Отладка программ в MPLAB X

Симуляция, SCL

Новые возможности отладки

Про что класс

Кто дойдет до конца, тот будет знать …

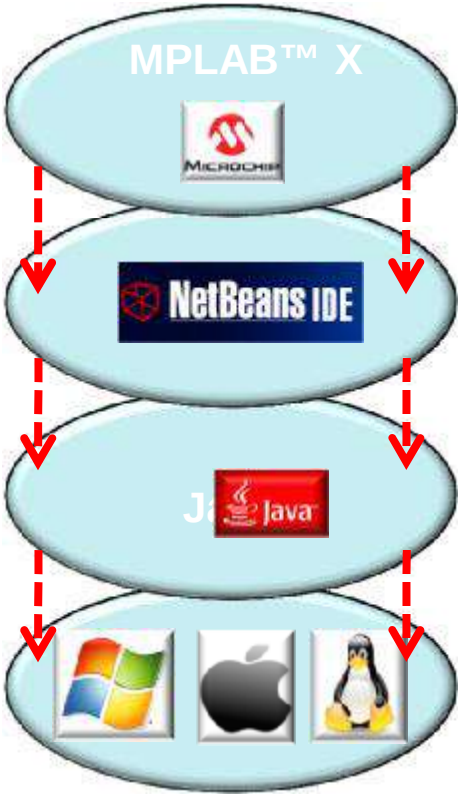
- Как более эффективно использовать Симулятор в MPLAB® X
- О Инжекции данных и входных сигналов во время симуляции
- Как использовать MPLAB Debugger (MDB) для автоматизации симуляции
- О новых отладочных функциях

План

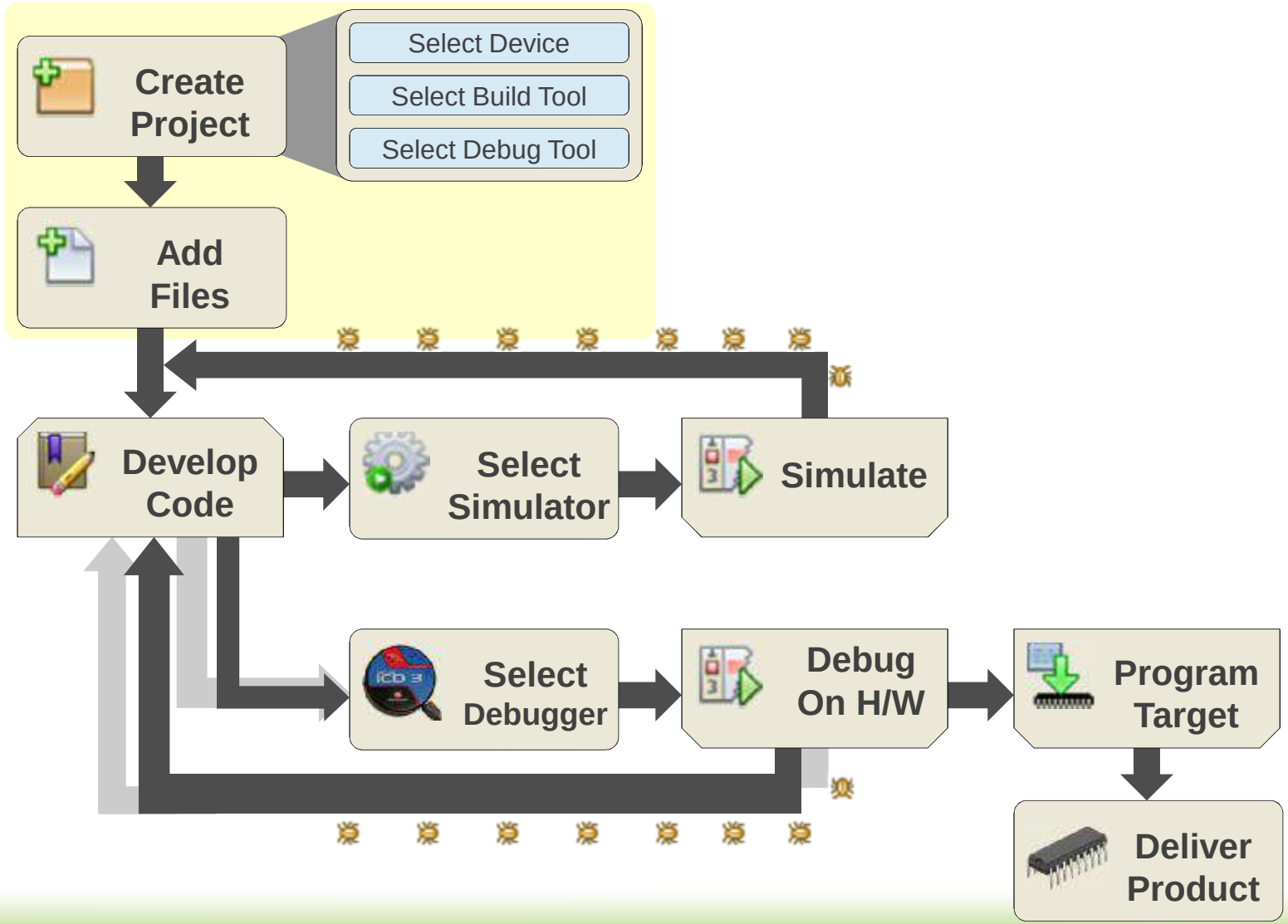
- **Основы Симулятора**
- Периферия
- Выводы Pin
- SCL/Stimulus
- MDB
- Новое в MPLAB X

Работа с MPLAB® X IDE
Installation

- **Поддержка различных платформ**
  - Windows XP*, Vista, 7, 8 (32 & 64-bit)
  - Linux
  - Mac OSX
- **Требует JRE (Java Runtime Environment)**
  - Включена в MPLAB X IDE (только для Windows)



MPLAB® X IDE
Процесс разработки



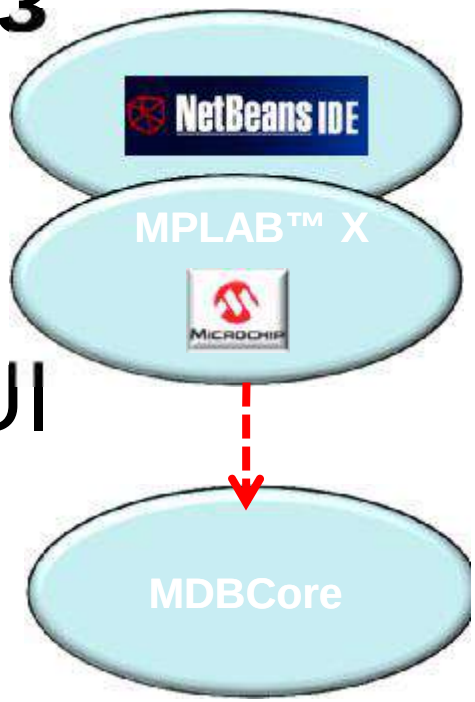
Основы Симулятора
Simulator

Что такое Симулятор?

- Отладочное средство, подобное внутрисхемным отладчикам MPLAB® REAL ICE™ и MPLAB ICD 3
- Программная симуляция PIC микроконтроллеров
- Может дать то, что не всегда доступно в железе
  - “Неограниченное” число точек останова
  - Возможность проведения тестов

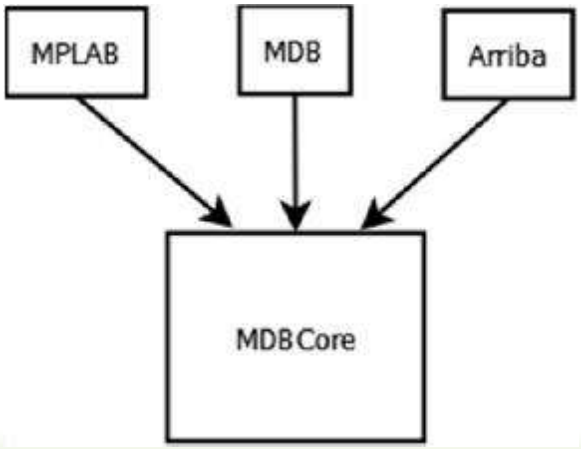
Основы
High Level View

- **MPLAB® X IDE состоит из двух частей**
- **MPLAB IDE**
  - Графическая оболочка GUI
- **MDBCore**
  - Ядро отладки



Основы
Headless

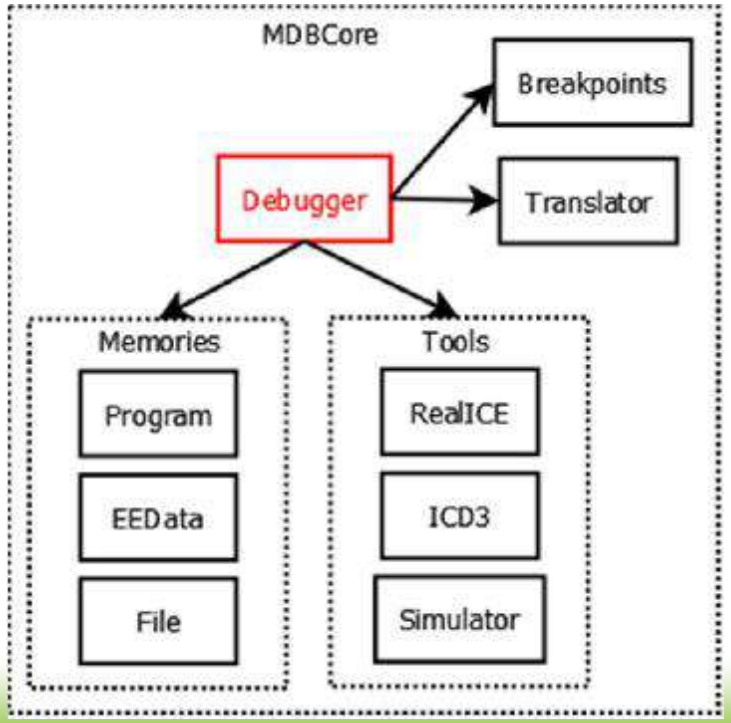
- **Позволяет менять пользовательский интерфейс**
  - MPLAB X
  - Microchip Debugger (MDB)
  - Arriba (Eclipse)



Основы
MDBCore

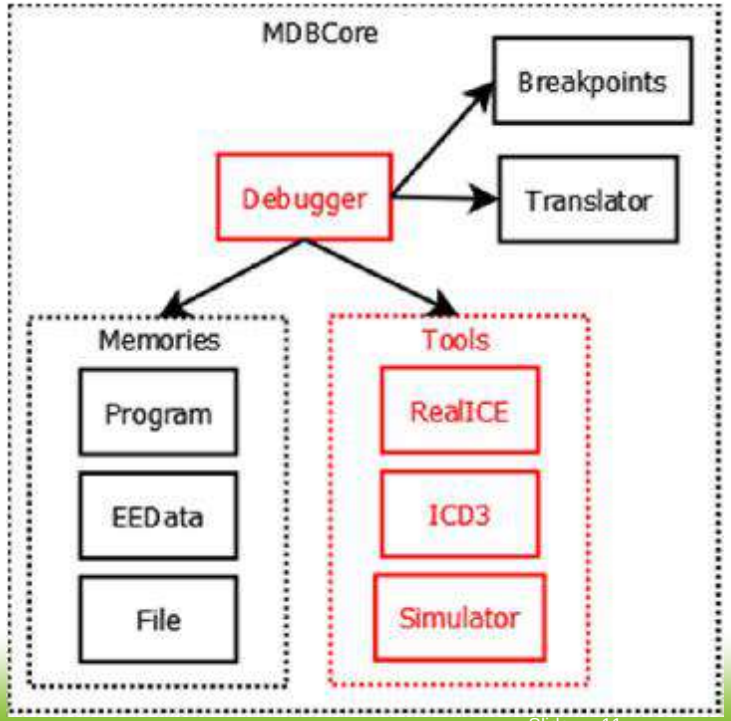
Более подробно о MDBCore

- Отладчик (Debugger) это то, с тем взаимодействует пользователь
  - Run
  - Instruction Step
  - Halt
- Так же
  - High Level Step
  - Step Over



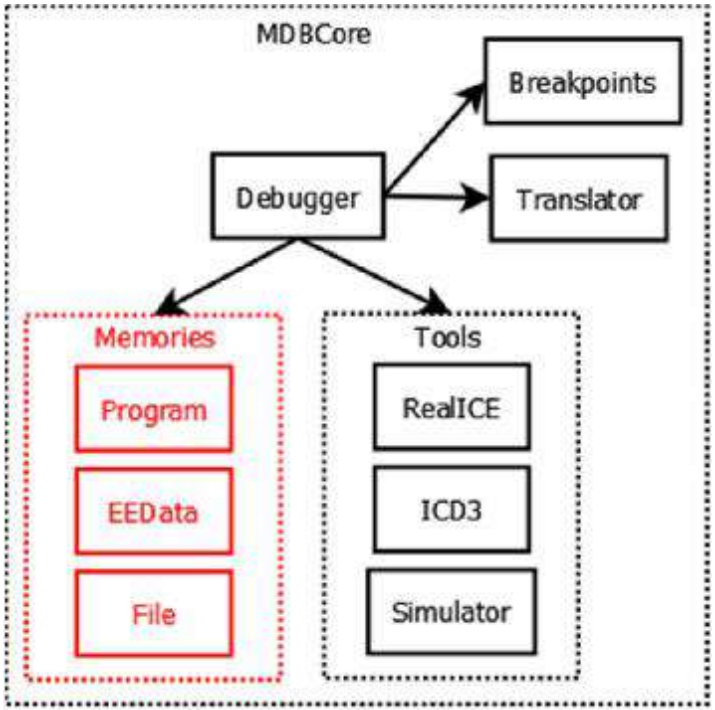
Основы
MDBCore

- Идея в том, чтобы Отладчик выступал Связующим звеном для всех инструментов в MPLAB®
- Инструменты отладки имеют однотипный интерфейс
  - Run
  - Step
  - Halt
  - Program
  - Read/Write



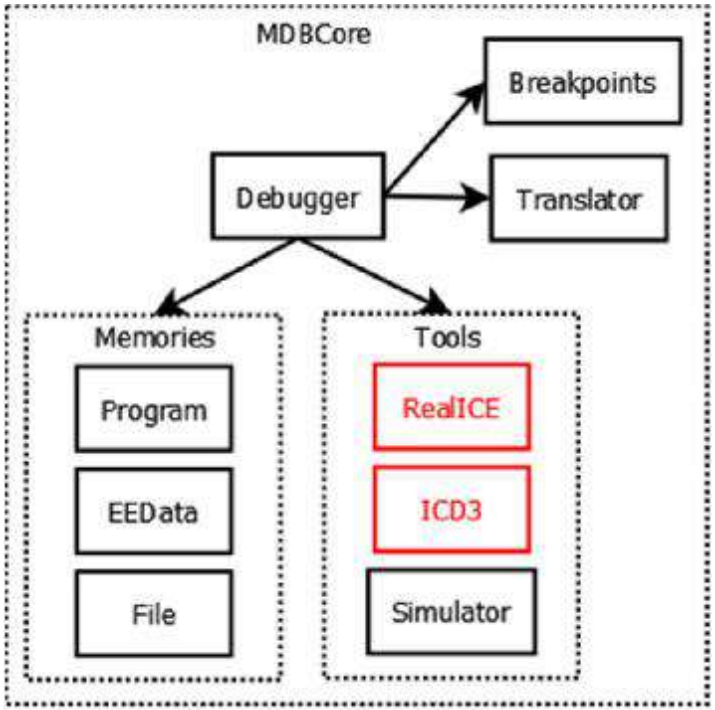
Основы
MDBCore

- Операция Build помещает образ программы в Память



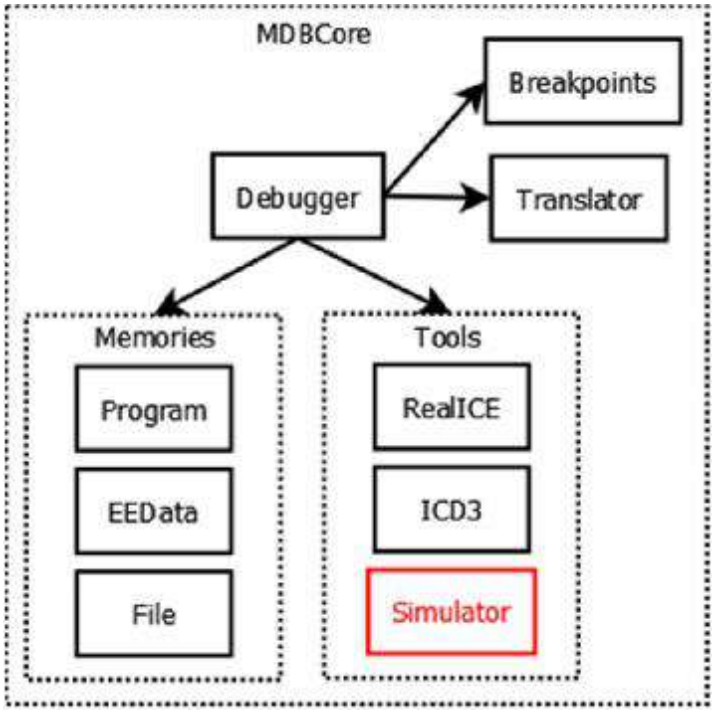
Основы
MDBCore

- Операция Build помещает образ программы в Память
- Аппаратные инструменты переносят образ в микроконтроллер



