ОПЕРАЦИОННАЯ СИСТЕМА РЕАЛЬНОГО ВРЕМЕНИ PICOS18

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 балщой, паласатый мух!
— Нэт! Балщой паласатый мух – эта шмел, а «ОС» - эта то, на чем вэртитса 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?

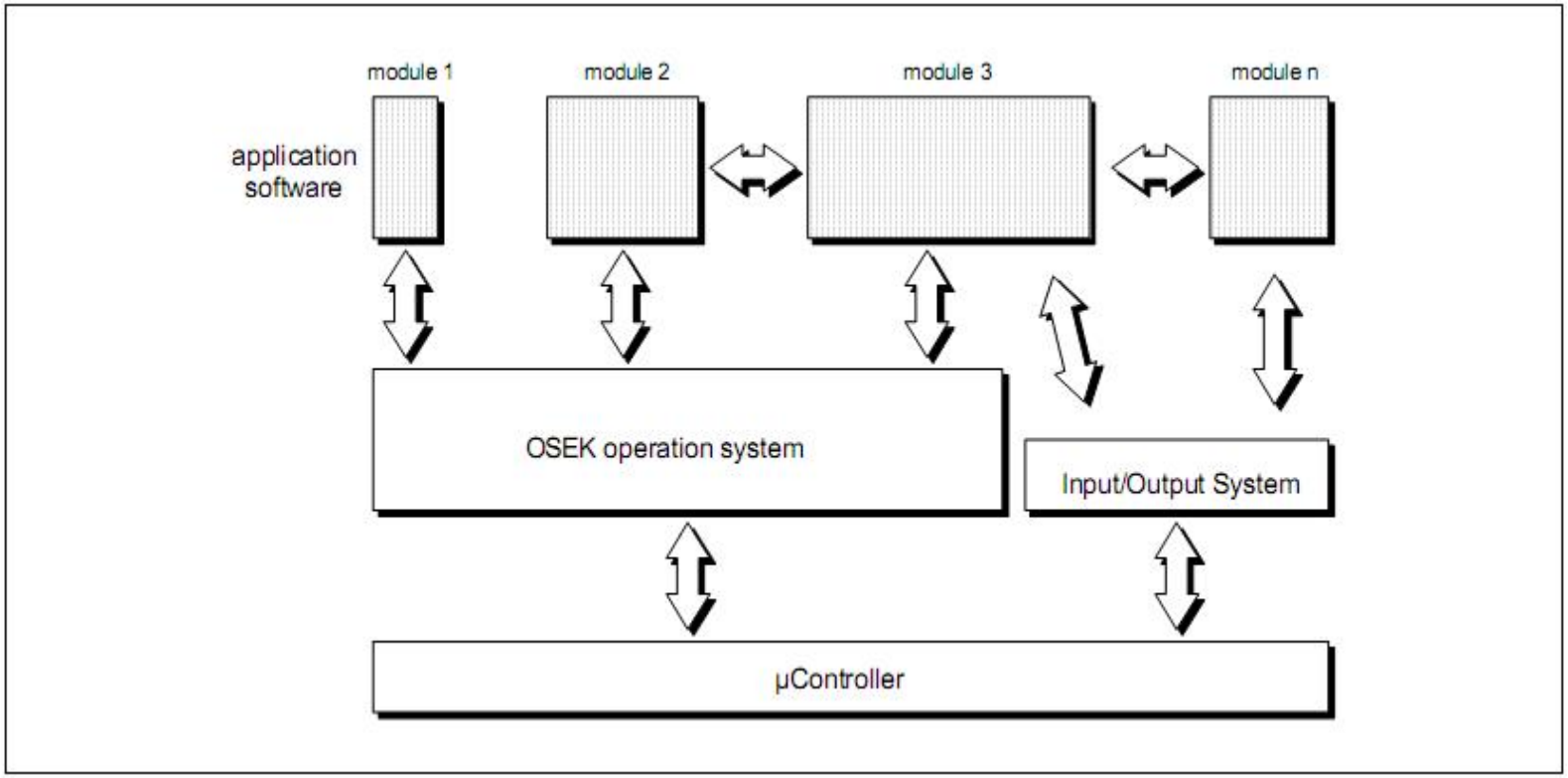
*По **объему** занимаемого **ОЗУ** (в сторону увеличения):*

- *без вытеснения*
- *с вытеснением*

PIC-КОНТРОЛЛЕРЫ И ОС

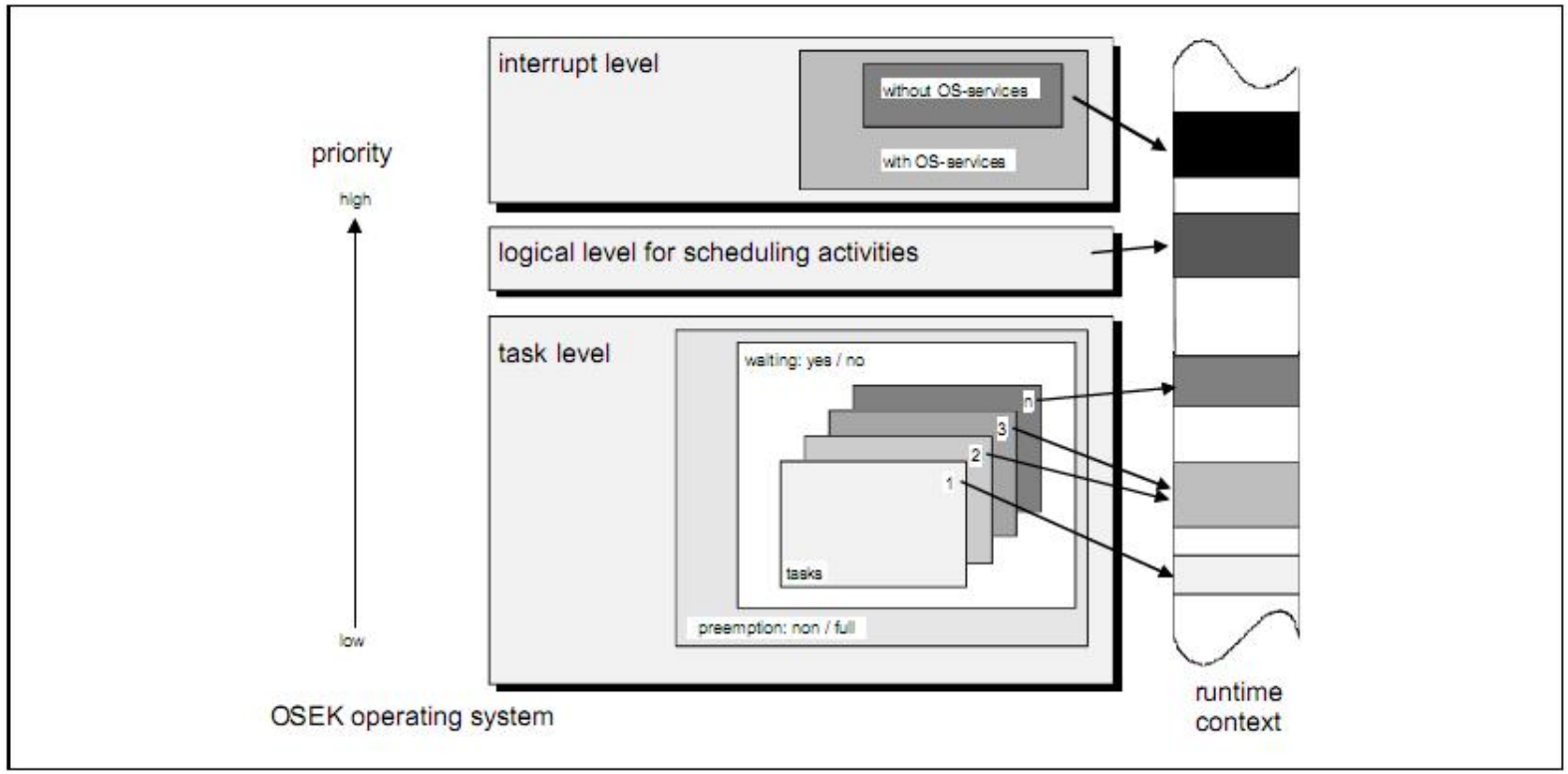
- *PIC10, PIC12* - *нет возможности*
- PIC16 - только кооперативные
- **PIC18** - **все типы**
- **PIC24, PIC30, PIC33** - **все типы**
- **PIC32** - **все типы**

