ИНТЕГРИРОВАННАЯ СРЕДА ДРАКОН. КУРС МОЛОДОГО БОЙЦА

Урок 2. ПЕРВЫЕ РЕЗУЛЬТАТЫ

Оглавление

§1. НЕСКОЛЬКО ЛИСТОВ В ПРОЕКТЕ, НЕСКОЛЬКО СХЕМ НА ЛИСТЕ

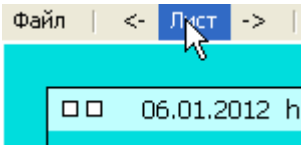
1) Все функции нижнего уровня мы поместим на отдельном листе. Это позволит «разделить сущности», и получить представление о работе программы с несколькими листами.

2) Каждая функция будет представлена отдельной схемой. Таким образом, на одном листе будет несколько схем.

3) Программные коды всех функций второго листа будут записаны в один файл.

§2. ДОБАВЛЯЕМ ВТОРОЙ ЛИСТ

4) Не закрывая первый лист, выбираем в меню «Файл» пункт «Новый лист». Редактор создаёт новый лист. Сохраняем его под именем «hard.drt». Теперь у нас открыто два листа. Переключаться между ними можно, используя стрелки в пункте главного меню «Лист»:

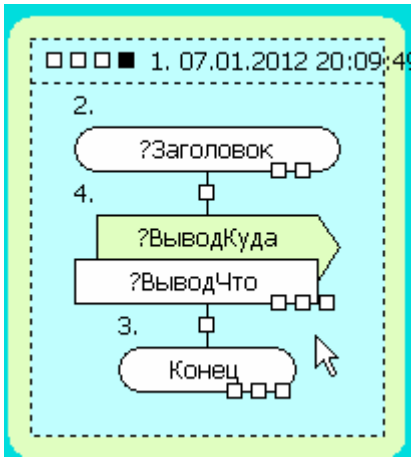


§3. ПРИМИТИВ ДЛЯ ПРОСТОЙ ФУНКЦИИ

5) Так как функции здесь будут очень простыми, то для них лучше подходит схема «Примитив»:

6) В заголовке функции пишем её назначение, например «Вкл. преобразователь». Надпись **крайне нежелательно вводить вручную**! Переключившись на первый лист, копируем текст из иконы «Вкл. преобразователь».
И берём себе за правило: **никогда не пишем от руки, то, что у нас уже где-то записано**. **Для этого есть Ctr+C / Ctr+V**!

7) Тело функции будет представлено одной иконой «Вывод». У этой иконы два поля для поясняющих надписей:



8) Записываем в поле «?ВыводКуда» - «PORTA бит 1», в поле «?ВыводЧто» - «Низкий уровень». Имеется в виду низкий уровень напряжения. С помощью контекстного меню изменяем ширину иконок, так, что бы они лучше смотрелись:

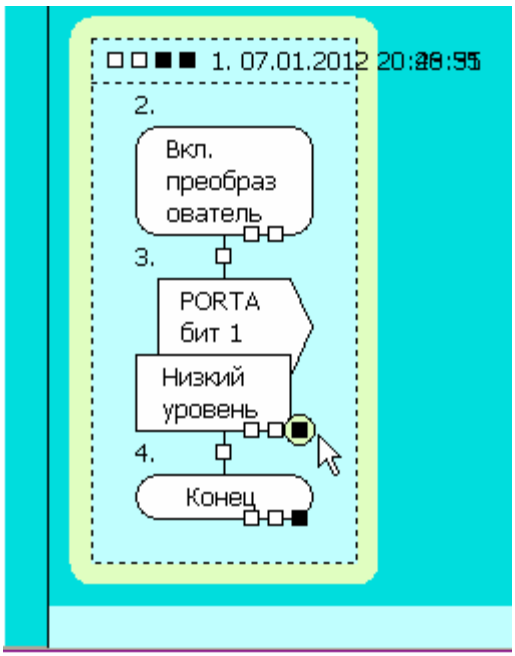
§4. УЧИТЫВАЕМ ОСОБЕННОСТИ ПРОЦЕССОРА

9) В программное приложение иконы записываем команды, которые приведут к включению нашего преобразователя. Подавляем в себе первое желание написать PORTA &= ~(1<<1).

10) У PIC процессоров такая операция **потенциально опасна**. Она выполняется по схеме чтение–модификация-запись. Чтение происходит не из внутреннего регистра–защёлки, а прямо с ножек процессора. Из-за этого вместе с битом, который мы меняем, могут изменится другие биты.

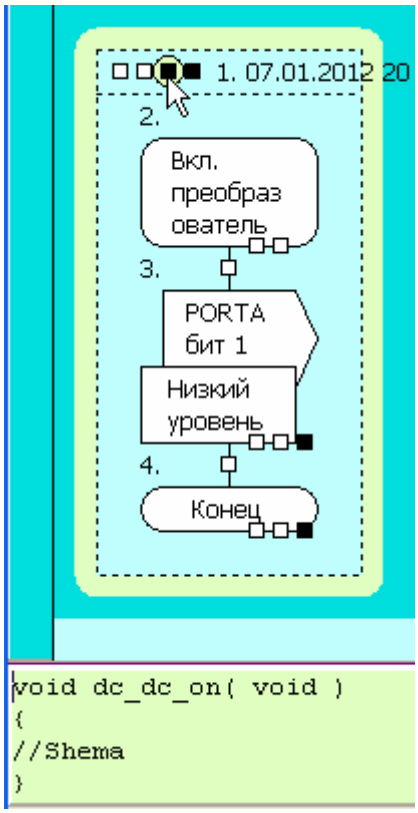
11) Обезопасим себя, и повысим надёжность устройства, применив дополнительную глобальную переменную – **зеркало порта**. Манипулировать битами будем только в этой переменной. А потом – копировать её в порт вывода.

12) В этой же функции будем устанавливать в правильное положение соответствующий бит в регистре управления. В итоге наша функция выглядит вот так:

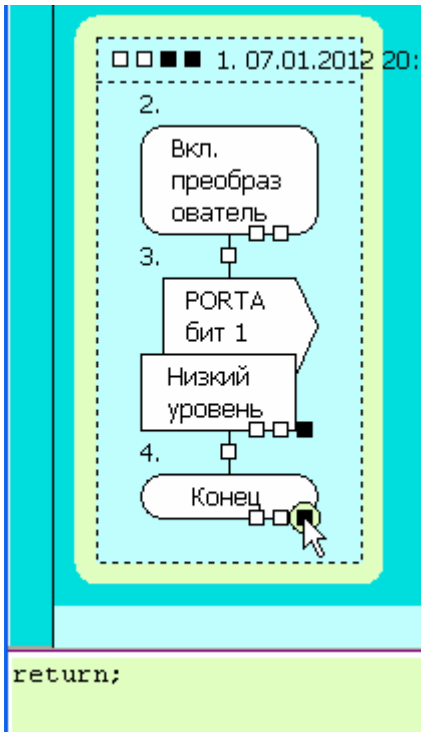



```
TRISA &= ~(1<<1);
mirror_porta &= ~(1<<1);
PORTA = mirror_porta;
```