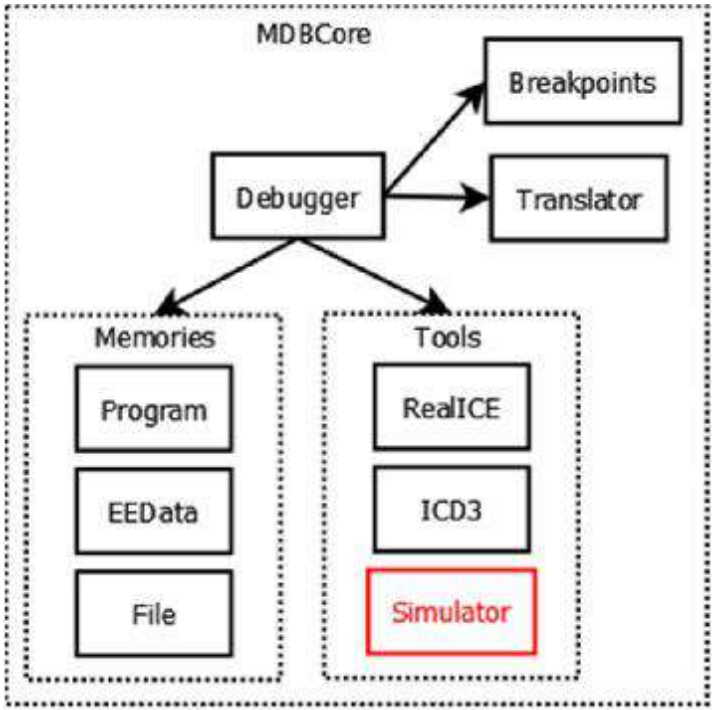
Основы
MDBCore

- Симулятор использует образ напрямую из памяти ПК
  - Reads/Writes – быстро!
  - Другие плюшки…



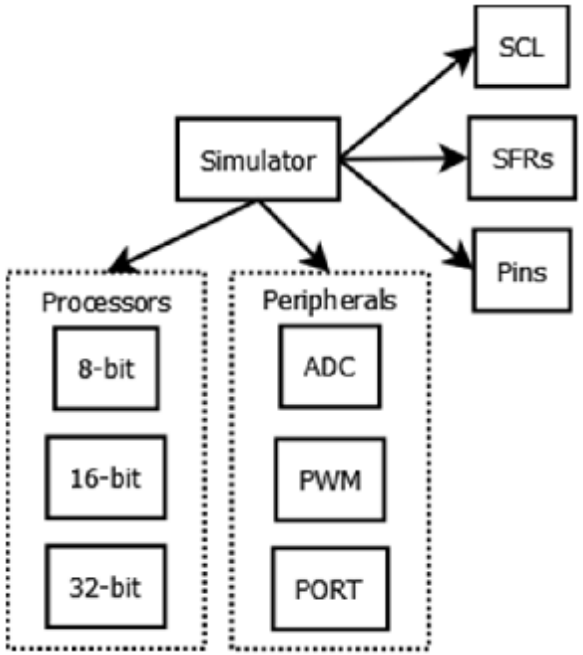
Основы
Simulator

- Давайте посмотрим на Симулятор «вооруженным взглядом» **J**



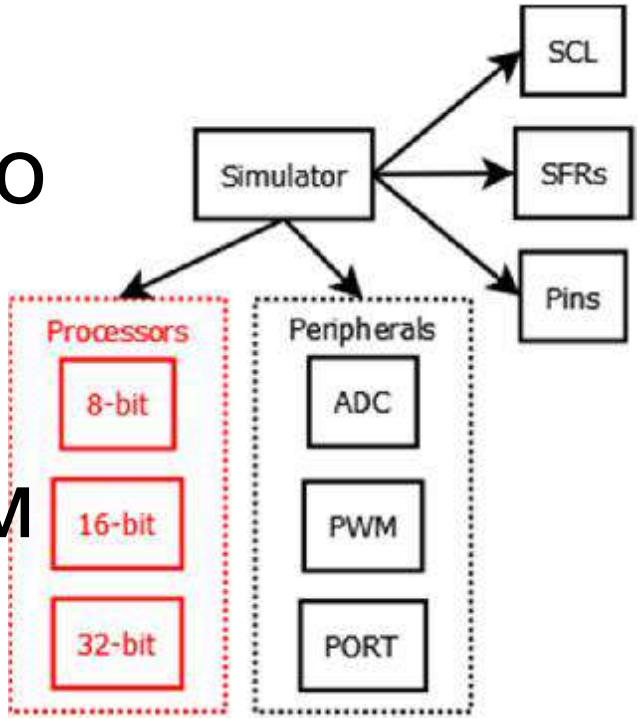
Основы
Simulator

- **Основная операция это шаг инструкции**
  - Run это повторяющиеся шаги
- **Результат с шагом выполнения инструкций**
  - Как и в реальном МК



Основы
Simulator

- **Ядро процессора**
  - Одинаковое для каждого семейства
  - Слежение за программным счетчиком (PC)
  - Декодирование команд
  - Обработка прерываний

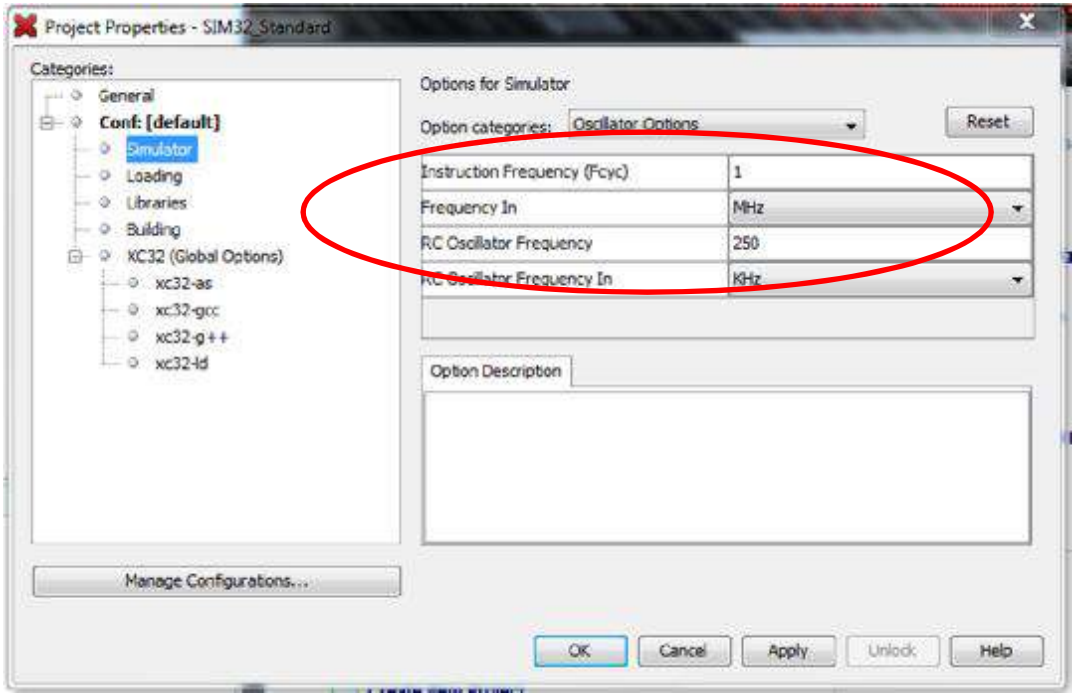


Основы
Simulator Timing

- **Симулятор оперирует с двумя типами скорости**
  - Simulated Time Per Instruction (TPI)
  - Executed Instructions Per Second (IPS)

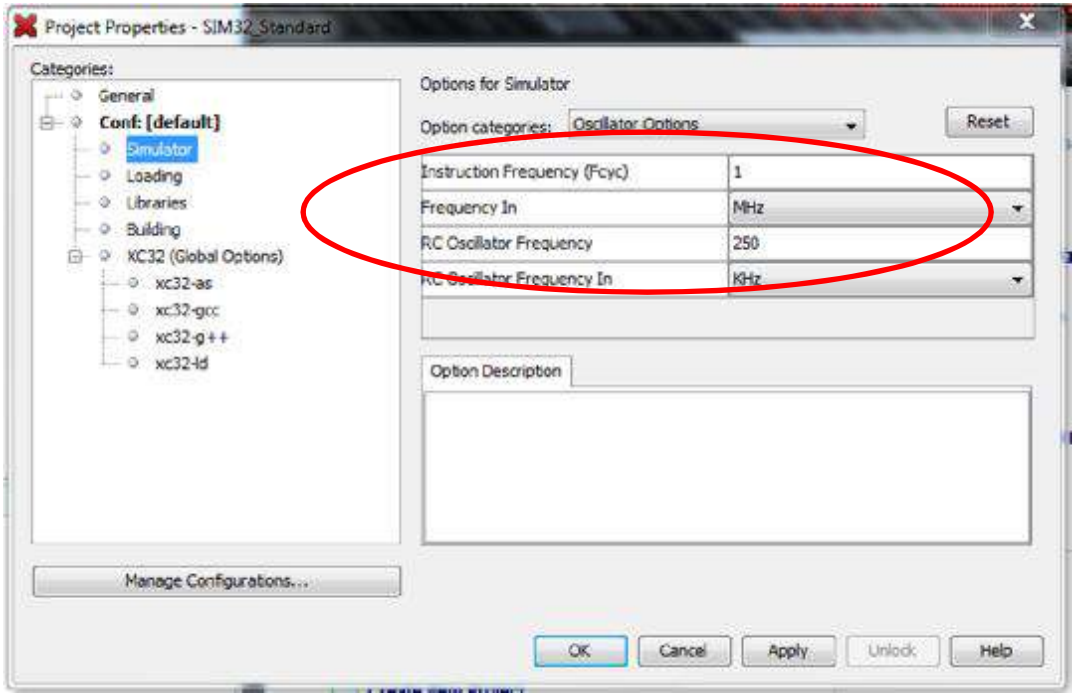
Основы
Time Per Instruction

- Симулируемое TPI устанавливается в свойствах проекта
  - Частота в ***Instructions*** per Second



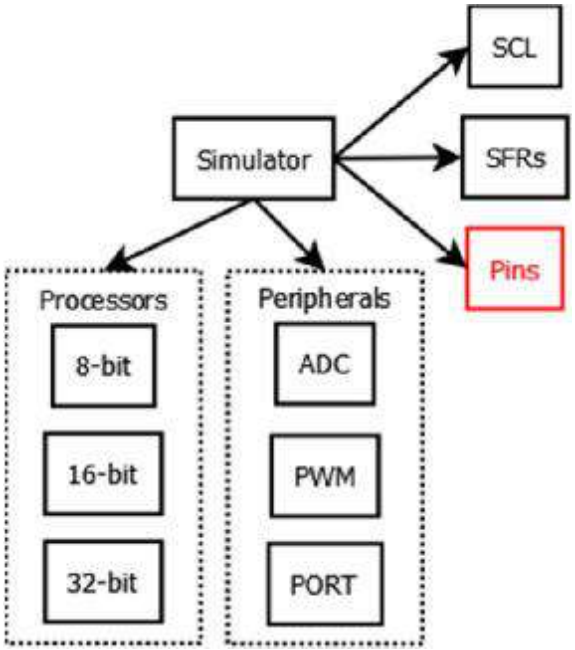
Основы
Time Per Instruction

- TPI используется для время-зависимой периферии (PWM, Timer, и т.п.).
- TPI не влияет на скорость симуляции (IPS)!



Основы
Pins

- **Моделирование входов (Pin) – новое MPLAB® X IDE**
  - Analog/Digital
  - Input/Output
  - Multiplexing

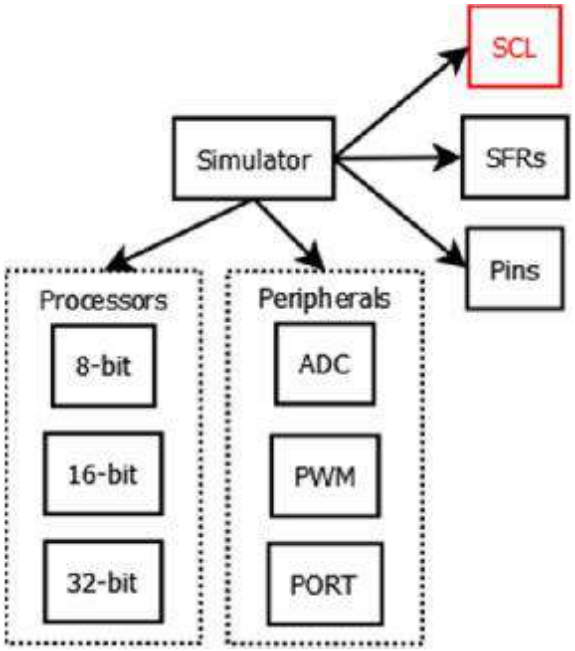


- **Периферия теперь использует pin**
  - Вместо битов регистров портов

Подробнее позже..

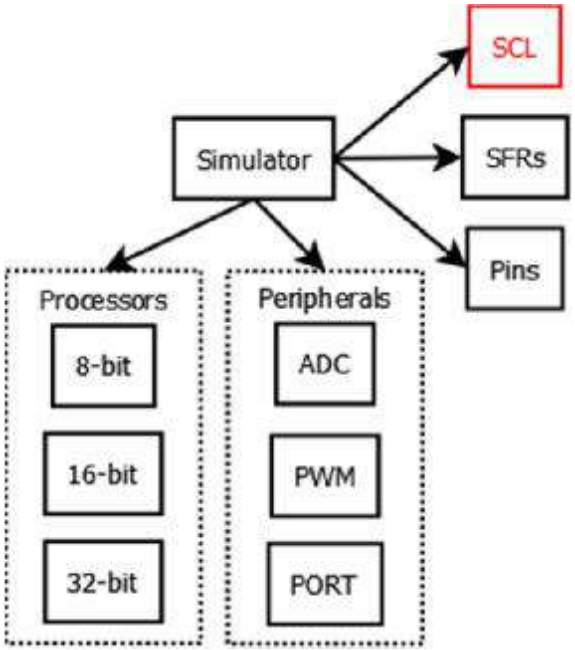
Основы
Stimulus Control Language

- **Инжектирует данные в процессе симуляции**
  - В входные Pin
  - Модифицирует память
  - Модифицирует переменные



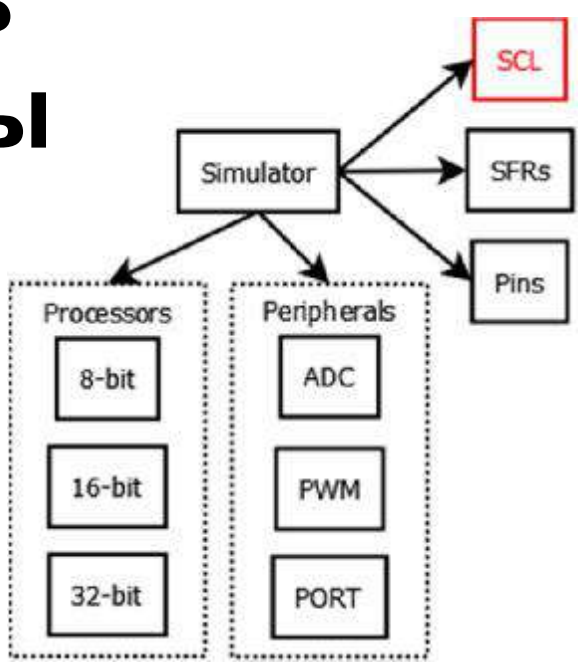
Основы
Stimulus Control Language

- **Инжекция привязывается к:**
  - Времени
  - Командным циклам
  - Событиям



Основы
Stimulus Control Language

- Это очень полезно!
- Позволяет проводить повторяющиеся тесты
- Позволяет моделировать различные ситуации
- и Много больше
  - Позже подробнее…