СИСТЕМНАЯ ФИЛОСОФИЯ OSEK/VDX

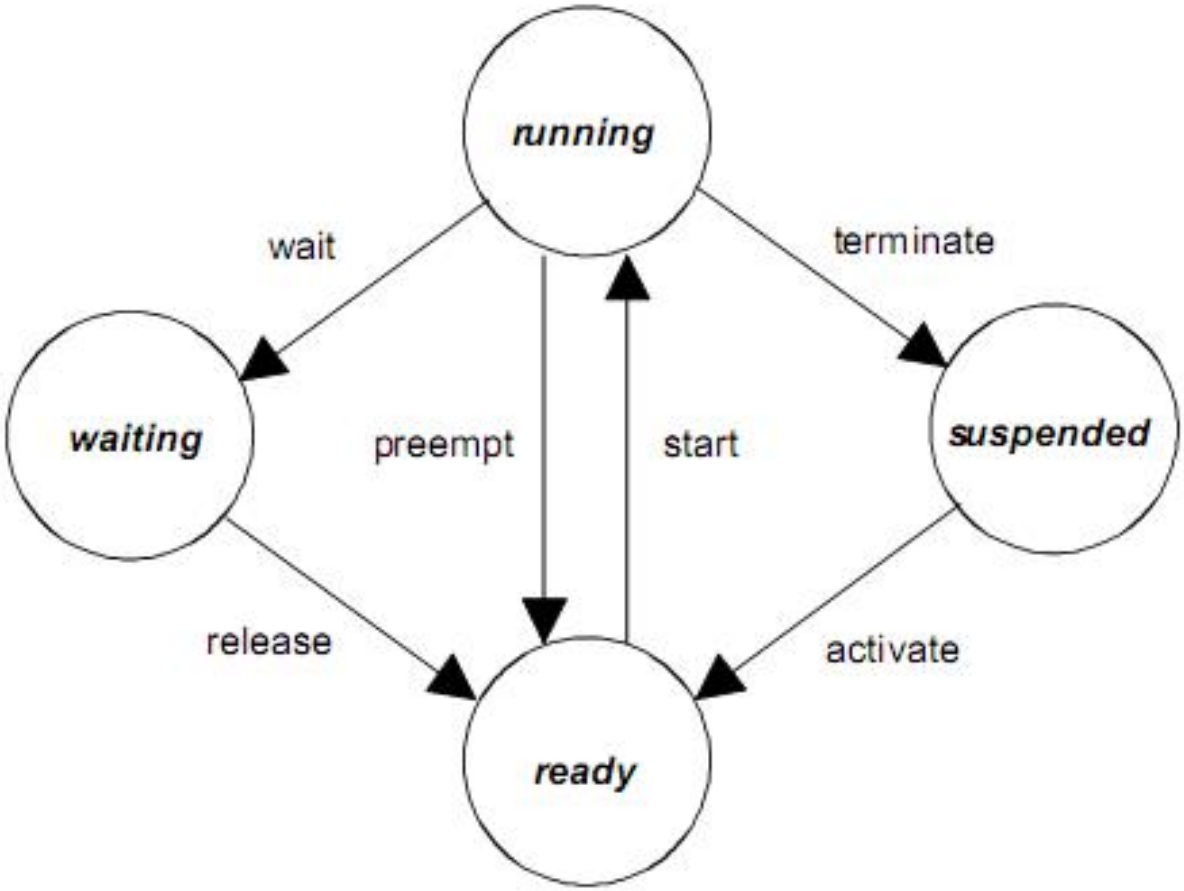


- Программные интерфейсы встроенной системы

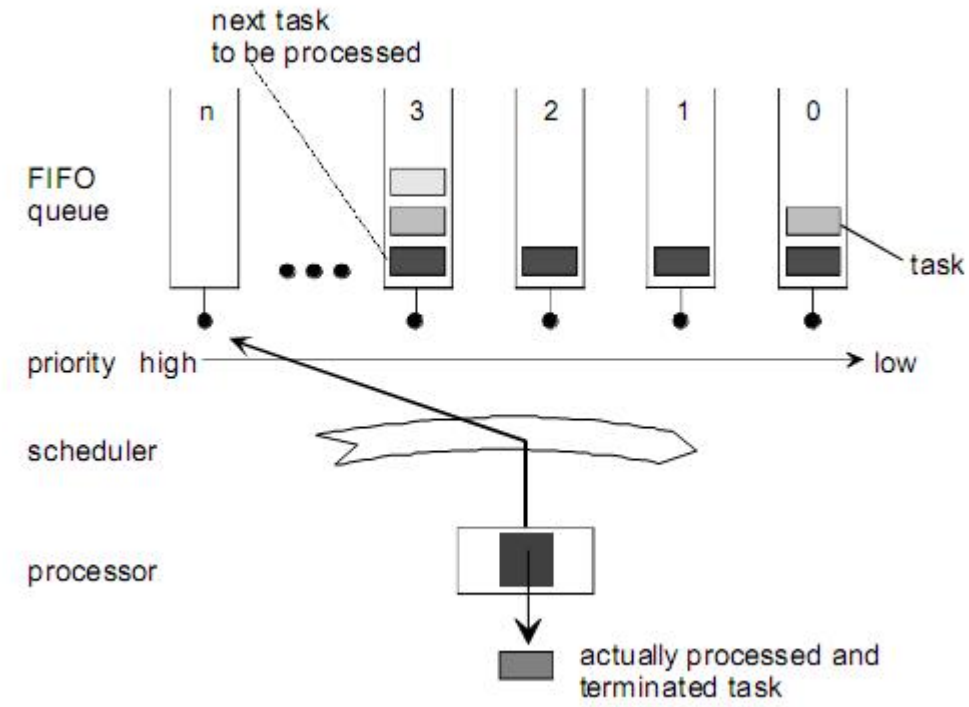
АРХИТЕКТУРА ОС

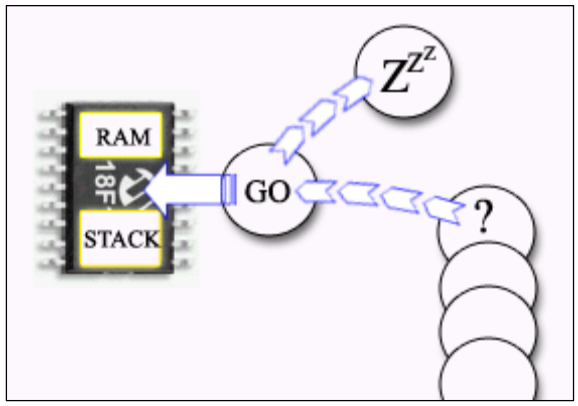


МОДЕЛЬ СОСТОЯНИЙ ЗАДАЧ



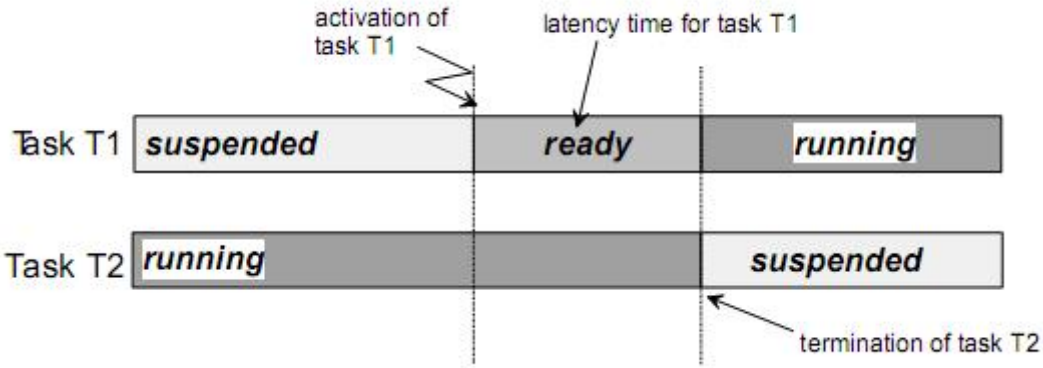
ПЛАНИРОВЩИК: ПОРЯДОК СОБЫТИЙ



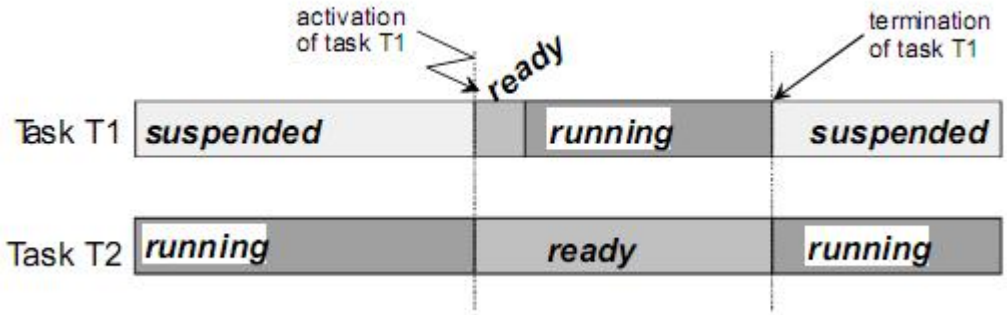


•Планировщик ищет задачи с состояниями ready/running.
•Из массива задач с состояниями ready/running планировщик выделяет набор задач с наивысшим приоритетом.
•В этом наборе планировщик находит самую старую(в смысле ожидания) задачу и запускает ее на выполнение.

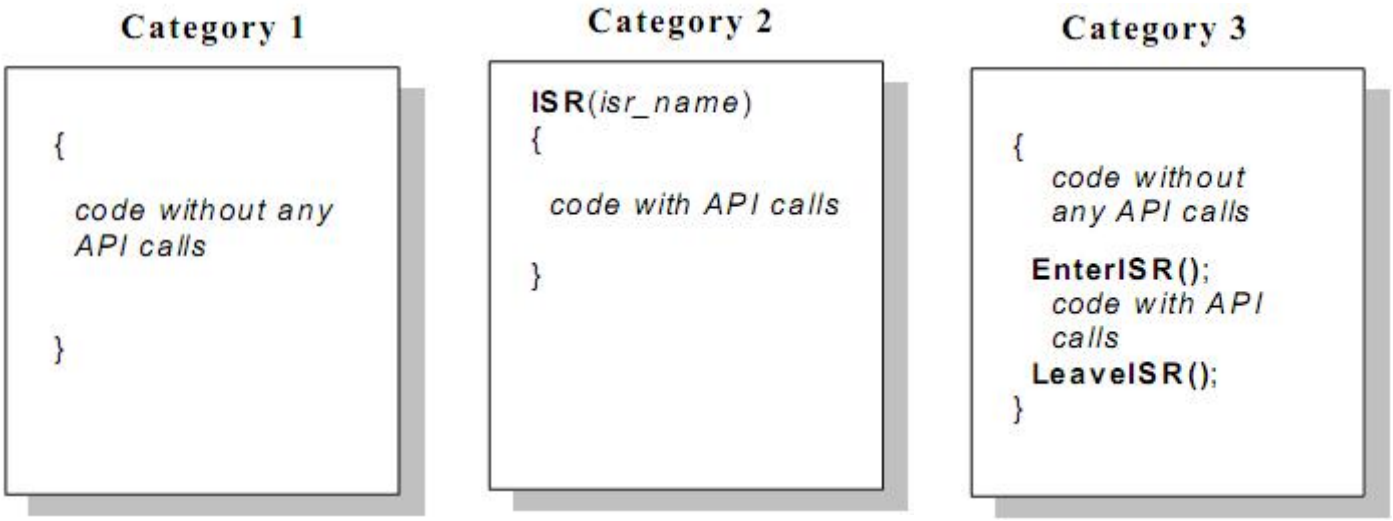
ПЛАНИРОВЩИК НЕВЫТЕСНЯЮЩЕГО ТИПА



ПЛАНИРОВЩИК ВЫТЕСНЯЮЩЕГО ТИПА



ОБСЛУЖИВАНИЕ ПРЕРЫВАНИЙ

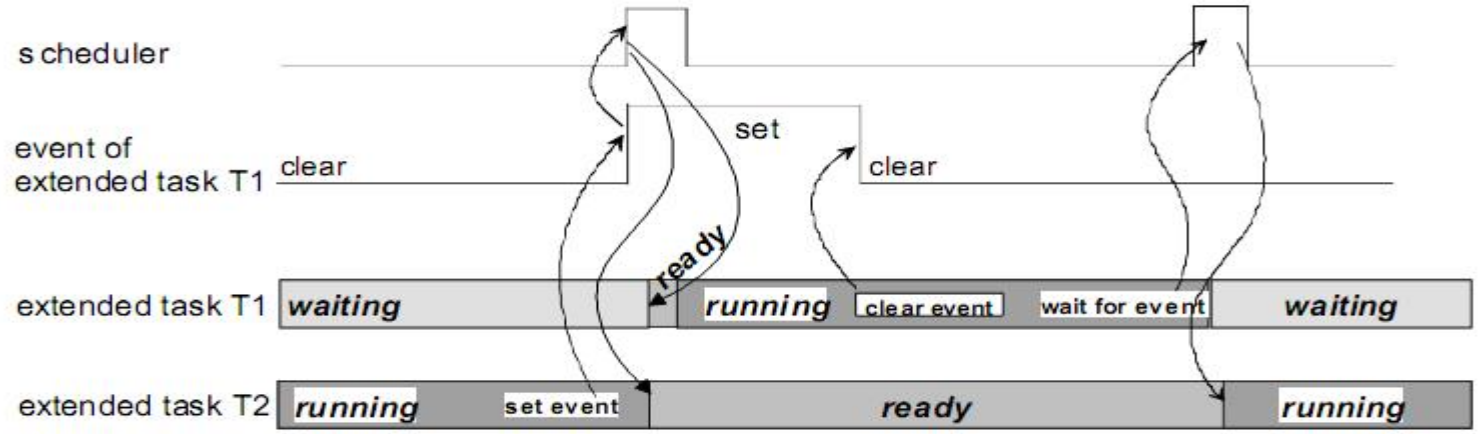


OSEK/VDX выделяет три категории обработчиков прерываний:

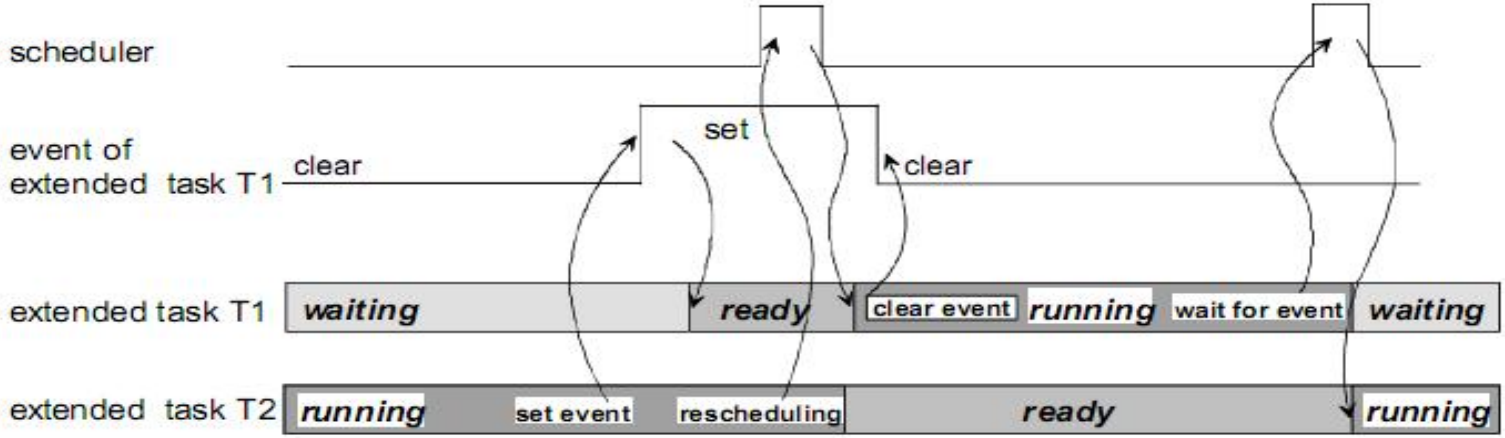
- ISR не использует сервисы ОС
- Используется предопределенный ISR как сервис ОС
- Используются сервисы ОС для входа в прерывание и выхода из него

МЕХАНИЗМ СОБЫТИЙ

СИНХРОНИЗАЦИЯ ЗАДАЧ С ВЫТЕСНЯЮЩИМ ПЛАНИРОВЩИКОМ…

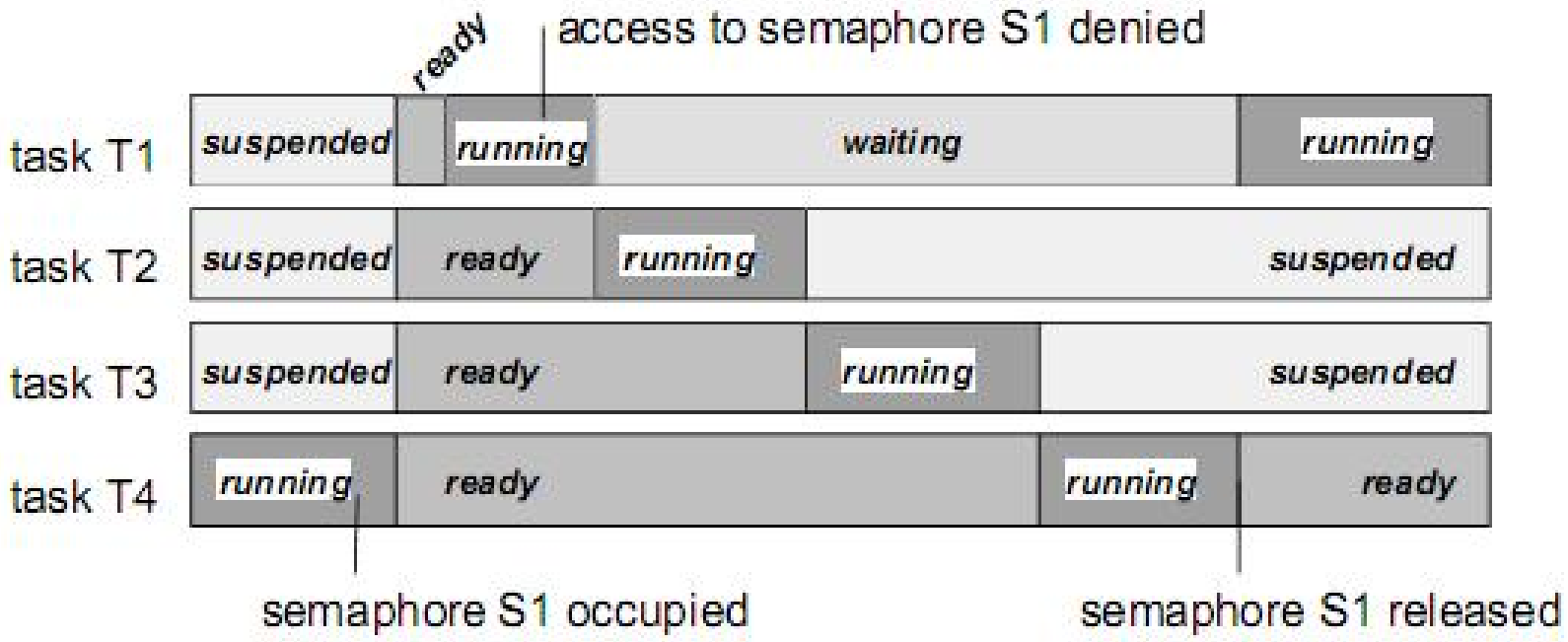


… И БЕЗ ВЫТЕСНЯЮЩЕГО ПЛАНИРОВЩИКА



УПРАВЛЕНИЕ РЕСУРСАМИ

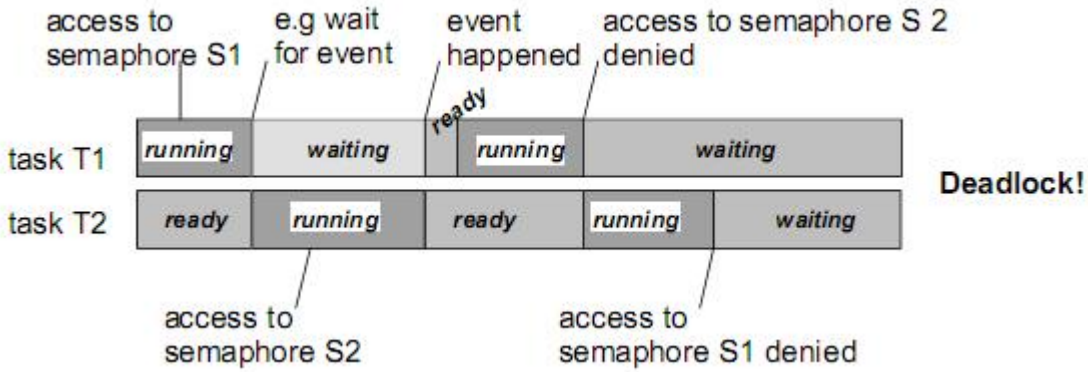
Главная проблема механизмов синхронизации



