

ОПЕРАЦИОННАЯ СИСТЕМА РЕАЛЬНОГО ВРЕМЕНИ PICOS18

Pragmatec

Докладчик: Кушнир Сергей Николаевич, ведущий инженер, ООО «НТЦ Системы контроля», Минск, Республика Беларусь
email: picos18@tut.by

ЧТО ТАКОЕ ОС?

- Гоги, ты знаешь, что такое «ОС»?
- Канэшна знаю: «ОС» - эта балшой, паласатый мух!
- Нэт! Балшой паласатый мух – эта шмел, а «ОС» - эта то, на чем вэртится наша Земла!

(анекдот)

**основная функция ОС – поддержка
параллельного асинхронного исполнения
разных процессов и взаимодействия между
ними**

КАКИЕ ОСРВ БЫВАЮТ?

*По способу организации работы планировщики
бывают:*

- ☛ с приоритетным вытеснением (**preemptive**)*
- ☛ с вытеснением без приоритетов (**round-robin**
или «карусельного» типа)*
- ☛ без вытеснения (**cooperative**)*
- ☛ кооперативный планировщик без приоритетов*
- ☛ комбинированные планировщики: например,
приоритетное планирование используется
наряду с карусельным*



КАКИЕ ОСРВ БЫВАЮТ?

По времени реакции на события (в сторону увеличения):

- ☛ *с приоритетным вытеснением (**preemptive**)*
- ☛ *с вытеснением без приоритетов (**round-robin** или «карусельного» типа)*
- ☛ *без вытеснения (**cooperative**)*
- ☛ *кооперативный планировщик без приоритетов*



КАКИЕ ОСРВ БЫВАЮТ?

По объему занимаемого ОЗУ (в сторону увеличения):

- ☞ *без вытеснения*
- ☞ *с вытеснением*

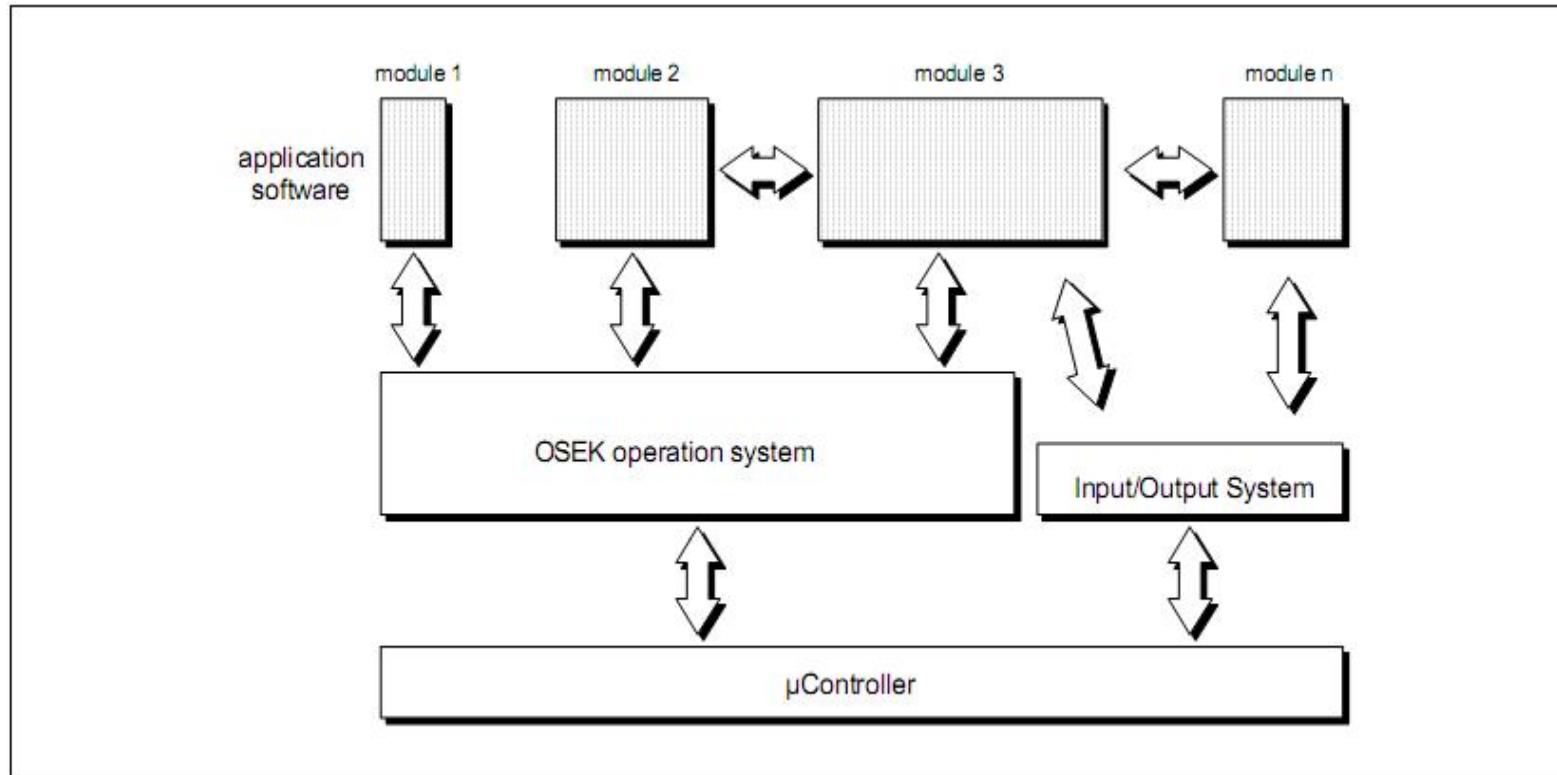


PIC-КОНТРОЛЛЕРЫ И ОС

- ☪ *PIC10, PIC12* - *нет возможности*
- ☪ PIC16 - только кооперативные
- ☪ PIC18 - **все типы**
- ☪ PIC24, PIC30, PIC33 - **все типы**
- ☪ PIC32 - **все типы**



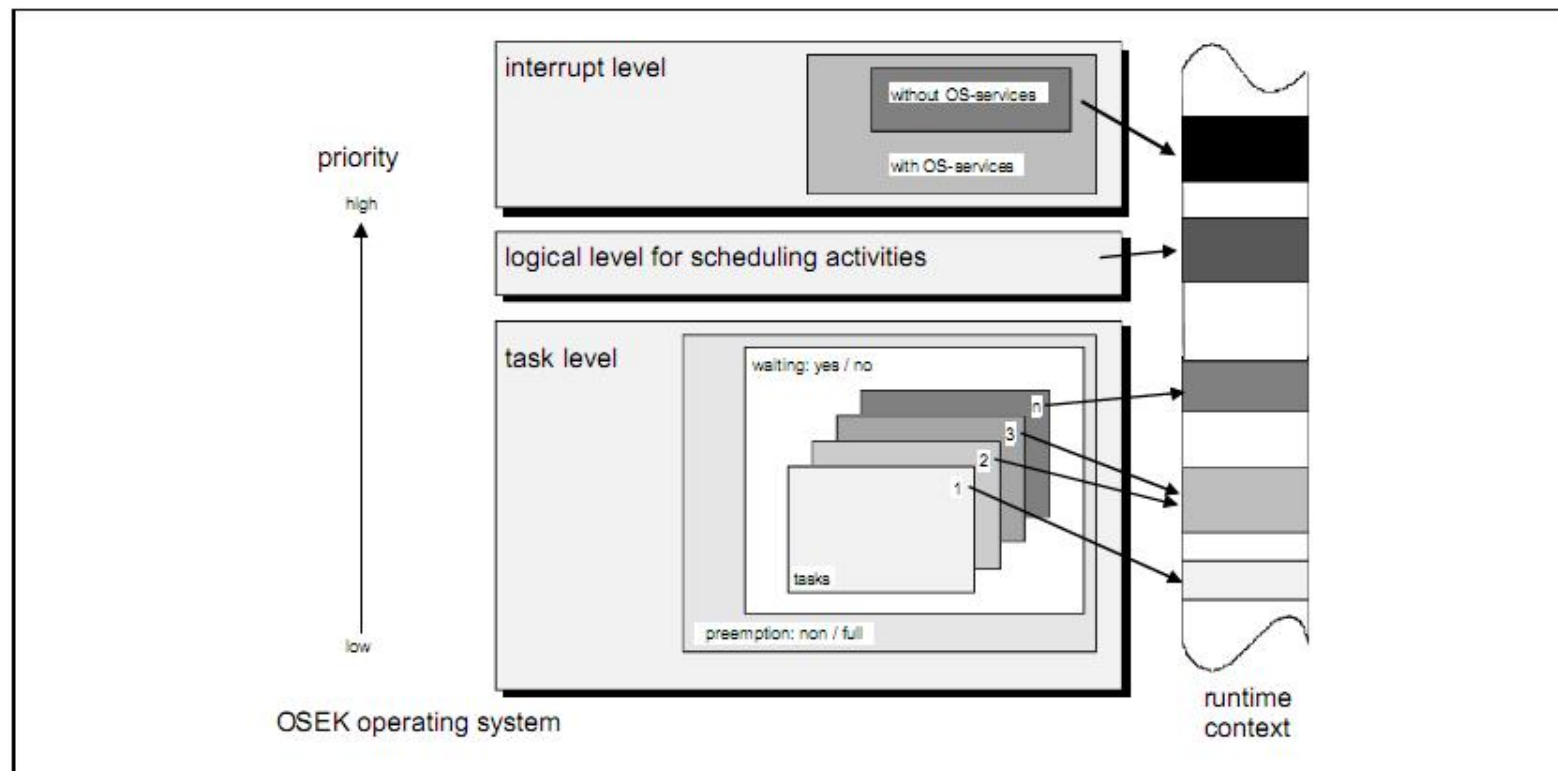
СИСТЕМНАЯ ФИЛОСОФИЯ OSEK/VDX



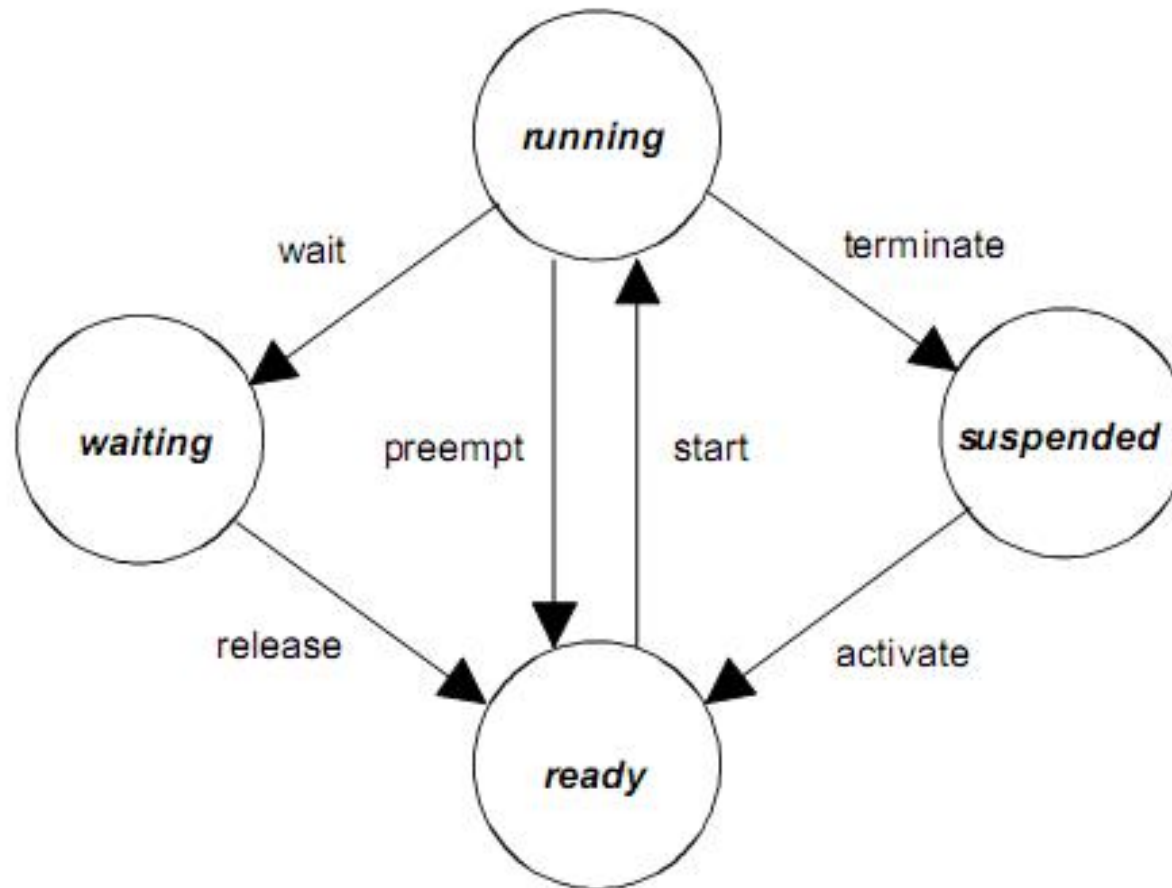
❖ Программные интерфейсы встроенной системы