План

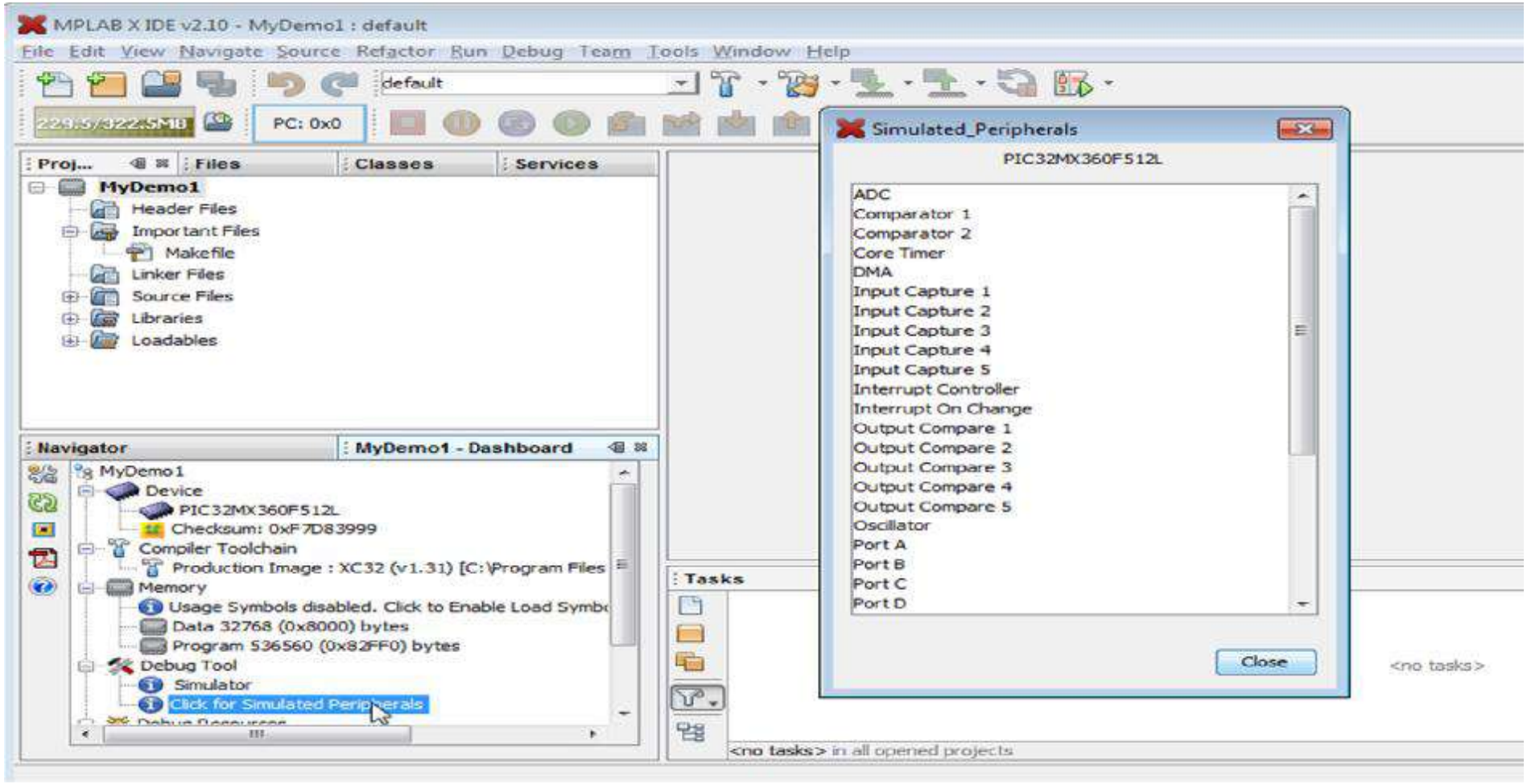
- Основы Симулятора
- **Периферия**
- Выводы Pin
- SCL/Stimulus
- MDB
- Новое в MPLAB X

Периферия
Coverage

- **Не вся периферия поддерживается (поддержка ~25 периферийных модулей)**

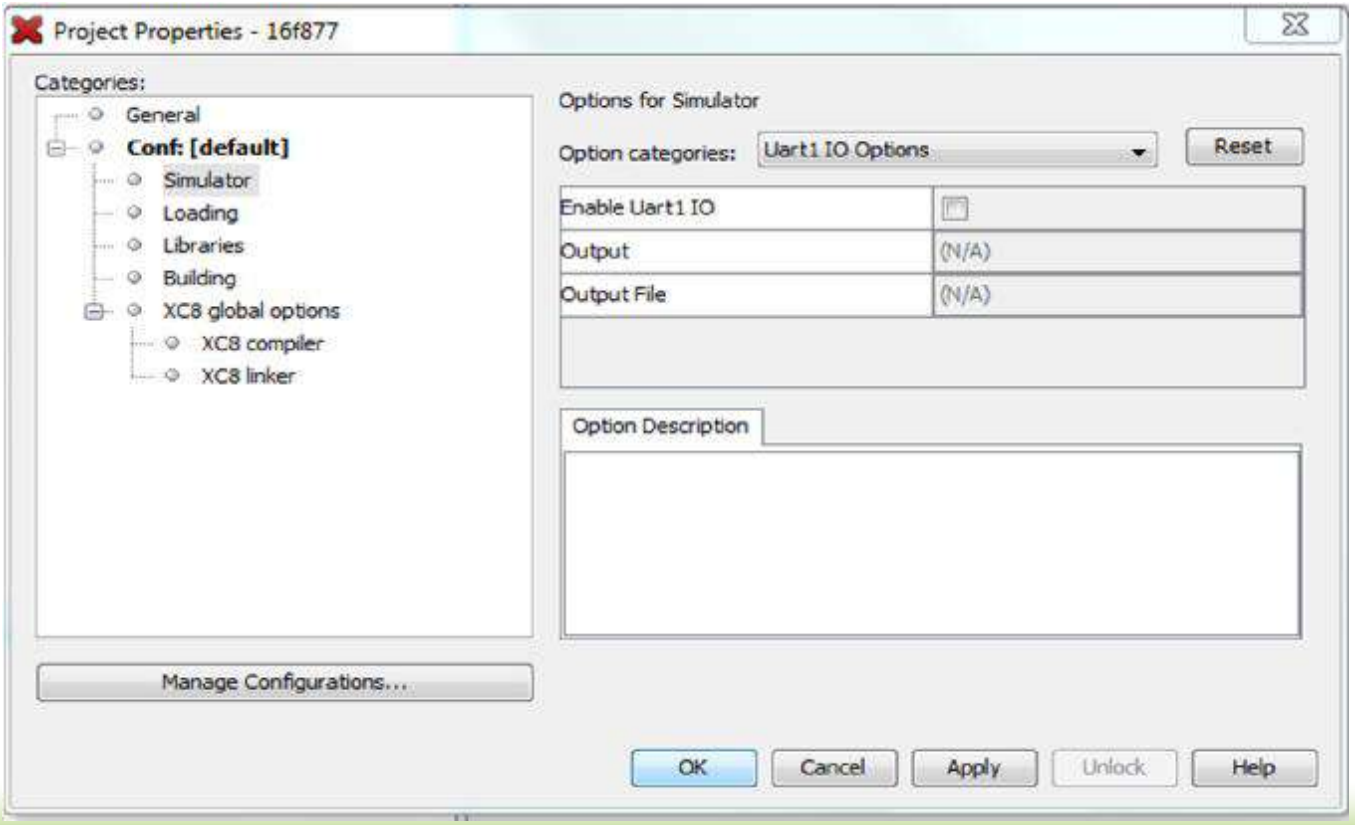
ADC	CCP	CodeGuard™
COG	COMP	CWG
DMA	ECCP	IC
INT	IOC	MCCP
NCO	OC	PORT
PPS	PWM	RTSP
TMR	UART	WDT

MPLAB® X IDE Simulator peripheral display



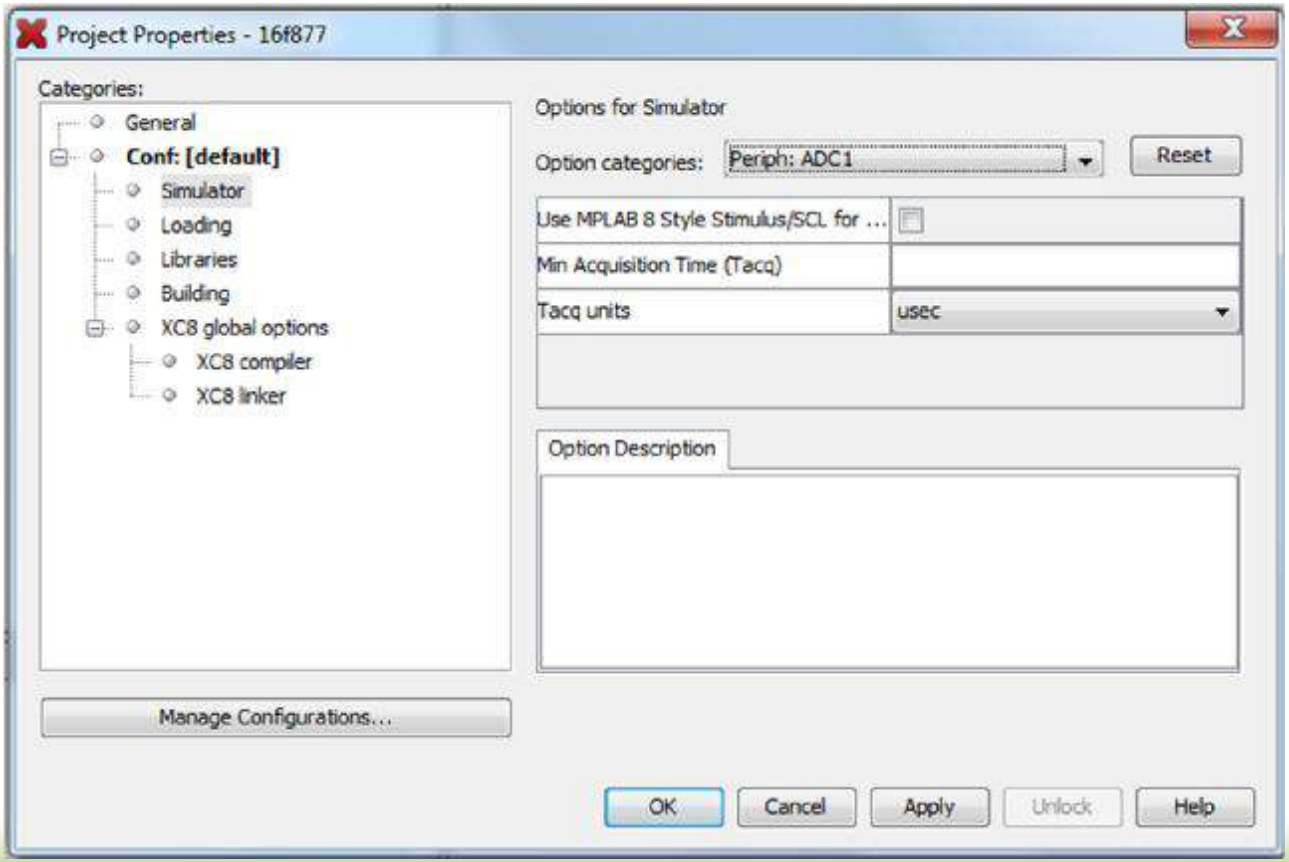
Периферия
Settings

- **UART имеет свои настройки**



Периферия
Settings

- **ADC еще один периф.модуль, имеющий свои настройки**



Периферия
ADC

ADC преобразует инжектируемое напряжение в pin

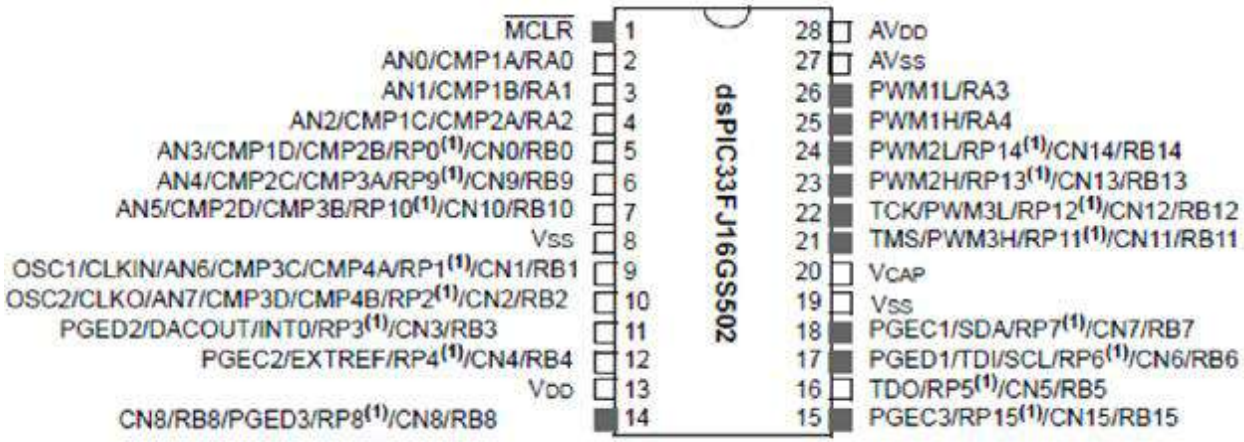
- **Раньше MPLAB® IDE не моделировал входы**
  - Вместо этого была доступна инжекция результата в буфер АЦП
  - “Use MPLAB 8 Style…” позволяет работать со старыми SCL (инжекция в буфер)

План

- Основы Симулятора
- Периферия
- **Выводы Pin**
- SCL/Stimulus
- MDB
- Новое в MPLAB X

Pin-ы

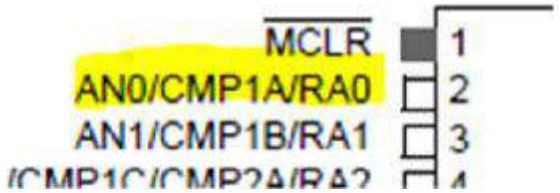
- PIC® работают с внешним миром через свои выводы



- Много функций мультиплексируются на выводы МикроКонтроллера

Pins

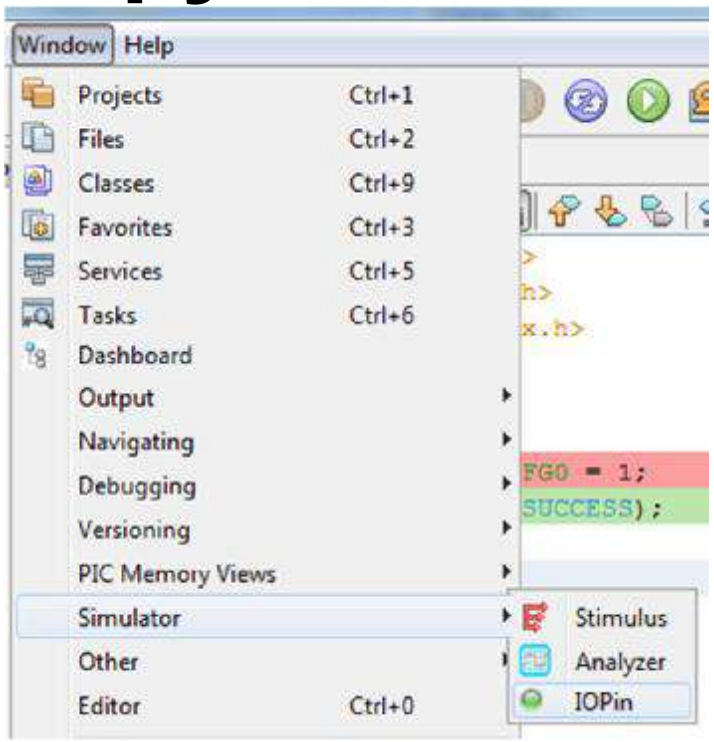
- **Мультиплексирование имеет приоритеты**

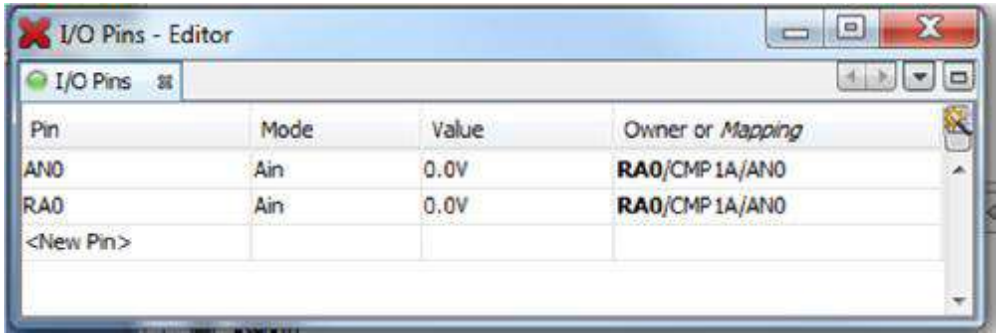


  - AN0 >> CMP1 >> RA0
  - Input или output

Pins

- **Симулятор позволяет отслеживать состояние и функции выводов**





Pins

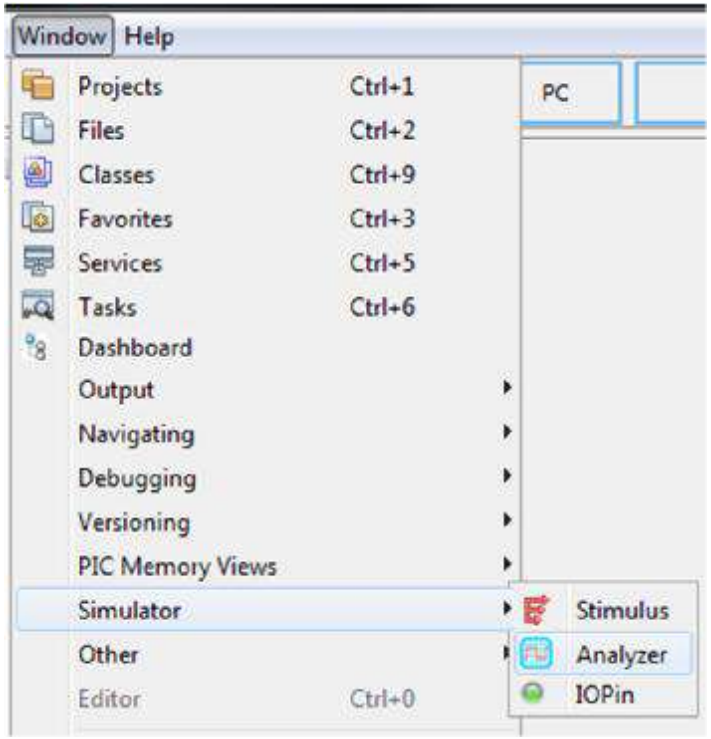
- **Отображается тип analog/digital и in/out**
- **Отображение приоритета**
  - низший приоритет – слева