ИНВЕРСИЯ ПРИОРИТЕТОВ ПРИ ИСПОЛЬЗОВАНИИ СЕМАФОРОВ

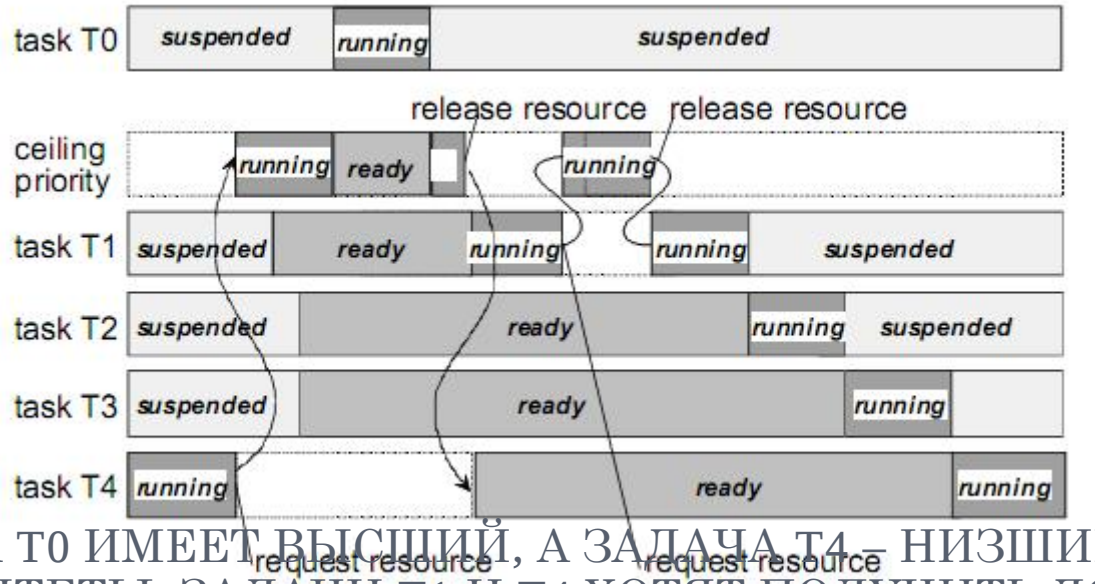
БЛОКИРОВКА СЕМАФОРОВ

Еще одна проблема механизмов синхронизации



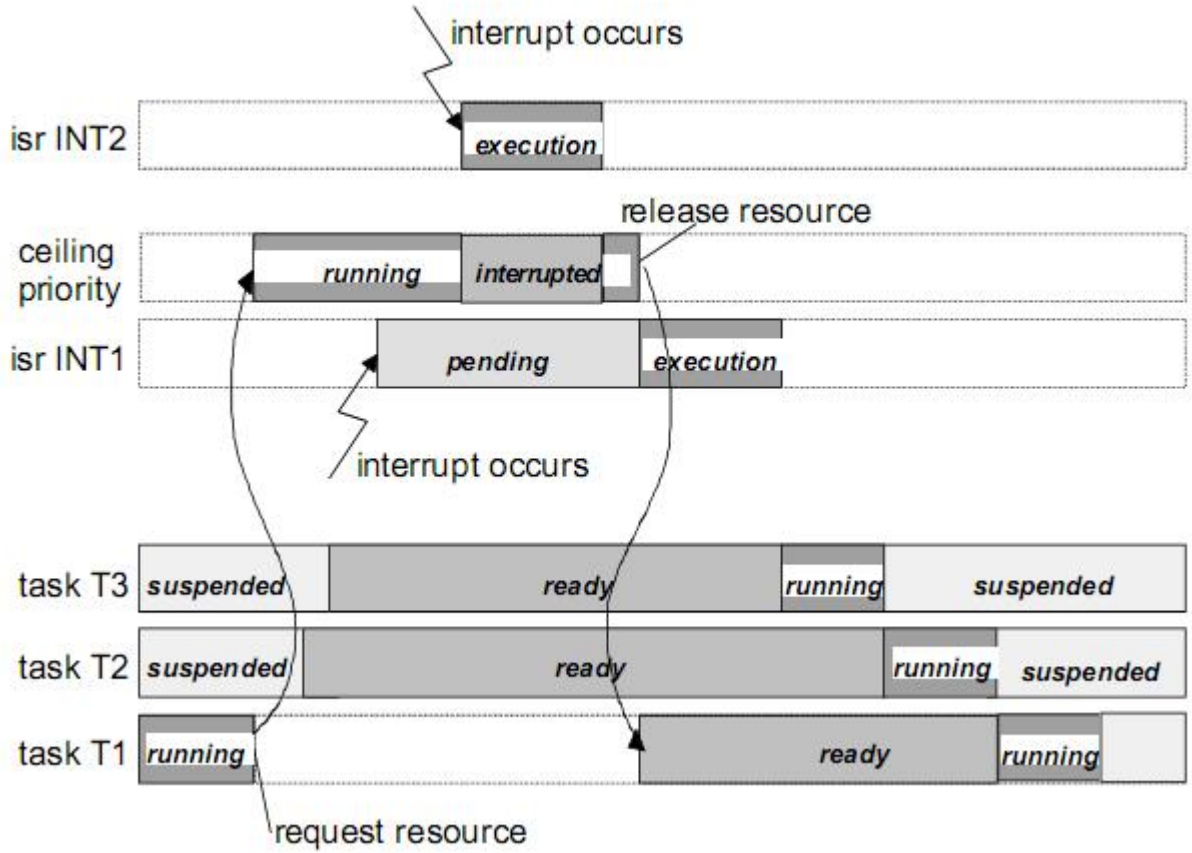
ВЗАИМНАЯ БЛОКИРОВКА ЗАДАЧ ПРИ ИСПОЛЬЗОВАНИИ СЕМАФОРОВ: ЗАДАЧА T1 ОЖИДАЕТ ДОСТУП К СЕМАФОРУ S2, ЗАНЯТЫЙ ЗАДАЧЕЙ T2, А ЗАДАЧА T2 ОЖИДАЕТ ДОСТУП К СЕМАФОРУ S1, ЗАНЯТЫЙ ЗАДАЧЕЙ T1!

РАСПРЕДЕЛЕНИЕ РЕСУРСОВ С ПРЕДЕЛЬНЫМ ПРИОРИТЕТОМ

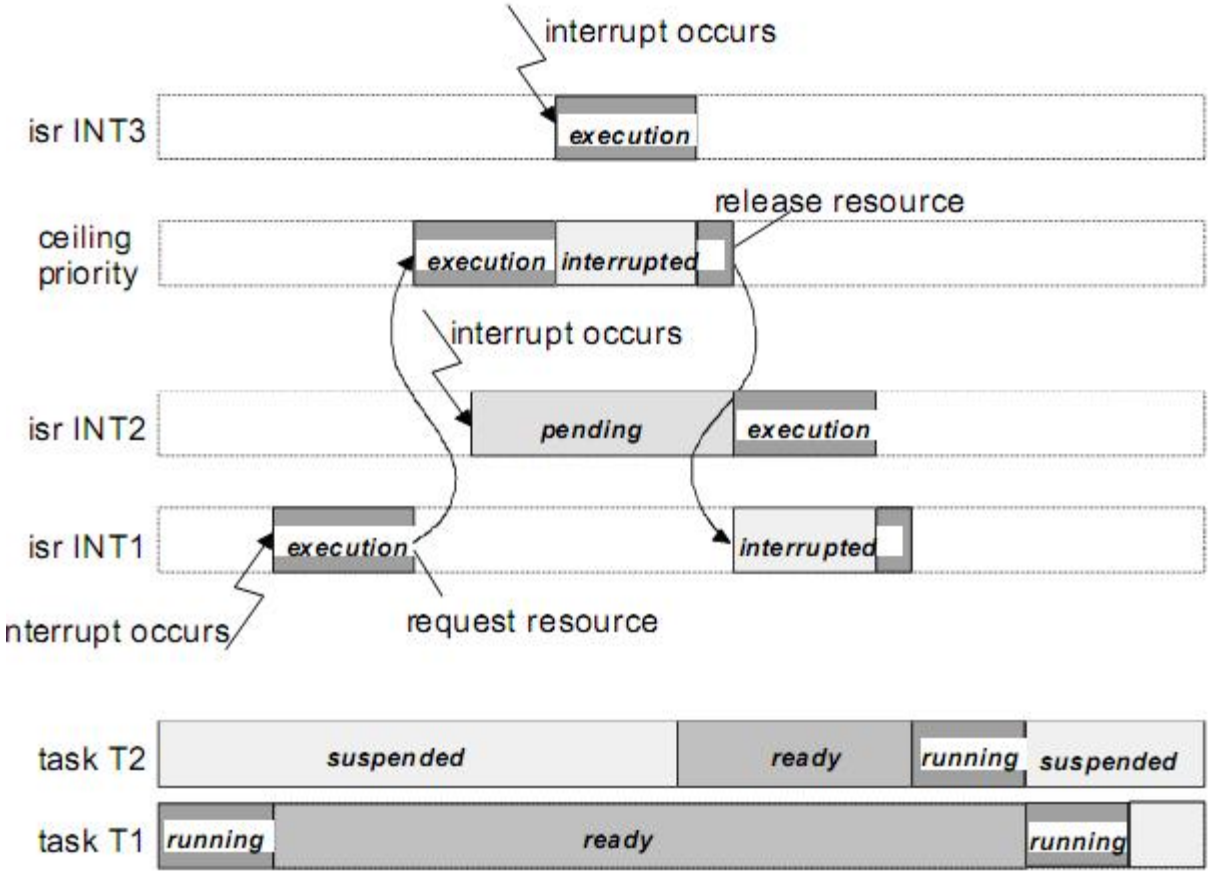


ЗАДАЧА T0 ИМЕЕТ ВЫСШИЙ, А ЗАДАЧА T4 – НИЗШИЙ ПРИОРИТЕТЫ. ЗАДАЧИ T1 И T4 ХОТЯТ ПОЛУЧИТЬ ДОСТУП К ОДНОМУ И ТОМУ ЖЕ РЕСУРСУ. НАГЛЯДНО ПОКАЗАНО, ЧТО ИНВЕРСИИ ПРИОРИТЕТОВ В ДАННОМ СЛУЧАЕ НЕ ПРОИСХОДИТ. ВЫСОКОПРИОРИТЕТНАЯ ЗАДАЧА T1 ОЖИДАЕТ МЕНЬШЕ ВРЕМЕНИ, ЧЕМ МАКСИМАЛЬНАЯ ПРОДОЛЖИТЕЛЬНОСТЬ ЗАНЯТИЯ РЕСУРСА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

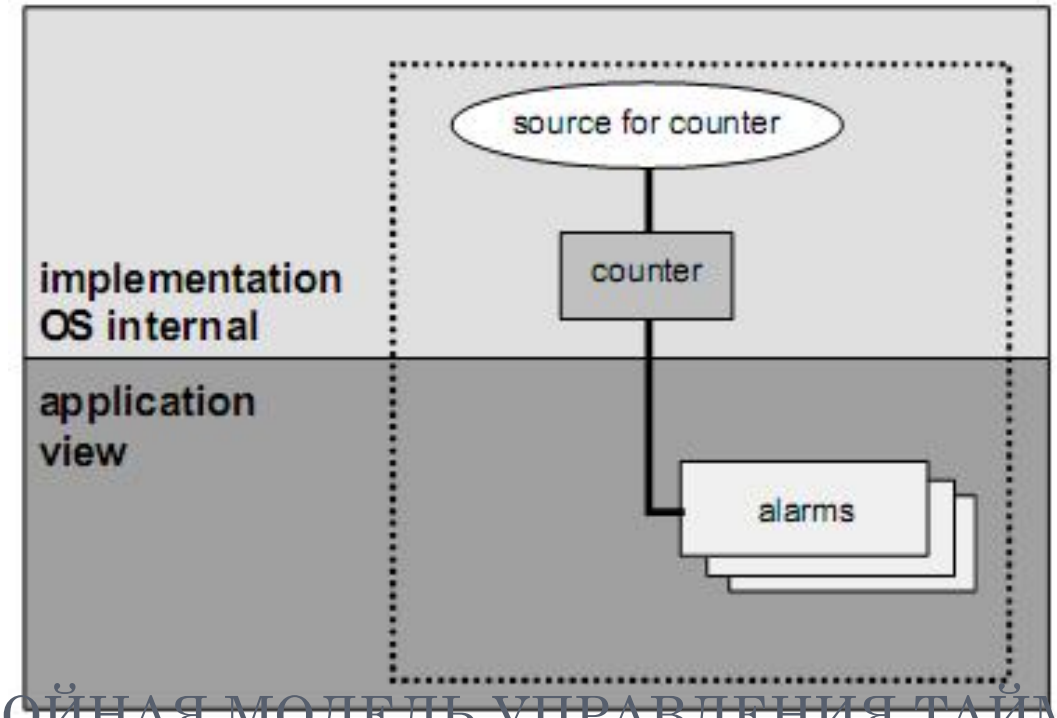
РАСПРЕДЕЛЕНИЕ РЕСУРСОВ С ПРЕДЕЛЬНЫМ ПРИОРИТЕТОМ И ОБРАБОТКА ПРЕРЫВАНИЙ



РАСПРЕДЕЛЕНИЕ РЕСУРСОВ МЕЖДУ ОБРАБОТЧИКАМИ ПРЕРЫВАНИЙ



ТАЙМЕРЫ И ТАЙМАУТЫ (ALARMS)