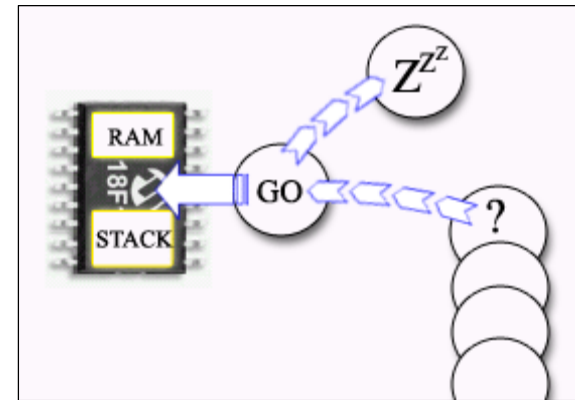
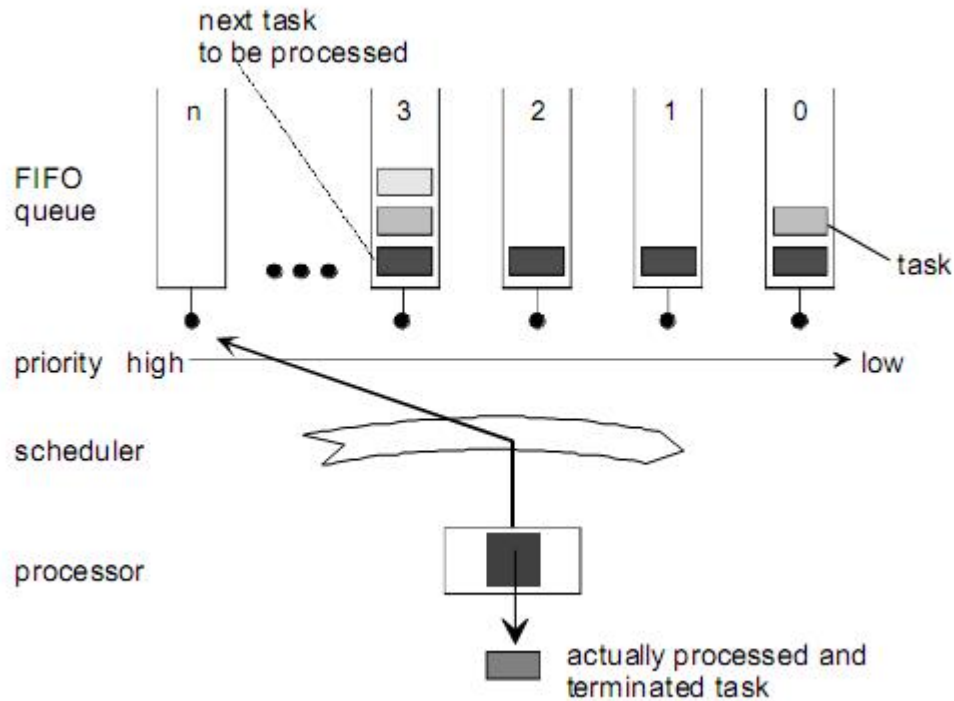
АРХИТЕКТУРА ОС



МОДЕЛЬ СОСТОЯНИЙ ЗАДАЧ



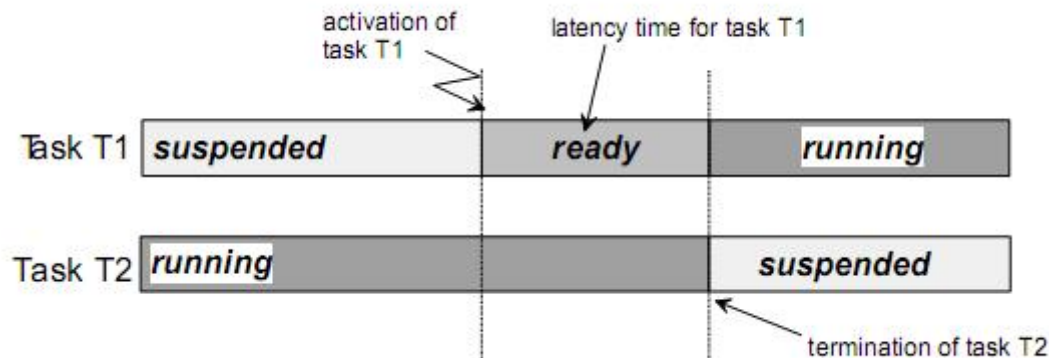
ПЛАНИРОВЩИК: ПОРЯДОК СОБЫТИЙ



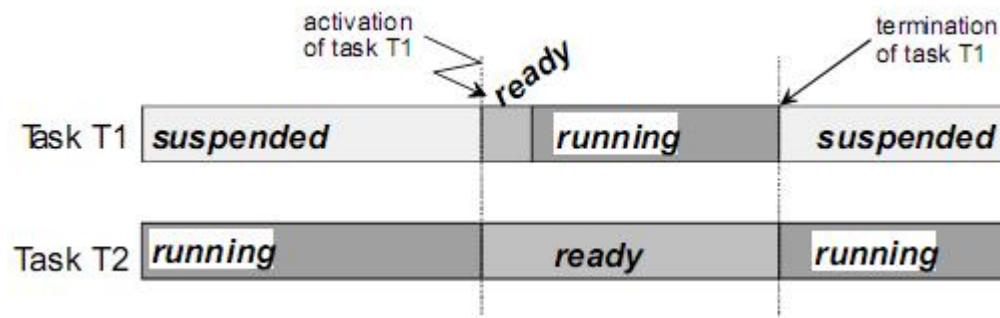
- Планировщик ищет задачи с состояниями **ready/running**.
- Из массива задач с состояниями **ready/running** планировщик выделяет набор задач с наивысшим приоритетом.
- В этом наборе планировщик находит самую старую(в смысле ожидания) задачу и запускает ее на выполнение.



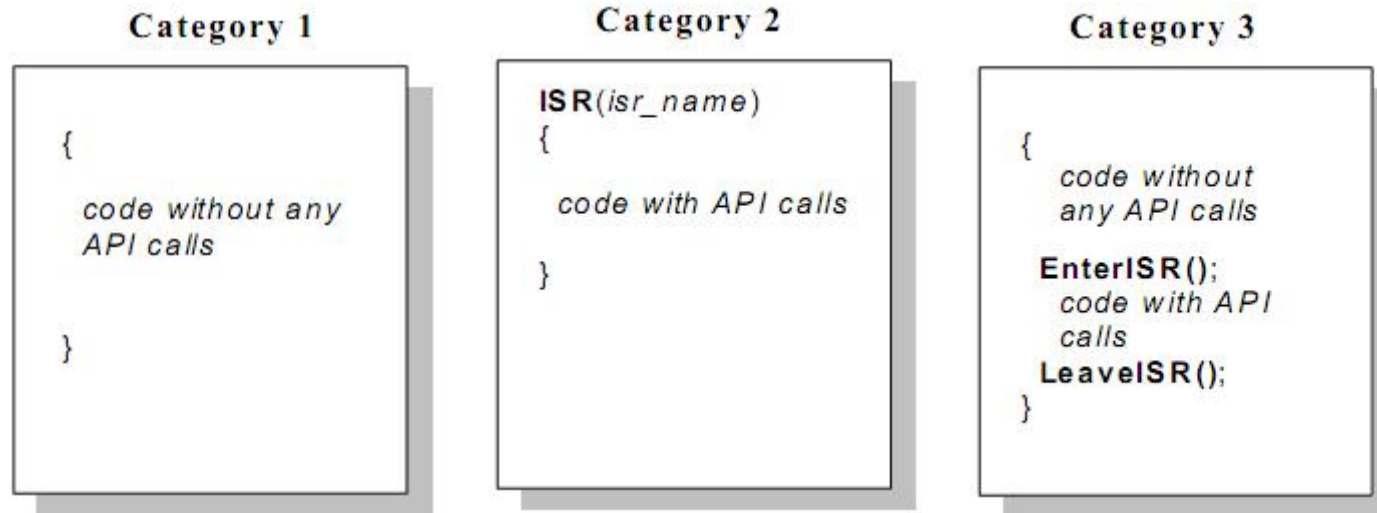
ПЛАНИРОВЩИК НЕВЫТЕСНЯЮЩЕГО ТИПА



ПЛАНИРОВЩИК ВЫТЕСНЯЮЩЕГО ТИПА



ОБСЛУЖИВАНИЕ ПРЕРЫВАНИЙ

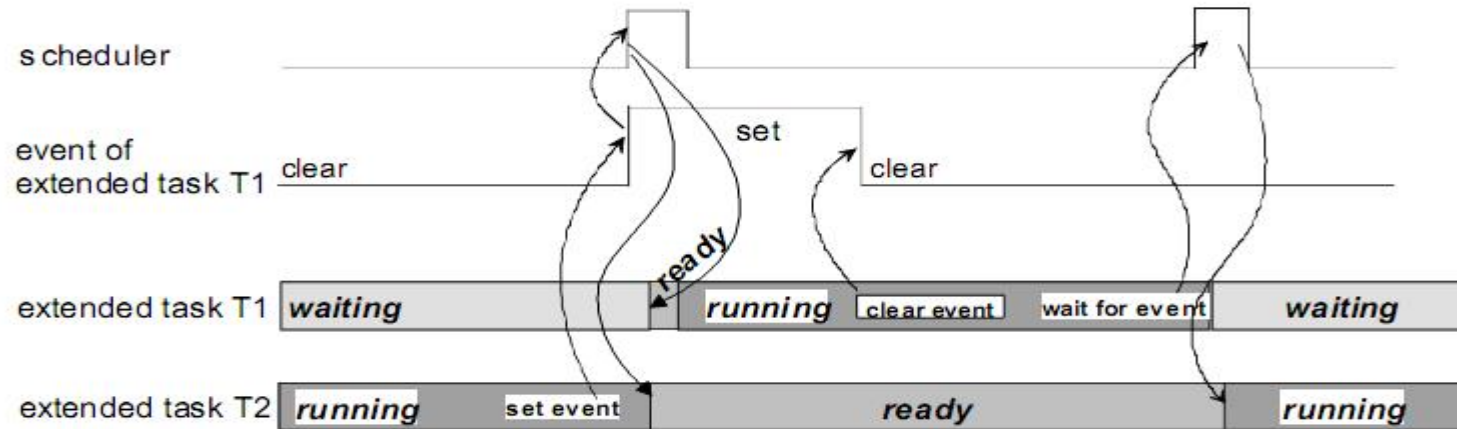


OSEK/VDX выделяет три категории обработчиков прерываний:

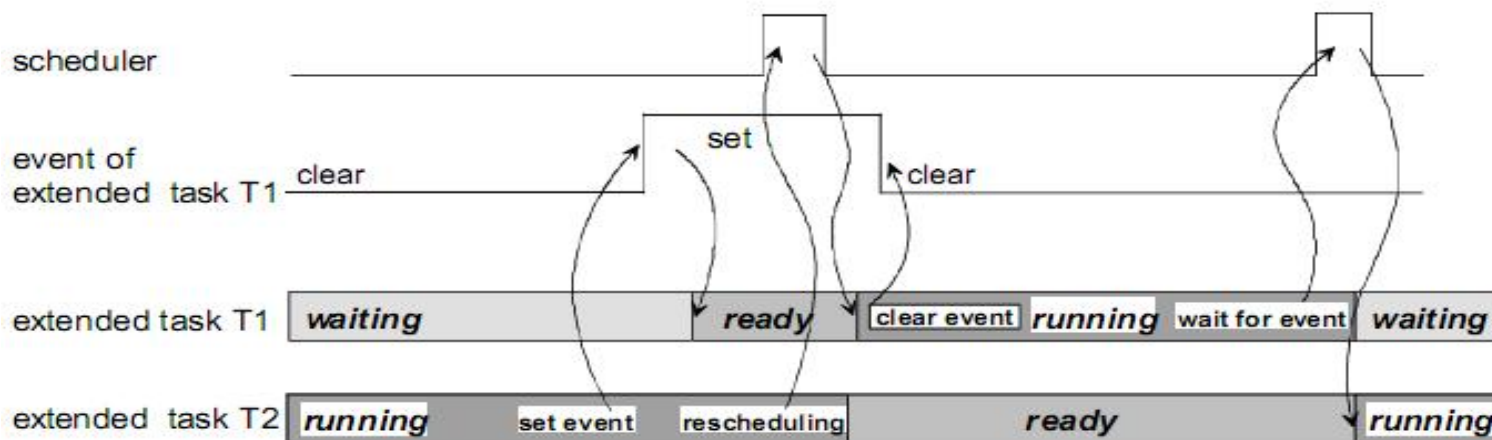
- ☛ **ISR** не использует сервисы ОС
- ☛ Используется predetermined **ISR** как сервис ОС
- ☛ Используются сервисы ОС для входа в прерывание и выхода из него

МЕХАНИЗМ СОБЫТИЙ

СИНХРОНИЗАЦИЯ ЗАДАЧ С ВЫТЕСНЯЮЩИМ ПЛАНИРОВОЩИКОМ...