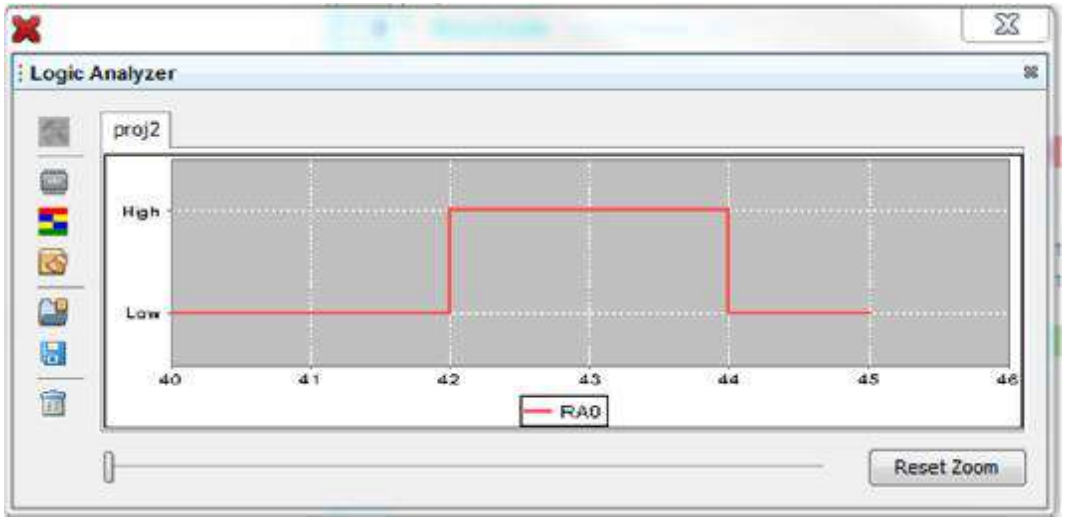
Pins

- **Отображается текущее состояние**
  - В Вольтах, если вход аналоговый
  - 0 или 1, если pin цифровой
- **Можно установить значение pin**

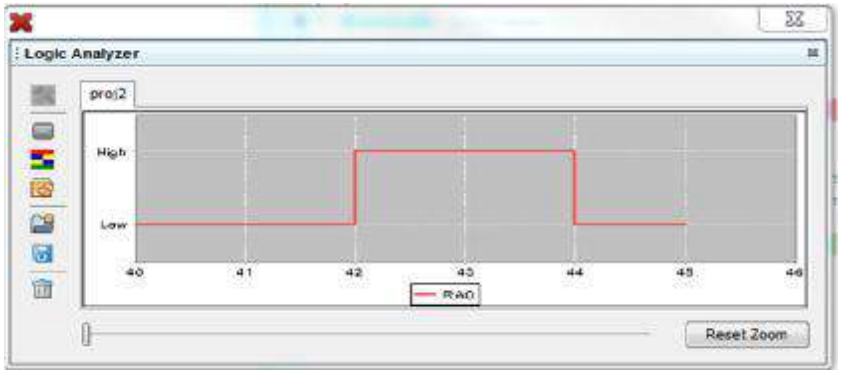
Pins

- **Logic Analyzer – инструмент, основанный на состоянии pin**





Pins



- **Logic Analyzer позволяет контролировать состояние pin во времени (подобно логическому анализатору)**

План

- Основы Симулятора
- Периферия
- Выводы Pin
- **SCL/Stimulus**
- MDB
- Новое в MPLAB X

SCL/Stimulus

В 2002 Microchip разработал Stimulus Control Language (SCL)

- VHDL – подобный язык, позволяющий инжектировать внешние данные в симулятор и не только…
- SCL несколько сложен…

SCL/Stimulus

В 2002 Microchip разработал Stimulus Control Language (SCL)

- Разработчик покинул Microchip вскоре после завершения реализации SCL
- Поддерживать SCL стало проблематично

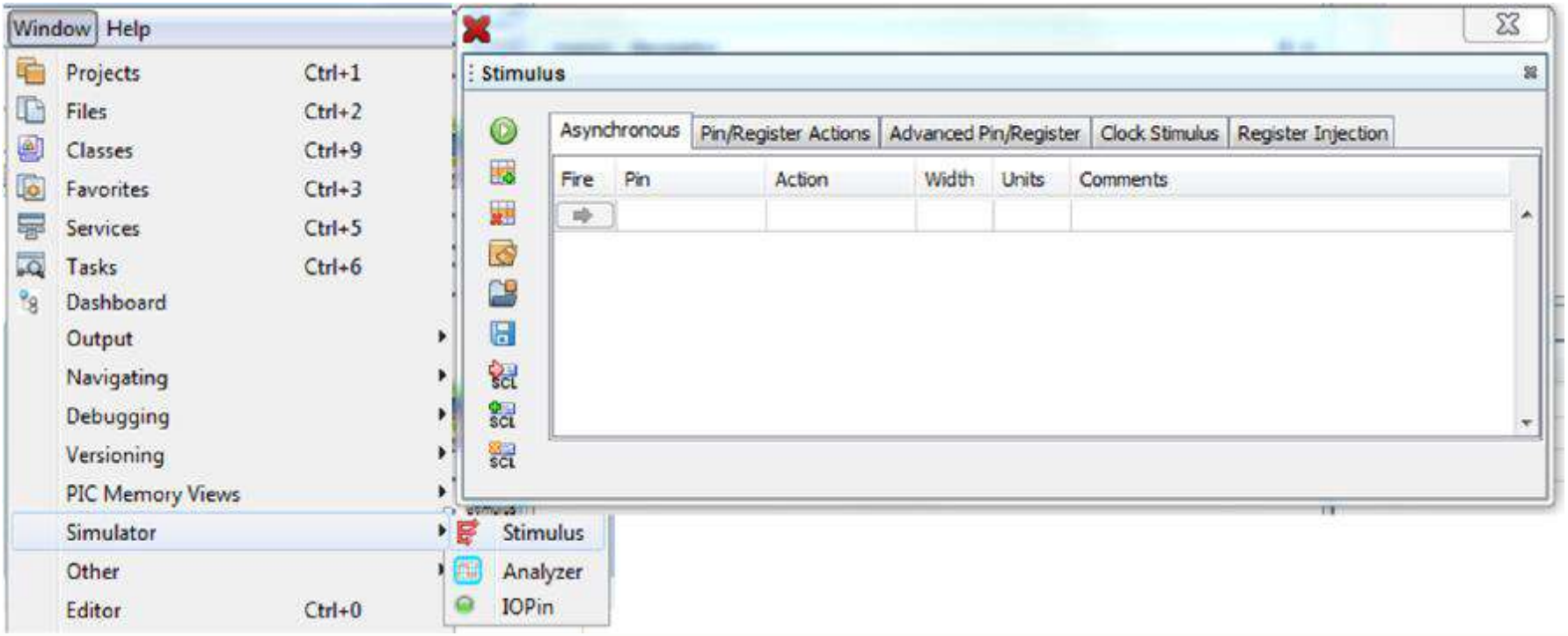
SCL/Stimulus

Microchip создал Графический интерфейс к SCL – Stimulus

- Позволяет незаметно для пользователя создавать SCL код
- GUI проще в понимании и проще документируется чем язык SCL

SCL/Stimulus

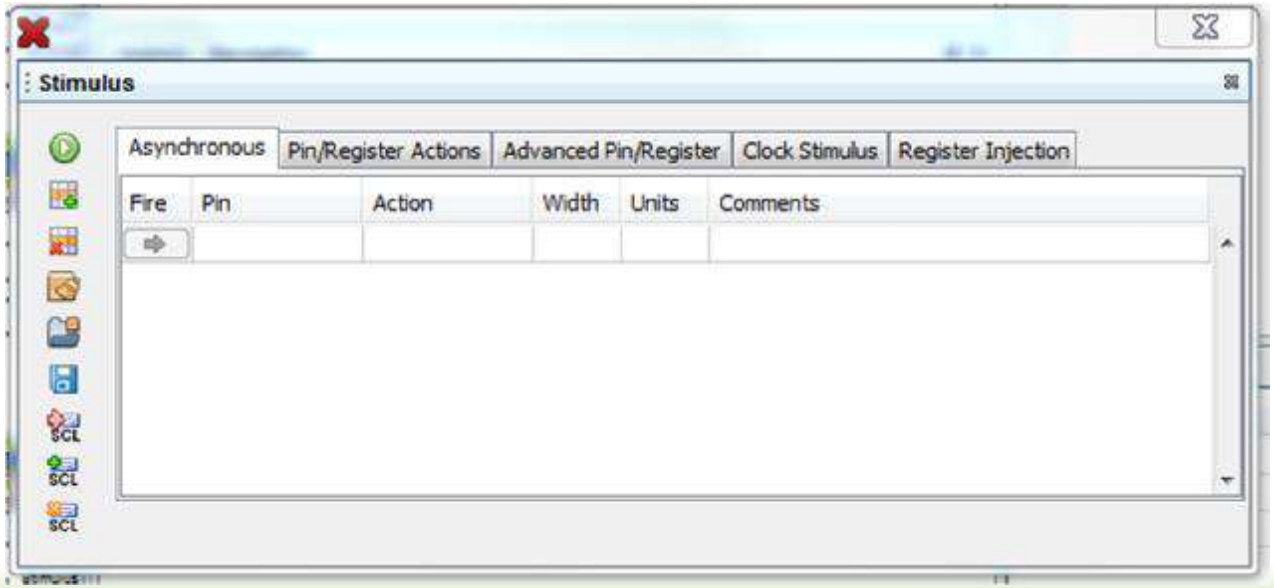
Stimulus активируется через Window -> Simulator -> Stimulus



SCL/Stimulus

5 вкладок содержат много возможностей

- Asynchronous – переключение Pin по требованию



SCL/Stimulus

5 вкладок содержат много возможностей

- Pin/Register Actions – переключение pin-ов или регистров по расписанию
- Advanced Pin/Register – больше возможностей по синхронному переключению pin/register
- Clock Stimulus – периодическая инжекция pin

SCL/Stimulus

5 вкладок содержат много возможностей

- Register Injection – инжекция регистров из файла

SCL/Stimulus

- **Asynchronous**

Первым делом выбирается pin для инжекции

- Ниспадающий список показывает все пины контроллера

SCL/Stimulus

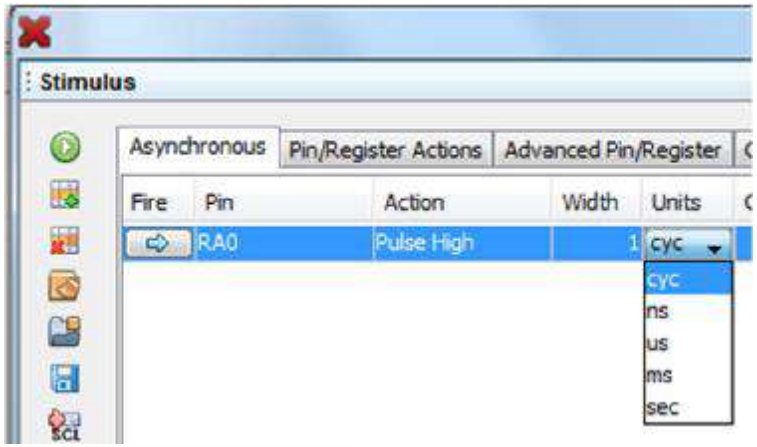
- **Далее – выбираем действие**
  - Set High – установить пин в 1
  - Set Low – установить пин в 0
  - Toggle – переключить состояние

SCL/Stimulus

- **… выбираем действие**
  - Pulse High – pin в 1 в течение заданного времени
  - Pulse Low – pin в 0 в течение заданного времени

SCL/Stimulus

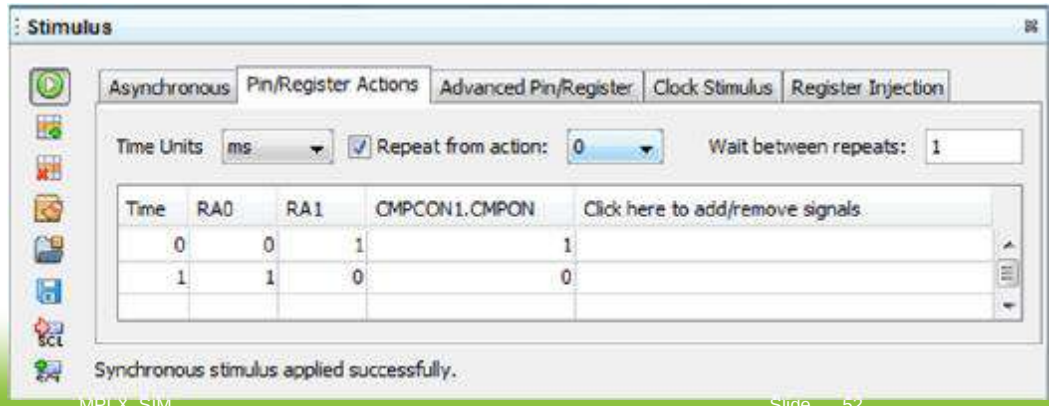
- **Если выбран Импульс – задается время**
  - В командных циклах или в “реальном” времени



SCL/Stimulus

Pin/Register Action позволяет инжектировать в pins/registers/fields основываясь на времени

- В командных циклах или в “реальном” времени
- Циклы могут повторяться



SCL/Stimulus

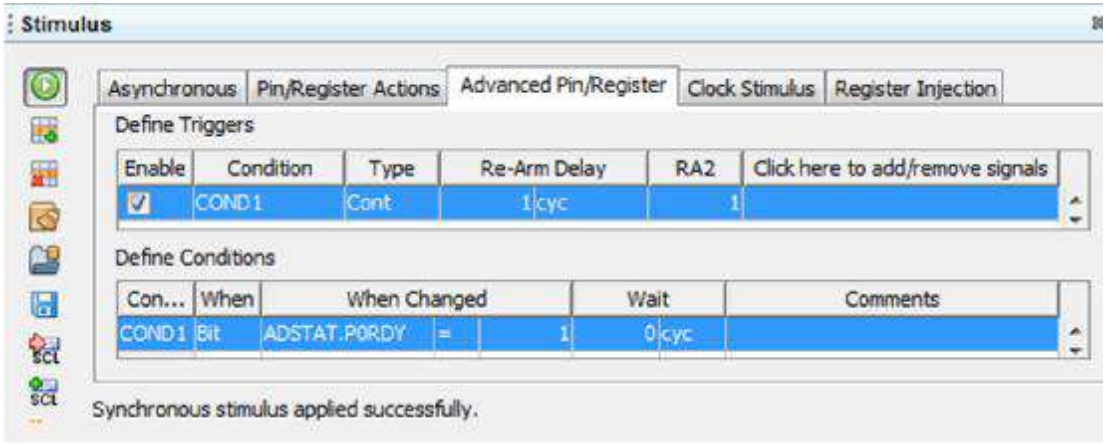
Pin/Register Action

Может менять значения регистров

- *Это симулятор - Не меняют данные в реальном МК!*
- Не асинхронные события требуют нажатия кнопки «применить» (apply)