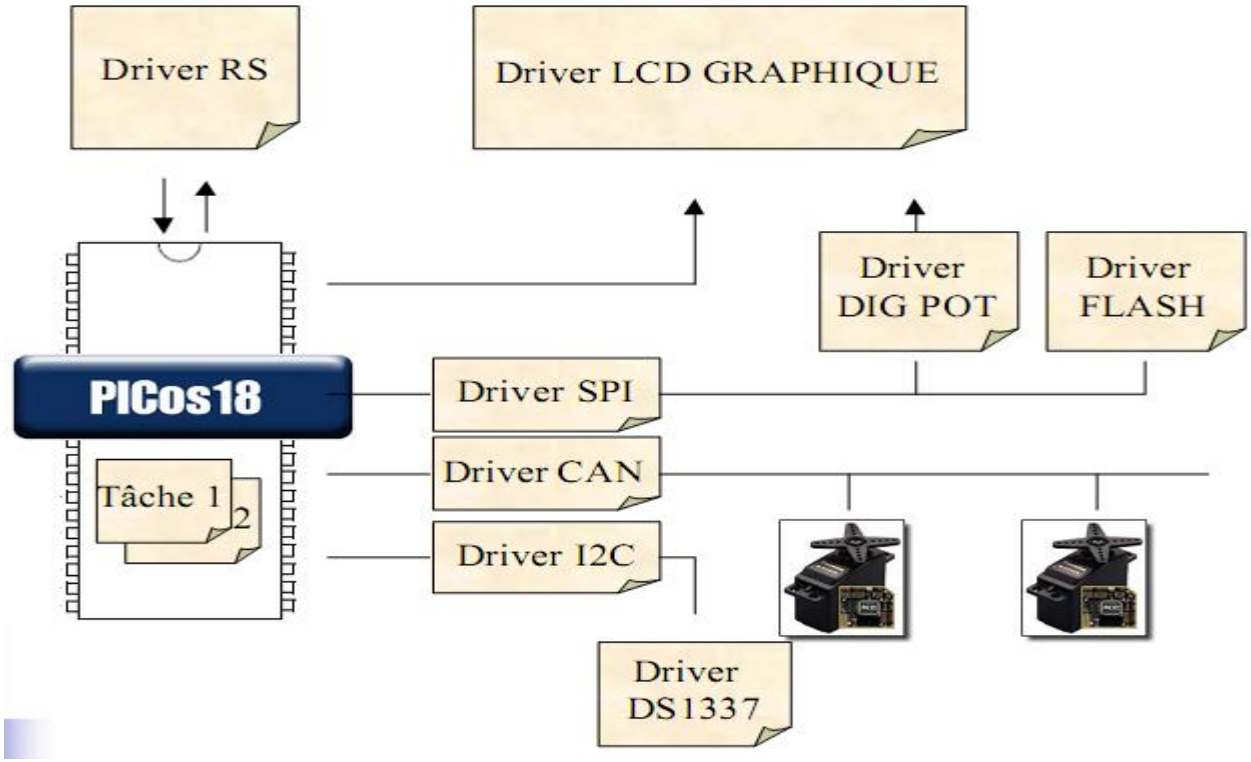


ПОСЛОЙНАЯ МОДЕЛЬ УПРАВЛЕНИЯ ТАЙМЕРАМИ И ТАЙМАУТАМИ.

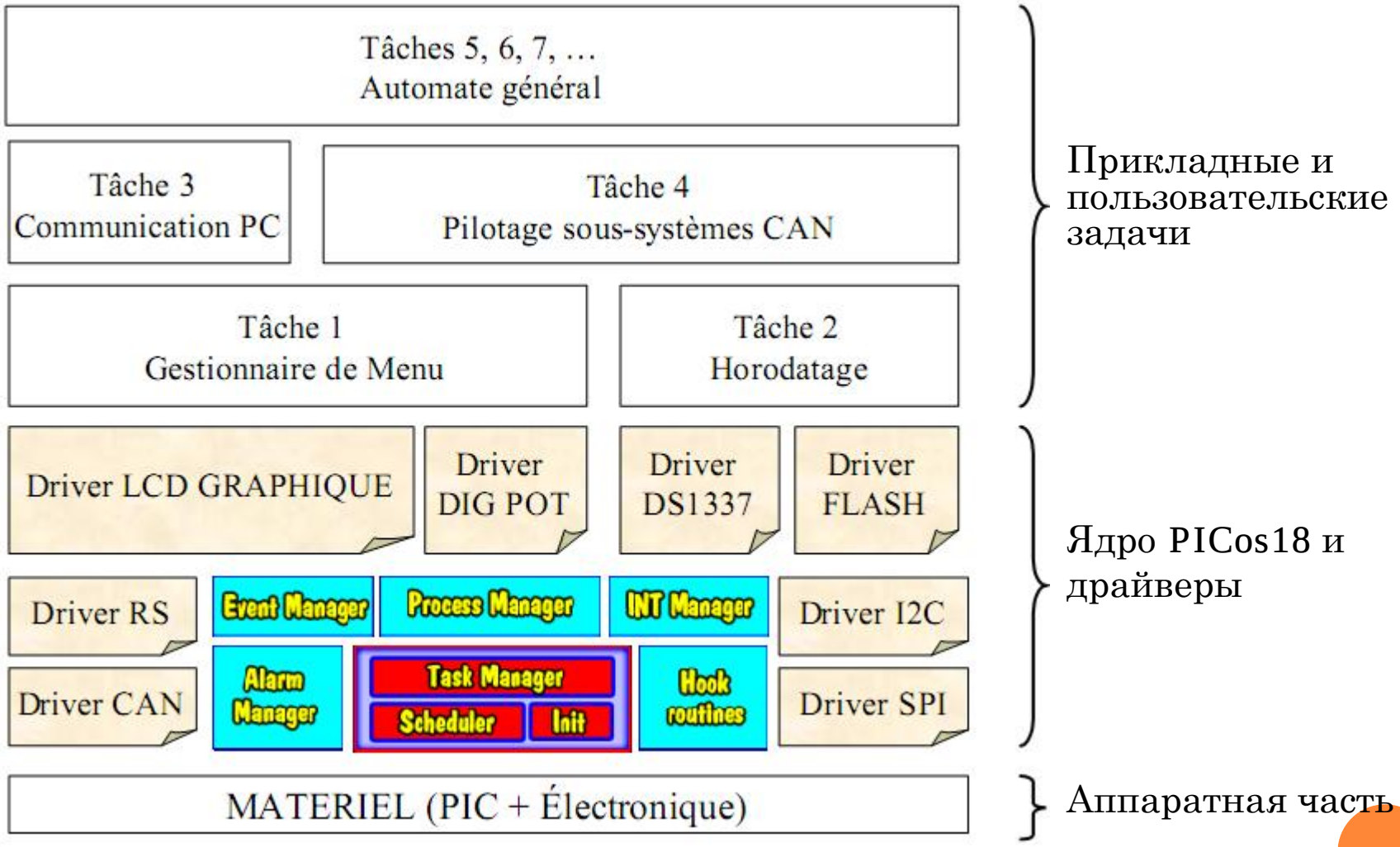
ОДИН СЧЕТЧИК ОС ОБЕСПЕЧИВАЕТ РАБОТУ ВСЕХ ТАЙМАУТОВ ВСЕХ ЗАДАЧ.

МНОГОЗАДАЧНАЯ ОС

- *Операционная система…*
- *…многозадачная…*
- *…реального времени…*



СТРУКТУРА ВСТРОЕННОИ СИСТЕМЫ



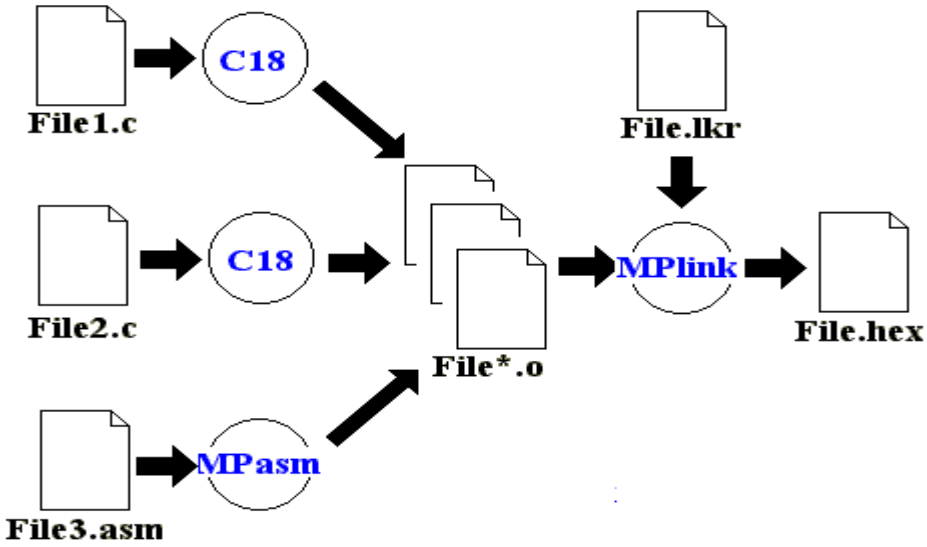
ЯДРО И СЕРВИСЫ PICOS18

Task Manager, Sheduler, Init составляют ядро ОС.