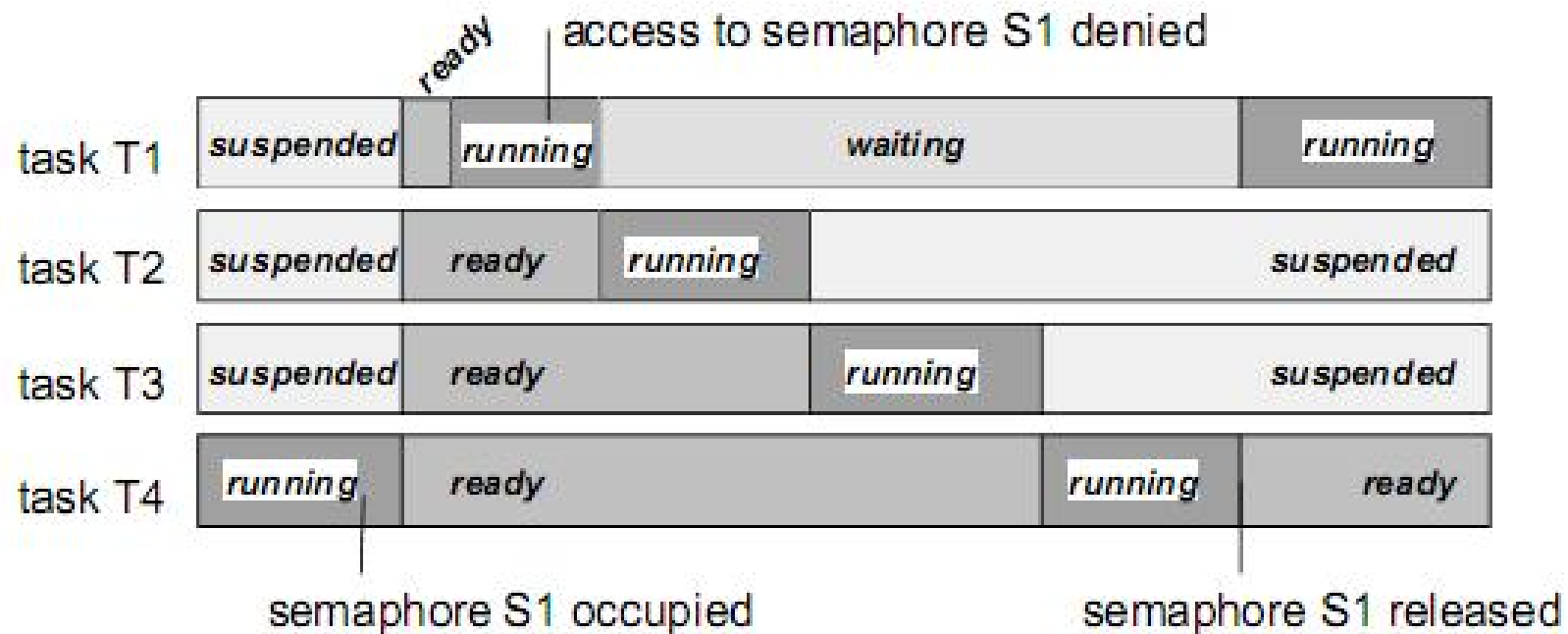


... И БЕЗ ВЫТЕСНЯЮЩЕГО ПЛАНИРОВОЩИКА



УПРАВЛЕНИЕ РЕСУРСАМИ

Главная проблема механизмов синхронизации

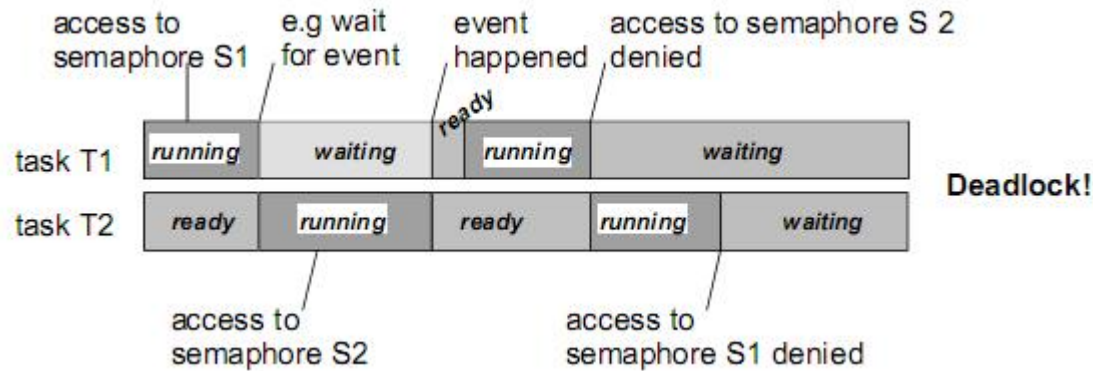


ИНВЕРСИЯ ПРИОРИТЕТОВ ПРИ ИСПОЛЬЗОВАНИИ СЕМАФОРОВ



БЛОКИРОВКА СЕМАФОРОВ

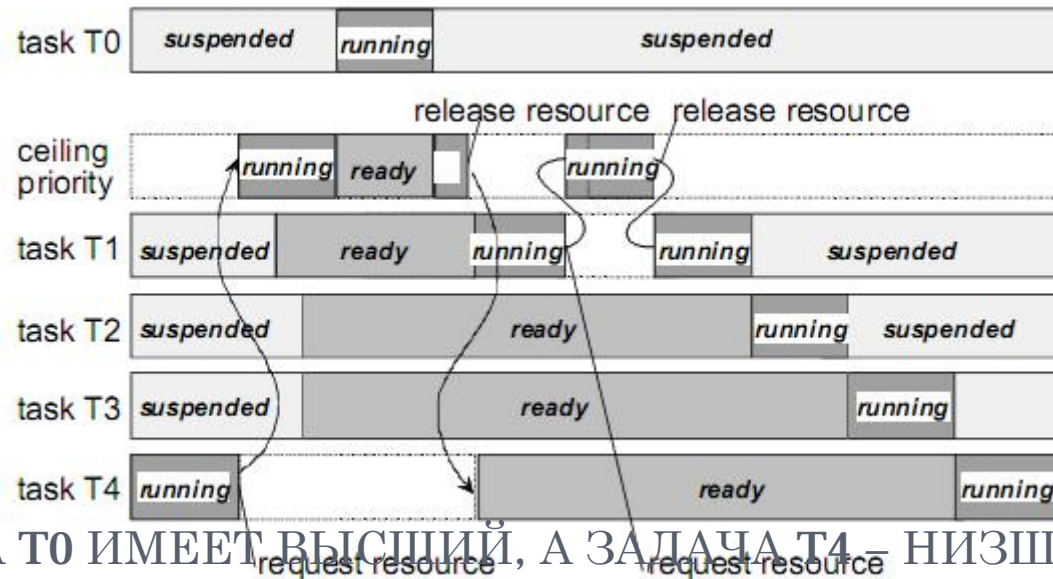
Еще одна проблема механизмов синхронизации



ВЗАИМНАЯ БЛОКИРОВКА ЗАДАЧ ПРИ ИСПОЛЬЗОВАНИИ СЕМАФОРОВ: ЗАДАЧА T1 ОЖИДАЕТ ДОСТУП К СЕМАФОРУ S2, ЗАНЯТЫЙ ЗАДАЧЕЙ T2, А ЗАДАЧА T2 ОЖИДАЕТ ДОСТУП К СЕМАФОРУ S1, ЗАНЯТЫЙ ЗАДАЧЕЙ T1!



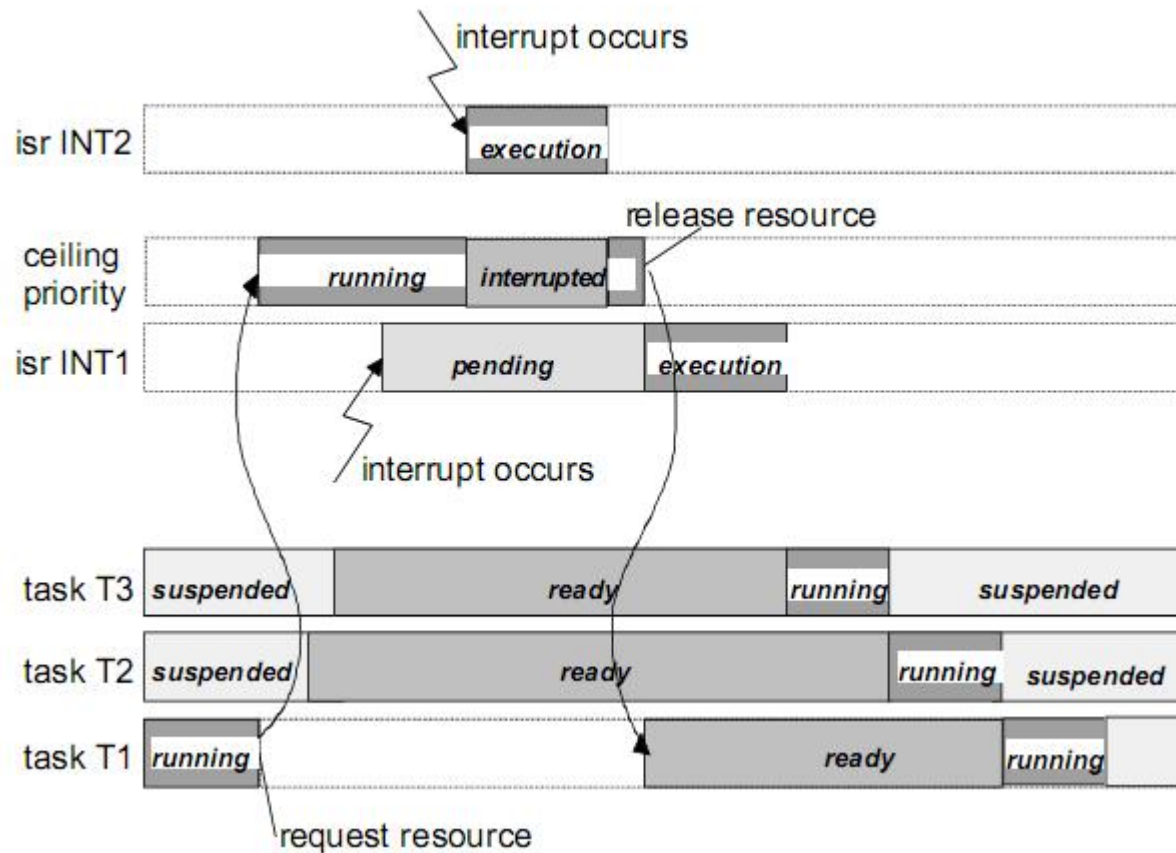
РАСПРЕДЕЛЕНИЕ РЕСУРСОВ С ПРЕДЕЛЬНЫМ ПРИОРИТЕТОМ



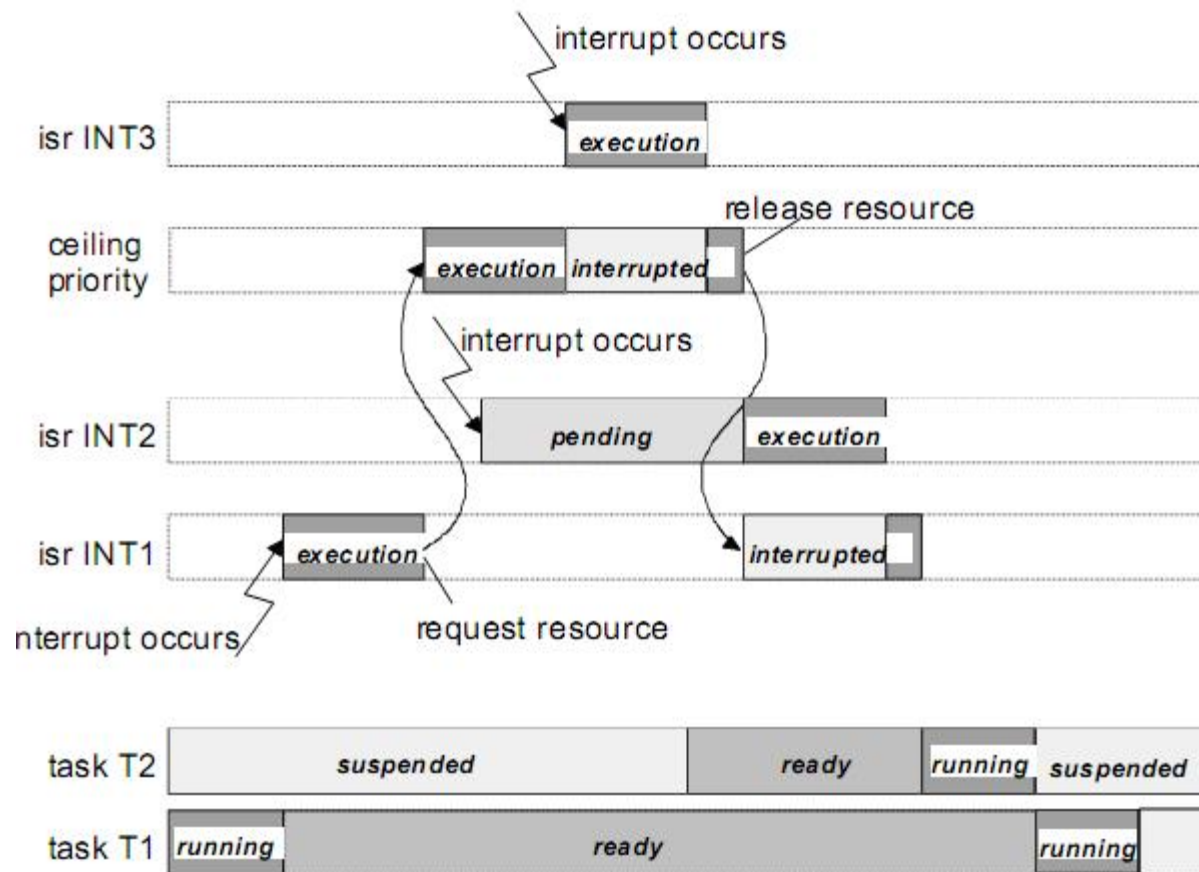
ЗАДАЧА T0 ИМЕЕТ ВЫСШИЙ, А ЗАДАЧА T4 – НИЗШИЙ ПРИОРИТЕТЫ. ЗАДАЧИ T1 И T4 ХОТЯТ ПОЛУЧИТЬ ДОСТУП К ОДНОМУ И ТОМУ ЖЕ РЕСУРСУ. НАГЛЯДНО ПОКАЗАНО, ЧТО ИНВЕРСИИ ПРИОРИТЕТОВ В ДАННОМ СЛУЧАЕ НЕ ПРОИСХОДИТ. ВЫСОКОПРИОРИТЕТНАЯ ЗАДАЧА T1 ОЖИДАЕТ МЕНЬШЕ ВРЕМЕНИ, ЧЕМ МАКСИМАЛЬНАЯ ПРОДОЛЖИТЕЛЬНОСТЬ ЗАНЯТИЯ РЕСУРСА ЗАДАЧЕЙ T4.