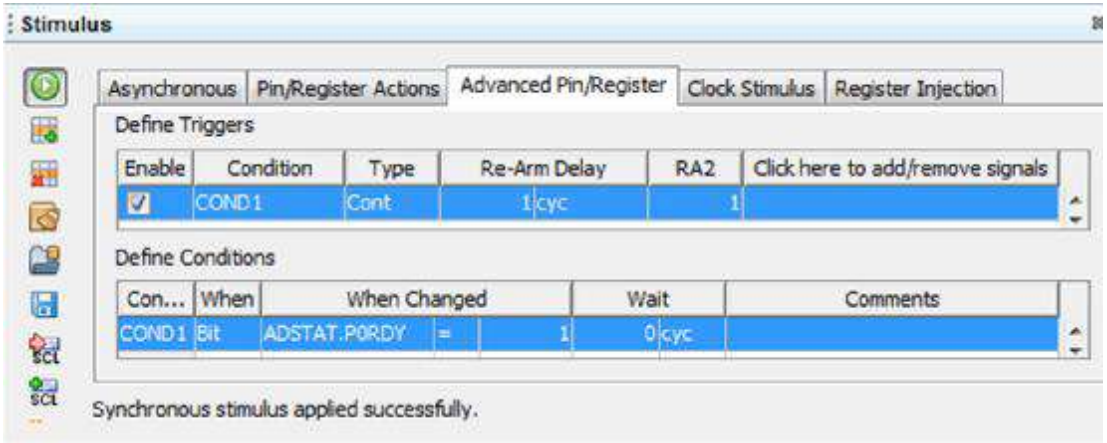
SCL/Stimulus

Advanced Pin/Register позволяют инжектировать данные основываясь на условиях



SCL/Stimulus

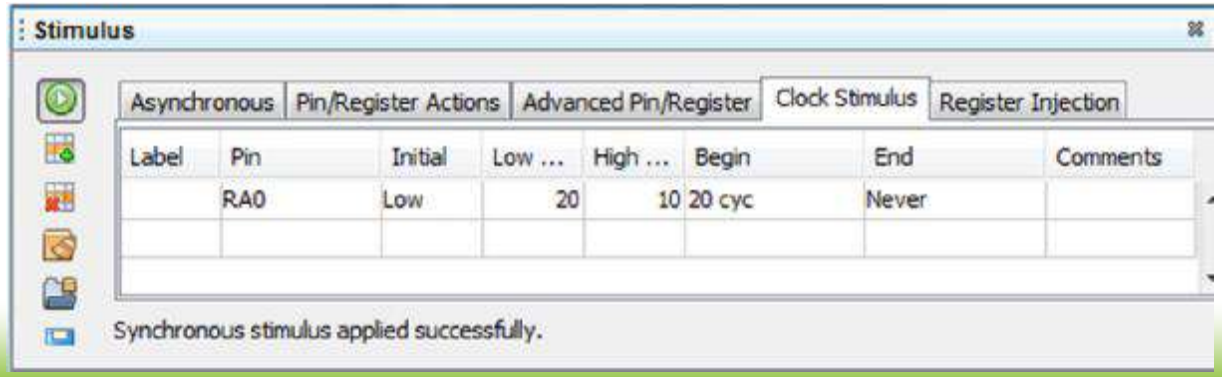
- **Два шага**
  - Определить условия для срабатывания триггера
  - Задать действия, выполняемые после срабатывания условия



SCL/Stimulus

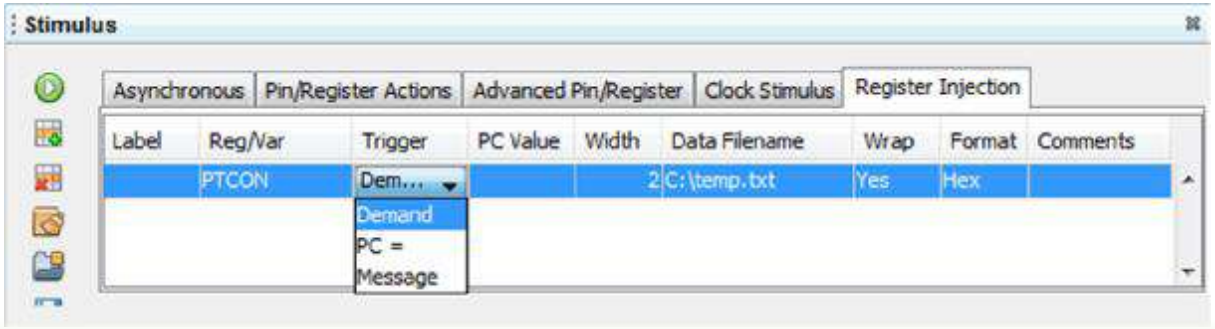
Clock Stimulus легкий путь для периодических инжекций

- **Определить**
  - Low/High счетчик циклов
  - Время Start/End



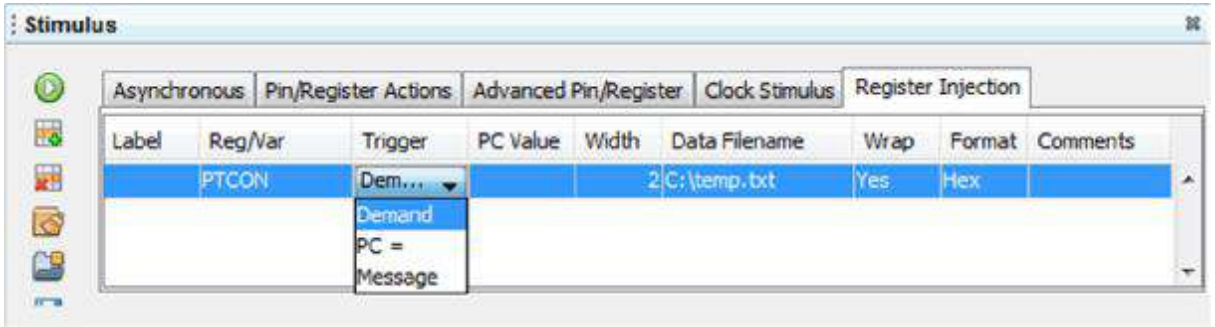
SCL/Stimulus

Register Injection предоставляет возможность инжекции данных из файла



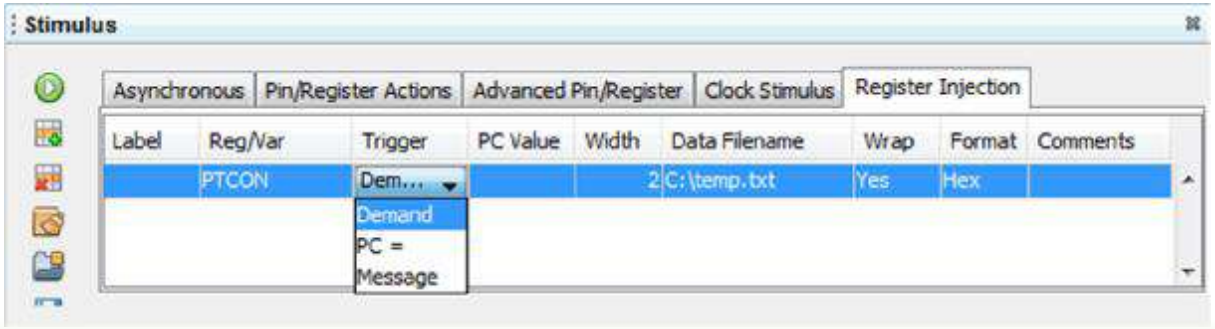
SCL/Stimulus

- **Три варианта**
  - Demand: когда читаем регистр
  - PC: когда PC = xxx
  - Message: триггер(ы) задан в файле данных



SCL/Stimulus

- **Применяется, например, при симуляции получения данных от периферии, подобной UART**



SCL/Stimulus

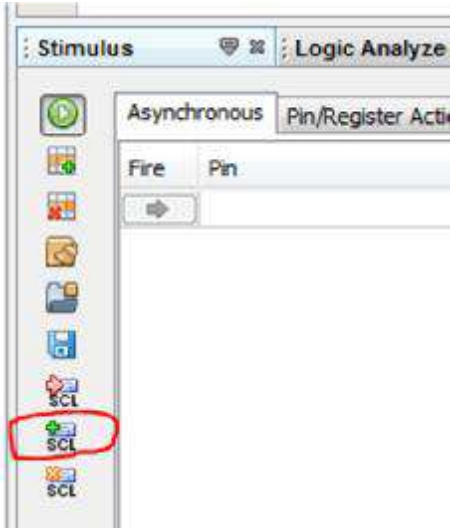
Stimulus подходит для простых задач

SCL намного мощнее

- Помните? Stimulus это только GUI (графическая оболочка) к языку SCL

SCL/Stimulus

Файл SCL это текстовый файл, который подключается к MPLAB® IDE через окно stimulus



SCL/Stimulus

При установке MPLAB® X IDE вы получаете SCL User's Guide

..\docs\SCL_Users_Guide

SCL/Stimulus

Простой SCL файл

```
testbench for "pic18f4620" is
  begin

    process is
      begin
        RA0 <= ‘1’;
        wait;
      end process;

  end testbench;
```

- Устанавливает RA0 в 1

SCL/Stimulus

В реальном МК действия происходят одновременно

А в Симуляторе все происходит последовательно

singleStep
- выполнение команды
  - инкремент stopwatch
  - обработка каждой периферии
  - выполнение SCL процессов

SCL/Stimulus

Симулируемая периферия и SCL немного замедляют время симуляции

singleStep
- выполнение команды
  - инкремент stopwatch
  - обработка каждой периферии
  - выполнение SCL процессов

SCL/Stimulus

SCL выполняется *после* выполнения инструкции и периферии

- Может преподнести сюрпризы
  - **Пример: переполнился таймер, выставлен флаг прерывания, затем SCL инжектирует новое значение в таймер**

SCL/Stimulus

Подобно C или Java в SCL запись

testbench for PIC32MX360F512L is end testbench;

это то же самое, что

testbench for PIC32MX360F512L is
end testbench;

SCL/Stimulus

Комментарии в стиле C или VHDL

```
// C style comment
-- VHDL style comment
```

SCL/Stimulus
Testbench

Каждая SCL программа содержит <u>один</u> testbench

```
testbench for PIC32MX360F512L is
    // processes, variables etc. go here
end testbench;
```

В MPLAB® X IDE имя контроллера носит декоративную роль

- Старые версии требовали соответствие имени МК установленному в проекте

SCL/Stimulus
Configuration

SCL программа *может* содержать конфигурацию

```
configuration for PIC32MX360F512L is
        // shared variables and labels go here
end configuration;
        // testbench follows
```

Содержит ссылки к variables и labels отлаживаемого кода

- Позволяет SCL процессам взаимодействовать с переменными и метками в коде

SCL/Stimulus
Process

Процессы это секции, где происходит вся работа

```
process is
    begin
        // process statements go here
    end process;
```

testbench может содержать несколько процессов

- Все активные процессы запускаются в каждый командный цикл

SCL/Stimulus
Process

Процессы это секции, где происходит вся работа

```
process is
    begin
        // process statements go here
    end process;
```

Выполняются последовательно, но псевдоконкурентно с точки зрения командного цикла

SCL/Stimulus
Process

Можно дать имя процессу

```
ProcessName: process is
    begin
        // process statements go here
    end process ProcessName;
```

- Имена должны быть разными
- Если нет имени, то SCL сгенерирует имя сам
- Возможно, что имена будут использоваться в следующих версиях

SCL/Stimulus
Wait

Оператор wait – наиболее важный оператор в SCL процессе

- **Когда выполнится последний оператор процесса, SCL начинает выполняться снова с первого оператора**

SCL/Stimulus
Wait

SCL процессы выполняются каждый командный цикл пока не столкнутся с wait

- отсутствие оператора wait означает бесконечное выполнение в одном командном цикле
- Результат – ошибка

SIM004: Failed to parse SCL SCL022: Process contains neither a wait statement nor a sensitivity list line(#)

SCL/Stimulus

Wait

4 типа оператора wait

```
wait;                    //unadorned
wait on sensitivity;     //sensitivity wait
wait until condition;    //condition wait
wait for timeout;        //timeout wait
```

SCL/Stimulus
Wait

wait;

- Сообщает SCL остановить процесс немедленно

```
process is
    begin
        wait; // terminate the process now!
    end process;
```

SCL/Stimulus
Wait

wait on *sensitivity*;

ждет *изменение* значения

```
process is
    begin
        wait on RD1;                    // pin RD1
        wait on uservar;                // user variable
        wait on STATUS;                 // STATUS register
        wait on PORTD.RD0;              // bit field RD0
    end process;
```

SCL/Stimulus
Wait

wait until *condition*;

- **Ждет когда условие станет верно**

```
process is
    begin
        wait until PORTA == 128;
        wait until RD1 == ‘1’;
        wait until ADCON.ADON == ‘1’;
        wait until PC == 4;
    end process;
```

SCL/Stimulus
Wait

wait for

- **Ждет заданное время**

```
process is
    begin
        wait for 10 ic;          // 10 instruction cycles
        wait for 10 ms;          // 10 milliseconds
    end process;
```

SCL/Stimulus
Wait

Можно комбинировать wait-типы

```
process is
    begin
        wait on RD1 for 10 ms;
        wait until PC == 10 for 20 ic;
    end process;
```

SCL/Stimulus
Время

```
10 ic;          // 10 instruction cycles
10 us;          // 10 microseconds
10 ms;          // 10 milliseconds
```

аналогично ps, ns, sec, min, hr

Время может быть случайным

```
random_time(lower,      // lower range value for random value
    upper,              // upper range value for random value
    units,              // units of desired time value (ns, ms, s, )
    seed1,              // random generator seed #1
    seed2,              // random generator seed #2
    result);            // result variable
```

SCL/Stimulus
Время

Функция **now()** *возвращает текущее время симуляции*

```
variable timeMark : cycle;

timeMark := now(); // зафиксировали время
...
if (now() > timeMark + 20 ic)
        then  // выполнить это если прошло
              // больше чем 20 Tcy после
              // зафиксированного времени
```

SCL/Stimulus
Assignment

Операции присвоения.

- **Присваивают значения на входы, регистры, память**

```
RA1 <= '1';                    // pin RA1 high
AN0 <= 3500 mv;                // pin AN0 to 3.5 V
ADDRESH <= 128;                // ADDRESH to 128
ADDRESH <= #16#80#;            // ADDRESH to 0x80
ADDRESH <= B"10000000";        // binary 10000000
SSPCON1.CKP <= '1';            // SSPCON1.CKP to 1
```

SCL/Stimulus

Wait

Присвоения значений переменным имеют чуть другой формат

```
intVar    := 123;       // set intVar to 123
timeVar := 10 ms;    // set timeVar to 10 ms
strVar    := "test";    // set strVar to "test"
```

SCL/Stimulus
If

Логика принятия решений представлена конструкцией

if-else

```
if booleanExp then
    scl statements
elsif booleanExp then        // optional
    scl statements
else                         // optional
    scl statements
end if;
```

SCL/Stimulus
If

Логика принятия решений представлена конструкцией

```
wait on RD0;            // make RD1 inverse of RD0
if RD0 == '0' then
    RD1 <= '1';
else
    RD1 <= '0';
end if;

if VDD <= 2500 mv then     // Simulate power down
    STATUS.PD <= '1';      // power down status
end if;
```

SCL/Stimulus
Loop

Циклы формируются с помощью loop

```
loop
    scl statements
    exit when booleanExp;          // optional
end loop;

// infinite loop (no exit when)
loop
    RD0 <= '0';
    wait on RD0;
end loop;
```

SCL/Stimulus
Loop

Циклы формируются с помощью loop

```
//clock until pc == foo label
loop
   RD0 <= '0';
   wait 4 ic;
   RD0 <= '1';
   wait 4 ic;
   exit when PC == foo;
end loop;
```

SCL/Stimulus
Loop

Циклы формируются с помощью loop

```
 //unadorned exit
loop
   AN1 <= 3500 mv;
   exit;              // unconditional exit
end loop;
```

SCL/Stimulus
Loop

Может иметь несколько точек выхода

```
//multiple exit conditions
loop
   wait on RD1;
   RD0 <= '0';
   exit when STATUS.Z == 0;
   exit when RD2 == '0';
end loop;
```

SCL/Stimulus
While

Выражение while. Подобно loop, но с проверкой логического условия

```
while booleanExpression loop
  scl statements
  exit when booleanExpression     // optional
end loop;
```

- Условие while может иметь выход when

SCL/Stimulus
While

Выражение while. Подобно loop, но с проверкой логического условия

```
//loop until PIR2.EEIF is cleared
while PIR2.EEIF == 1 loop
   RD1 <= '1';
   wait on RD0;
   exit when PC == 0;   // device reset
end loop;
```

SCL/Stimulus
Types

SCL позволяет пользовательские типы и переменные *(types и variables)*

- Перечисление

```
type color is (red, blue, green);
```

- Массивы

```
type bit_vector is array (integer range <>) of bit;
```

SCL/Stimulus
Types

Диапазоны

```
type Byte is range 0 to #16#FF#;
```

- Диапазоны могут иметь определение в виде размерности

```
type voltage is range -10000000 to 10000000
   units mV;
      V  = 1000 mV;
      kV = 1000 V;
   end units;
```

SCL/Stimulus
Variables

Определение переменных

variable *varName* : *type*;

variable intVar : integer;

- Тип может быть пользовательским или предопределенным
- Переменные для SFRs, поля и pin- автоматически создаются SCL

SCL/Stimulus

Predefined Types

bit
bit_vector
boolean
byte
character
cycle
daddress
file_open_kind
file_open_status
frequency
integer
line
paddress
severity_level
string
text
time
vector_file_mode
voltage
word

SCL/Stimulus
Constants

Представление констант в SCL

```
intVar    = 1;                  // decimal 1
intVar    = B"10";              // binary 10
intVar    = #8#70#;             // octal 70
intVar    = #10#10#;            // decimal 10
intVar    = #16#FFFFFFFF#;      // hexadecimal
intVar    = #16#FFFF_FFFF#;     // '_'s ignored!
intVar    = 1e9;                // scientific notation
```

SCL/Stimulus
Constants

Представление констант в SCL

```
charVar   = 'a';          // character 'a'
stringVar = "SCL";        // String
boolVar   = true;         // enumeration
pin       = '1';          // pin hi
pin       = '0';          // pin lo
```

SCL/Stimulus
Constants

SCL не поддерживает десятичные дроби

- Как установить на pin напряжение 3.5В ?