ЛИЦЕНЗИЯ_GPL


```
/*********************************************************************/
/*                                                                   */
/* 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 фалы 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

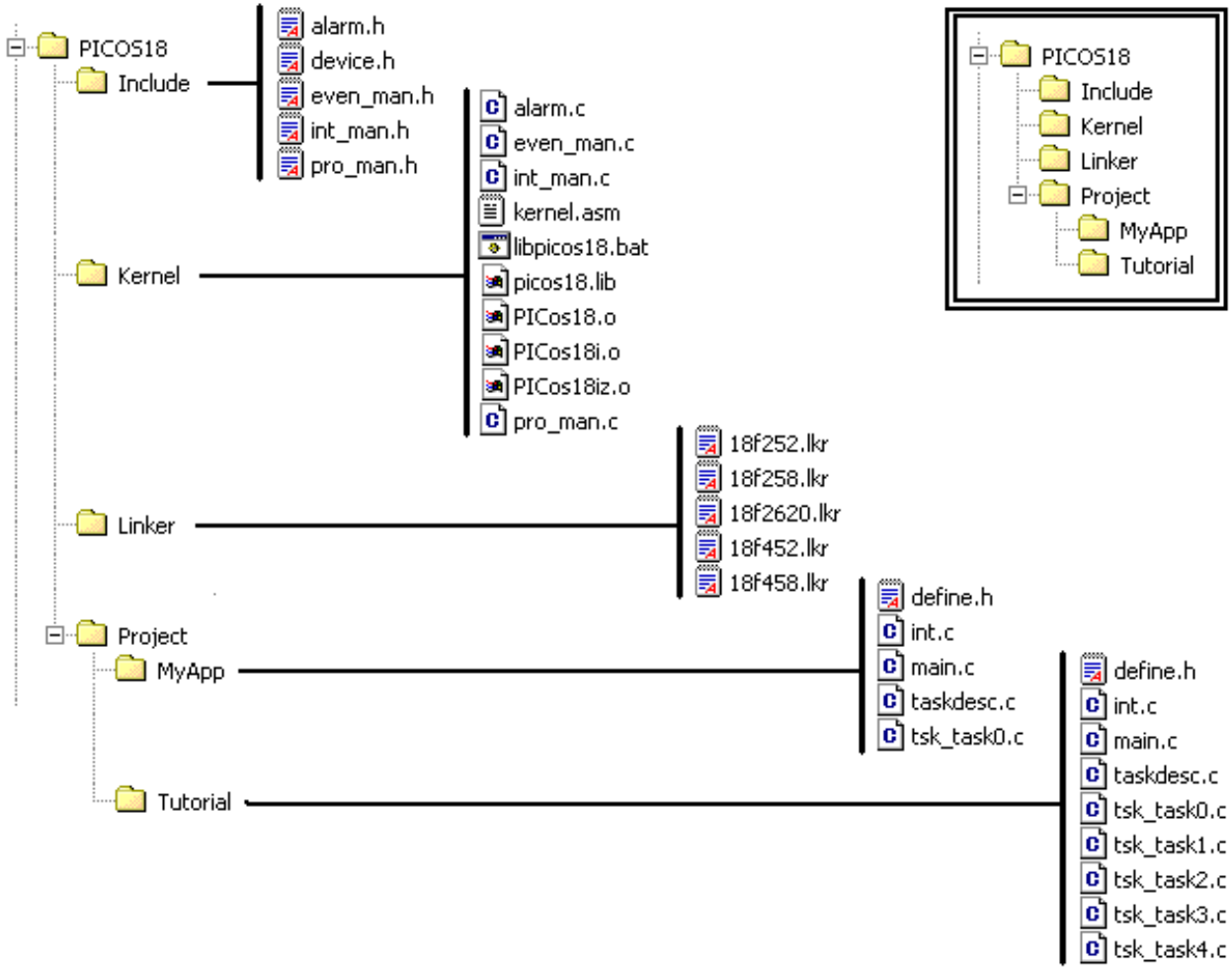
Использование драйверов

Пример приложения

СРЕДСТВА РАЗРАБОТКИ

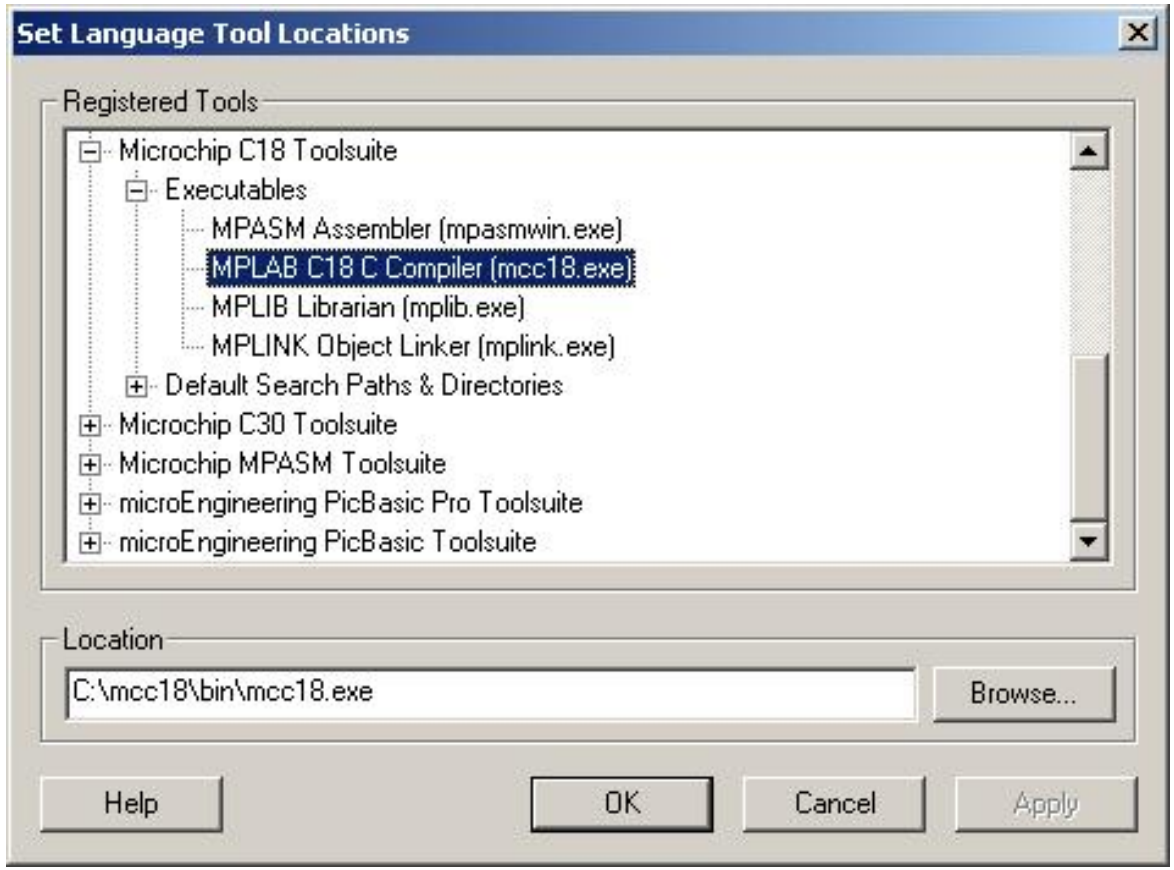
- Зайдите на сайт www.picos18.com и скачайте последнюю версию ОС.

СРЕДСТВА РАЗРАБОТКИ



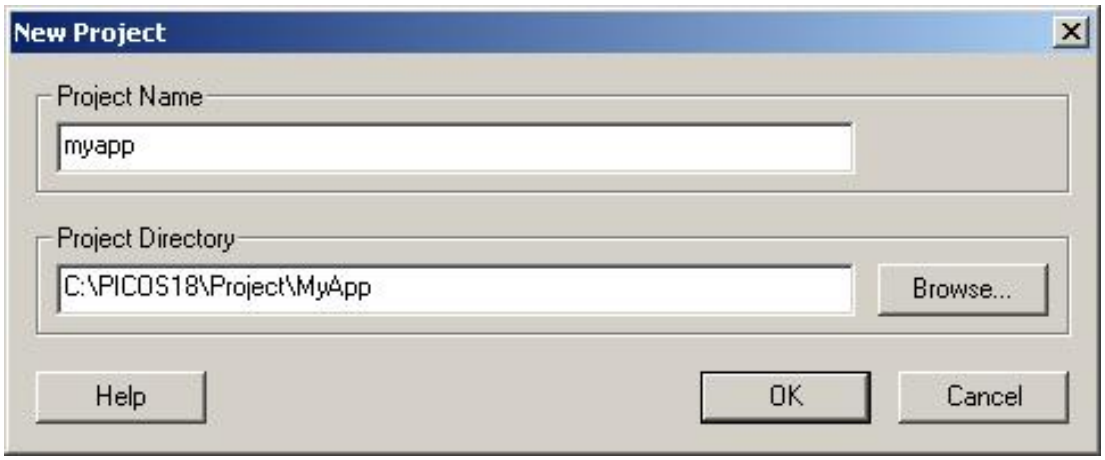
- После распаковки получится примерно такое дерево директориев с файлами ОС

КОМПИЛЯТОР C18



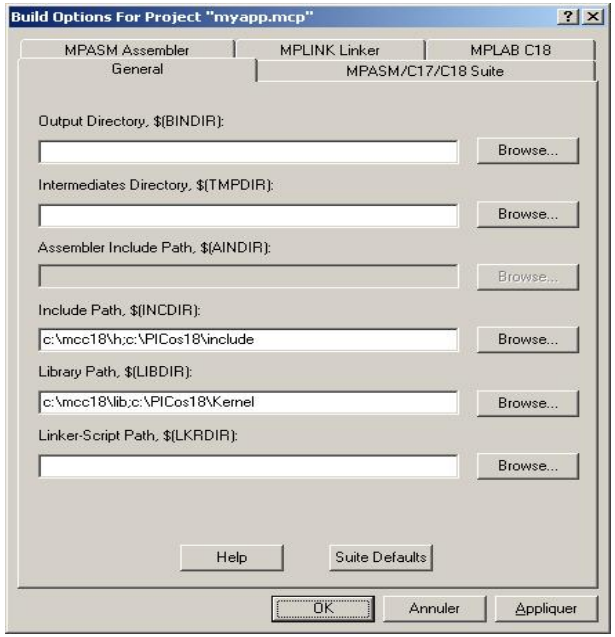
- Для компиляции проекта с использованием PICos18 необходим компилятор MPLAB C18. Student Edition также неплохо работает.

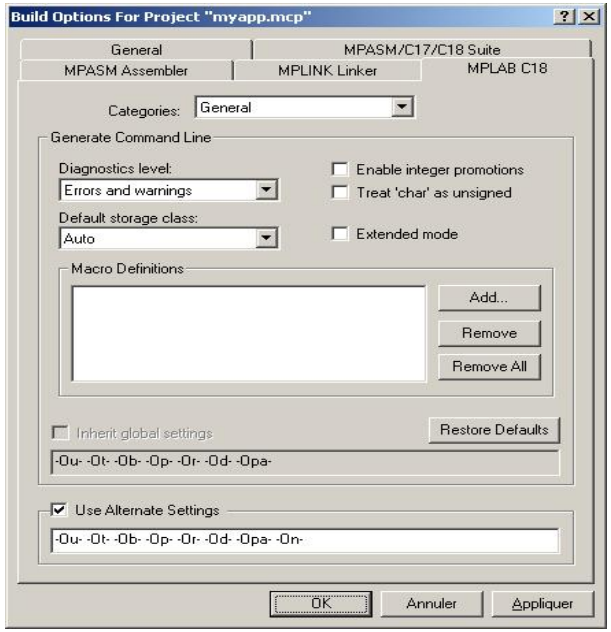
ОТКРЫТИЕ ПРОЕКТА



- Задайте имя и месторасположение Вашего проекта с PICos18.

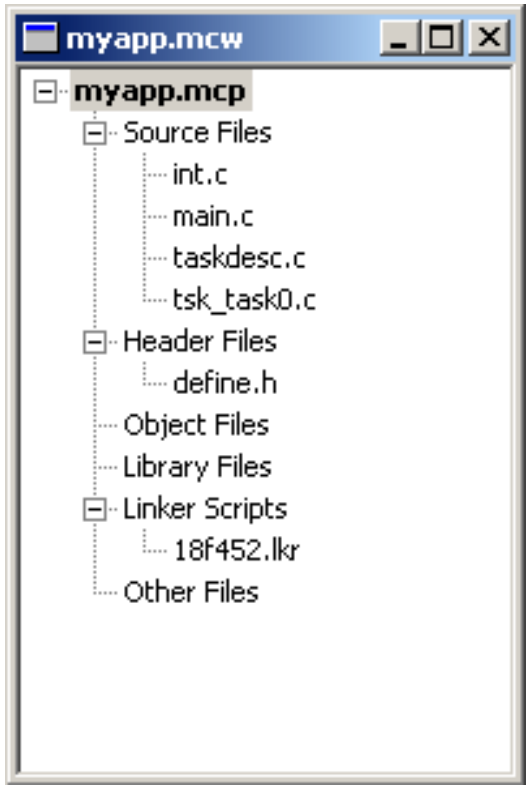
НАСТРОЙКИ ПРОЕКТА





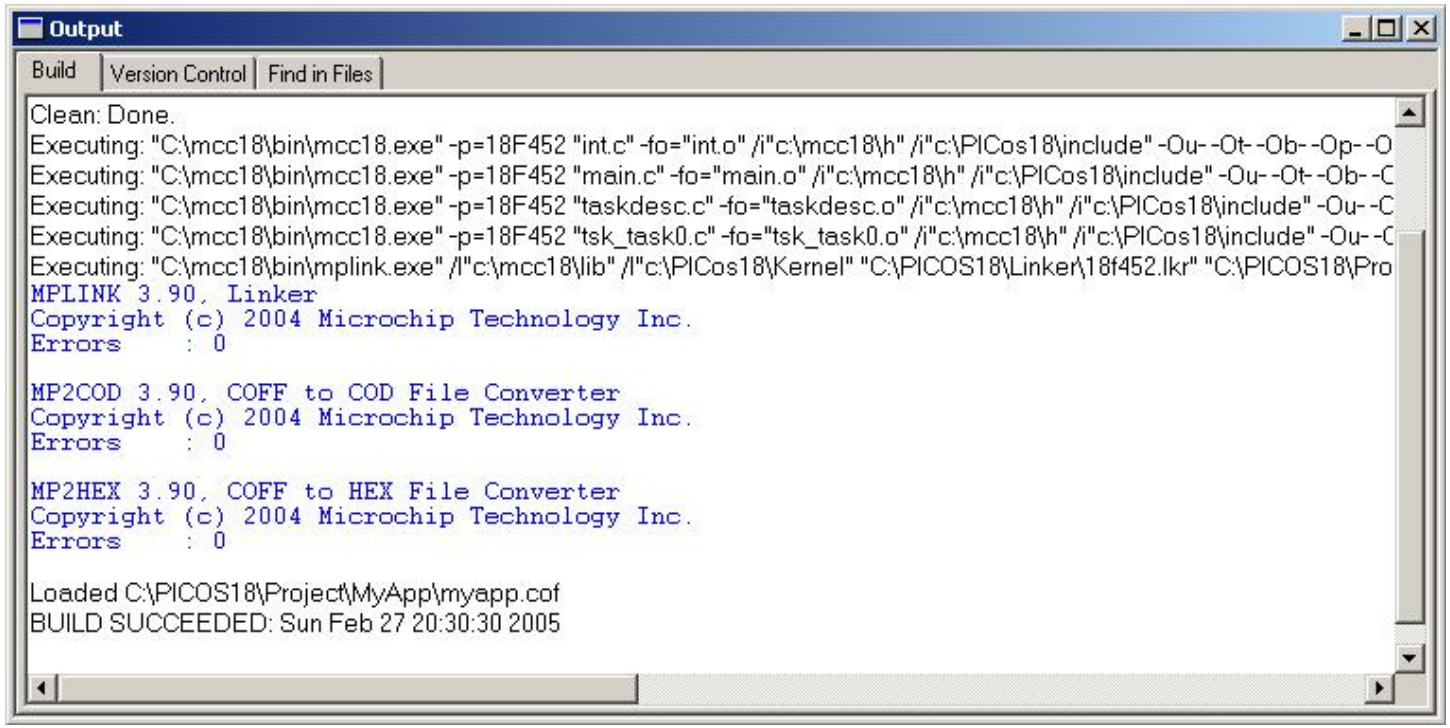
- Укажите пути к рабочим и вспомогательным директориям.
- Задайте режимы компиляции и компоновки.

ПРОЕКТ



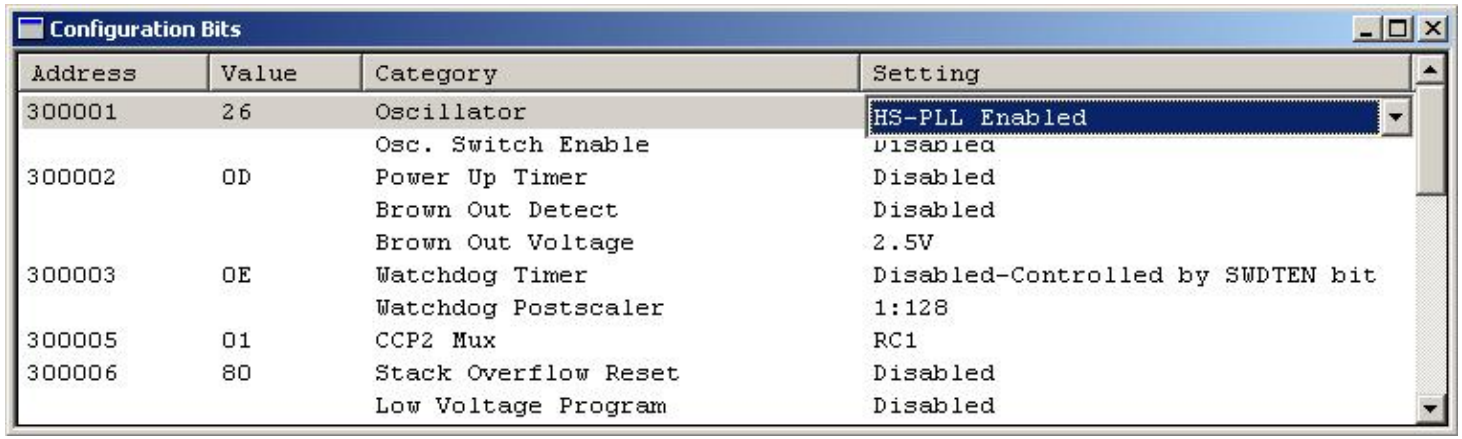
- Добавьте в проект файлы ОС

КОМПИЛЯЦИЯ ПРОЕКТА



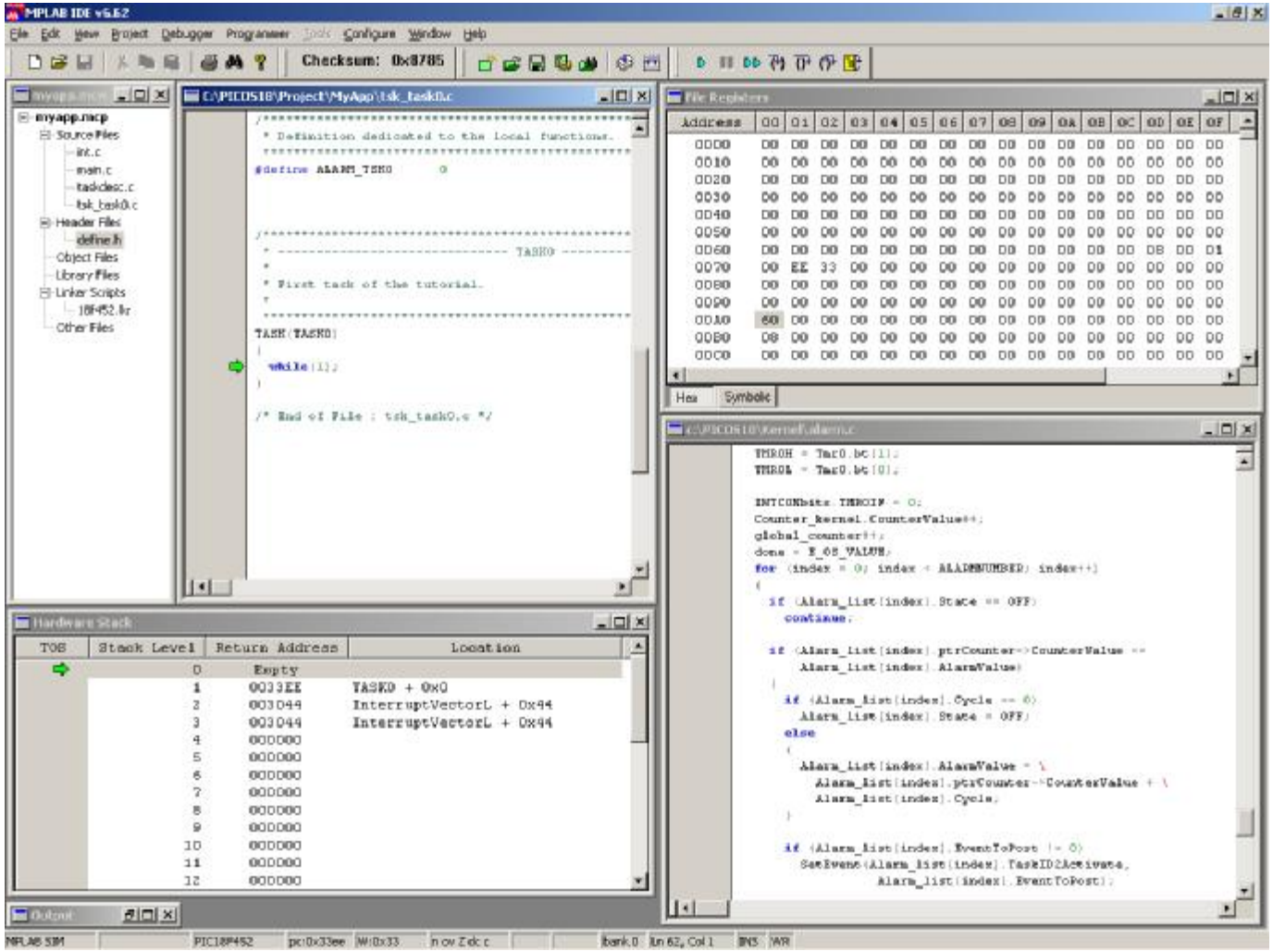
- Скомпилируйте проект

ТАЙМЕРА



- 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



- Симулятор позволяет выполнить программу без микроконтроллера!

ПЕРВАЯ ЗАДАЧА

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

Задача

