec   _dma_isr                ; 08 DMA Interrupt
PROG2:      .ivec   _i2s1_mmc1_tx_isr       ; 09 Programmable 2 Interrupt
FFT:        .ivec   _coprocfft_isr          ; 10 Coprocessor FFT Module Interrupt
PROG3:      .ivec   _i2s1_mmc1_rx_isr       ; 11 Programmable 3 Interrupt
LCD:        .ivec   _lcd_isr                ; 12 LCD Interrupt
SAR:        .ivec   _saradc_isr             ; 13 SAR ADC Interrupt
XMT2:       .ivec   _i2s2_tx_isr            ; 14 I2S2 Tx Interrupt
RCV2:       .ivec   _i2s2_rx_isr            ; 15 I2S2 Rx Interrupt
```

Секция vectors

```
XMT3:    .ivec   _i2s3_tx_isr        ; 16 I2S3 Tx Interrupt
RCV3:    .ivec   _i2s3_rx_isr        ; 17 I2S3 Rx Interrupt
RTC:     .ivec   _rtc_isr            ; 18 RTC interrupt
SPI:     .ivec   _spi_isr            ; 19 SPI Receive Interrupt
USB:     .ivec   _usb_isr            ; 20 USB Transmit Interrupt
GPIO:    .ivec   _gpio_isr           ; 21 GPIO Interrupt
EMIF:    .ivec   _emif_error_isr     ; 22 EMIF Error Interrupt
I2C:     .ivec   _i2c_isr            ; 23 IIC interrupt
BERR:    .ivec   _berr_isr           ; 24 Bus Error Interrupt
DLOG:    .ivec   _dlog_isr           ; 25 Emulation Interrupt - DLOG
RTOS:    .ivec   _rtos_isr           ; 26 Emulation Interrupt - RTOS
RTDRCV:  .ivec   _rtdxrcv_isr        ; 27 Emulation Interrupt - RTDX receive
RTDXMT:  .ivec   _rtdxxmt_isr        ; 28 Emulation Interrupt - RTDX transmit
EMUINT:  .ivec   _emuint_isr         ; 29 Emulation monitor mode interrupt
SINT30:  .ivec   _no_isr             ; 30 Software Interrupt #30
SINT31:  .ivec   _sint31_isr         ; 31 Software Interrupt #31

         .text
_no_isr: nop
label:   jmp     label
```

Описание регистров

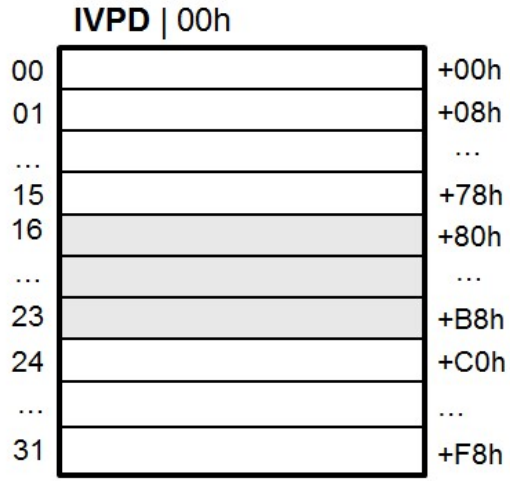
```
#define Uint16 unsigned int
typedef struct {
    volatile Uint16   IER0;        /*0000h – разрешения прерываний 0 */
    volatile Uint16   IFR0;        /*0001h – флаги прерываний 0 */
    volatile Uint16   ST0_55;      /*0002h – состояние процессора 0 */
    volatile Uint16   ST1_55;      /*0003h – состояние процессора 1 */
    volatile Uint16   ST3_55;      /*0004h – состояние процессора 3 */
    volatile Uint16   Res0[64];    /*0008h – регистры микропроцессора */
    volatile Uint16   IER1;        /*0045h – разрешения прерываний 1 */
    volatile Uint16   IFR1;        /*0046h – флаги прерываний 1 */
    volatile Uint16   DBIER0;      /*0047h – разрешения отладки 0 */
    volatile Uint16   DBIER1;      /*0048h – разрешения отладки 1 */
    volatile Uint16   IVPD;        /*0049h – таблица прерываний D */
    volatile Uint16   IVPH;        /*004Ah – таблица прерываний H */
    volatile Uint16   ST2_55;      /*004Bh – состояние процессора 2 */
    volatile Uint16   Res1[4];     /*004Ch – регистры микропроцессора */
} INT_Registers;
#define   INT_REGS  ((INT_Registers*)0x0000)
INT_REGS->IVPD =  (Uint16)(VECTSTART>>8);
INT_REGS->IVPH =  INT_REGS->IVPD;
```

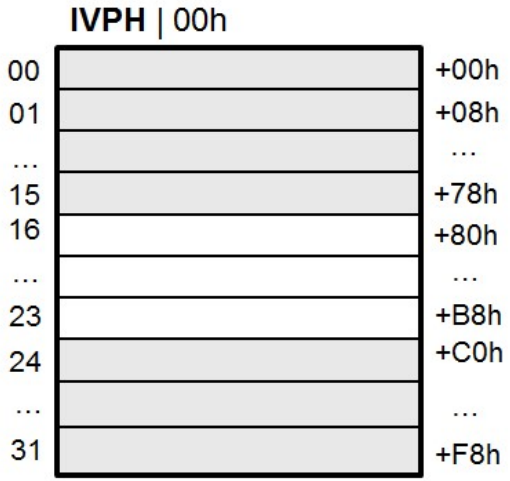
Установка вектора (C)