VDD <= 3500 mv;

SCL/Stimulus
Shared

SCL имеет доступ к переменным и меткам отлаживаемого кода

- **Нужно объявить в конфигурации общие переменные и метки**

shared variable *varName*;
shared label *labelName*;

SCL/Stimulus
Shared

SCL в MPLAB X так же позволяет иметь доступ к GPR регистрам через shared locations

shared location *varName* := *locationAddress*;

SCL/Stimulus
Shared - Variable

```
configuration for "PIC16F877" is
   shared variable stateVar;
   end configuration;

testbench for "PIC16F877" is
   begin
   process is
      begin
      if stateVar == 0 then
         RD0 <= 1;
      else
         RD1 <= 0;
      end if;
      end process;
   end testbench;
```

SCL/Stimulus

Shared - Label

```
configuration for "PIC16F877" is
   shared label main;
   end configuration;

testbench for "PIC16F877" is
   begin
   process is
      begin
      wait until PC == main;
      report("reached main");
      end process;
   end testbench;
```

SCL/Stimulus

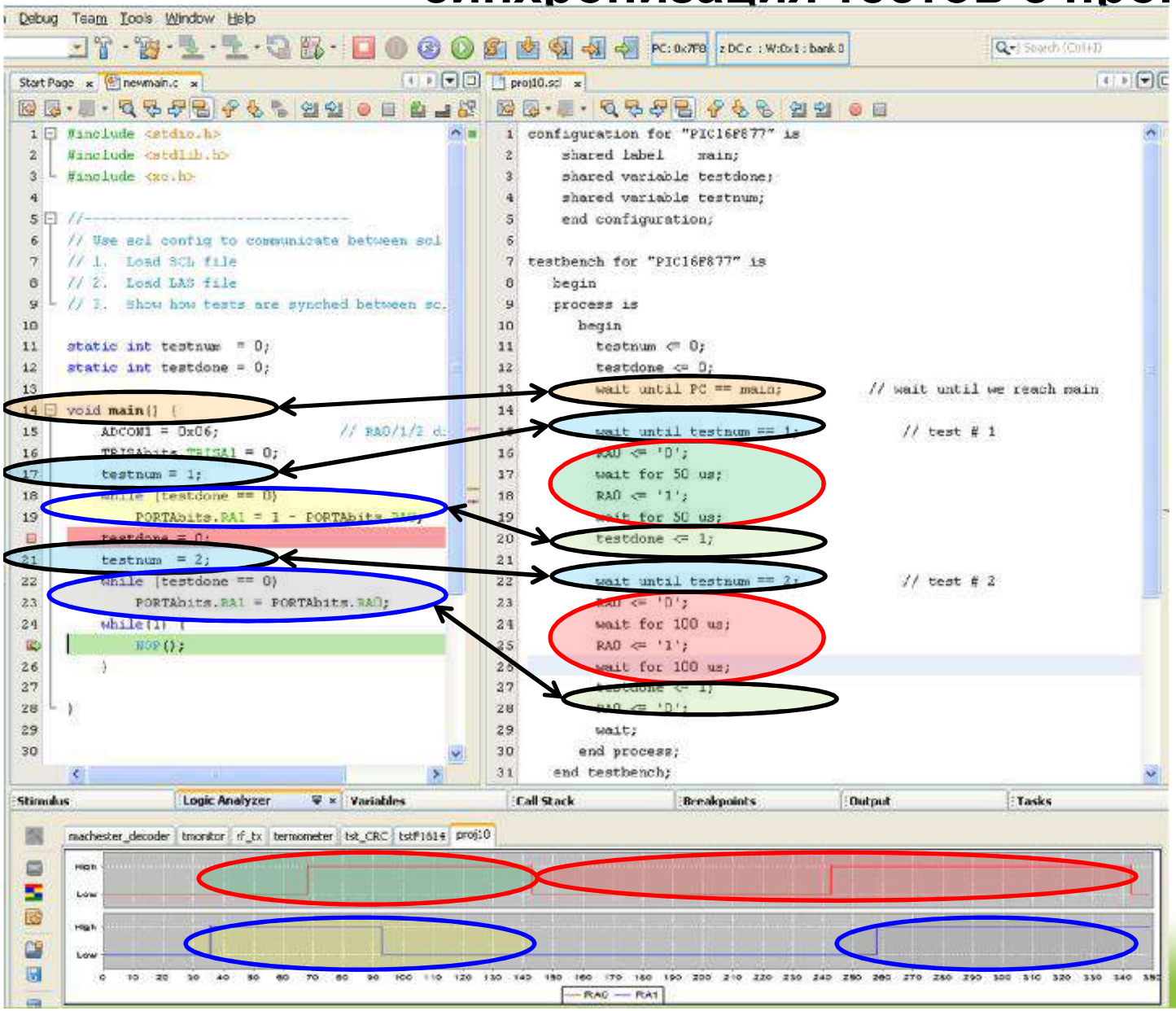
Shared - Location

```
configuration for "PIC16F877" is
   shared location loc70 := #16#70#;
   end configuration;
testbench for "PIC16F877" is
   begin
   process is
      begin
      wait on loc70;
      if loc70 == 0 then
         loc70 <= 1;
      else
         loc70 <= 0;
      end if;
      end process;
   end testbench;
```

MICROCHIP
MASTERs 2014

SCL/Stimulus

синхронизация тестов с программой



Тест 1

1. Ждем main
2. Ждем testnum == 1
3. Симулируем переключение RA0
4. Тест 1 завершен

Тест 2

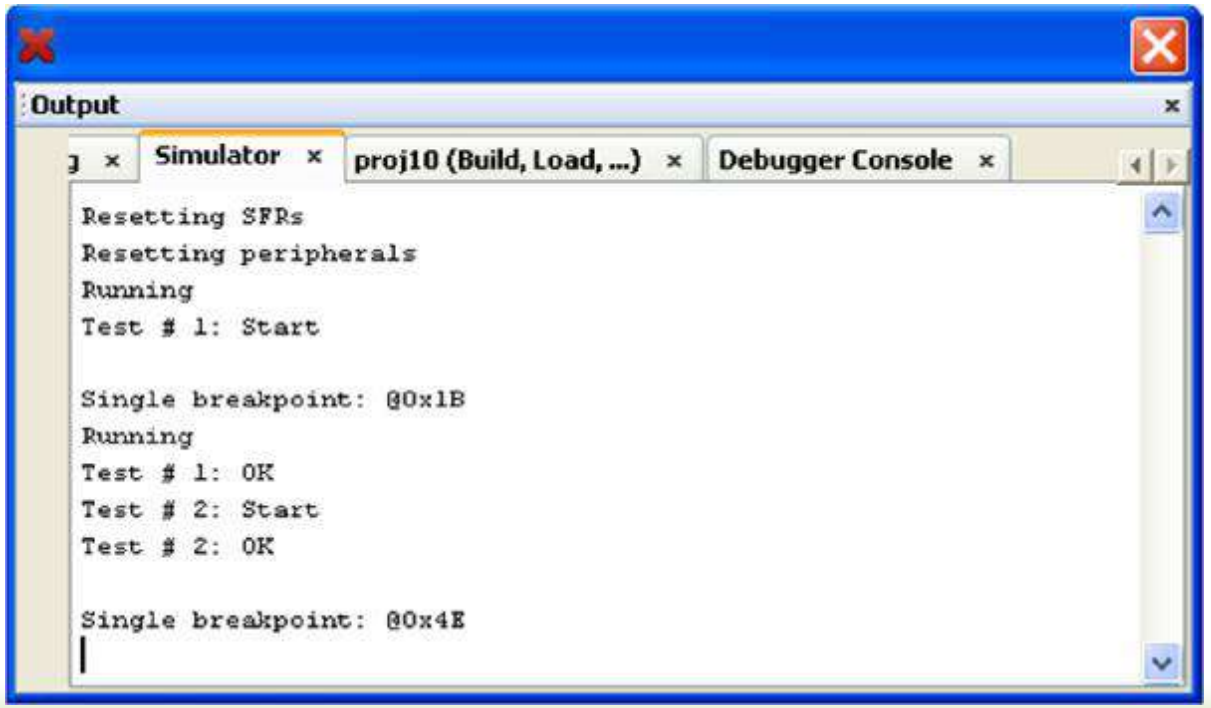
1. Симулируем переключение RA0
2. Пока длится - RA1 переключается в программе
3. Завершаем Тест 2

SCL/Stimulus
report

SCL может выводить сообщения в окно Output в MPLAB X

report(string); *// string is output to simulator console*

report(“Test 1: Start”);
report(“Test 1: OK”);



SCL/Stimulus
работа с файлами

Открытие файла

```
file_open(fileOpenStatus,   // file_open_status
    variable fileVar,       // file variable
    fileName,               // file name,
    String variable fileMode) // file mode (must be read_mode)
```

Закрытие файла

```
file_close(fileVar)
```

SCL/Stimulus
работа с файлами

Содержимое файла инжектируется в SFR по «запросу» – каждое обращение к SFR читает следующее значение из файла

```
accessin(filename,   // filename (String)
         mode,       // data format
         sfr,        // sfr to receive file data
         rewind);    // true => reuse file data after eof
accessin("C:/MyProj/MyData.dat", hex_mode, RXB0D0, true);
```

Отсылает строку в SFR

```
packetin(inString,   // String to be parsed
         sfr,        // SFR to be injected
         append);    // boolean, if false, clear previous packets
packetin(inString, RCREG, true);
```

Удобно для инжекции данных в UART. Может эмулировать потерю пакетов

SCL/Stimulus
работа с файлами

Чтение строки

```
readline(fileVar,      // file variable of file to read
         inStr)        // string variable to receive line
```

Чтение данных (3 варианта):

```
read(inStr,    // String to parse
     result)   // variable to receive newly parsed value

read(inStr,    // String to parse
     result,   // variable to receive newly parsed value
     status)   // boolean, true if parse was successful

read(fileVar, // file variable of file to be read
     location) // sfr or shared location to receive value
```

SCL/Stimulus
работа с файлами

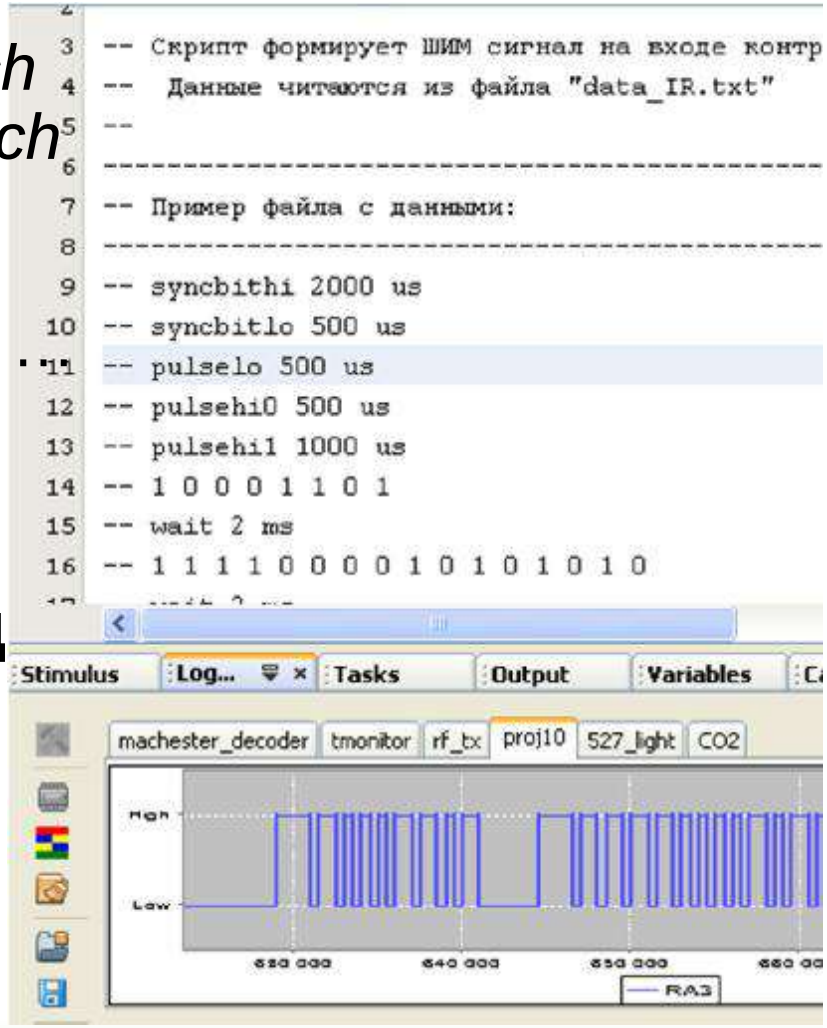
Нахождение subString в inString

```
match(inString,      // string to search
    subString)       // pattern to match
```

Разбор файла

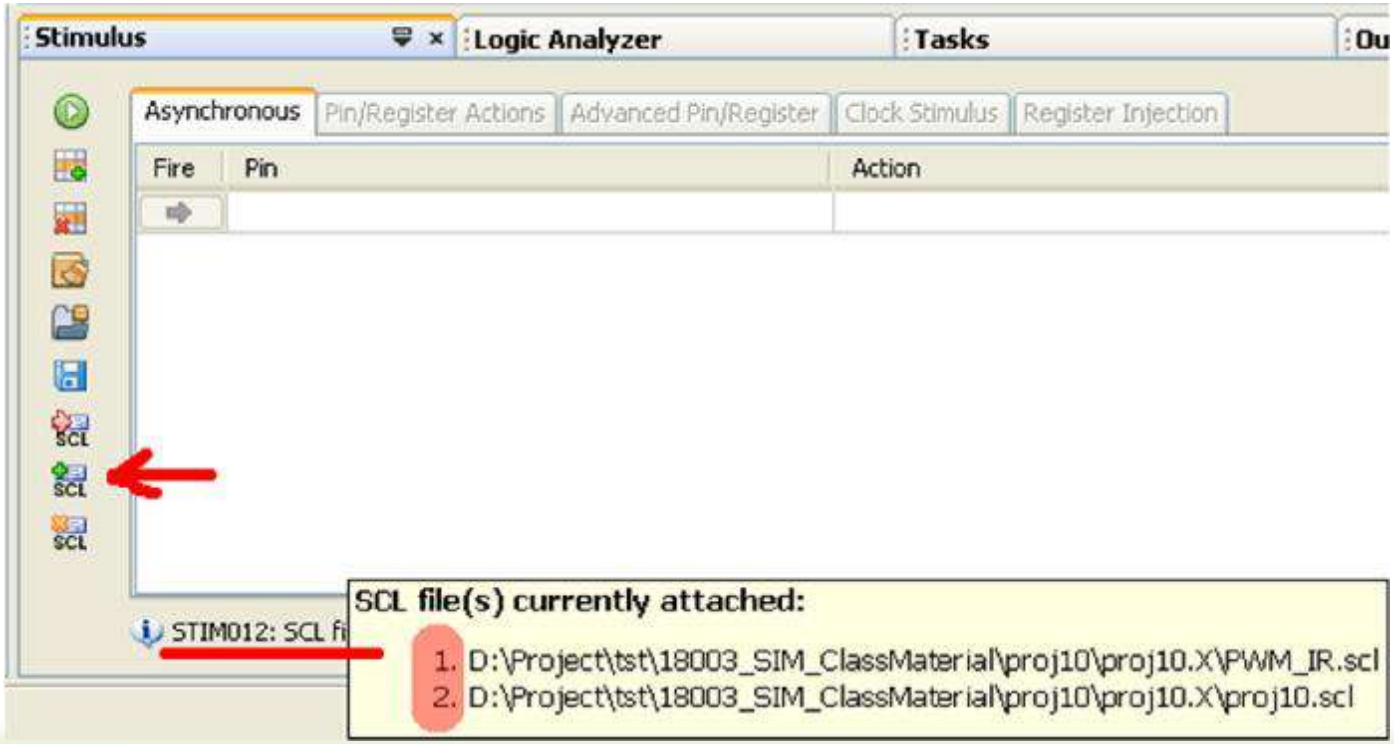
```
if match(lineVar, "wait") == true then ...

    read(lineVar, Variable);
```