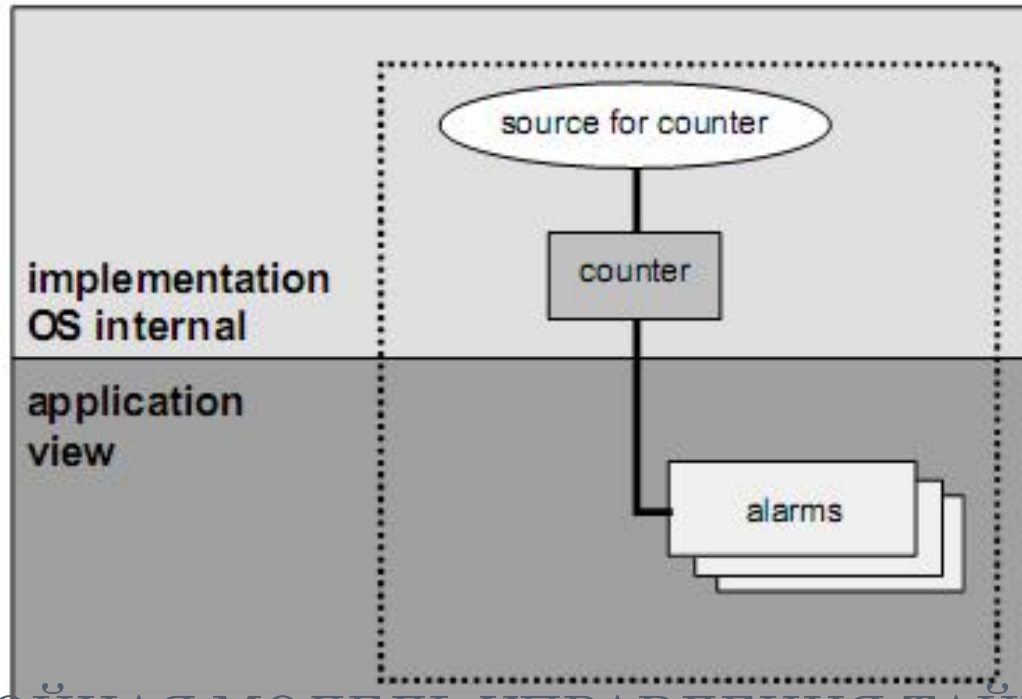
РАСПРЕДЕЛЕНИЕ РЕСУРСОВ С ПРЕДЕЛЬНЫМ ПРИОРИТЕТОМ И ОБРАБОТКА ПРЕРЫВАНИЙ



РАСПРЕДЕЛЕНИЕ РЕСУРСОВ МЕЖДУ ОБРАБОТЧИКАМИ ПРЕРЫВАНИЙ



ТАЙМЕРЫ И ТАЙМАУТЫ (ALARMS)



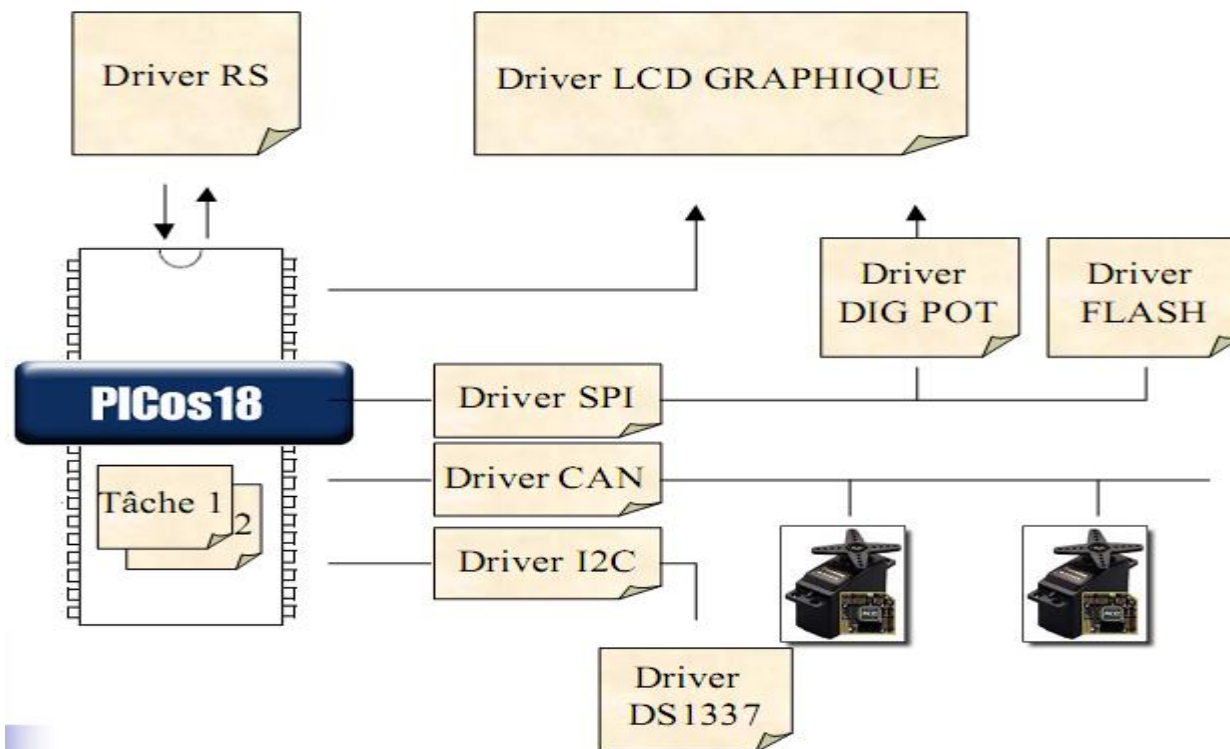
ПОСЛОЙНАЯ МОДЕЛЬ УПРАВЛЕНИЯ ТАЙМЕРАМИ И ТАЙМАУТАМИ.

ОДИН СЧЕТЧИК ОС ОБЕСПЕЧИВАЕТ РАБОТУ ВСЕХ ТАЙМАУТОВ ВСЕХ ЗАДАЧ.

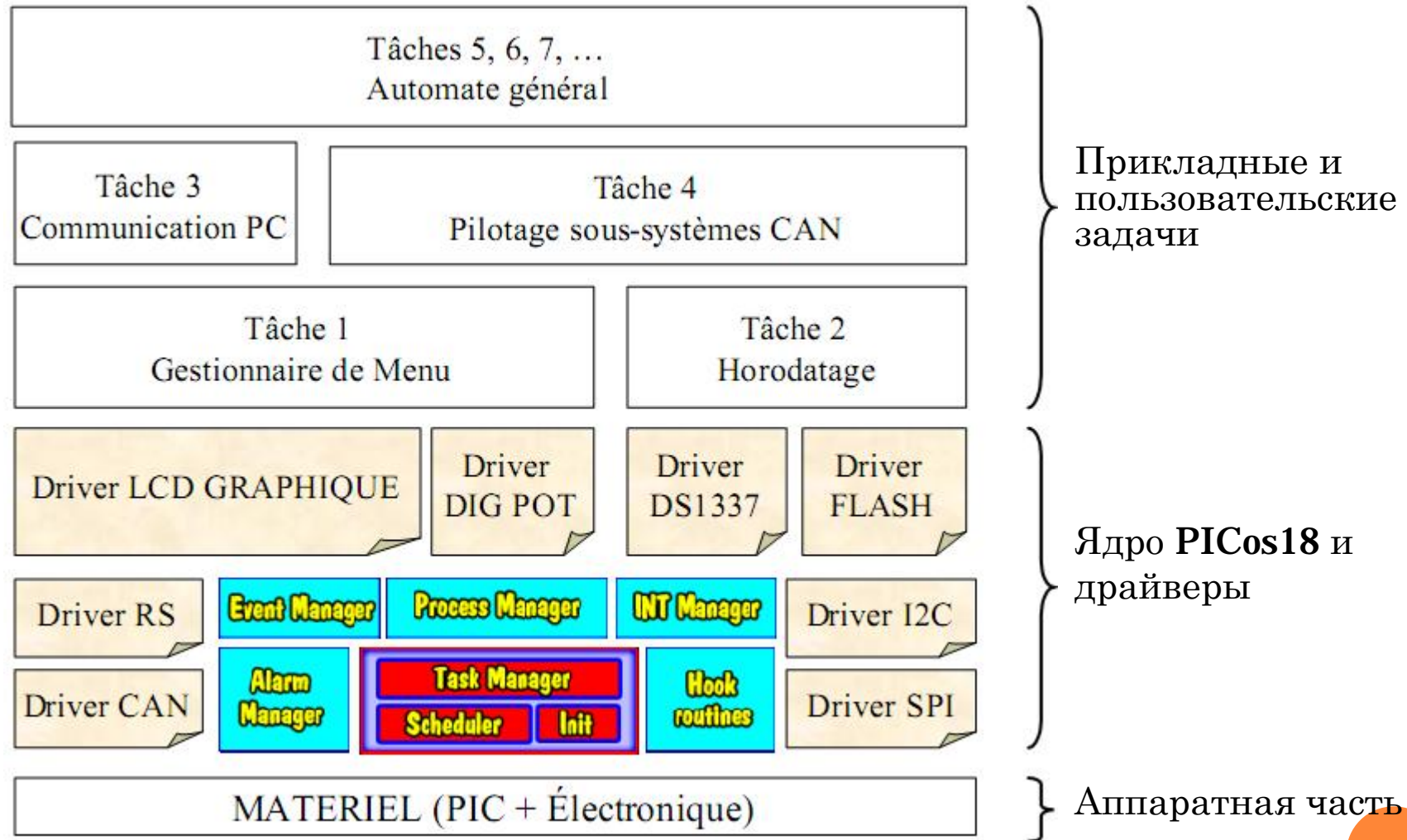


МНОГОЗАДАЧНАЯ ОС

- *Операционная система...*
- *...многозадачная...*
- *...реального времени...*



СТРУКТУРА ВСТРОЕННОЙ СИСТЕМЫ



ЯДРО И СЕРВИСЫ PICOS18



Task Manager, Scheduler, Init составляют ядро ОС.



ЛИЦЕНЗ ИЯ_GPL



Pragmatec

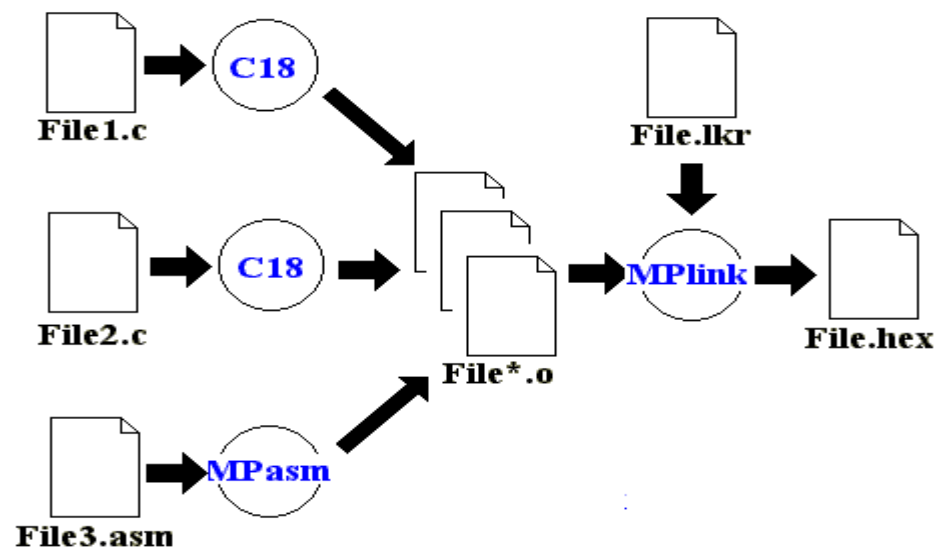
```

/*****
/*
/* File name: filename of the source file
/*
/* Since: date of creation
/*
/* Version: 3.xx (current release of PICos18)
/*
/* Author: Designed by Pragmatec S.A.R.L. www.pragmatec.net
/* MONTAGNE Xavier [XM] xavier.montagne@pragmatec.net
/* LASTNAME Firstname [xx]
/*
/* Purpose: file content explanation
/*
/* The GPL licence from Boston (USA)(see « Free Software Foundation)
/* Distribution: This file is part of PICos18.
/* PICos18 is free software; you can redistribute it
/* and/or modify it under the terms of the GNU General
/* Public License as published by the Free Software
/* Foundation; either version 2, or (at your option)
/* any later version.
/*
/* PICos18 is distributed in the hope that it will be
/* useful, but WITHOUT ANY WARRANTY; without even the
/* implied warranty of MERCHANTABILITY or FITNESS FOR A
/* PARTICULAR PURPOSE. See the GNU General Public
/* License for more details.
/*
/* You should have received a copy of the GNU General
/* Public License along with gpsim; see the file
/* COPYING.txt. If not, write to the Free Software
/* Foundation, 59 Temple Place - Suite 330,
/* Boston, MA 02111-1307, USA.
/*
/* > A special exception to the GPL can be applied should
/* you wish to distribute a combined work that includes
/* PICos18, without being obliged to provide the source
/* code for any proprietary components.
/*
/* History:
/* 2004/09/20 [XM] Create this file.
/*
/*****

```



КОМПИЛЯТОР MICROCHIP



Ядро PICos18 состоит из файлов на C и файлов на ассемблере (**kernel.asm**) Эти разные файлы будут компилироваться и компоноваться все вместе с целью получить один **HEX**-файл, который будет загружен в ваш **PIC18**.