13) Из файла «hard.h» копируем прототип функции, и записываем на его основе «скелет» нашей функции:



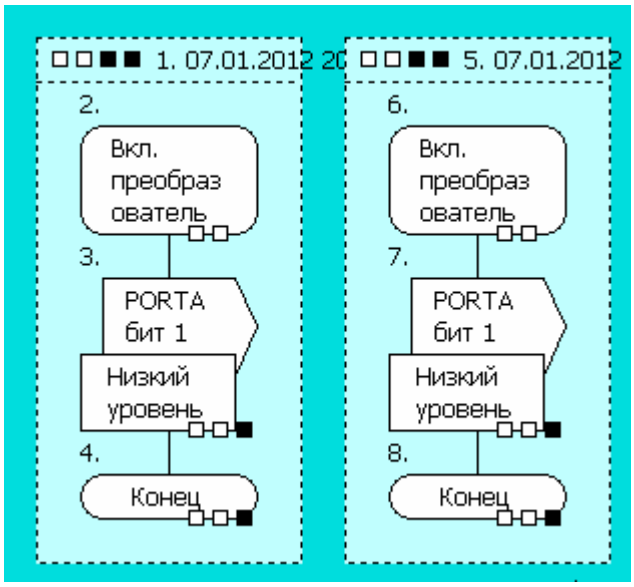
14) В Икону «Конец» вписываем оператор «return»:



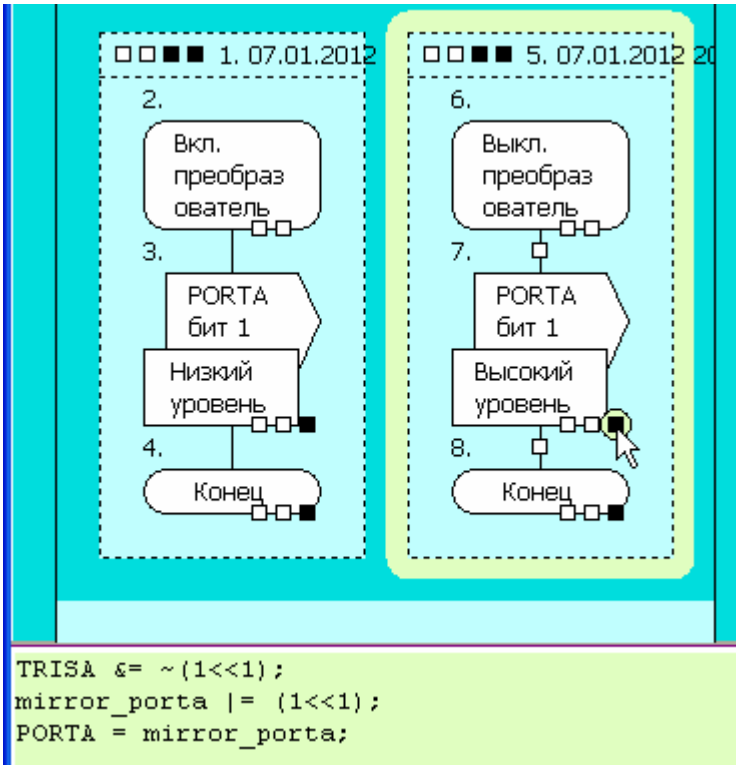
Функция готова.

§5. ВТОРАЯ ФУНКЦИЯ. ДА ЗДРАВСТВУЕТ CTR+C!

15) Приступаем к созданию функции выключения преобразователя. Тут нам поможет умение Интегрированной Среды копировать целые схемы. Щёлкаем в поле нашей схемы, и нажимаем Ctr+C. Щёлкаем на поле листа за пределами схемы, и нажимаем Ctr+V:



16) Меняем несколько символов в имени схемы, имени функции и операторах:

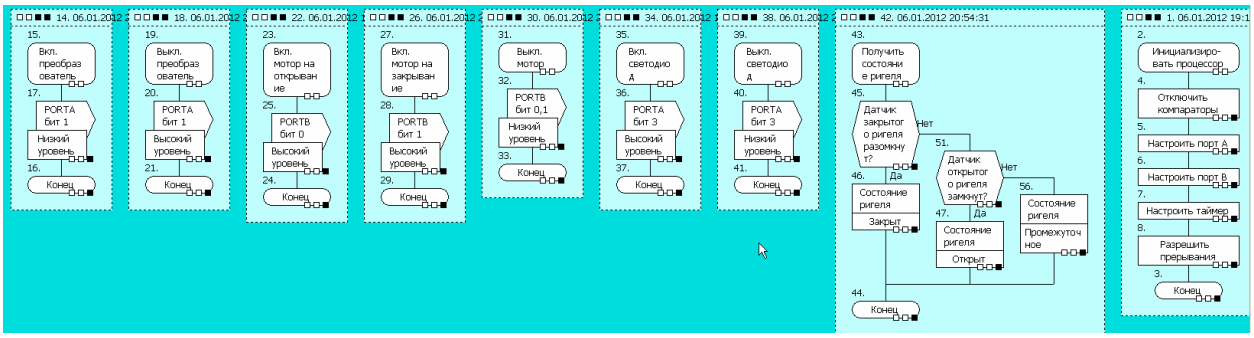


Вторая функция готова.

§6. ВСЕ ОСТАЛЬНЫЕ ФУНКЦИИ

17) Так, или примерно так, усиленно налегая на Ctr+C/Ctr+V, создаём на листе остальные функции. При этом имена для иконок «?Заголовок» аккуратно копируем из икон первого листа, а имена функций - из файла «hard.h».

В результате получился вот такой лист:



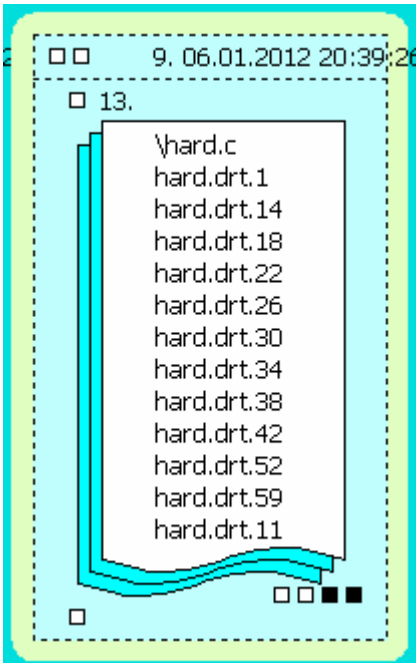
§7. ГНОМ – ВСЕМУ ГОЛОВА.