```
typedef void (*isr)(void);
isr ISR_plug(int num, isr iep) {
    unsigned long old, *ivp;
    unsigned int *reg;
    reg = (unsigned int*)(num>=16 && num<=23 ? 0x49 : 0x4A);
    ivp = (unsigned long*)(*reg << 3 + num << 2);
    asm(" BSET INTM");
    old = *ivp & 0xFFFFFF;
    *ivp = old & 0xFF000000 | (unsigned long)iep;
    asm(" BCLR INTM");
    return (isr)old;
}
```

IVPD 00 0049h

IVPH 00 004Ah





Установка вектора (asm)

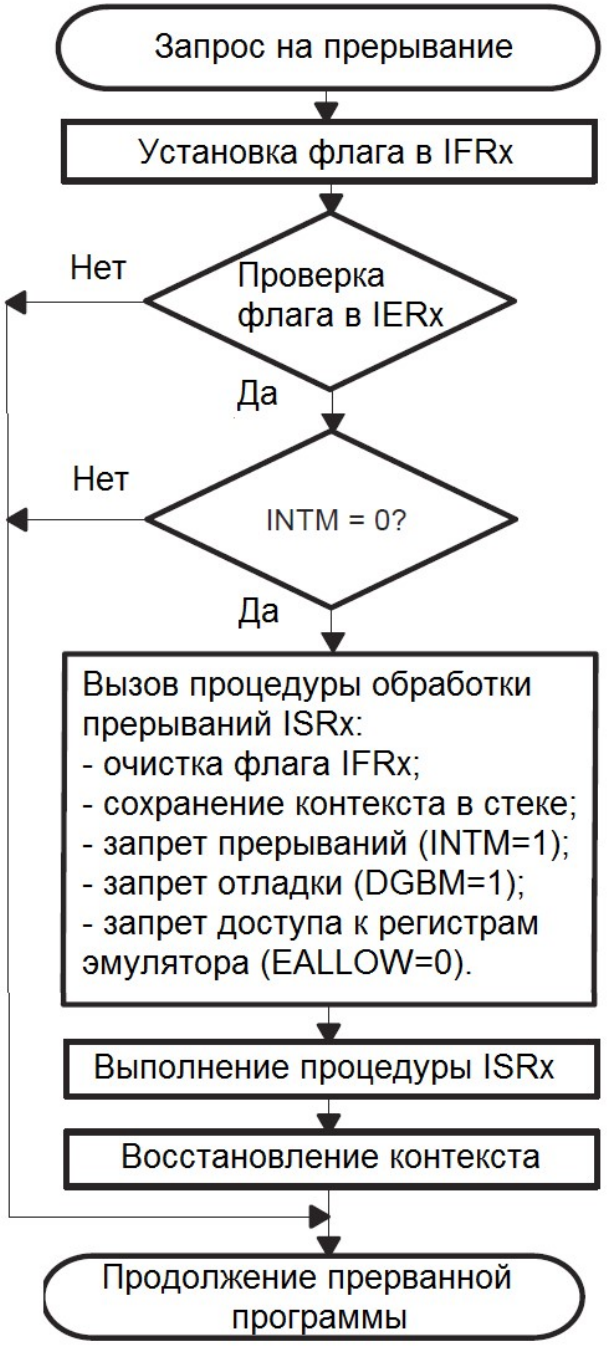
AC0 **T0** **AC0**

```
; long ISR_plug(int num, long iep)

_ISR_plug:
    AMAR    *(#049h), XAR2      ; XAR2 - IVPD
    MOV     T0, AC1             ; AC1 – номер прерывания xxxxxb
    AND     #018h, AC1, AC2     ; AC2 – xx000b
    SUB     #010h, AC2, AC2     ; AC2 – xx000b – 10000b
    XCC     AC2 == #0           ; Если IVPD, то пропуск команды
    AMAR    *(#04ah), XAR2      ; XAR2 - IVPH
    SFTS    AC1, #3, AC1        ; AC1 – смещение вектора в байтах
    BSET    INTM                ; Запретить прерывания
    MOV     *AR2, AC2           ; AC2 – регистр таблицы прерываний
    SFTS    AC2, #4, AC2        ; Сдвиг два раза по 4 бита
    SFTS    AC2, #4, AC2        ; AC2 – адрес таблицы прерываний
    OR      AC1, AC2            ; AC2 – адрес вектора в таблице (байта)
    SFTS    AC2, #-1, AC2       ; AC2 – адрес вектора в таблице (слово)
    MOV     AC2, XAR3           ; XAR3 – адрес вектора прерывания
    MOV     dbl(*AR3), AC1      ; AC1 – адрес старой процедуры
    MOV     AC0, dbl(*AR3)      ; Запись вектора в таблицу
    BCLR    INTM                ; Разрешить прерывания
    MOV     AC1, AC0            ; Возврат старого адреса в AC0
    RET
```

Пример программы

```
#define IER0 (*(volatile unsigned int *)0x00)
volatile int ISR_Count = 0;
interrupt void ADC_ISR(void)
{
    /* Ввода данных из регистра АЦП */
    ISR_Count++;
}
void adc_test()
{
    isr ISR_old = 0, ISR_new = &ADC_ISR;
    /* Задать вектор прерывания */
    ISR_old = ISR_plug(13, ISR_new );
    /* Инициализация АЦП */
    IER0 |= 0x2000;
    /* Ожидание завершения ввода данных */
    while( ISR_Count < 32 );
    /* Восстановить старый вектор прерывания */
    ISR_old = ISR_plug(13, ISR_old );
    return;
}
```



Обработчик прерывания