PICOS18 : ОБУЧЕНИЕ



Средства разработки



Первая задача



Вытеснение



Многозадачное приложение



Прерывания



Использование драйверов

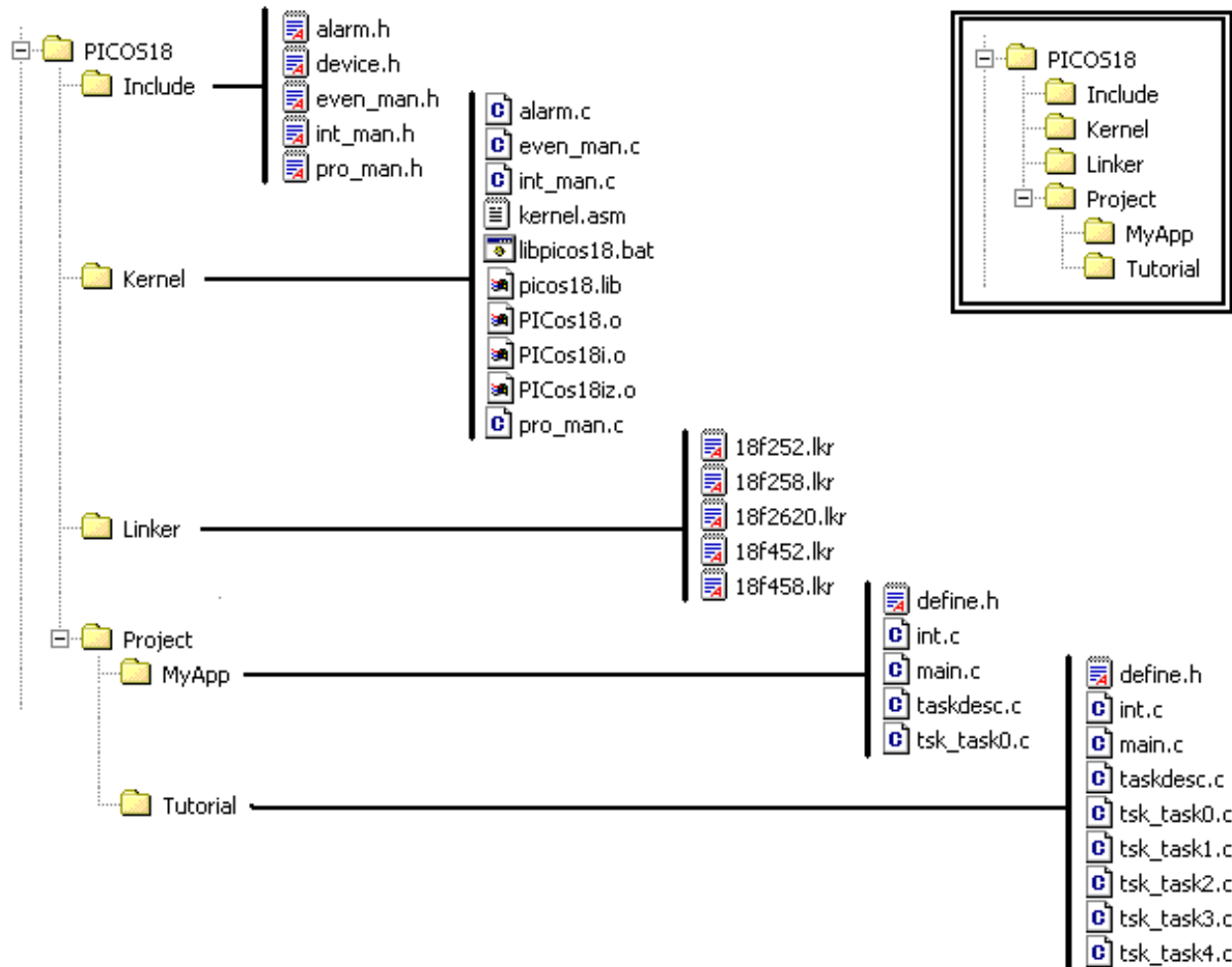


Пример приложения





УГЕДУСІДА РАЗРАБОТКИ

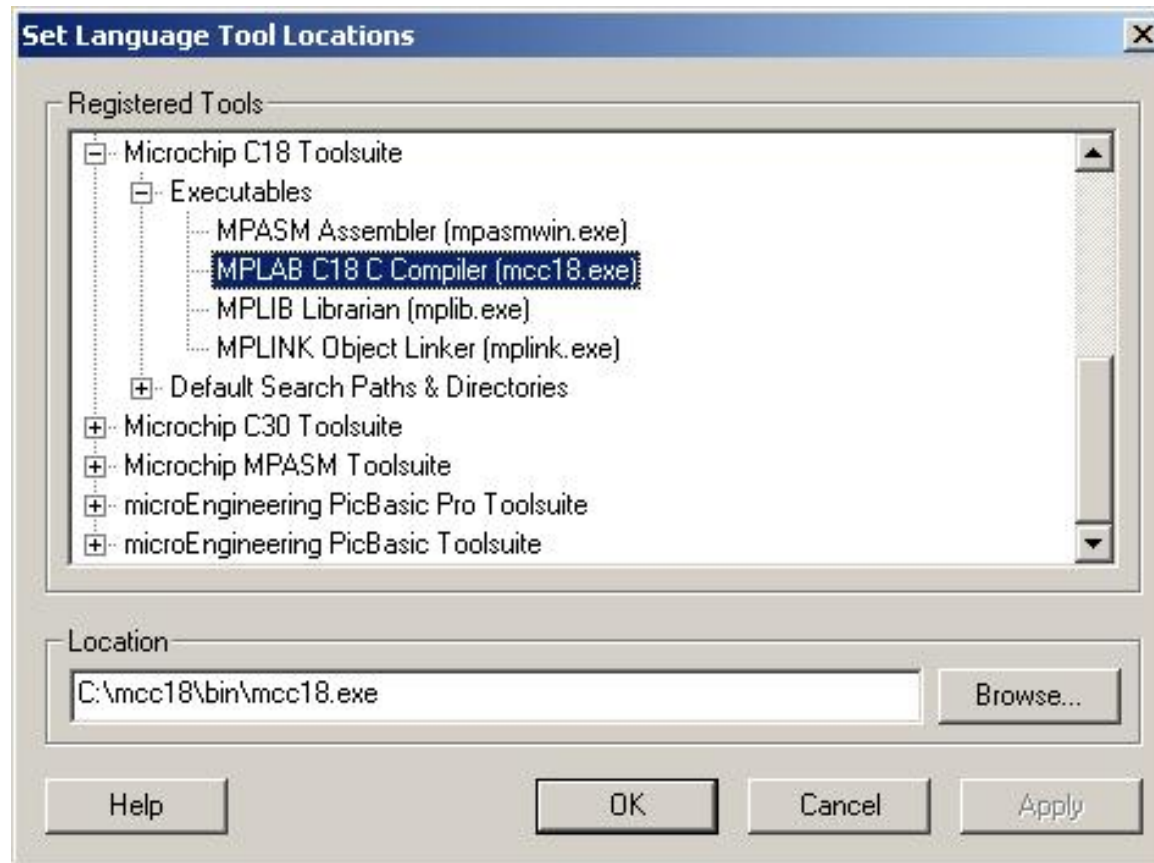


- ❖ После распаковки получится примерно такое дерево директориев с файлами ОС





КОМПИЛЯТОР C18

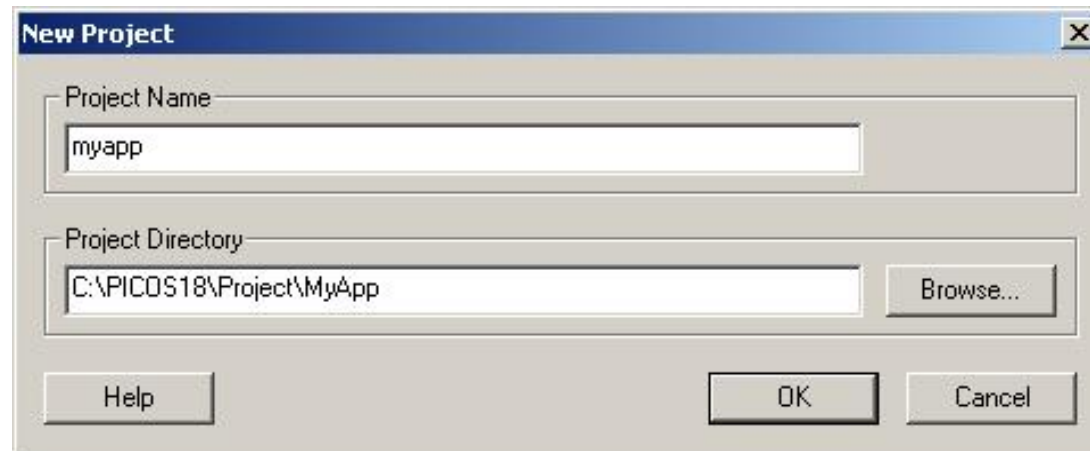


- Для компиляции проекта с использованием PICos18 необходим компилятор MPLAB C18. Student Edition также неплохо работает.





ОТКРЫТИЕ ПРОЕКТА

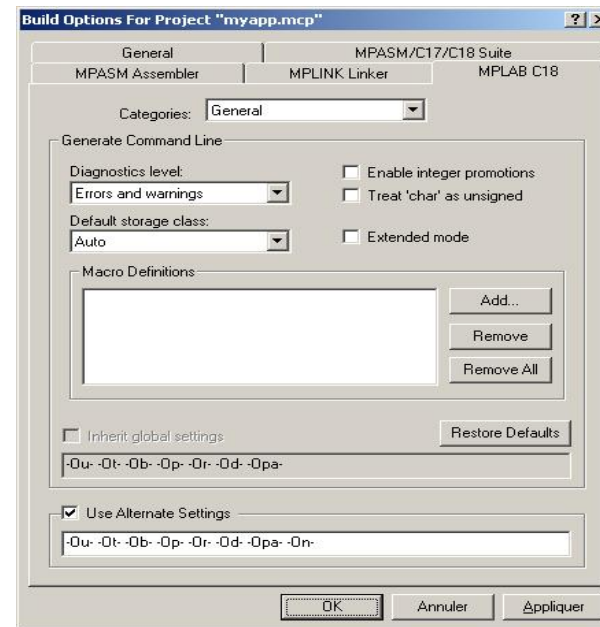
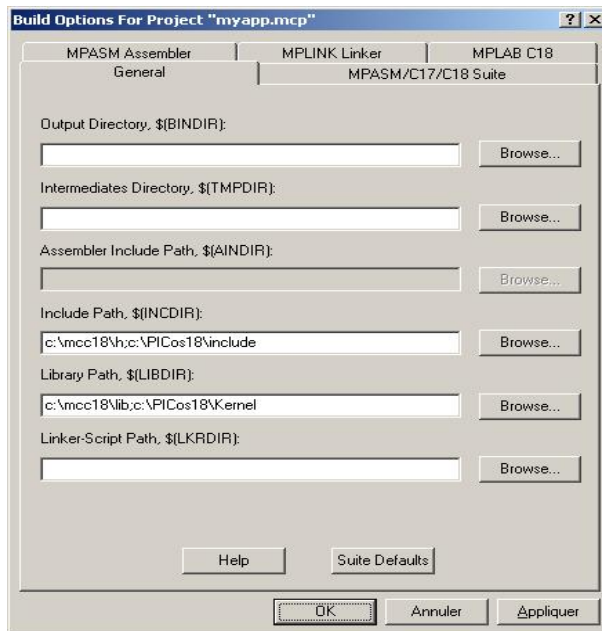


- ☛ **Задайте имя и месторасположение Вашего проекта с PICos18.**





НАСТРОЙКИ ПРОЕКТА

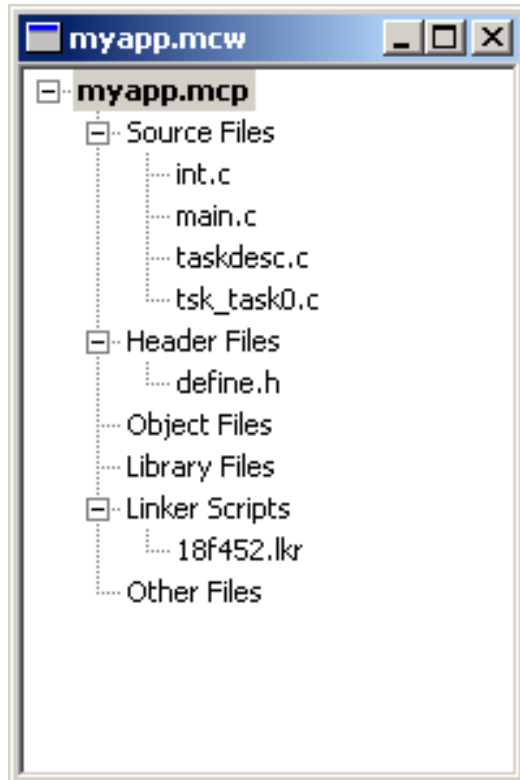


- ☛ Укажите пути к рабочим и вспомогательным директориям.
- ☛ Задайте режимы компиляции и компоновки.





ПРОЕКТ



➤ Добавьте в проект файлы ОС





КОМПИЛЯЦИЯ ПРОЕКТА

```
Output
Build | Version Control | Find in Files
Clean: Done.
Executing: "C:\mcc18\bin\mcc18.exe" -p=18F452 "int.c" -fo="int.o" /i"c:\mcc18\h" /i"c:\PICos18\include" -Ou- -Ot- -Ob- -Op- -O
Executing: "C:\mcc18\bin\mcc18.exe" -p=18F452 "main.c" -fo="main.o" /i"c:\mcc18\h" /i"c:\PICos18\include" -Ou- -Ot- -Ob- -C
Executing: "C:\mcc18\bin\mcc18.exe" -p=18F452 "taskdesc.c" -fo="taskdesc.o" /i"c:\mcc18\h" /i"c:\PICos18\include" -Ou- -C
Executing: "C:\mcc18\bin\mcc18.exe" -p=18F452 "tsk_task0.c" -fo="tsk_task0.o" /i"c:\mcc18\h" /i"c:\PICos18\include" -Ou- -C
Executing: "C:\mcc18\bin\mplink.exe" /i"c:\mcc18\lib" /i"c:\PICos18\Kernel" "C:\PICOS18\Linker\18f452.lkr" "C:\PICOS18\Pro
MPLINK 3.90, Linker
Copyright (c) 2004 Microchip Technology Inc.
Errors      : 0

MP2COD 3.90, COFF to COD File Converter
Copyright (c) 2004 Microchip Technology Inc.
Errors      : 0

MP2HEX 3.90, COFF to HEX File Converter
Copyright (c) 2004 Microchip Technology Inc.
Errors      : 0

Loaded C:\PICOS18\Project\MyApp\myapp.cof
BUILD SUCCEEDED: Sun Feb 27 20:30:30 2005
```

☛ Скомпилируйте проект





ТАЙМЕРА

Address	Value	Category	Setting
300001	26	Oscillator	HS-PLL Enabled
		Osc. Switch Enable	Disabled
300002	0D	Power Up Timer	Disabled
		Brown Out Detect	Disabled
		Brown Out Voltage	2.5V
300003	0E	Watchdog Timer	Disabled-Controlled by SWDTEN bit
		Watchdog Postscaler	1:128
300005	01	CCP2 Mux	RC1
300006	80	Stack Overflow Reset	Disabled
		Low Voltage Program	Disabled

- ❖ Сторожевой таймер будет только мешать работе, периодически вызывая сброс.
- ❖ Если Вам необходим **Watchdog Timer**, включите его после отладки программы, не забыв добавить команду его сброса. Где – определите сами!