18) Помещаем на лист Гном, на Гном помещаем Модуль. В модуле записываем имя файла для сохранения текста программы.

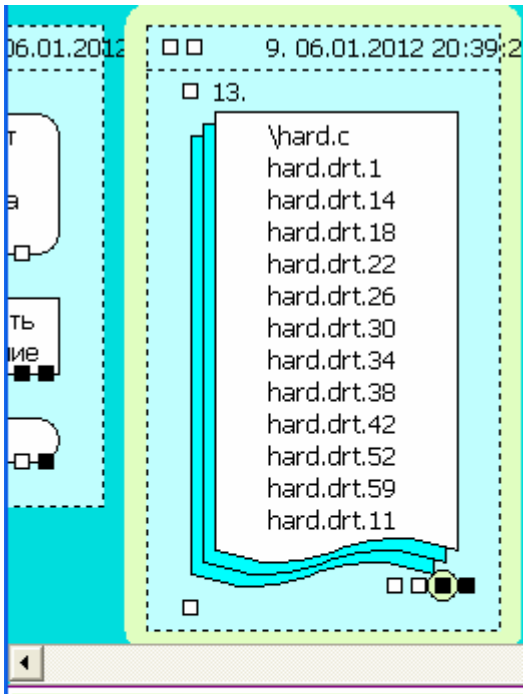
19) Щёлкаем по первой из наших схем правой кнопкой, выбираем «Взять ссылку в системный буфер», переходим к Модулю, вставляем ссылку из буфера на вторую строчку. Эту операцию надо проделать для всех схем на листе.

20) Можно облегчить и ускорить этот труд. Все ссылки содержат одинаковое имя листа, и различаются только номером схемы. Несколько раз щёлкаем Ctr+V, потом вручную исправляем номера схем. Номер схемы – это первая цифра в заголовке схемы:

В итоге Гном выглядит вот так:



21) Переходим к наполнению программного приложения Гнома. Щёлкаем третий квадрат. Вписываем подключаемые файлы, глобальные переменные и необходимые «магические символы»:



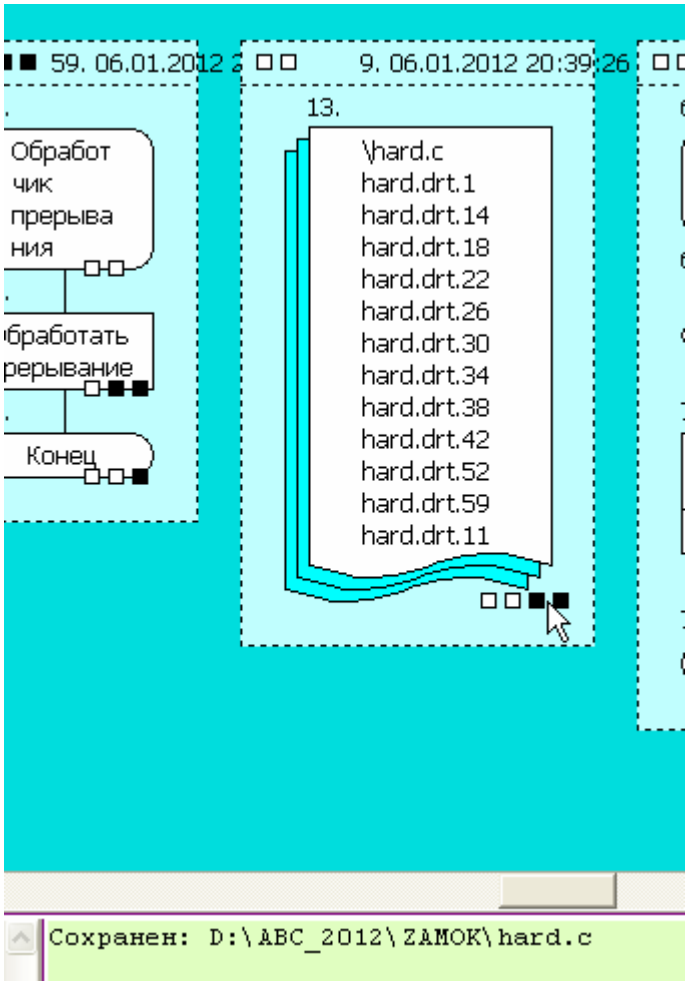

```
#include <pic16f6x.h>
#include "hard.h"

char mirror_porta;
char mirror_portb;
unsigned int base_time;

//Shema
//
```

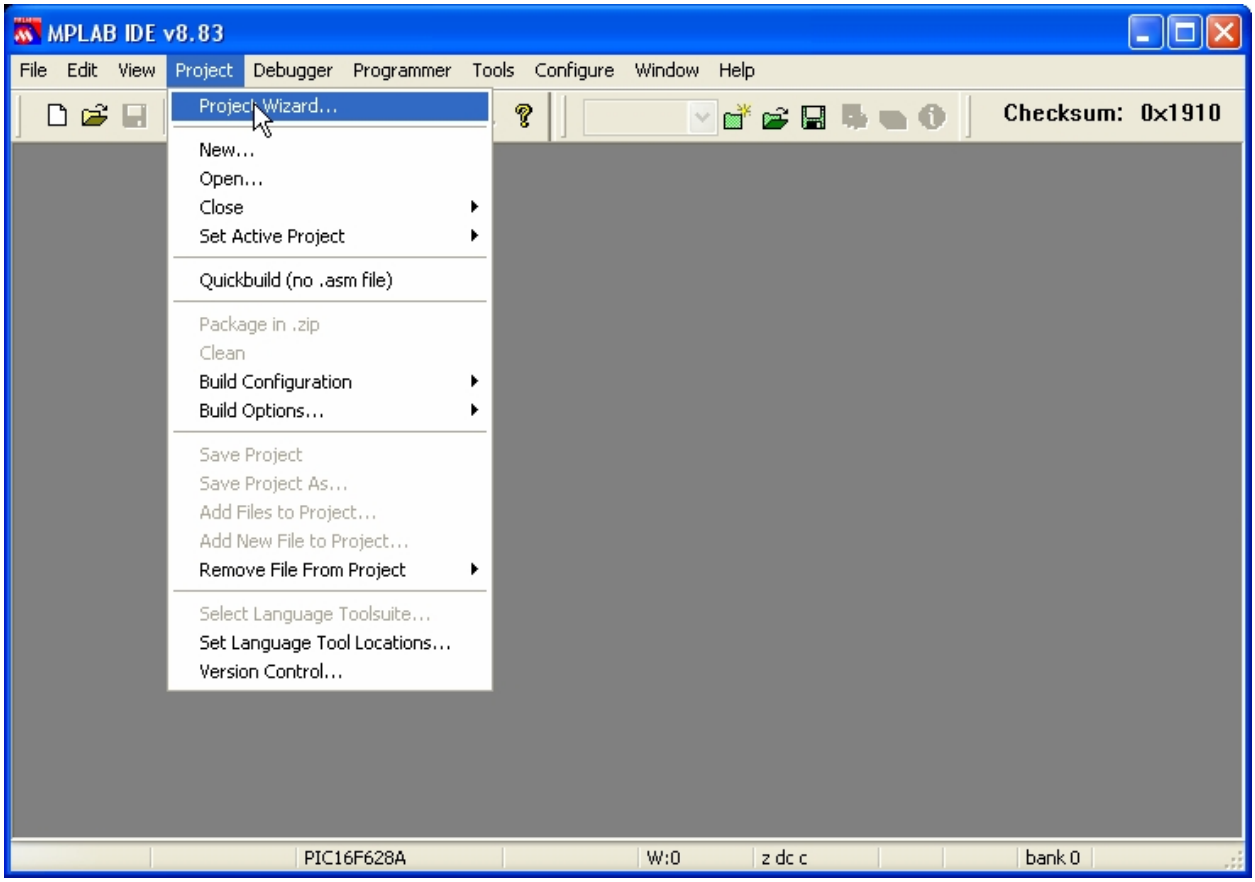
Всё готово для получения текста программы.