Удобно для выдачи команд SCL -программе



SCL/Stimulus

Можно подключать несколько SCL – файлов
- подключение «типовых» тестов



SCL/Stimulus Литература

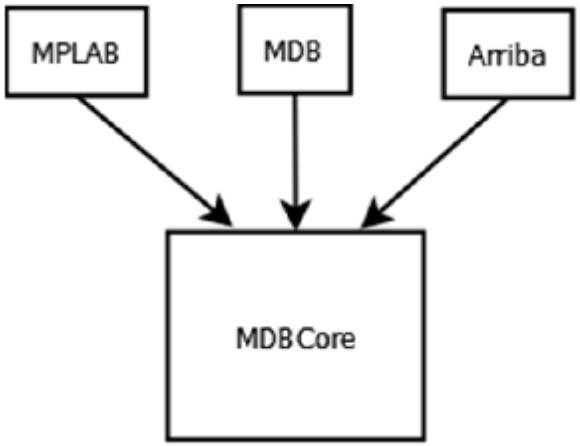
- SCL User's Guide (в составе документации пакета MPLAB X IDE) www.microchip.com/mplabx/
- Язык SCL http://pic24.ru/doku.php/osa/articles/scl
- SCL Code repository http://www.microchip.com/forums/m109149.aspx
- SCL PRIMER / TUTORIAL http://www.microchip.com/forums/m111255.aspx

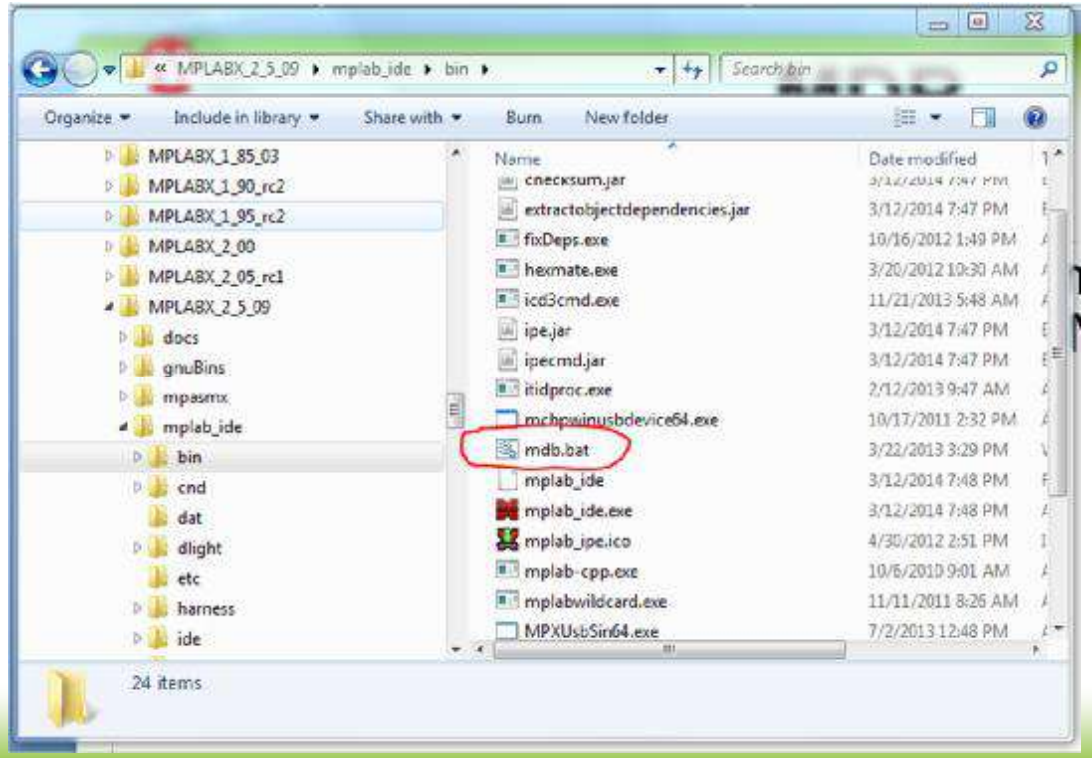
План

- Основы Симулятора
- Периферия
- Выводы Pin
- SCL/Stimulus
- **MDB**
- Новое в MPLAB X

MDB

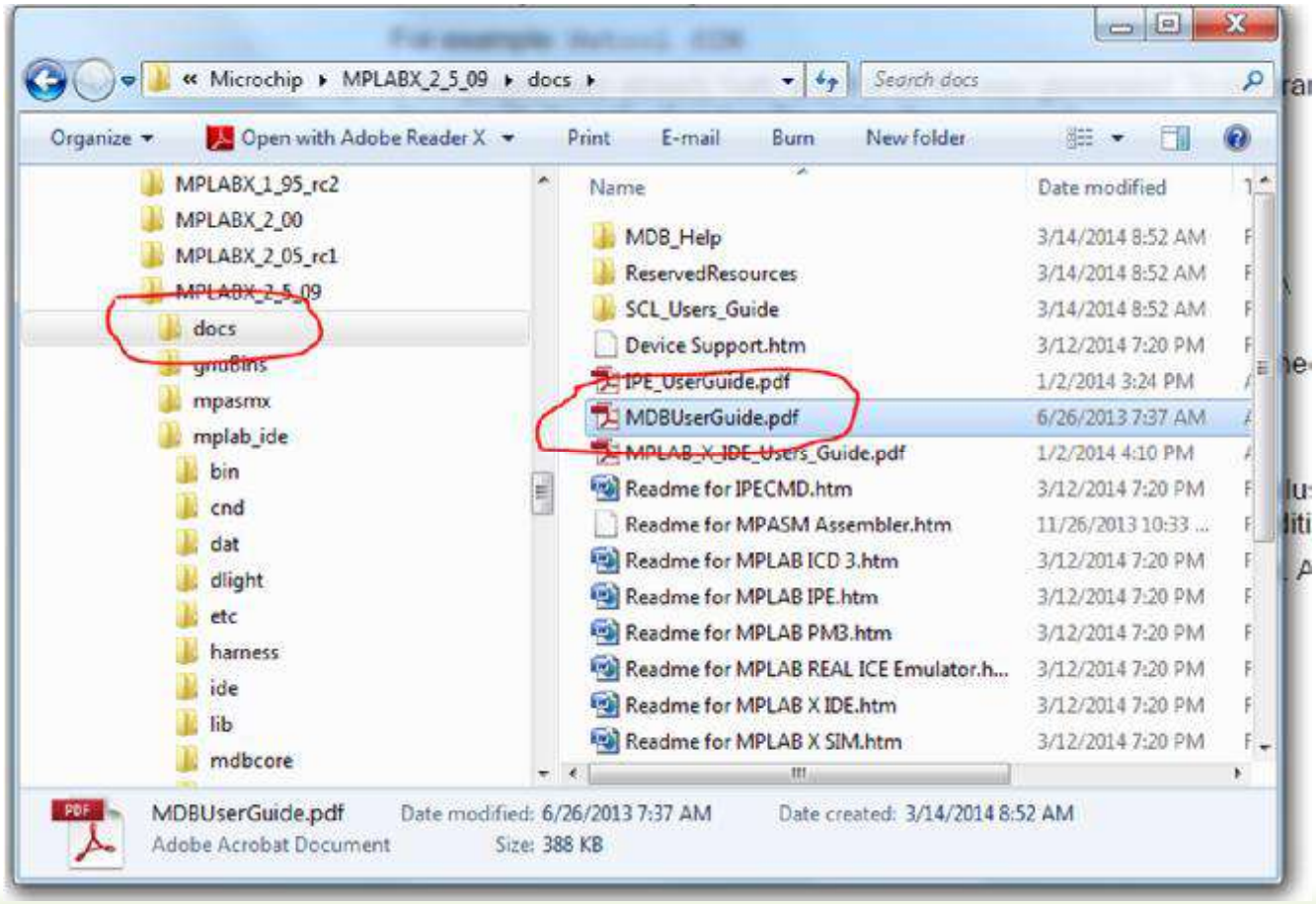
- **Microchip Debugger (MDB) это command line обертка для MDBCore, которая поставляется с MPLAB® IDE**





MDB

Есть User Guide



MDB

- **Первый шаг – задается тип МК**

```
mdb
> device PIC16F877
```

- **Затем инструмент**
  - MDB поддерживает те же инструменты, что MPLAB® X IDE

```
> hwtool sim
```

MDB

Далее загружаем образ программы для отладки

- MDB не компилирует проект
- Используйте MPLAB X IDE для компиляции

```
> program “proj10.X.debug.elf”
```

- **Установить точки останова если необходимо**

```
> break main.c:53
> break *0x108
```

MDB

- **Затем запустить программу на выполнение**

  > run

- **Ждем точку останова или ручной останов (halt)**

  > halt

MDB

Можно тестировать состояние памяти (ОЗУ) или состояние портов с помощью команды print

- MDB не поддерживает pin (пока)

> print PORTA

Команда Stim позволяет прикреплять SCL файлы

> stim “proj10.scl”

MDB

MDB поддерживает все инструменты MPLAB® X IDE, наиболее пригоден для работы с симулятором

- Не требует «железа»
- MDB поддерживает скрипты
- Таким образом – пригоден для автоматического тестирования

MDB

Просто создайте текстовый файл с MDB командами и запустите его с командной строки

mdb proj10mdb.txt

Удобнее перенаправить вывод данных в файл для записи результатов

mdb proj10mdb.txt > proj10results.txt

MDB

Теперь можно запустить тест на всю ночь!!

и пойти спать J

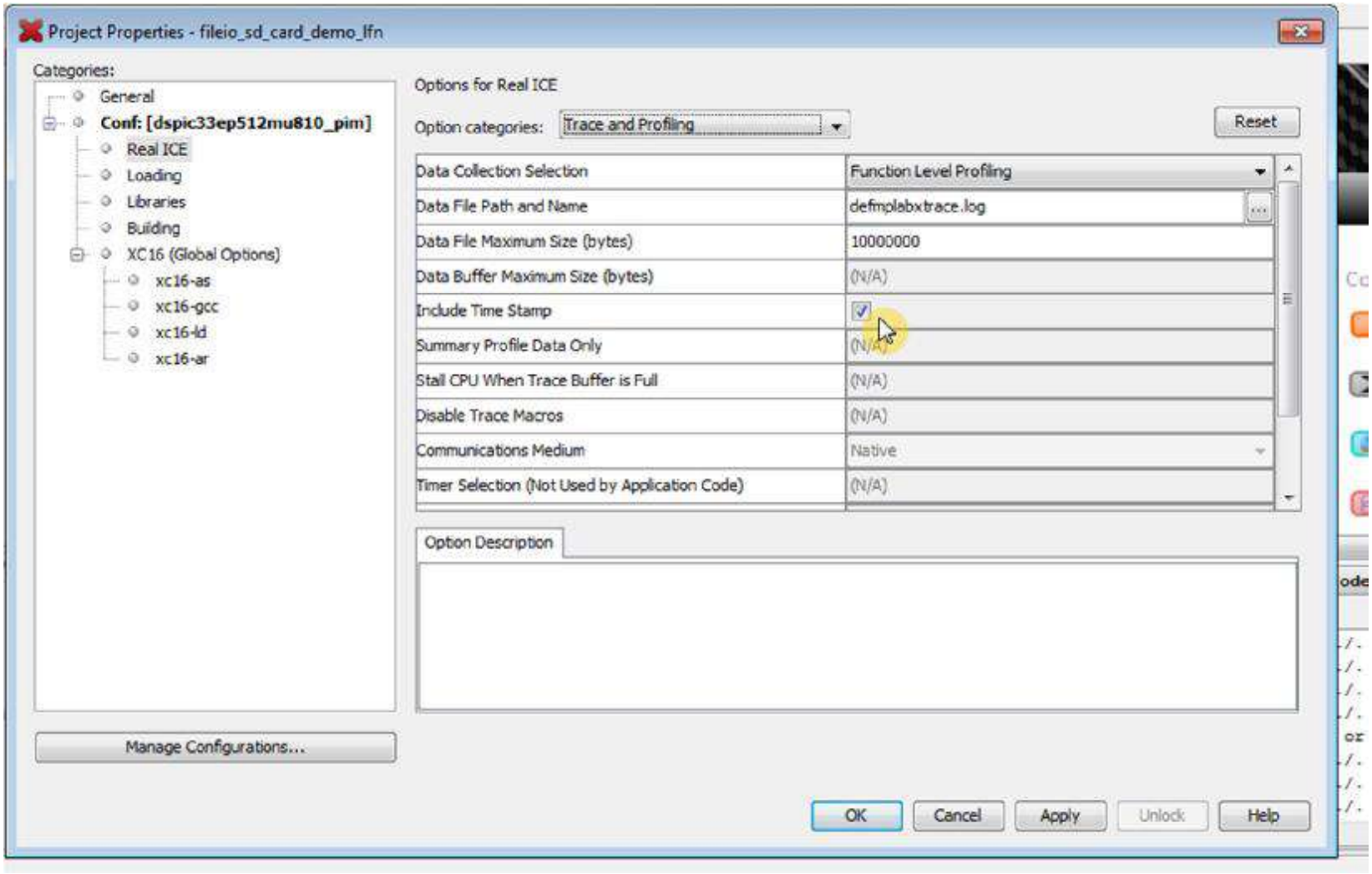
План

- Основы Симулятора
- Периферия
- Выводы Pin
- SCL/Stimulus
- MDB
- **Новое в MPLAB X**

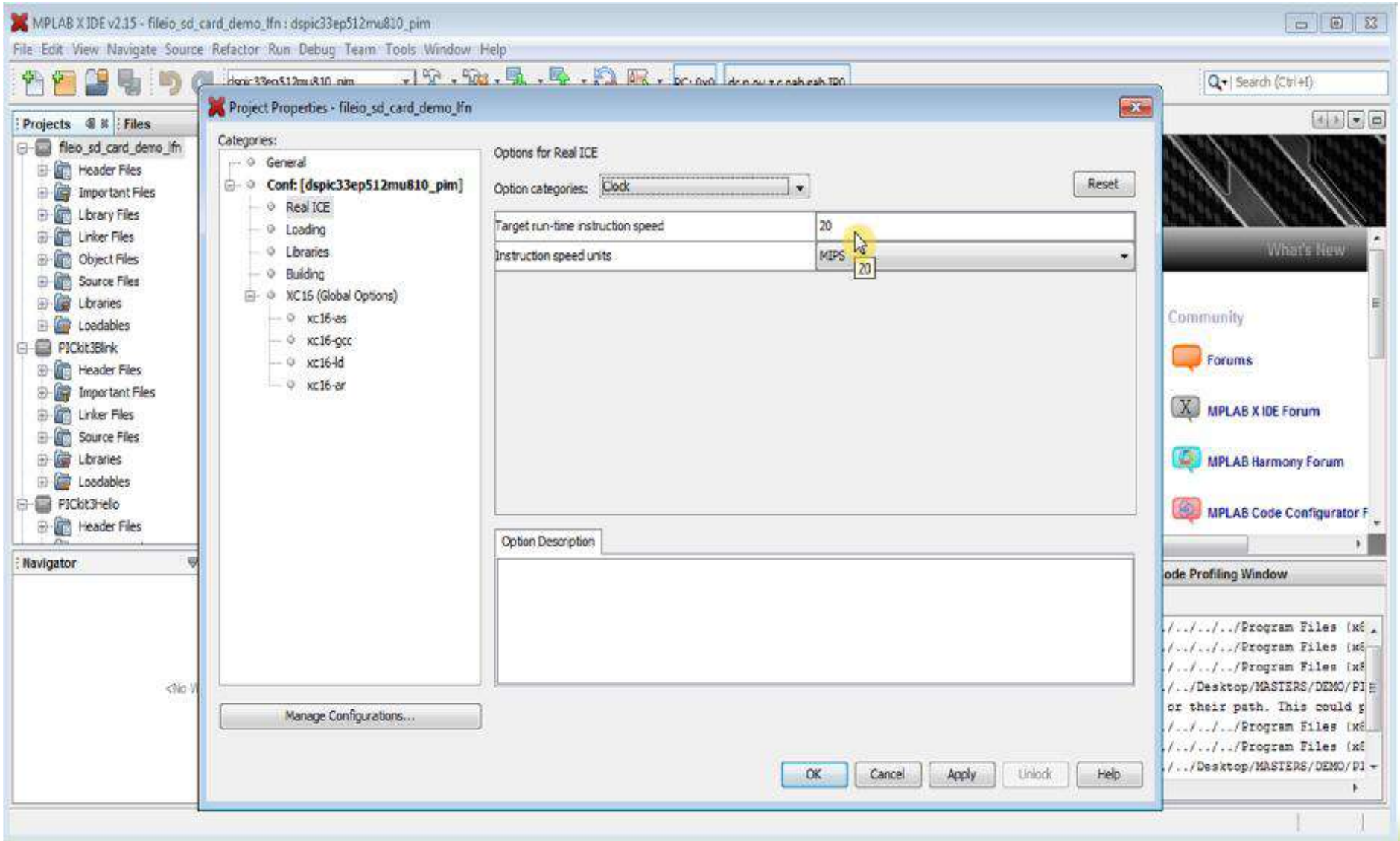
MPLAB® X IDE - Function Level Profiling

- Поддержка большинства МК (с data capture)
- Работает с MPLAB® REAL ICE™
- Устанавливается как plug-in
- Позволяет строить профиль использования функций с временными отметками
- Compiler instrumented trace – добавляет код до и после всех функций (только для сессии отладки)
- Будет доступен с www.embeddedcodesource.com