СИМУЛЯТОРЕ

The screenshot displays the MPLAB IDE v8.62 interface. The main editor window shows a C program with a task definition. The File Registers window shows memory addresses and values. The Hardware Stack window shows the current stack state.

```
/* Definition dedicated to the local functions.
.....
#define ALARM_TMR0 0

.....
/* First task of the tutorial.
.....
TASK (TASK0)
{
    while (1) {}

/* End of file : task_task0.c */
```

Address	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	QA	QB	QC	QD	QE	QF
0000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0070	00	EE	33	00	00	00	00	00	00	00	00	00	00	00	00	00
0080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

```
INTCONbits.INTCONIF = 0;
Counter_kernel.CounterValue++;
global_counter++;
done = _E_OS_VALRSH;
for (index = 0; index < ALARMBUMBER; index++)
{
    if (Alarm_list[index].State == OFF)
        continue;

    if (Alarm_list[index].ptrCounter->CounterValue ==
        Alarm_list[index].AlarmValue)

        if (Alarm_list[index].Cycle == 0)
            Alarm_list[index].State = OFF;
        else
        {
            Alarm_list[index].AlarmValue = \
            Alarm_list[index].ptrCounter->CounterValue + \
            Alarm_list[index].Cycle;
        }

    if (Alarm_list[index].EventToPost != 0)
        SetEvent(Alarm_list[index].TaskID2Activate,
            Alarm_list[index].EventToPost);
```

TOS	Stack Level	Return Address	Location
→	0	Empty	
	1	0033EE	TASK0 + 0x0
	2	003044	InterruptVectorL + 0x44
	3	003044	InterruptVectorL + 0x44
	4	000000	
	5	000000	
	6	000000	
	7	000000	
	8	000000	
	9	000000	
	10	000000	
	11	000000	
	12	000000	

☛ Симулятор позволяет выполнить программу без микроконтроллера!





ПЕРВАЯ ЗАДАЧА

```
TASK(TASK0)
{
    while(1);
}

/*****
 * ----- task 0 -----
 *****/
rom_desc_tsk rom_desc_task0 = {
    TASK0_PRIO,          /* prioinit from 0 to 15      */
    stack0,             /* stack address (16 bits)    */
    TASK0,              /* start address (16 bits)    */
    READY,              /* state at init phase       */
    TASK0_ID,           /* id_tsk from 1 to 15       */
    sizeof(stack0)     /* stack size (16 bits)      */
};

AlarmObject Alarm_list[] =
{
/*****
 * ----- First task -----
 *****/
{
    OFF,                /* State                      */
    0,                  /* AlarmValue                  */
    0,                  /* Cycle                       */
    &Counter_kernel,   /* ptrCounter                  */
    TASK0_ID,          /* TaskID2Activate            */
    ALARM_EVENT,       /* EventToPost                 */
    0,                  /* CallBack                    */
},
};
```

} Задача

} Бланк
описания
задачи

} Бланк
описания
таймаутов
задачи

☞ У задачи есть состояние, таймауты...





ОПИСАНИЕ ЗАДАЧИ

```
/* *****  
 * ----- COUNTER & ALARM DEFINITION -----  
 * *****  
Resource Resource_list[] =  
{  
  {  
    10,          /* priority          */  
    0,          /* Task prio         */  
    0,          /* lock              */  
  }  
};  
  
/* *****  
 * ----- TASK & STACK DEFINITION -----  
 * *****  
#define DEFAULT_STACK_SIZE    128  
DeclareTask(TASK0);  
volatile unsigned char stack0[DEFAULT_STACK_SIZE];
```

Бланк
описания
ресурсов
задачи

Бланк
описания
стека
задачи

☞ ... ресурсы, стек.





MC

```
#pragma code _INTERRUPT_VECTORL = 0x003000
void InterruptVectorL(void)
{
    SAVE_TASK_CTX(stack_low, stack_high);
    EnterISR();

    if (INTCONbits.TMR0IF == 1)
        AddOneTick();
    /*Here is the next interrupts you want to manage */
    /*if (INTCONbits.RBIF == 1) */
    /* MyOwnISR(); */

    LeaveISR();
    RESTORE_TASK_CTX;
}
#pragma code
```

Обработчик
«медленных»
прерываний
3-й категории

The screenshot shows a code editor window titled "C:\PIC018\Project\MyApp\int.c" with the following code:

```
#pragma interruptlow InterruptVectorL =
void InterruptVectorL(void)
{
    EnterISR();

    if (INTCONbits.TMR0IF == 1)
        AddOneTick();
    /*Here is the next interrupts you want
    /*if (INTCONbits.RBIF == 1)
    /* MyOwnISR();
```

Next to the code editor is a "Stopwatch" window with the following data:

	Stopwatch	Total Simulated
Synch Instruction Cycles	10009	57884
Zero Time (mSecs)	1.000900	5.788400
Processor Frequency (MHz)		40.000000

There is also a checkbox labeled "Clear Simulation Time On Reset" which is checked.

- ❖ Поставьте точку останова и проследите, что прерывания происходят через 1 мс, т.е. 1 тик = 1 мс.





ЗАПУСК ПРИЛОЖЕНИЯ

```
void Init(void)
{
    FSR0H = 0;
    FSR0L = 0;

    /* User setting : actual PIC frequency */
    Tmr0.lt = TMR0_PRESET;

    /* Timer OFF - Enabled by Kernel */
    TOCON = 0x08;
    TMR0H = Tmr0.bt[1];
    TMR0L = Tmr0.bt[0];
}
```

Сервис ОС -
функция
инициализации
и периферии

- ☛ Если необходима другая длительность тика, то переопределите константу **TMR0_PRESET**





ВЫТЕСНЕНИЕ

Периодическое просыпание задачи

```
TASK(TASK0)
{
    TRISBbits.TRISB4 = 0;
    LATBbits.LATB4   = 0;

    SetRelAlarm(ALARM_TSK0, 1000, 200);

    while(1)
    {
        WaitEvent(ALARM_EVENT);
        ClearEvent(ALARM_EVENT);

        LATBbits.LATB4 = ~LATBbits.LATB4;
    }
}

AlarmObject Alarm_list[] =
{
    /******
    * ----- First task -----
    *****/
    {
        OFF,                /* State          */
        0,                  /* AlarmValue     */
        0,                  /* Cycle          */
        &Counter_kernel,   /* ptrCounter     */
        TASK0_ID,          /* TaskID2Activate */
        ALARM_EVENT,      /* EventToPost    */
        0,                  /* Callback       */
    },
};
```

☛ Задача Task0 и описание ее таймаута





СИМУЛЯТОРЕ

The screenshot shows a simulation environment with two windows. The left window, titled 'C:\...\tsk_task0.c', displays the following C code:

```
while (1)
{
    WaitEvent (ALARM_EVENT);
    ClearEvent (ALARM_EVENT);

    LATBbits.LATB4 = ~LATBbits.LATB4;
}
```

The right window, titled 'Watch', shows a table of variables being monitored:

Address	Symbol Name	Value
0F81	PORTB	0x10
01A2	global_counter	2401

At the bottom of the Watch window, there are four tabs labeled 'Watch 1', 'Watch 2', 'Watch 3', and 'Watch 4'.

- ❖ Поставьте точку прерывания и проследите, как меняется значение счетчика таймаутов.





ОТКРЫТИЕ ВТОРОЙ ЗАДАЧИ

```
#define DEFAULT_STACK_SIZE 128
DeclareTask(TASK0);
DeclareTask(TASK1);

volatile unsigned char stack0[DEFAULT_STACK_SIZE];
volatile unsigned char stack1[DEFAULT_STACK_SIZE];

/*****
 * ----- TASK DESCRIPTOR SECTION -----
 *****/
#pragma          romdata          DESC_ROM
const rom unsigned int descromarea;
/*****
 * ----- task 0 -----
 *****/
rom_desc_tsk rom_desc_task0 = {
    TASK0_PRIO,          /* prioinit from 0 to 15      */
    stack0,              /* stack address (16 bits)    */
    TASK0,               /* start address (16 bits)    */
    READY,               /* state at init phase       */
    TASK0_ID,            /* id_tsk from 1 to 15       */
    sizeof(stack0)       /* stack size (16 bits)       */
};
/*****
 * ----- task 1 -----
 *****/
rom_desc_tsk rom_desc_task1 = {
    TASK1_PRIO,          /* prioinit from 0 to 15      */
    stack1,              /* stack address (16 bits)    */
    TASK1,               /* start address (16 bits)    */
    READY,               /* state at init phase       */
    TASK1_ID,            /* id_tsk from 1 to 15       */
    sizeof(stack1)       /* stack size (16 bits)       */
};
```

☛ К задаче Task0 добавьте задачу Task1.





ОТКРЫТИЕ ВТОРОЙ ЗАДАЧИ

```
unsigned char hour, min, sec;

/*****
 * ----- TASK1 -----
 *
 * Second task of the tutorial.
 *
 *****/
TASK(TASK1)
{
    hour = min = sec = 0;

    while(1)
    {
        WaitEvent(TASK1_EVENT);
        ClearEvent(TASK1_EVENT);

        sec++;
        if (sec == 60)
        {
            sec = 0;
            min++;
            if (min == 60)
            {
                min = 0;
                hour++;
            }
        }
    }
}
```

☛ Задача **Task1** будет вести счет времени





ОТКРЫТИЕ ВТОРОЙ ЗАДАЧИ

```
SetRelAlarm(ALARM_TSK0, 1000, 1000);
while(1)
{
    WaitEvent(ALARM_EVENT);
    ClearEvent(ALARM_EVENT);

    LATBbits.LATB4 = ~LATBbits.LATB4;
    SetEvent(TASK1_ID, TASK1_EVENT);
}
```

```
/*
 * ----- Events -----
 */
#define ALARM_EVENT 0x80
#define TASK1_EVENT 0x10

/*
 * ----- Task ID -----
 */
#define TASK0_ID 1
#define TASK1_ID 2

#define TASK0_PRIO 7
#define TASK1_PRIO 6
```

- ☞ Измените задачу Task0
- ☞ И обновите определения





СИМУЛЯТОРЕ

The screenshot shows a debugger interface with two windows. The left window, titled 'C:\...\tsk_task1.c', displays the following C code:

```
while(1)
{
    WaitEvent(TASK1_EVENT);
    ClearEvent(TASK1_EVENT);

    sec++;
    if (sec == 60)
    {
        sec = 0;
        min++;
        if (min == 60)
```

The right window, titled 'Watch', shows a table of variables being monitored:

Address	Symbol Name	Value
0F81	PORTB	0x10
0222	global_counter	7001
0233	sec	7
0232	min	0
0231	hour	0

At the bottom of the Watch window, there are buttons for 'Watch 1', 'Watch 2', 'Watch 3', and 'Watch 4'.

- ☛ Поставьте точку останова и наблюдайте за счетом времени!





ВЫТЕСНЕНИЕ

```
C:\...\tsk_task0.c
while (1)
{
    WaitEvent (ALARM_EVENT);
    ClearEvent (ALARM_EVENT);

    LATBbits.LATB4 = ~LATBbits.LATB4;
    SetEvent (TASK1_ID, TASK1_EVENT);
}

C:\...\tsk_task1.c
while (1)
{
    WaitEvent (TASK1_EVENT);
    ClearEvent (TASK1_EVENT);

    sec++;
    if (sec == 60)
    {
        sec = 0;
    }
}
```

WaitEvent в задаче 0
SetEvent в задаче 0
WaitEvent в задаче 1
ClearEvent в задаче 1

☞ Поставьте точки останова и наблюдайте за последовательностью остановок





ВЫТЕСНЕНИЕ

```
/* *****  
 * ----- Task ID -----  
 * ***** */  
#define TASK0_ID      1  
#define TASK1_ID      2  
  
#define TASK0_PRIO    7  
#define TASK1_PRIO    10
```

WaitEvent в задаче 0
SetEvent в задаче 0
ClearEvent в задаче 1
WaitEvent в задаче 0

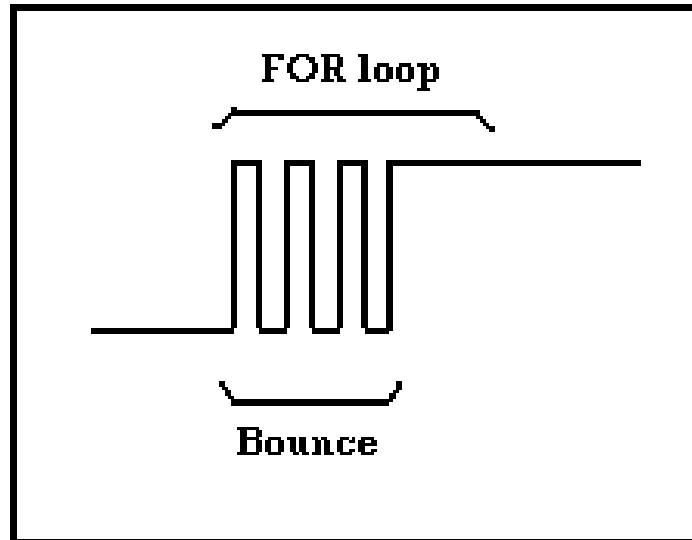
- ☛ Измените значение приоритета задачи **Task1** и наблюдайте за изменившимся порядком остановок





ПРИЛОЖЕНИЕ

Создание фоновой задачи



- ☛ Типичный сигнал с дребезгом от кнопки на входе микроконтроллера





ДОБАВЛЕНИЕ ЗАДАЧИ

```
#define DEFAULT_STACK_SIZE 128
DeclareTask(TASK0);
DeclareTask(TASK1);
DeclareTask(TASK2);

volatile unsigned char stack0[DEFAULT_STACK_SIZE];
volatile unsigned char stack1[DEFAULT_STACK_SIZE];
volatile unsigned char stack2[DEFAULT_STACK_SIZE];

/*****
 * ----- TASK DESCRIPTOR SECTION -----
 *****/
#pragma          romdata          DESC_ROM
...
/*****
 * ----- task 1 -----
 *****/
rom_desc_tsk rom_desc_task2 = {
    TASK2_PRIO,          /* prioinit from 0 to 15      */
    stack2,              /* stack address (16 bits)    */
    TASK2,               /* start address (16 bits)   */
    READY,               /* state at init phase       */
    TASK2_ID,            /* id_tsk from 1 to 15      */
    sizeof(stack2)      /* stack size (16 bits)      */
};
```