22) Нажимаем клавишу F9. Наш программный файл записан:

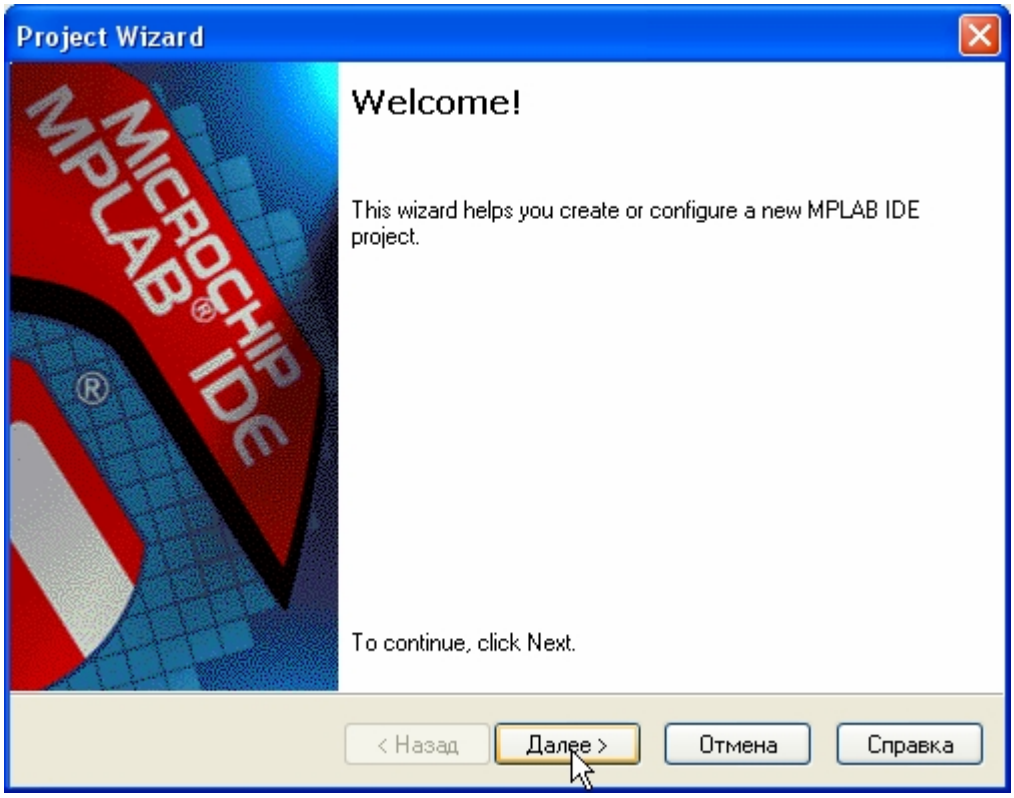


§8. КОМПИЛИРУЕМ ПРОГРАММУ.

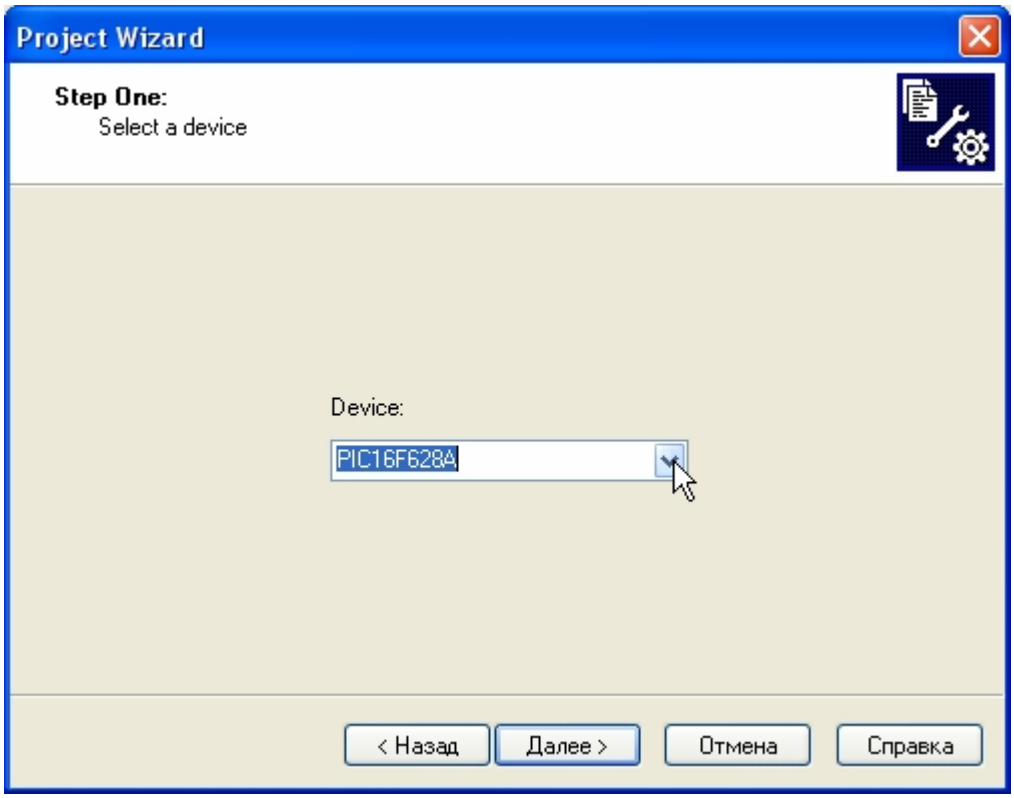
23) Запускаем интегрированную среду для работы с нашим процессором – MPLAB IDE:



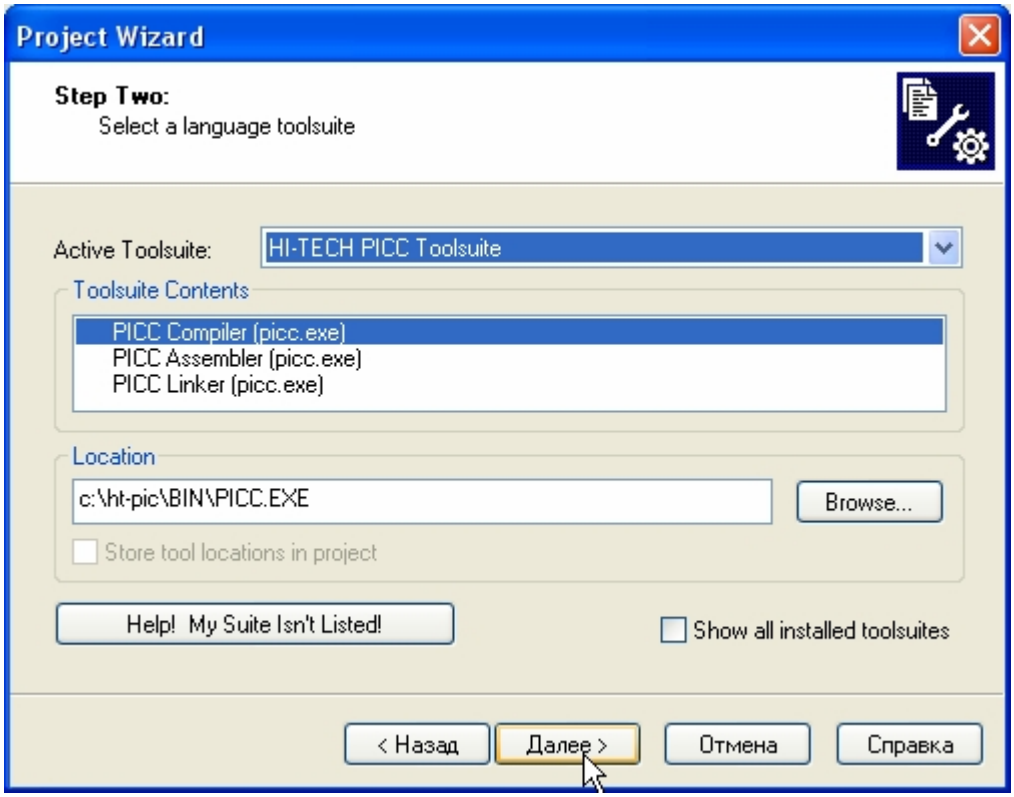
24) Запускаем мастер создания проектов:



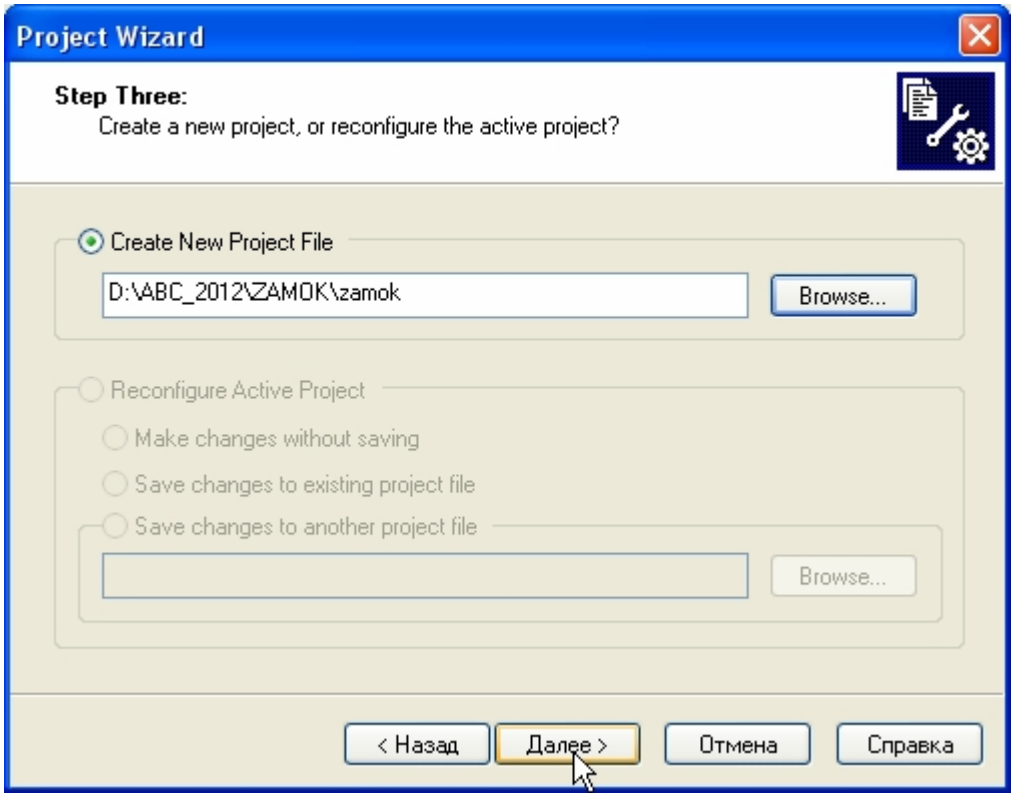
25) Указывает тип нашего процессора:



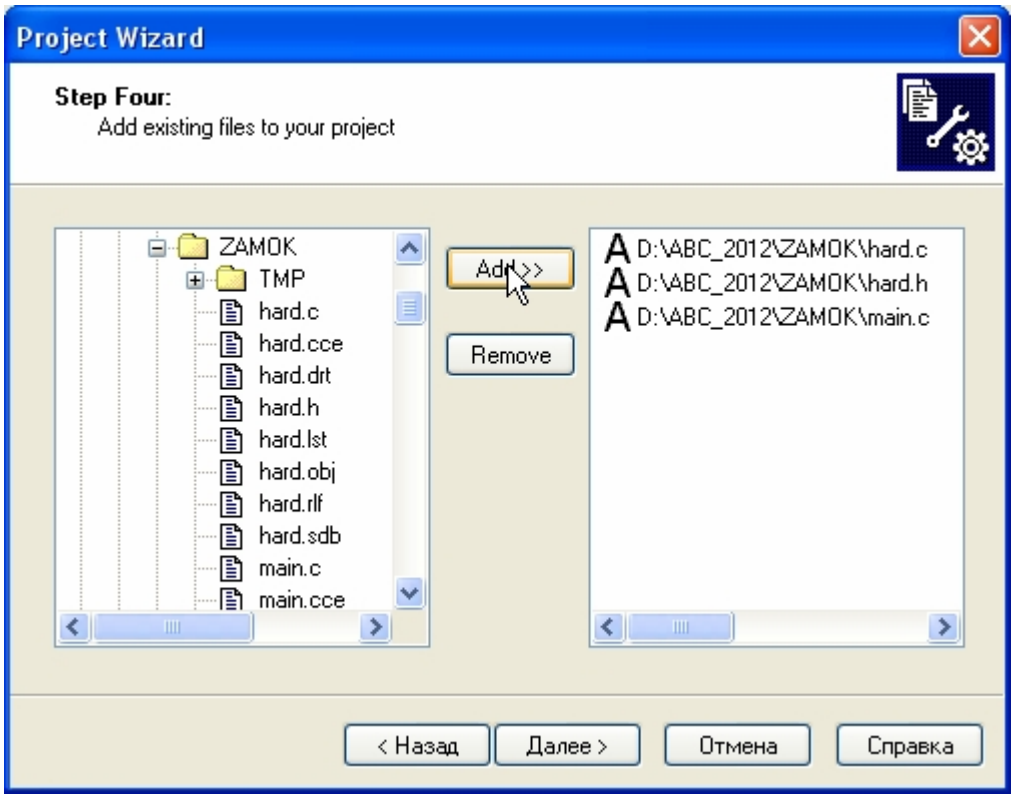
26) Указываем тип и расположение нашего компилятора:



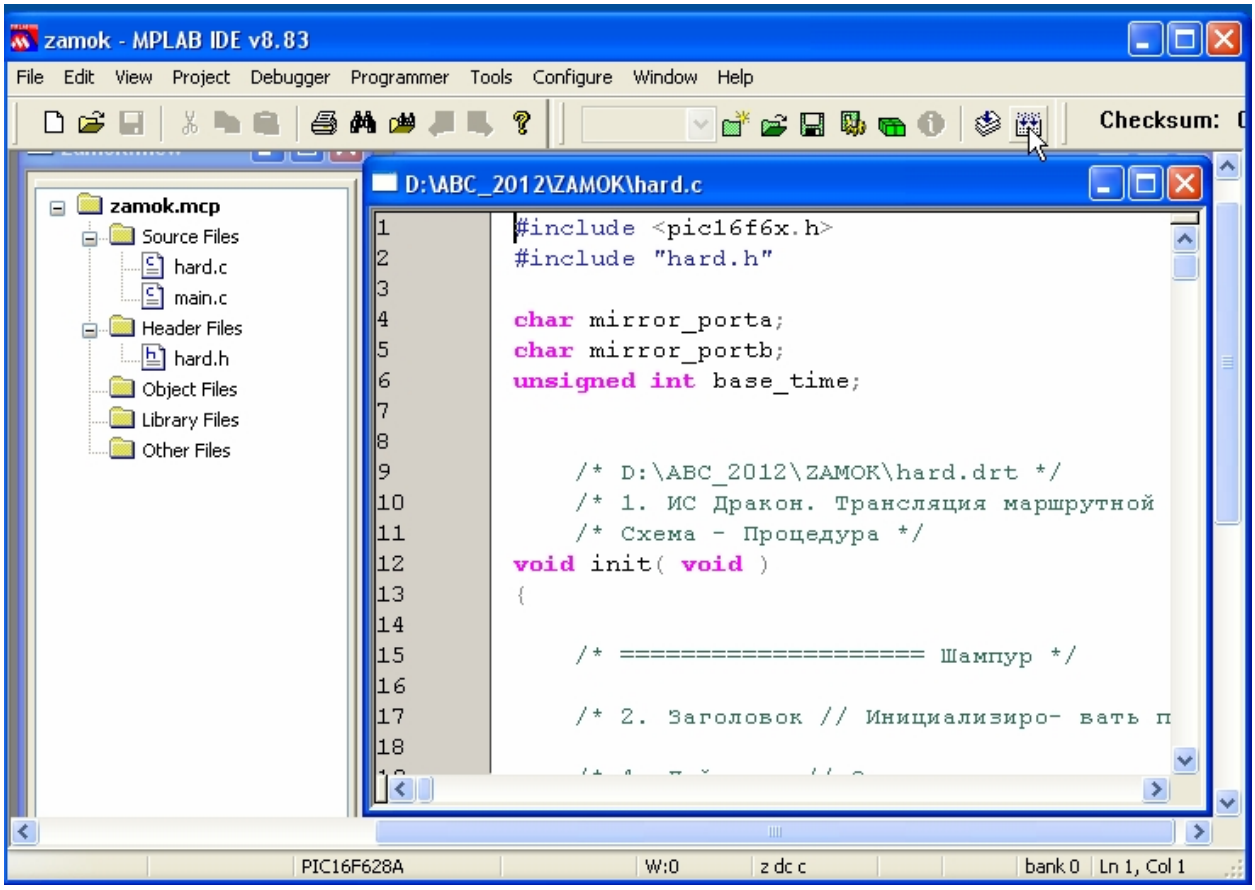
27) Задаём имя проекта:



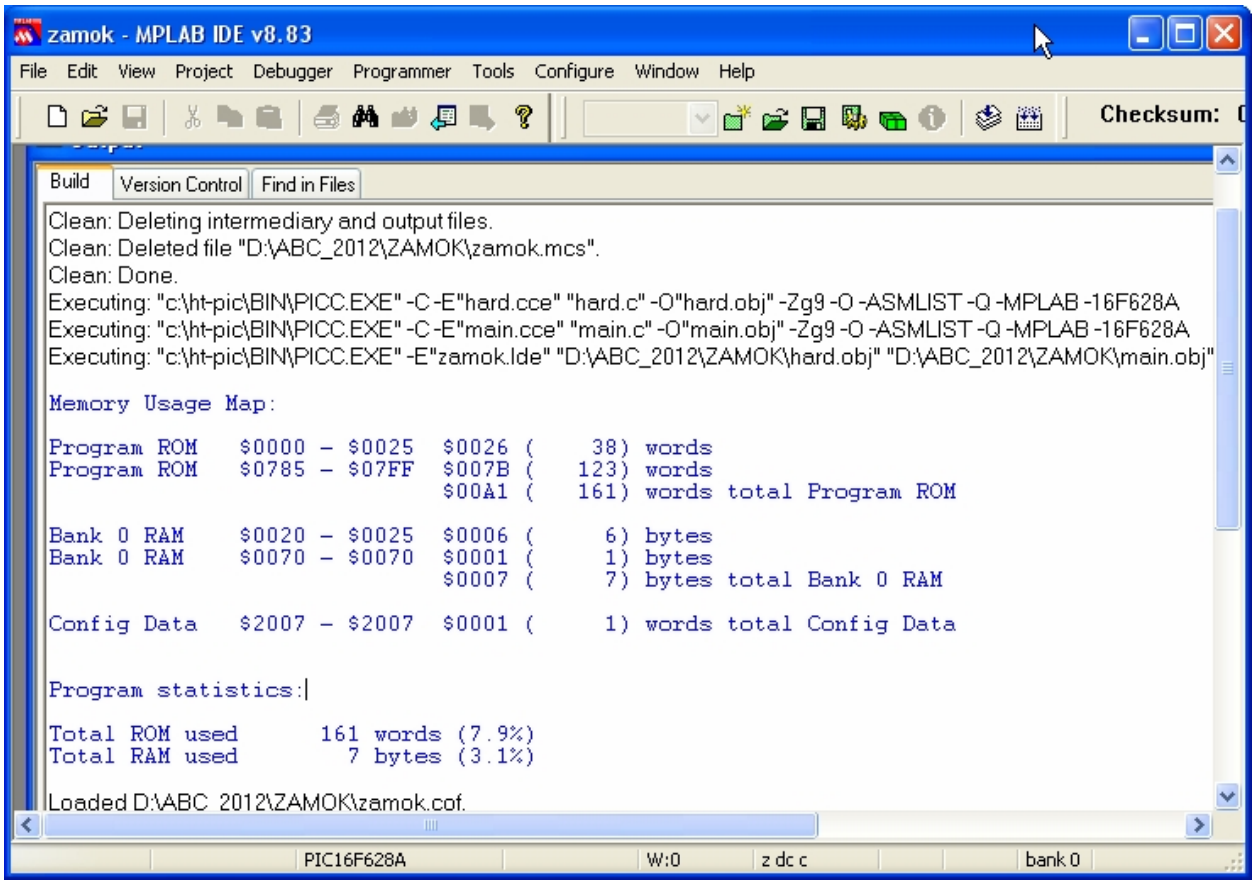
28) Добавляем в проект наши файлы с исходными текстами:



29) Проект создан, нажимаем на кнопку компиляции:

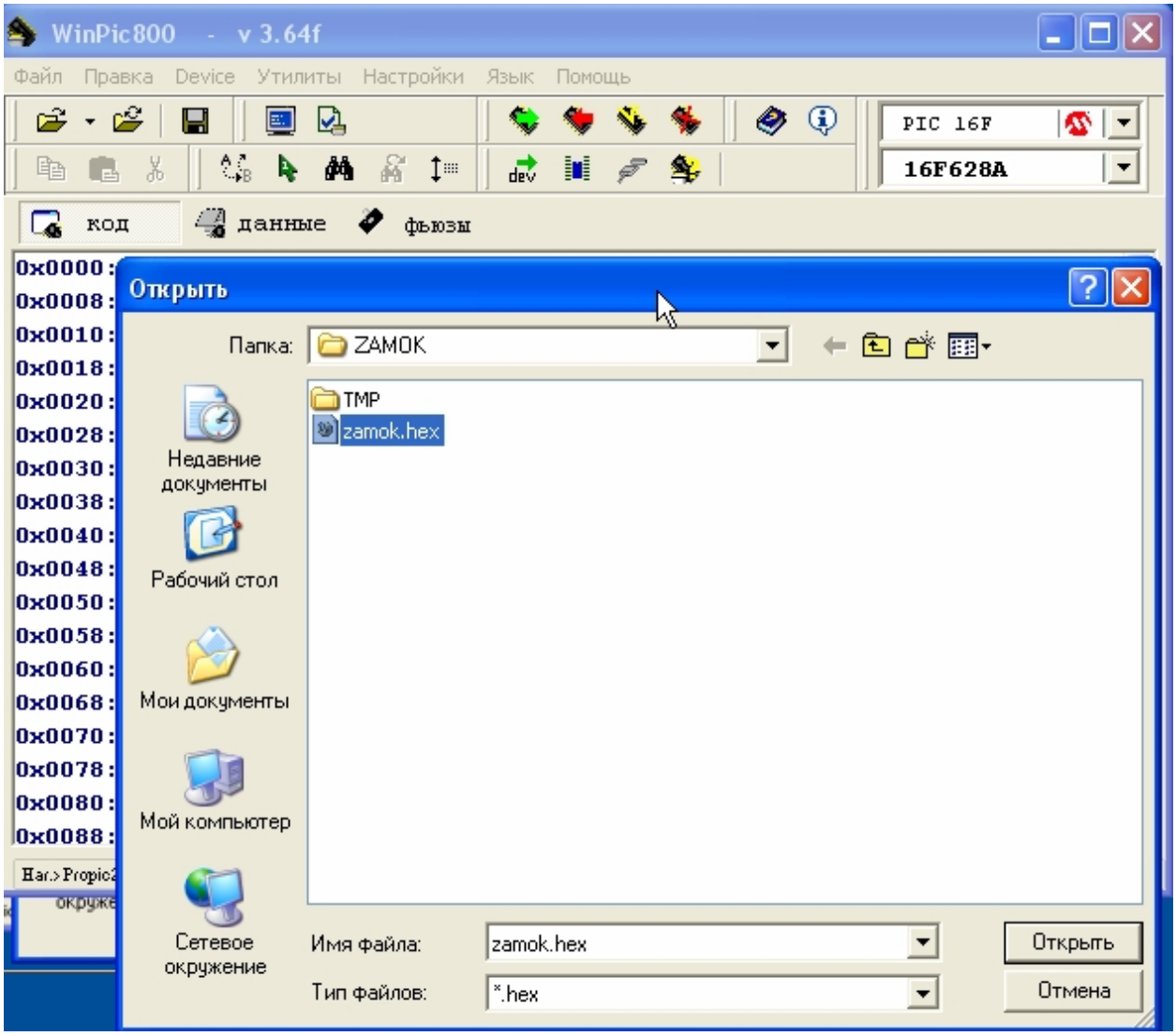


30) Проект откомпилирован, файл прошивки, пригодный для загрузки в микропроцессор, создан:

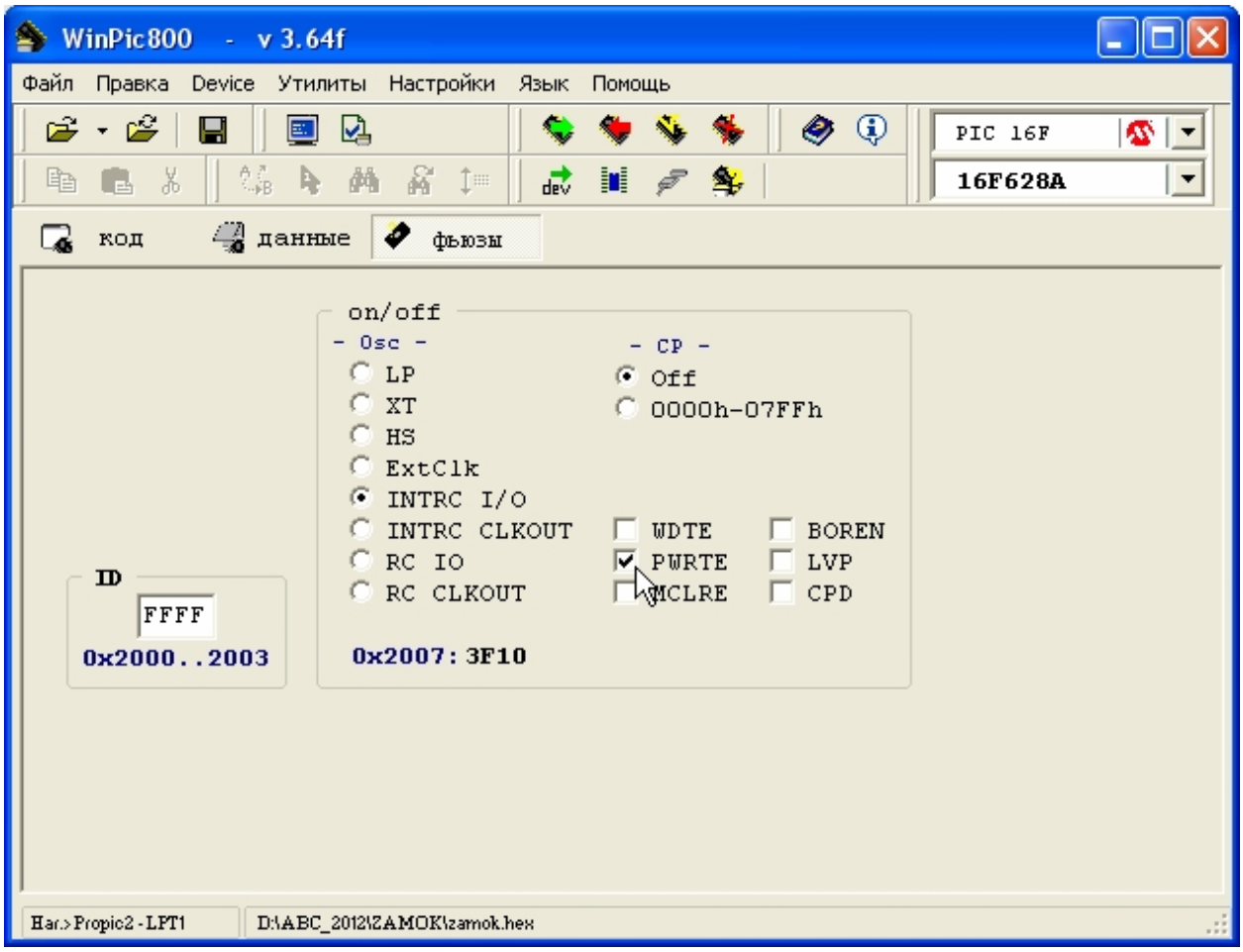


§9. ПРОШИВАЕМ ПРОЦЕССОР.

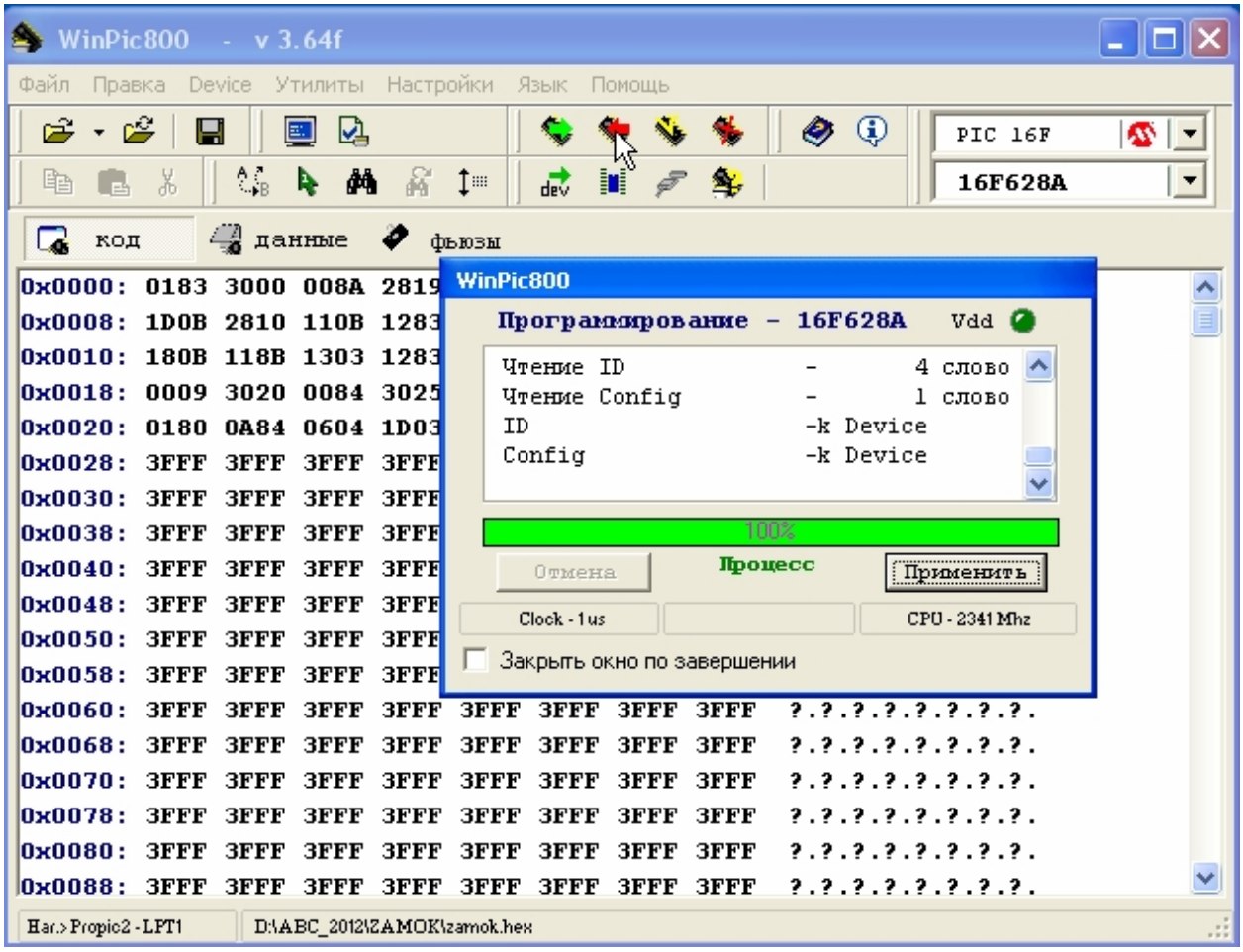
31) Запускаем программатор, загружаем в него файл прошивки:



32) Проверяем и устанавливаем биты конфигурации процессора:



33) Прошиваем процессор:



§10. ПРОВЕРЯЕМ В «ЖЕЛЕЗЕ».

34) Вставляем запрограммированную микросхему в контроллер замка. Подаём питание. Нажимаем кнопку. Ригель послушно втягивается. Нажимаем ещё раз – ригель запирает дверь. Нажимаем кнопку и держим её нажатой – ригель совершает возвратно-поступательные движения. Всё работает так, как нарисовано в нашем алгоритме.