```
/*********************************************************************
* ---------------------------- 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)        */
};
```

Бланк описания задачи

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

Бланк описания таймаутов задачи

- У задачи есть состояние, таймауты…

ОПИСАНИЕ ЗАДАЧИ

```
/*****************************************************************
 * ---------------------- COUNTER & ALARM DEFINITION ---------------------
 *****************************************************************/
Resource Resource_list[] =
{
  {
    10,                                 /* priority                 */
    0,                                  /* Task prio                */
    0,                                  /* lock                     */
  }
};
```

Бланк описания ресурсов задачи

```
/*****************************************************************
 * ------------------------ TASK & STACK DEFINITION --------------------
 *****************************************************************/
#define DEFAULT_STACK_SIZE      128
DeclareTask(TASK0);
volatile unsigned char stack0[DEFAULT_STACK_SIZE];
```

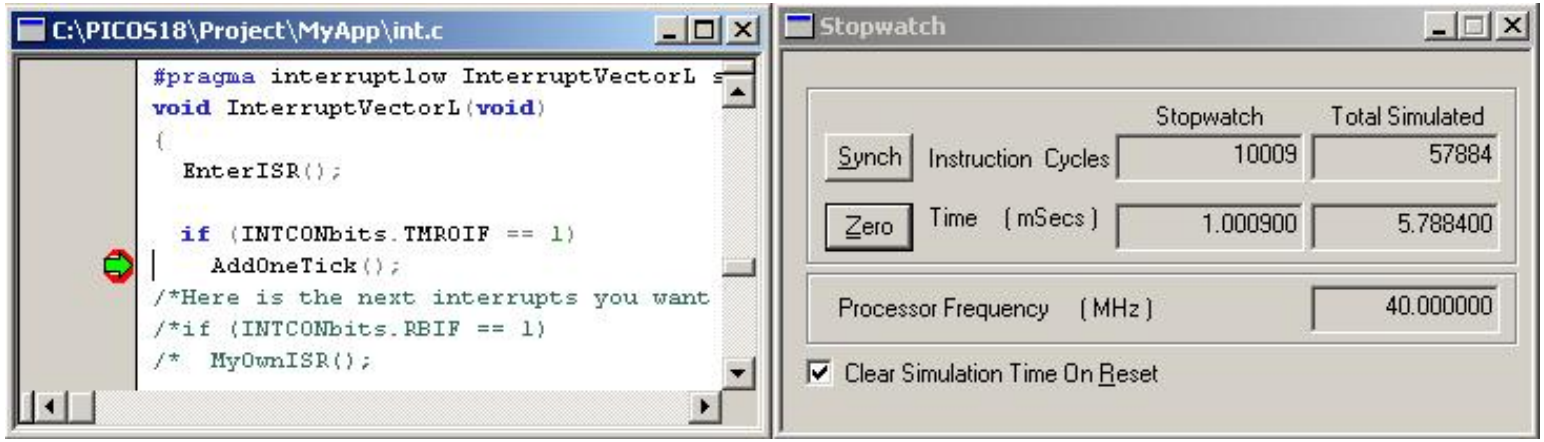
Бланк описания стека задачи

- … ресурсы, стек.

МС

```
#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



- 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 */
  T0CON = 0x08;
  TMR0H = Tmr0.bt[1];
  TMR0L = Tmr0.bt[0];
}
```

Сервис ОС - функция инициализации периферии

- Если необходима другая длительность тика, то переопределите константу TMR0_PRESET

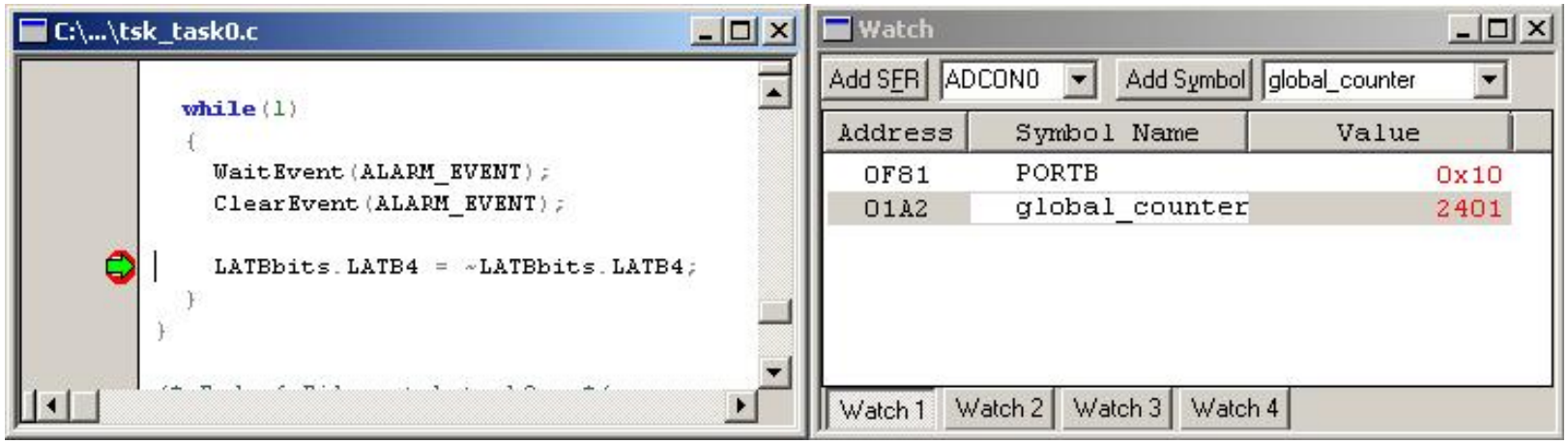
ВЫТЕСНЕНИЕ

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



- 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

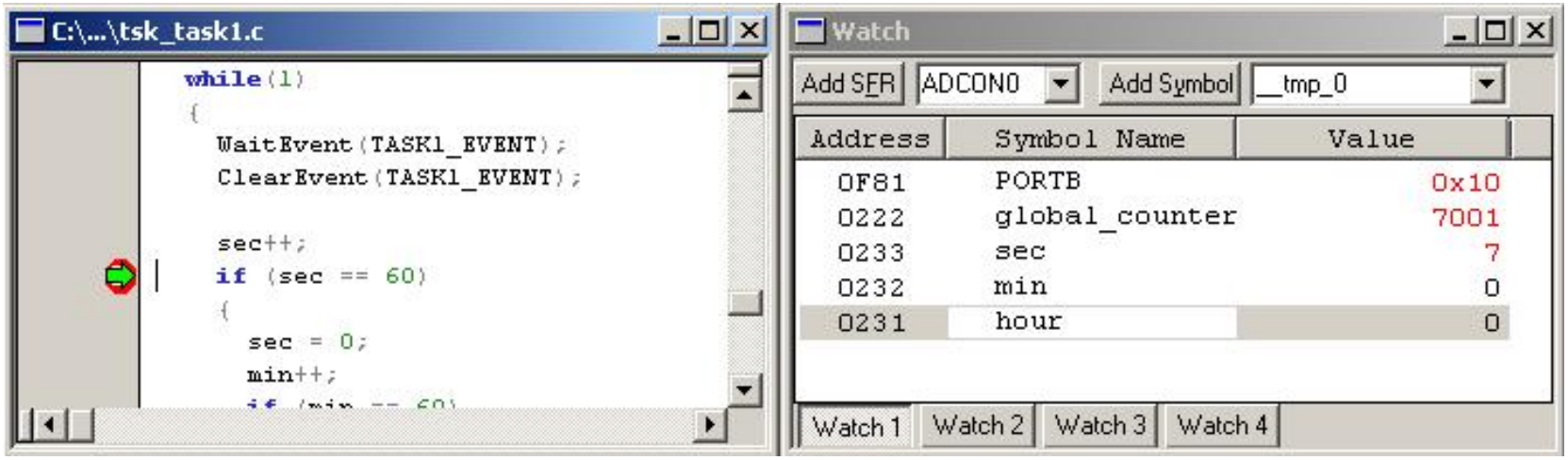
ОТКРЫТИЕ ВТОРОЙ ЗАДАЧИ

```
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



- 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 0
ClearEvent в задаче 1

- Поставьте точки останова и наблюдайте за последовательностью остановок

ВЫТЕСНЕНИЕ

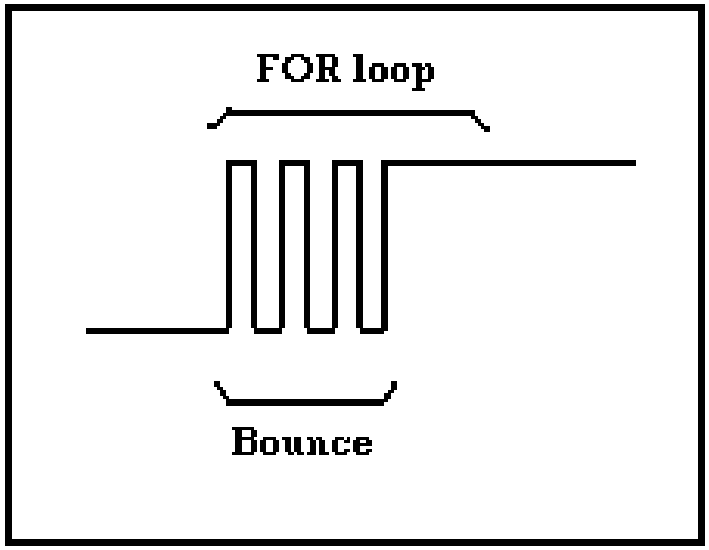
```
/**********************************************************************
 * ---------------------------- Task ID -------------------------------
 **********************************************************************/
#define TASK0_ID        1
#define TASK1_ID        2

#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

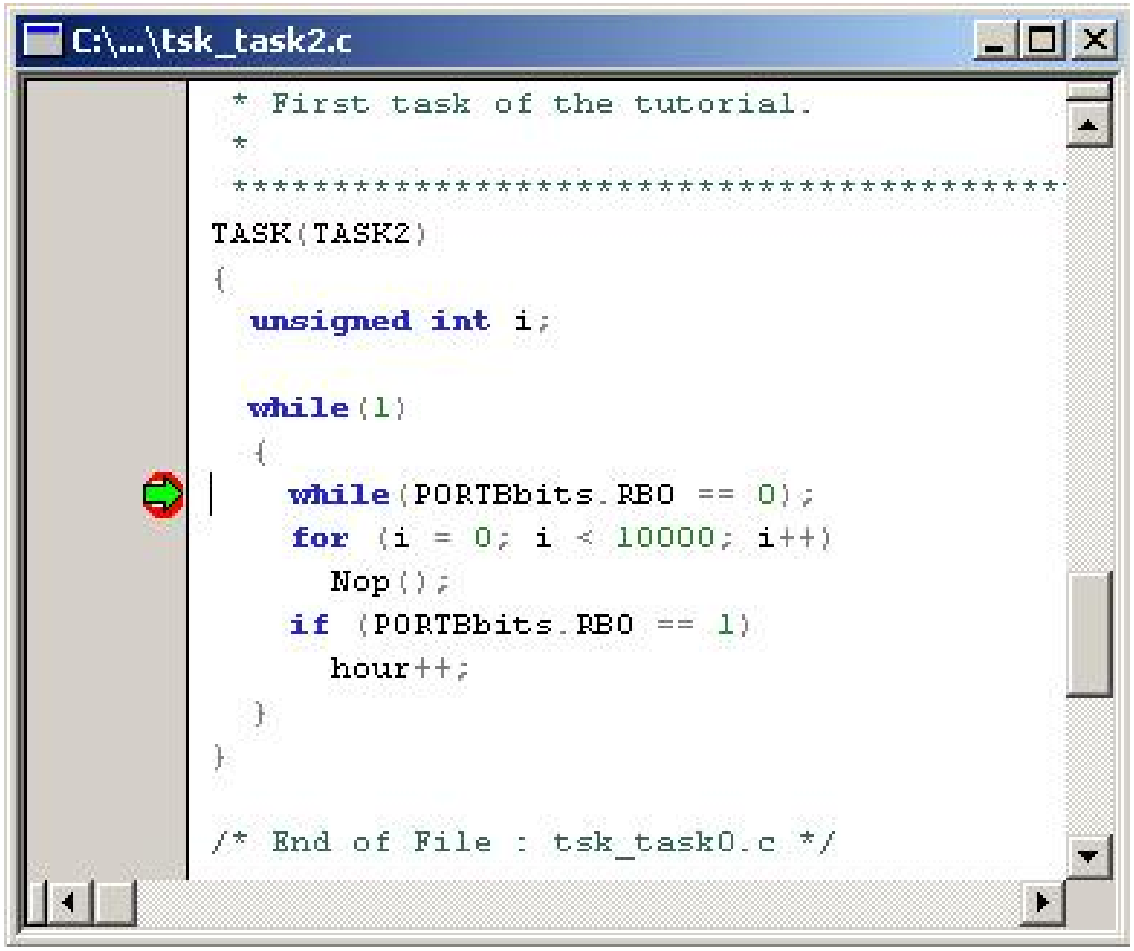
```
#define DEFAULT_STACK_SIZE 128
DeclareTask(TASK0);
DeclareTask(TASK1);
DeclareTask(TASK2);