☛ Добавим задачу Task2





ДОБАВЛЕНИЕ ЗАДАЧИ

```
extern unsigned char hour;

/*****
 * ----- TASK2 -----
 *
 * Third task of the tutorial.
 *
 *****/
TASK(TASK2)
{
    unsigned int i;

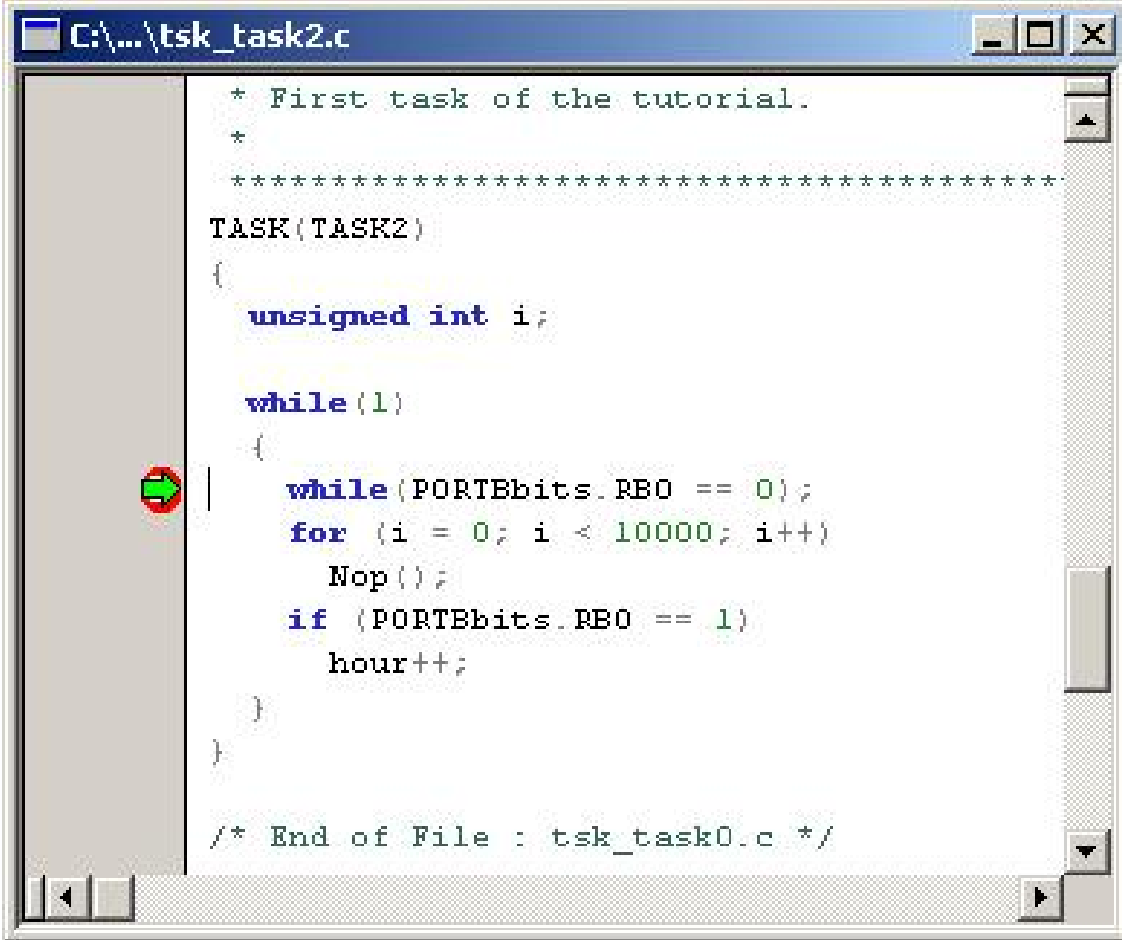
    while(1)
    {
        while(PORTBbits.RB0 == 0);
        for (i = 0; i < 10000; i++)
            Nop();
        if (PORTBbits.RB0 == 1)
            hour++;
    }
}

/*****
 * ----- Task ID -----
 *****/
#define TASK0_ID 1
#define TASK1_ID 2
#define TASK2_ID 3

#define TASK0_PRIO 7
#define TASK1_PRIO 10
#define TASK2_PRIO 1
```

☛ Добавим задачу Task2





```
C:\...\tsk_task2.c
* First task of the tutorial.
*
*****
TASK(TASK2)
{
    unsigned int i;

    while (1)
    {
        while (PORTBbits.RB0 == 0);
        for (i = 0; i < 10000; i++)
            Nop();
        if (PORTBbits.RB0 == 1)
            hour++;
    }
}

/* End of File : tsk_task0.c */
```

☛ Простой способ подавления дребезга





КАРУСЕЛЬ

```
/*
 * ----- Task ID -----
 */
#define TASK0_ID 1
#define TASK1_ID 2
#define TASK2_ID 3
#define TASK3_ID 4

#define TASK0_PRIO 7
#define TASK1_PRIO 10
#define TASK2_PRIO 1
#define TASK3_PRIO 1
```

The image shows two side-by-side code editor windows. The left window is titled 'C:\...\tsk_task2.c' and contains the following code:

```
TASK(TASK2)
{
    unsigned int i;
    TRISBbits.TRISB0 = 1;

    while(1)
    {
        while(PORTBbits.RB0 == 0);
        for (i = 0; i < 10000; i++)
            Nop();
        if (PORTBbits.RB0 == 1)
            hour++;
    }
}
```

The right window is titled 'C:\...\tsk_task3.c' and contains the following code:

```
TASK(TASK3)
{
    unsigned int i;
    TRISBbits.TRISB1 = 1;

    while(1)
    {
        while(PORTBbits.RB1 == 0);
        for (i = 0; i < 10000; i++)
            Nop();
        if (PORTBbits.RB1 == 1)
            min++;
    }
}
```

- ☞ Добавьте задачу **Task3**, такую же, как **Task2** и с тем же приоритетом.
- ☞ Наблюдайте за последовательностью остановок.





ТРАССИРОВКА В СИМУЛЯТОРЕ

The screenshot shows a simulator interface with two main windows. The left window, titled 'C:\...\tsk_task2.c', displays the following C code:

```
TASK(TASK2)
{
    unsigned int i;
    TRISBbits.TRISB0 = 1;

    while(1)
    {
        while(PORTBbits.RB0 == 0);
        for (i = 0; i < 10000; i++)
            Nop();
        if (PORTBbits.RB0 == 1)
            hour++;
    }
}
```

The right window, titled 'Watch', displays a table of variables being monitored:

Address	Symbol Name	Value
0F81	PORTB	0x01
0322	global_counter	37
0333	sec	0
0332	min	0
0331	hour	0

At the bottom of the Watch window, there are four tabs labeled 'Watch 1', 'Watch 2', 'Watch 3', and 'Watch 4'.

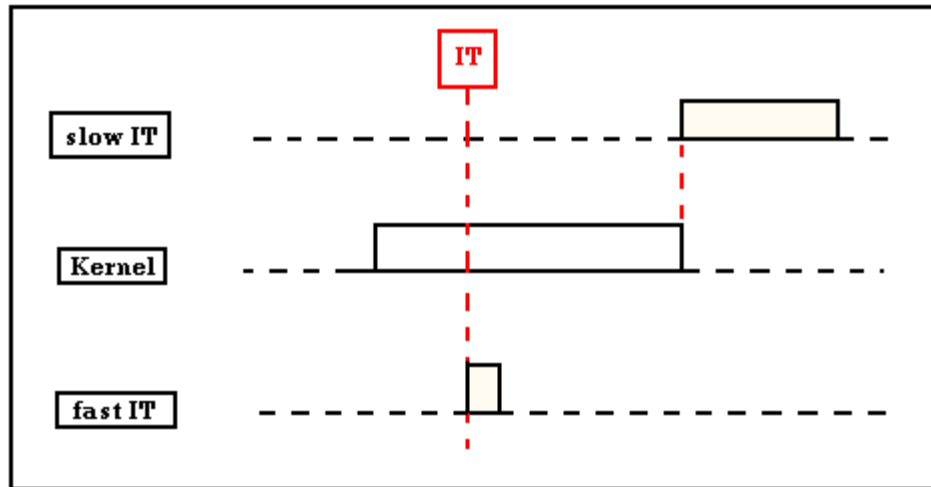
☛ Установите точки останова в разных задачах и проконтролируйте моменты остановок





ПРЕРЫВАНИЯ

Особенности файла int.c





ОСОБЕННОСТИ ФАЙЛА INT.C

```
/* *****  
 * Function you want to call when an IT occurs.  
 * ***** */  
extern void AddOneTick(void);  
/*extern void MyOwnISR(void); */  
void InterruptVectorL(void);  
void InterruptVectorH(void);  
/* *****  
 * General interrupt vector. Do not modify.  
 * ***** */  
#pragma code IT_vector_low=0x18  
void Interrupt_low_vec(void)  
{  
    _asm goto InterruptVectorL _endasm  
}  
#pragma code  
  
#pragma code IT_vector_high=0x08  
void Interrupt_high_vec(void)  
{  
    _asm goto InterruptVectorH _endasm  
}  
#pragma code
```





ОСОБЕННОСТИ ФАЙЛА INT.C

```
/*
*****
* General ISR router. Complete the function core with the if or switch
* case you need to jump to the function dedicated to the occurring IT.
*****
#pragma code _INTERRUPT_VECTORL = 0x003000
void InterruptVectorL(void)
{
    SAVE_TASK_CTX(stack_low, stack_high);
    EnterISR();

    if (INTCONbits.TMR0IF == 1)
        AddOneTick();
    /*Here is the next interrupts you want to manage */
    /*if (INTCONbits.RBIF == 1) */
    /* MyOwnISR(); */

    LeaveISR();
    RESTORE_TASK_CTX;
}
#pragma code

/* BE CARREFULL : ONLY BSR, WREG AND STATUS REGISTERS ARE SAVED */
/* DO NOT CALL ANY FUNCTION AND USE PLEASE VERY SIMPLE CODE LIKE */
/* VARIABLE OR FLAG SETTINGS. CHECK THE ASM CODE PRODUCED BY C18 */
/* IN THE LST FILE. */
#pragma code _INTERRUPT_VECTORH = 0x003300
#pragma interrupt InterruptVectorH
void InterruptVectorH(void)
{
    if (INTCONbits.INT0IF == 1)
        INTCONbits.INT0IF = 0;
}
#pragma code
```





ПРЕРЫВАНИЙ»

Address	Symbol Name	Value
0F81	PORTB	0x00
0322	global_counter	1
0333	sec	0
0332	min	0
0331	hour	0
0FF2	INTCON	0x72
0FF1	INTCON2	0x00
0FF0	INTCON3	0xC0

[INTCON3] INT2IP INT1IP - INT2IE INT1IE - INT2IF INT1IF
1 1 - 0 0 - 0 0

Режим «Медленных прерываний»

```
void Init(void)
{
    FSR0H = 0;
    FSR0L = 0;

    /* User setting : actual PIC frequency */
    Tmr0.lt = _40MHZ;

    /* Timer OFF - Enabled by Kernel */
    TOCON = 0x08;
    THROH = Tmr0.bt[1];
    THROL = Tmr0.bt[0];

    INTCON3bits.INT1IP = 0;
    INTCON3bits.INT2IP = 0;
    INTCON3bits.INT1IE = 1;
    INTCON3bits.INT2IE = 1;
    TRISBbits.TRISB1 = 1;
    TRISBbits.TRISB2 = 1;
}
```





МОДИФИКАЦИЯ ПРЕРЫВАНИЯ

```
/* *****  
 * Function you want to call when an IT occurs.  
 * ***** */  
extern void AddOneTick(void);  
extern void MyOwnISR(void);  
void InterruptVectorL(void);  
void InterruptVectorH(void);  
  
...  
/* *****  
 * General ISR router. Complete the function core with the if or switch  
 * case you need to jump to the function dedicated to the occurring IT.  
 * ***** */  
#pragma code _INTERRUPT_VECTORL = 0x003000  
void InterruptVectorL(void)  
{  
    SAVE_TASK_CTX(stack_low, stack_high);  
    EnterISR();  
    if (INTCONbits.TMR0IF == 1)  
        AddOneTick();  
/*Here is the next interrupts you want to manage */  
    if (INTCON3bits.INT1IF || INTCON3bits.INT2IF)  
        MyOwnISR();  
    LeaveISR();  
    RESTORE_TASK_CTX;  
}  
#pragma code
```

☛ Добавьте обработку прерываний от INT1 и INT2





ОБРАБОТЧИК ПРЕРЫВАНИЯ

```
void MyOwnISR(void)
{
    if (INTCON3bits.INT1IF)
    {
        INTCON3bits.INT1IF = 0;
        /* ... */
    }
    if (INTCON3bits.INT2IF)
    {
        INTCON3bits.INT2IF = 0;
        /* ... */
    }
}
```

```
#include "define.h"
void MyOwnISR(void);
/*****
 * Definition dedicated to the local functions.
 *****/
#define ALARM_TSK0      0
```

☞ Добавьте функцию обработки прерывания





ЗАДАЧАМИ

```
TASK(TASK2)
{
    unsigned int i;
    for (i = 0; i < 10000; i++)
        Nop();
    if (PORTBbits.RB1 == 1)
        hour++;
    TerminateTask();
}

TASK(TASK3)
{
    unsigned int i;
    for (i = 0; i < 10000; i++)
        Nop();
    if (PORTBbits.RB2 == 1)
        min++;
    TerminateTask();
}

/*****
 * ----- task 2 -----
 *****/
rom_desc_tsk rom_desc_task2 = {
    TASK2_PRIO,          /* prioinit from 0 to 15      */
    stack2,             /* stack address (16 bits)    */
    TASK2,              /* start address (16 bits)    */
    SUSPENDED,         /* state at init phase       */
    TASK2_ID,           /* id_tsk from 1 to 15       */
    sizeof(stack2)      /* stack size (16 bits)      */
};

/*****
 * ----- task 3 -----
 *****/
rom_desc_tsk rom_desc_task3 = {
    TASK3_PRIO,          /* prioinit from 0 to 15      */
    stack3,             /* stack address (16 bits)    */
    TASK3,              /* start address (16 bits)    */
    SUSPENDED,         /* state at init phase       */
    TASK3_ID,           /* id_tsk from 1 to 15       */
    sizeof(stack2)      /* stack size (16 bits)      */
};
```

☛ Измените задачи Task2 и Task3 и их описание





ЗАДАЧАМИ

```
void MyOwnISR(void)
{
    TaskStateType State;
    if (INTCON3bits.INT1IF)
    {
        INTCON3bits.INT1IF = 0;
        GetTaskState(TASK2_ID, &State);
        if (State == SUSPENDED)
            ActivateTask(TASK2_ID);
    }
    if (INTCON3bits.INT2IF)
    {
        INTCON3bits.INT2IF = 0;
        GetTaskState(TASK3_ID, &State);
        if (State == SUSPENDED)
            ActivateTask(TASK3_ID);
    }
}
```

☛ Используя сервисы, измените состояние задач





ЗАДАЧАМИ

The screenshot shows a debugger interface with two source code windows and a watch window.