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



- 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
```

C:\...\tsk_task2.c

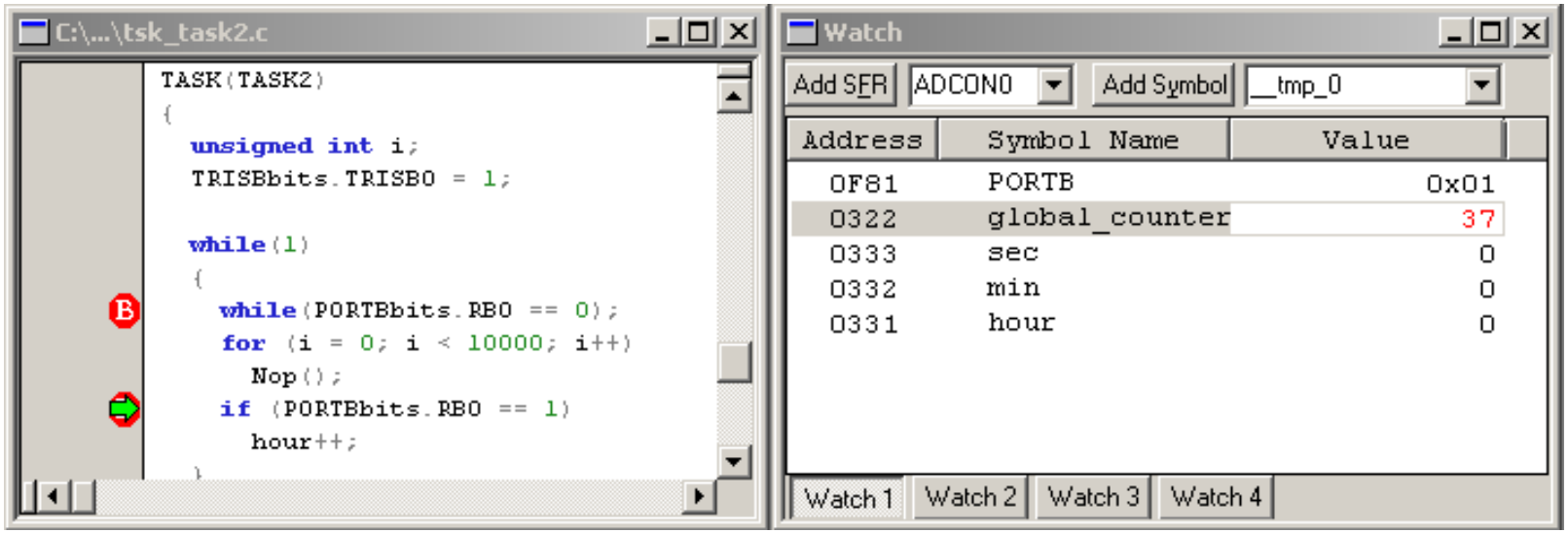
```
TASK(TASK2)
{
  unsigned int i;
  TRISBbits.TRISB0 = 1;

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

C:\...\tsk_task3.c

```
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.

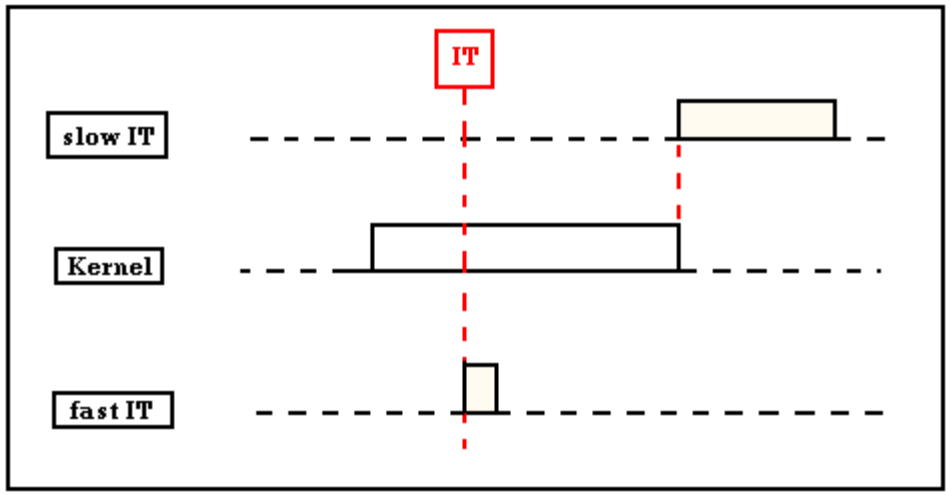
ТРАССИРОВКА В СИМУЛЯТОРЕ



- Установите точки останова в разных задачах и проконтролируйте моменты остановок

ПРЕРЫВАНИЯ

Особенности файла int.c



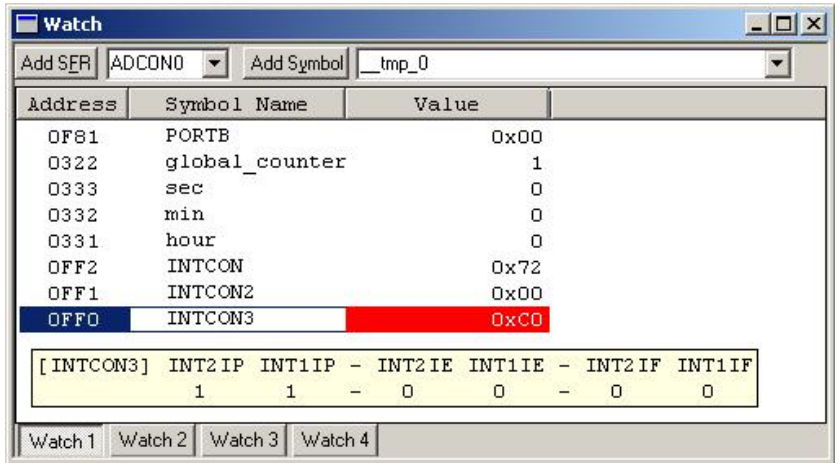
ОСОБЕННОСТИ ФАЙЛА INT.C

```
/*********************************************************************
 * 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
```

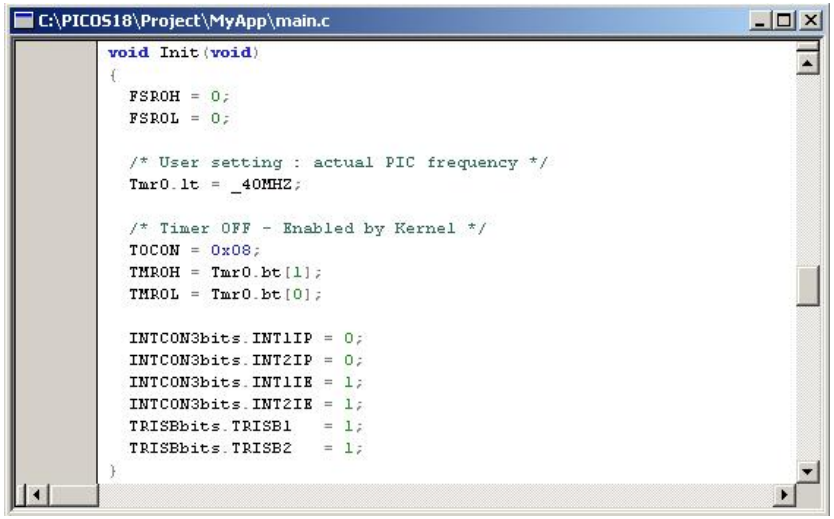
ОСОБЕННОСТИ ФАЙЛА INT.C

```
/*********************************************************************
 * General ISR router. Complete the function core with the if or switch
 * case you need to jump to the function dedicated to the occuring 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»



Режим «медленных прерываний»



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 occuring 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

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 */
```

C:\...\tsk_task3.c

```
TASK(TASK3)
{
  unsigned int i;

  for (i = 0; i < 10000; i++)
    Nop();
  if (PORTBbits.RB2 == 1)
    min++;
  TerminateTask();
}
```

Watch

Add SFR | ADCON0 | Add Symbol | __tmp_0

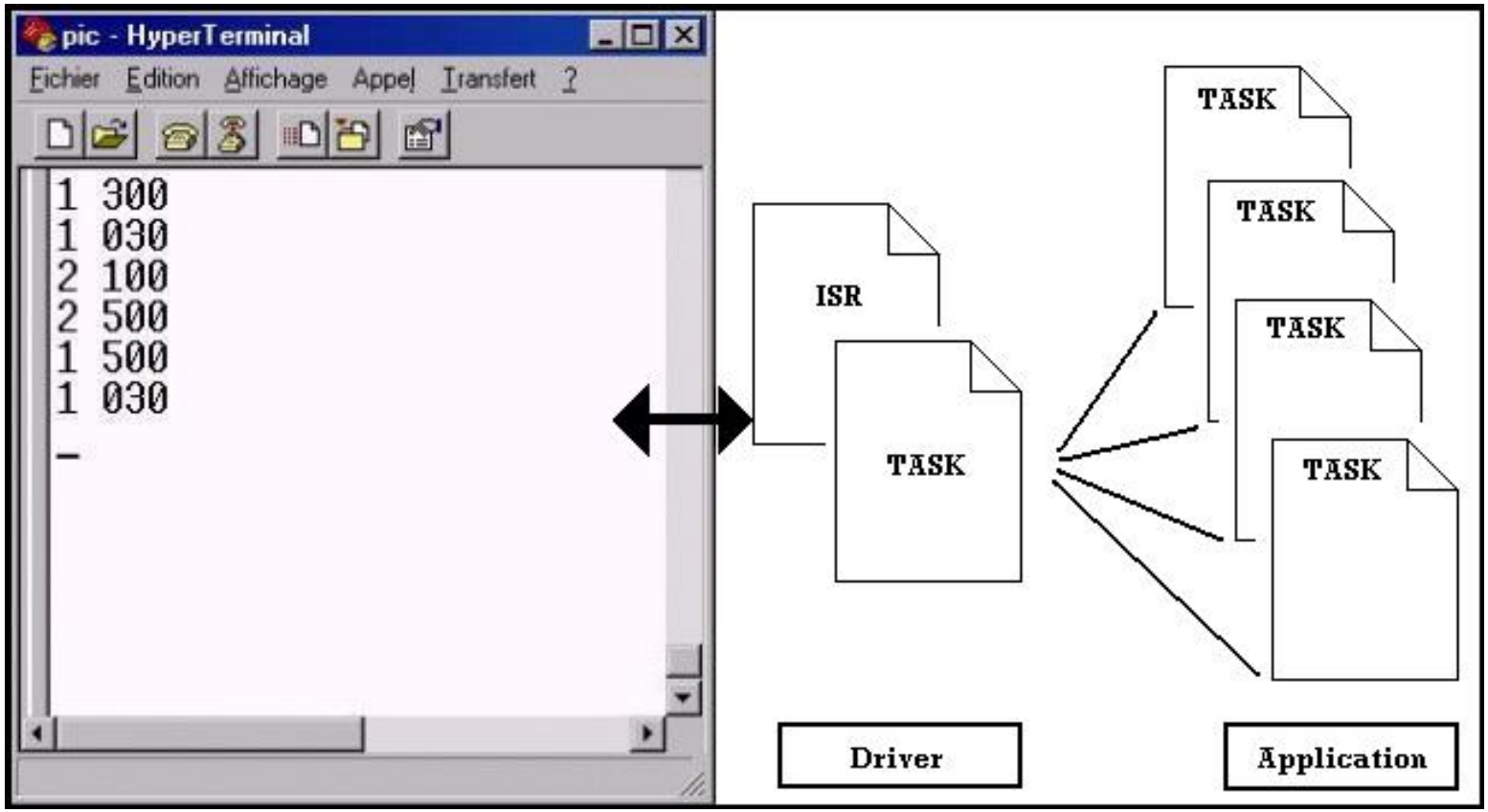
Address	Symbol Name	Value
0FF0	INTCON3	0x1A

Watch 1 | Watch 2 | Watch 3 | Watch 4

- Проследите, что задачи выполняются

ИСПОЛЬЗОВАНИЕ ДРАЙВЕРОВ

Особенности драйверов



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

ПРОГРАММНОГО СЛОЯ

> Drivers

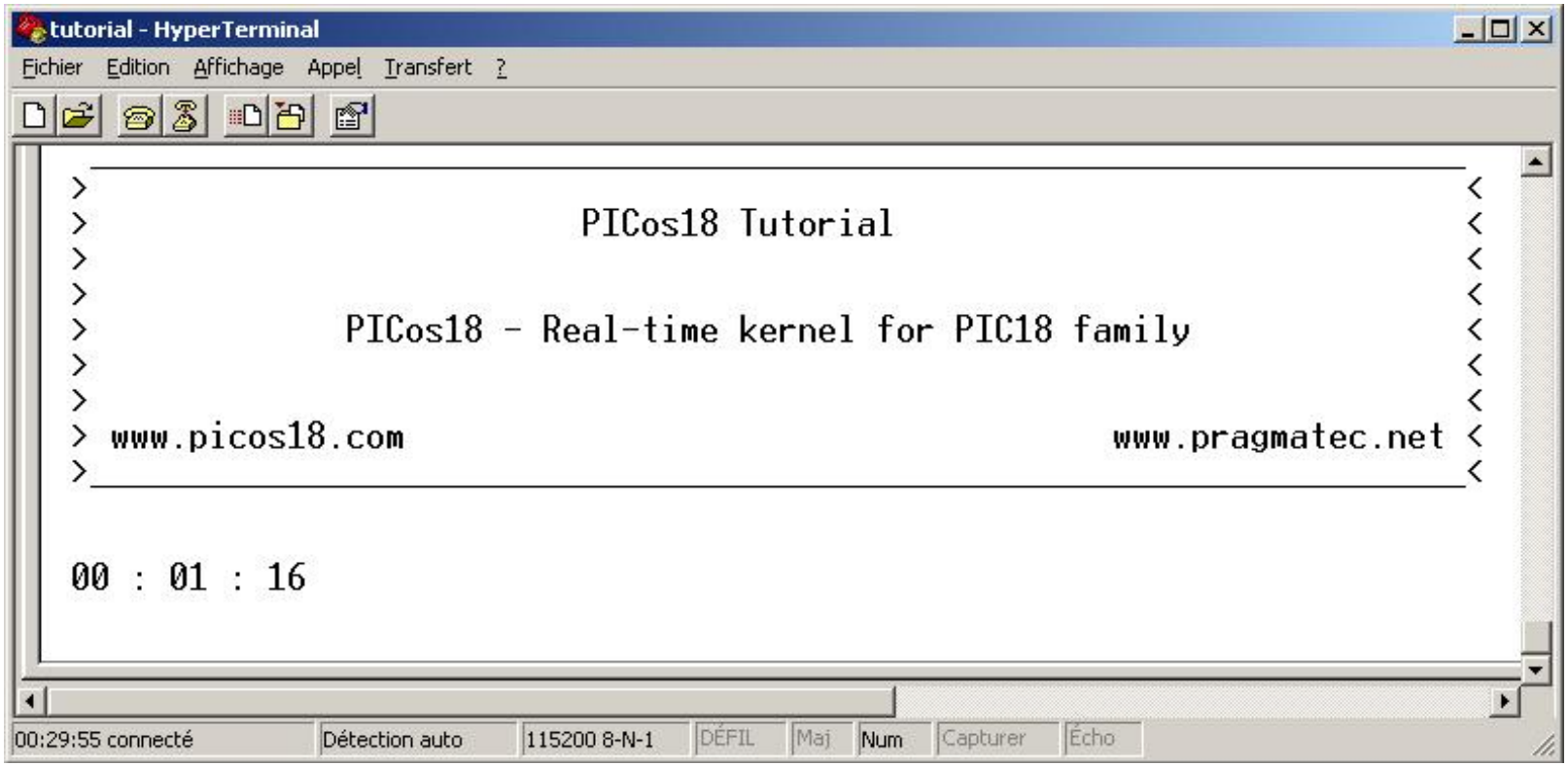
	Version	*Document*	*Sources*
LCD HD4478	v1.01	TXT	
Librairie HD4478	en cours...		
Driver I2C (master)	v1.03	TXT	
Driver I2C (slave)	en cours...		
Driver CAN	v1.00	TXT	
DS1337 (RTC)	en cours...		
DS1624 (T°)	en cours...		
Driver RS232	v1.02	TXT	

- Драйверы некоторых устройств и периферии доступны на сайте.

ИСПОЛЬЗОВАНИЕ ДРАЙВЕРА

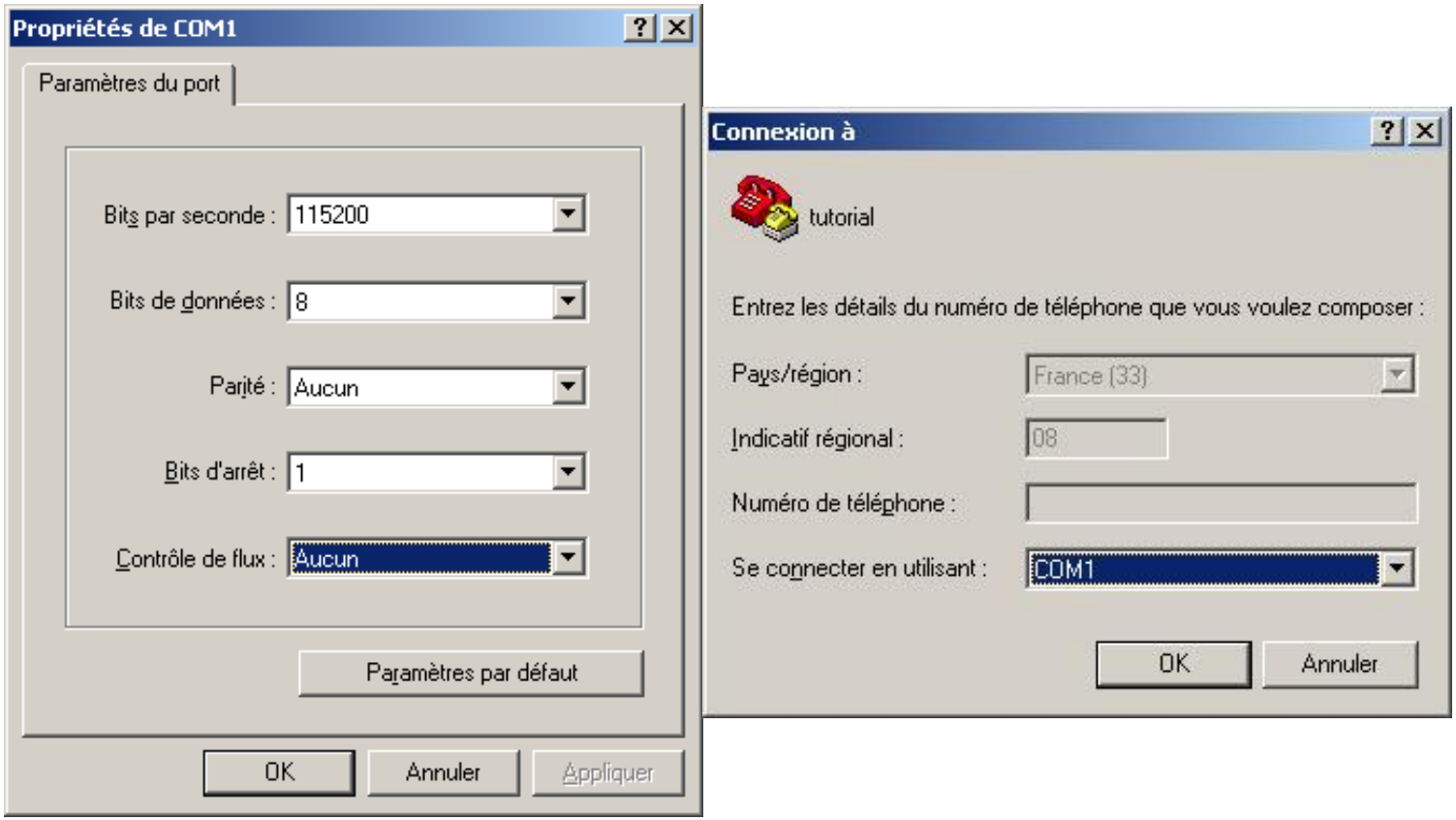
```
#include "define.h"
#include "drv_rs.h"
#include <stdio.h>
#include <string.h>
#define ALARM_TSK4 1
int Printf (const rom char *fmt, ...);
extern unsigned char hour;
extern unsigned char min;
extern unsigned char sec;
/*****************************************************************
 * Definition 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(">                                                                           <\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_enqMsg(&RS_msg, buffer, 50);
  va_start (ap, fmt);
  n = vfprintf (_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



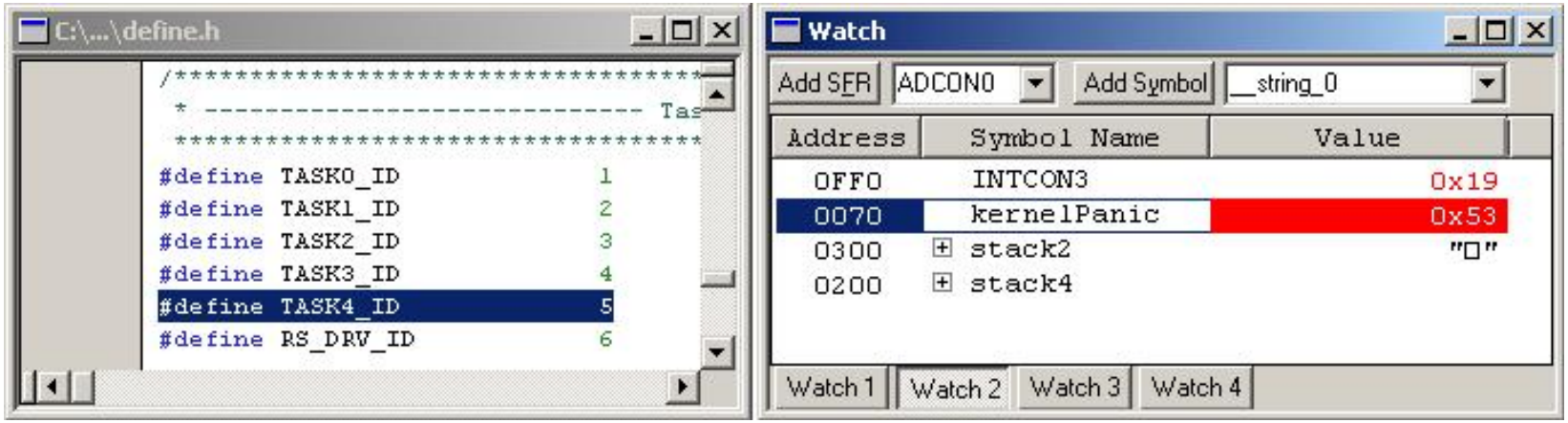
- Результат работы драйвера

НАСТРОЙКИ HYPERTERMINAL



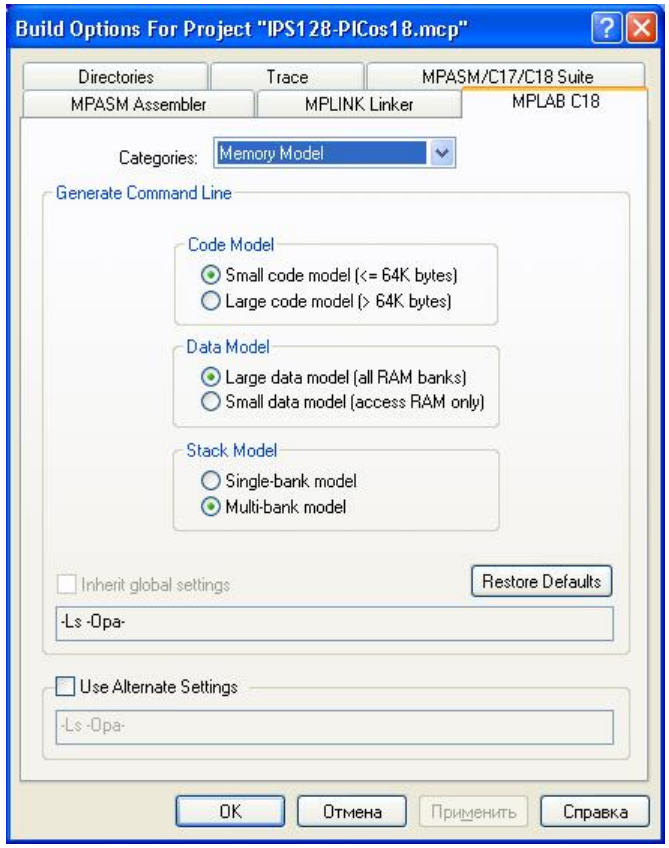
ПРИМЕР ПРИЛОЖЕНИЯ

Множественные прерывания

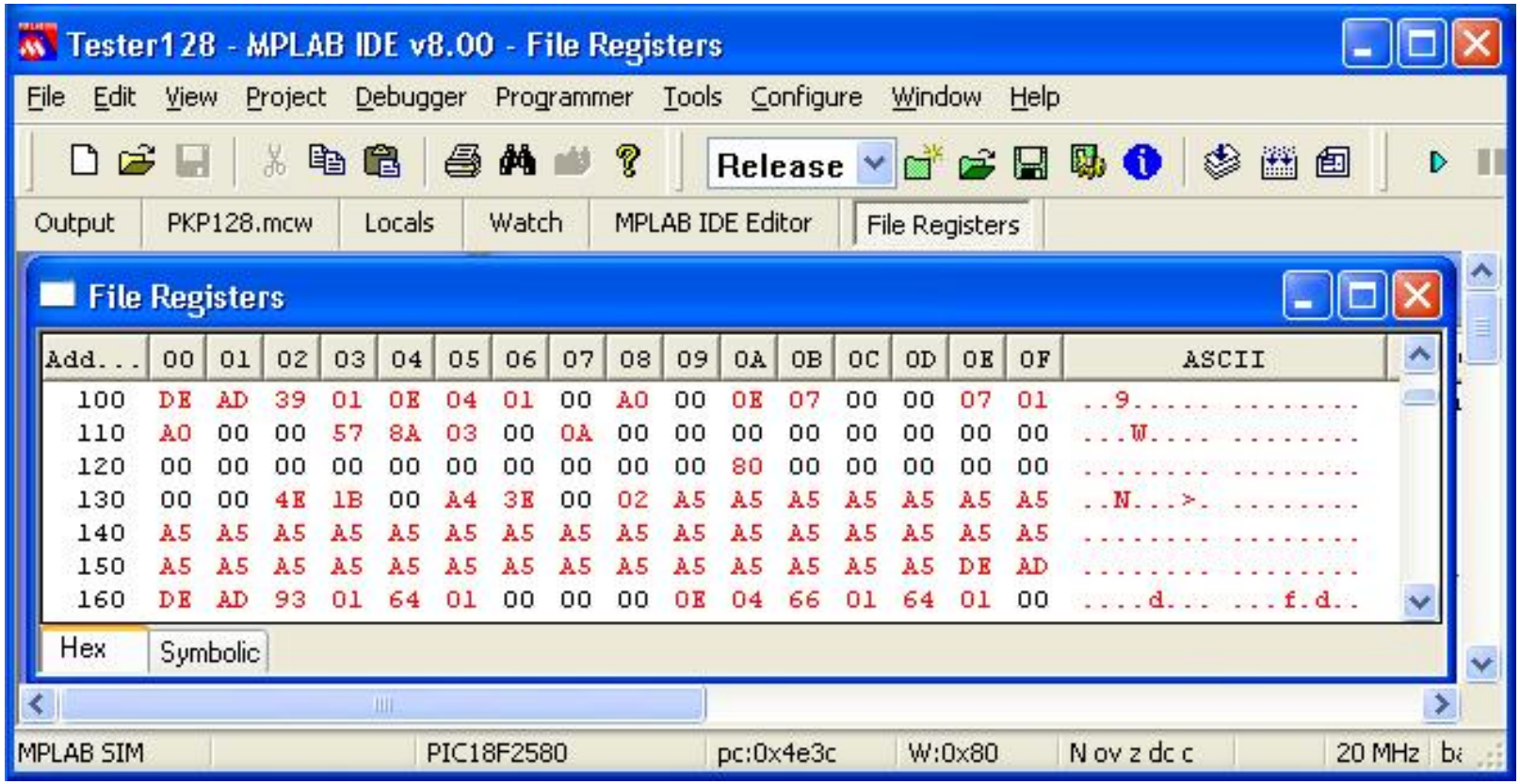


- Иногда контроллер вдруг «сходит с ума». На самом деле, происходит разрушение стеков задач.
- Переменная kernelPanic позволяет увидеть, какая задача явилась причиной разрушения стека

ПРОЦЕСС ОТЛАДКИ



- Определите модели Вашей программы



- Стек задач инициализируется кодом a5. 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