Left Window (C:\...\tsk_task2.c):

```
if (INTCON3bits.INT1IF)
{
    INTCON3bits.INT1IF = 0;
    GetTaskState(TASK2_ID, &State);
    if (State == SUSPENDED)
        ActivateTask(TASK2_ID);
}
if (INTCON3bits.INT2IF)
{
    INTCON3bits.INT2IF = 0;
    GetTaskState(TASK3_ID, &State);
    if (State == SUSPENDED)
        ActivateTask(TASK3_ID);
}
/* End of File : tsk_task0.c */
```

Right Window (C:\...\tsk_task3.c):

```
TASK(TASK3)
{
    unsigned int i;
    for (i = 0; i < 10000; i++)
        Nop();
    if (PORTEbits.RB2 == 1)
        min++;
    TerminateTask();
}
```

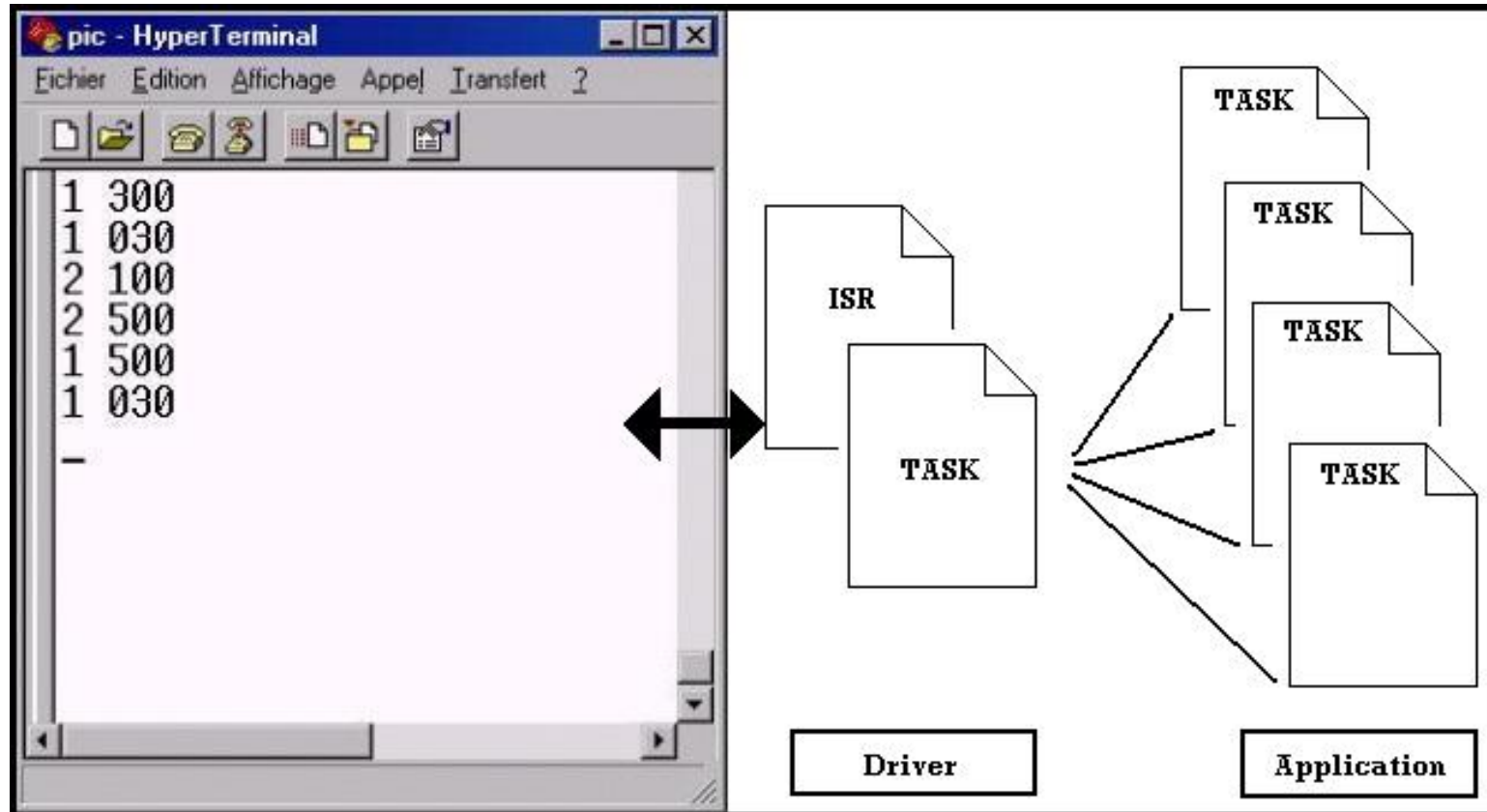
Watch Window:

Address	Symbol Name	Value
0FF0	INTCON3	0x1A

☞ Проследите, что задачи выполняются



Особенности драйверов



- ❖ Драйвер обычно состоит из **ISR (Interrupt Service Routine, Обработчик Прерывания)** и задачи.



> Drivers			
	<i>Version</i>	<i>Document</i>	<i>Sources</i>
LCD HD4478	v1.01	<u>TXT</u>	
Librairie HD4478	en cours...		
Driver I2C (master)	v1.03	<u>TXT</u>	
Driver I2C (slave)	en cours...		
Driver CAN	v1.00	<u>TXT</u>	
DS1337 (RTC)	en cours...		
DS1624 (T°)	en cours...		
Driver RS232	v1.02	<u>TXT</u>	

- ☛ Драйверы некоторых устройств и периферии доступны на сайте.





ИСПОЛЬЗОВАНИЕ ДРАЙВЕРА

```
#include "define.h"
#include "drv_rs.h"
#include <stdio.h>
#include <string.h>
#define ALARM_TSK4 1
int Printf (const rom char *fmt, ...);
extern unsigned char hour;
extern unsigned char min;
extern unsigned char sec;
/*****
 * Definition dedicated to the local functions.
 *****/
RS_message_t RS_msg;
unsigned char buffer[80];
/*****
 * ----- TASK4 -----
 *
 * Fifth task of the tutorial.
 *****/
TASK(TASK4)
{
    SetRelAlarm(ALARM_TSK4, 1000, 1000);
    Printf(" _____ \r\n");
    Printf("> <\r\n");
    Printf("                PICos18 Tutorial          <\r\n");
    Printf("> <\r\n");
    Printf("                PICos18 - Real-time kernel for PIC18 family <\r\n");
    Printf("> <\r\n");
    Printf("> <\r\n");
    Printf("> www.picos18.com                www.pragmatec.net <\r\n");
    Printf("> _____ <\r\n");
    Printf(" \r\n");
    Printf(" \r\n");
    while(1)
    {
        WaitEvent(ALARM_EVENT);
        ClearEvent(ALARM_EVENT);
        Printf("%02d : %02d : %02d\r", (int)hour, (int)min, (int)sec);
    }
}
/*****
 * Function in charge of structure registration and buffer transmission.
 *
 * @param string IN const string send to the USART port
 * @return void
 *****/
int Printf (const rom char *fmt, ...)
{
    va_list ap;
    int n;
    RS_engMsg(&RS_msg, buffer, 50);
    va_start (ap, fmt);
    n = vprintf (_H_USER, fmt, ap);
    va_end (ap);
    SetEvent(RS_DRV_ID, RS_NEW_MSG);
    WaitEvent(RS_QUEUE_EMPTY);ClearEvent(RS_QUEUE_EMPTY);
    return n;
}
```

☞ Так выглядит
текст драйвера





```
tutorial - HyperTerminal
Fichier Edition Affichage Appel Transfert ?
[Icons]
>
> PICos18 Tutorial
>
> PICos18 - Real-time kernel for PIC18 family
>
> www.picos18.com www.pragmatec.net
>
00 : 01 : 16
[Progress bar]
00:29:55 connecté Détection auto 115200 8-N-1 DÉFIL Maj Num Capturer Écho
```

☛ Результат работы драйвера





НАСТРОЙКИ HYPERTERMINAL

Propriétés de COM1

Paramètres du port

Bits par seconde : 115200

Bits de données : 8

Parité : Aucun

Bits d'arrêt : 1

Contrôle de flux : Aucun

Paramètres par défaut

OK Annuler Appliquer

Connexion à

tutorial

Entrez les détails du numéro de téléphone que vous voulez composer :

Pays/région : France (33)

Indicatif régional : 08

Numéro de téléphone :

Se connecter en utilisant : COM1

OK Annuler





ПРИМЕР ПРИЛОЖЕНИЯ

Множественные прерывания

The screenshot shows two windows from a debugger. The left window, titled 'C:\...\define.h', displays the following code:

```
/*----- Task -----*/
#define TASK0_ID 1
#define TASK1_ID 2
#define TASK2_ID 3
#define TASK3_ID 4
#define TASK4_ID 5
#define RS_DRV_ID 6
```

The right window, titled 'Watch', shows a table of watched variables:

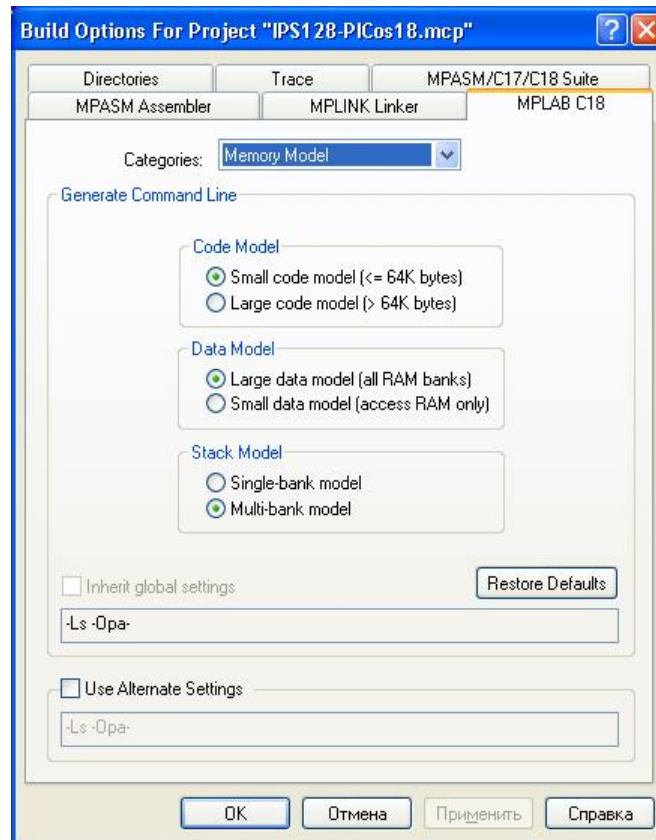
Address	Symbol Name	Value
0FF0	INTCON3	0x19
0070	kernelPanic	0x53
0300	stack2	"□"
0200	stack4	"□"

- ☛ Иногда контроллер вдруг «сходит с ума». На самом деле, происходит разрушение стеков задач.
- ☛ Переменная **kernelPanic** позволяет увидеть, какая задача явилась причиной разрушения стека





ПРОЦЕСС ОТЛАДКИ



☛ Определите модели Вашей программы



The screenshot shows the 'File Registers' window in MPLAB IDE v8.00. The window title is 'Tester128 - MPLAB IDE v8.00 - File Registers'. The menu bar includes File, Edit, View, Project, Debugger, Programmer, Tools, Configure, Window, and Help. The toolbar contains various icons, including a 'Release' dropdown. The window tabs show 'Output', 'PKP128.mcw', 'Locals', 'Watch', 'MPLAB IDE Editor', and 'File Registers'. The 'File Registers' window displays a table of memory addresses and their contents in hexadecimal and ASCII.

Addr...	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
100	DE	AD	39	01	0E	04	01	00	A0	00	0E	07	00	00	07	01	...9.....
110	A0	00	00	57	8A	03	00	0A	00	00	00	00	00	00	00	00	...W.....
120	00	00	00	00	00	00	00	00	00	00	80	00	00	00	00	00
130	00	00	4E	1B	00	A4	3E	00	02	A5	A5	A5	A5	A5	A5	A5	...N...>..
140	A5	A5	A5	A5	A5	A5	A5	A5	A5	A5	A5	A5	A5	A5	A5	A5
150	A5	A5	A5	A5	A5	A5	A5	A5	A5	A5	A5	A5	A5	A5	DE	AD
160	DE	AD	93	01	64	01	00	00	00	0E	04	66	01	64	01	00	...d...f.d..

At the bottom of the window, there are tabs for 'Hex' and 'Symbolic'. The status bar at the bottom shows 'MPLAB SIM', 'PIC18F2580', 'pc:0x4e3c', 'W:0x80', 'N ov z dc c', and '20 MHz b:...'.

☛ **Стек задач инициализируется кодом а5. Это позволяет оптимизировать использование памяти.**



ЗАКЛЮЧЕНИЕ

PICos18 – это:

- ☞ Современное решение задач для встроенных систем на базе **PIC18** по стандарту **OSEK/VDX**
- ☞ Полностью отлаженное ядро, начиная с **v3.00b4**
- ☞ Большое число драйверов периферии и устройств
- ☞ Разработка и отладка в **MPLAB + C18 (Student Edition)** с использованием стандартных отладчиков: **ICD2, ICD3, PICkit2, PICkit3, RealICE**
- ☞ Документация на английском, французском, португальском и русском языках.
- ☞ Реально работающие обучающие примеры
- ☞ Доступность и бесплатность как пакета **PICos18**, так и среды разработки
- ☞ Сайт и форум разработчиков www.picos18.com