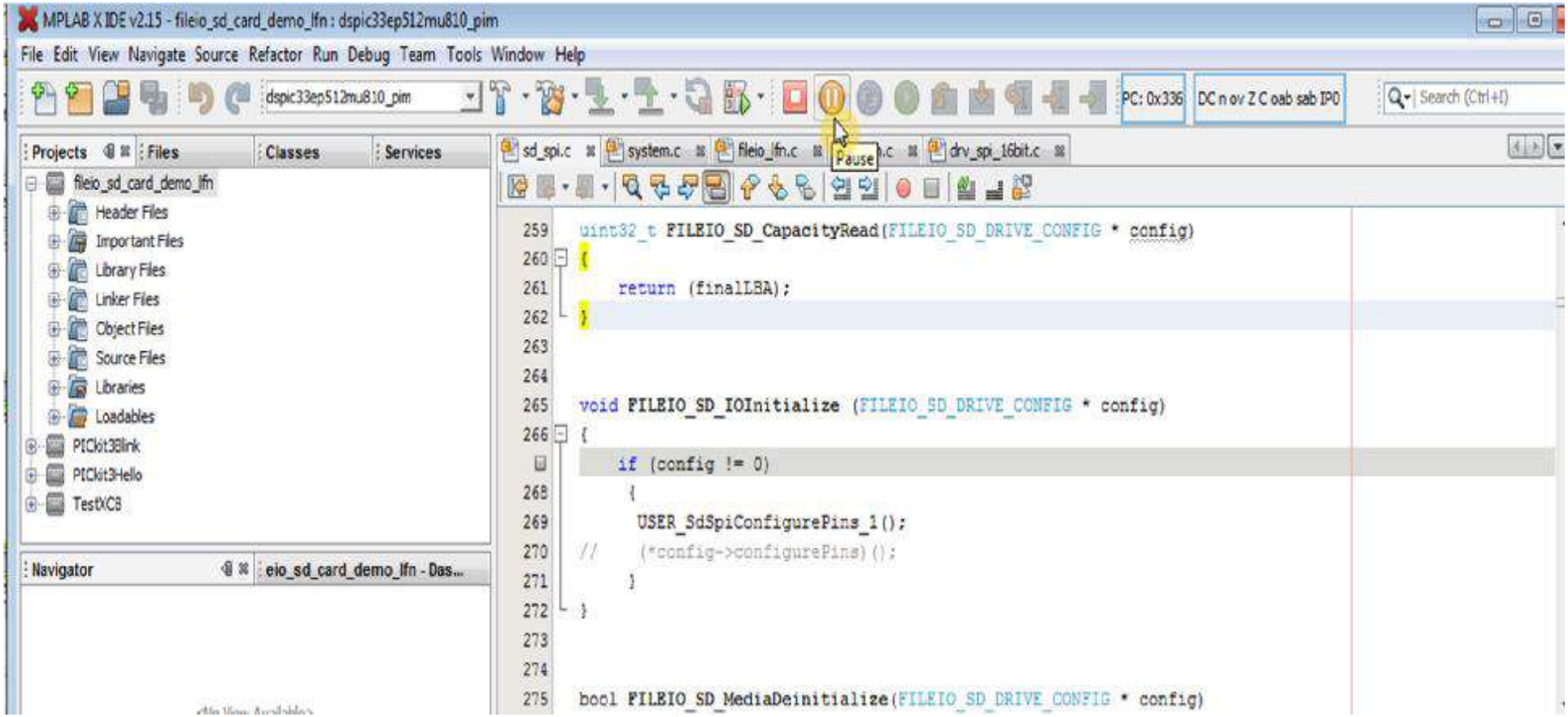
MPLAB® X IDE - Function Level Profiling



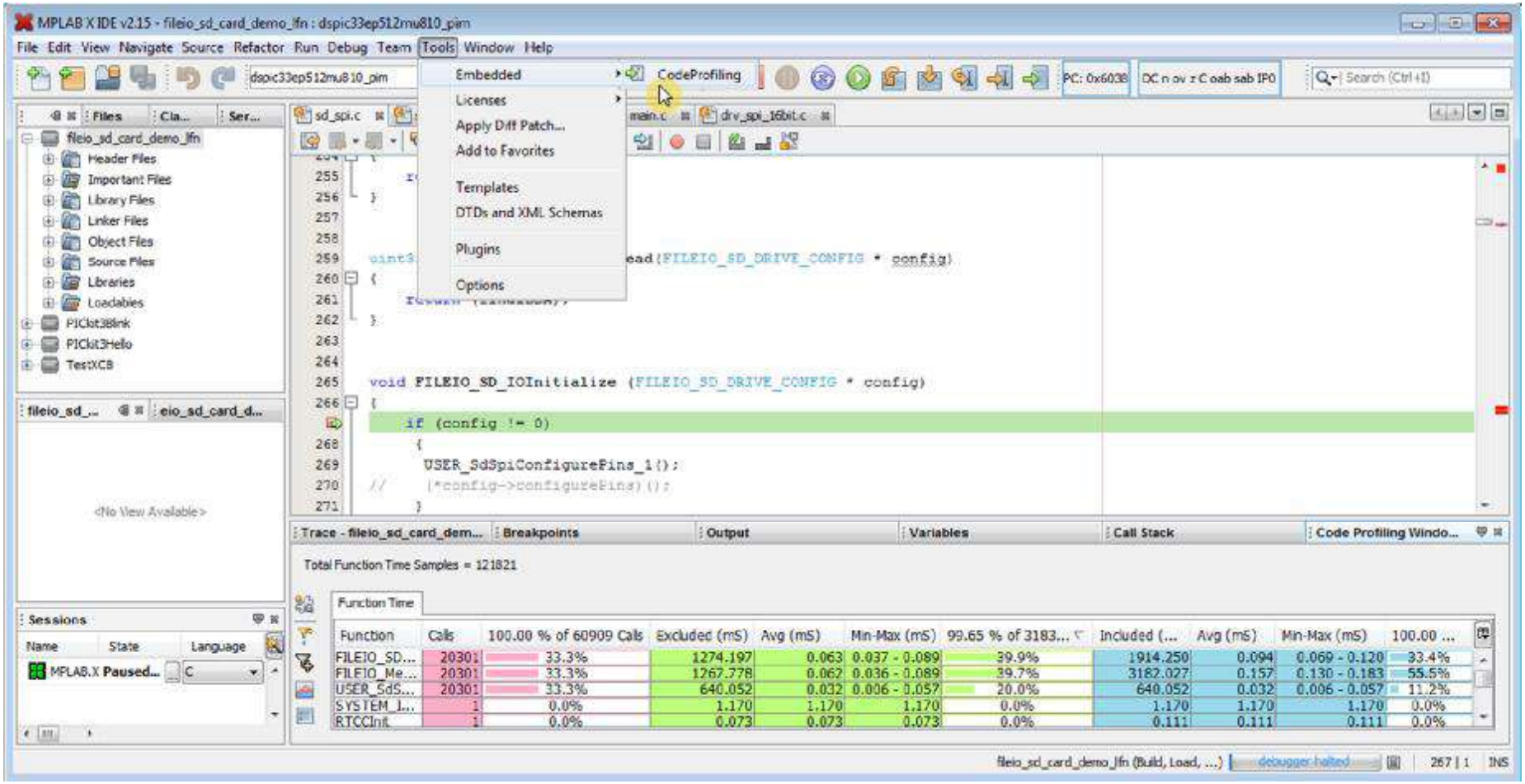
MPLAB® X IDE - Function Level Profiling



MPLAB® X IDE - Function Level Profiling



MPLAB® X IDE - Function Level Profiling



MPLAB® X IDE - Function Level Profiling

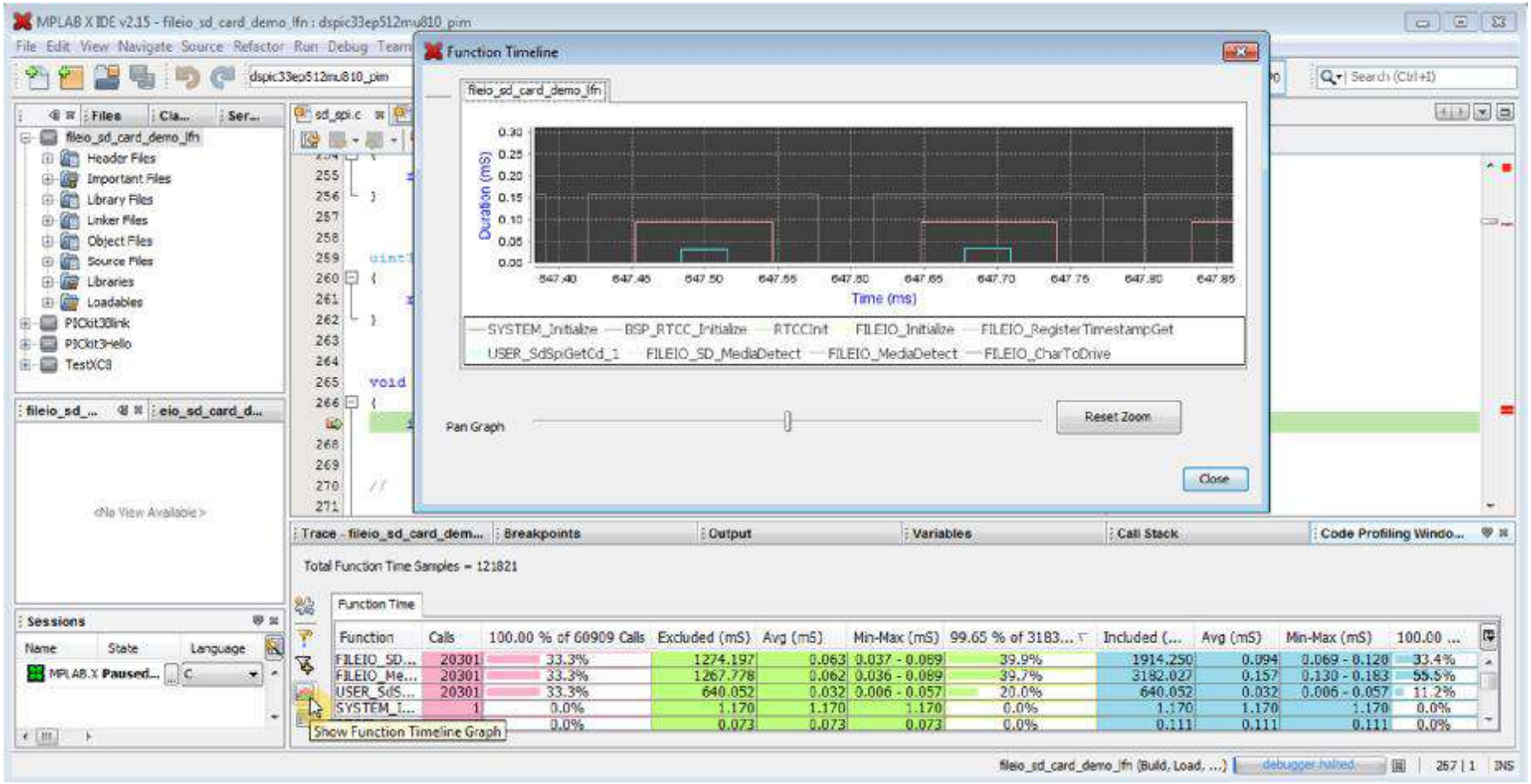
Code Profiling Window - fileio_sd_card_demo_lfn

Total Function Time Samples = 314175

Function Time

Function	Calls	100.00 % of 157086 Calls	Excluded (mS)	Avg (mS)	Min-Max (mS)	99.66 % of 8215.09...	Included (mS)	Avg (mS)	Min-Max (mS)	100.00 % of 14806.07...
FILEIO_SD_Medi...	52360	33.3%	3286.413	0.063	0.037 - 0.094	39.9%	4938.596	0.094	0.069 - 0.120	33.4%
FILEIO_MediaDet...	52360	33.3%	3268.776	0.062	0.036 - 0.089	39.7%	8207.372	0.157	0.130 - 0.183	55.4%
USER_SdSpiGetC...	52360	33.3%	1658.523	0.032	0.006 - 6.340	20.1%	1658.679	0.032	0.006 - 6.496	11.2%
SYSTEM_Initialize	1	0.0%	1.170	1.170	1.170	0.0%	1.170	1.170	1.170	0.0%
RTCCInit	1	0.0%	0.074	0.074	0.074	0.0%	0.112	0.112	0.112	0.0%
FILEIO_Initialize	1	0.0%	0.039	0.039	0.039	0.0%	0.039	0.039	0.039	0.0%
BSP_RTCC_Initialize	1	0.0%	0.038	0.038	0.038	0.0%	0.038	0.038	0.038	0.0%
FILEIO_CharToDr...	1	0.0%	0.032	0.032	0.032	0.0%	0.032	0.032	0.032	0.0%
FILEIO_Register...	1	0.0%	0.031	0.031	0.031	0.0%	0.031	0.031	0.031	0.0%

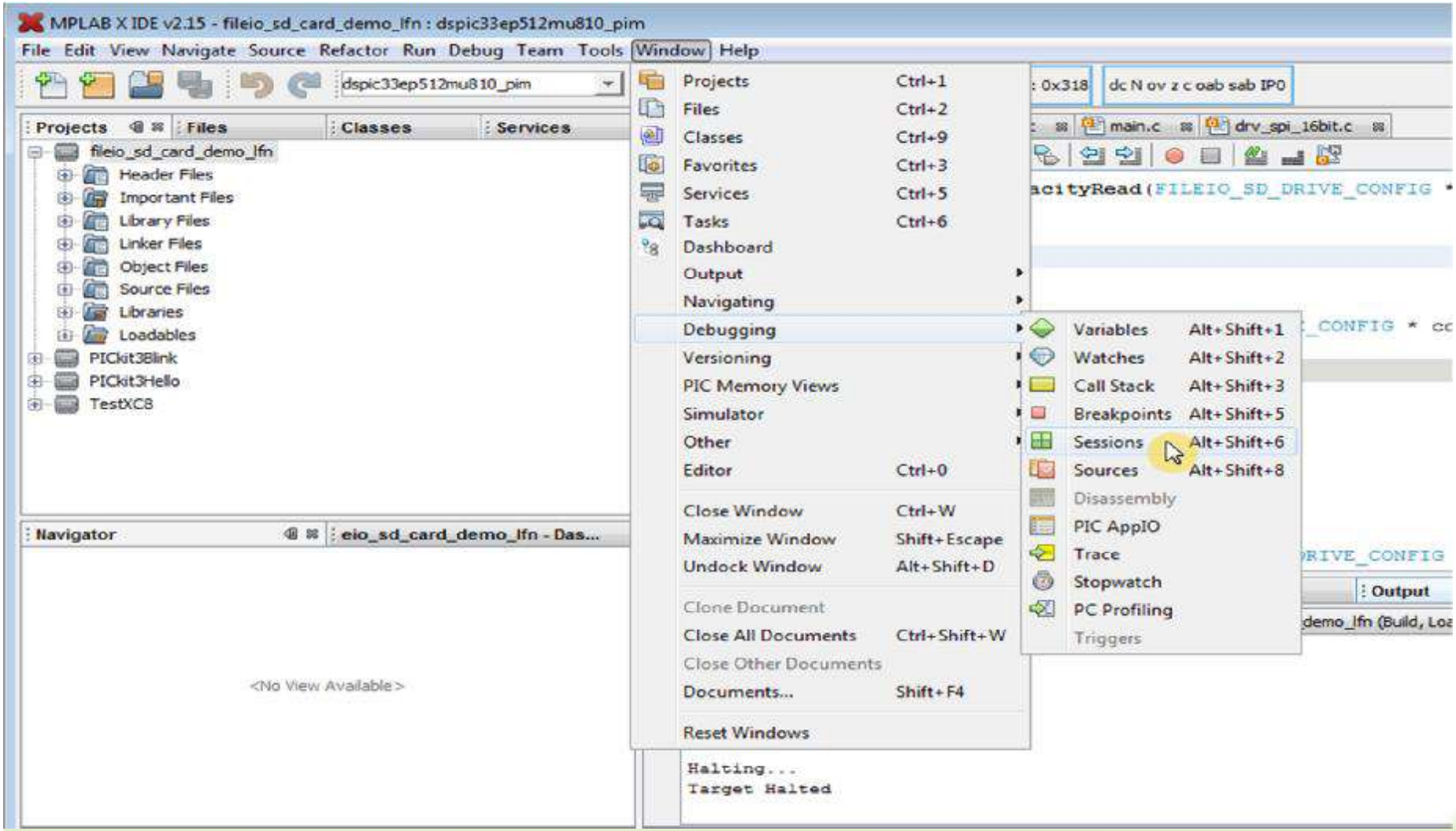
MPLAB® X IDE - Function Level Profiling



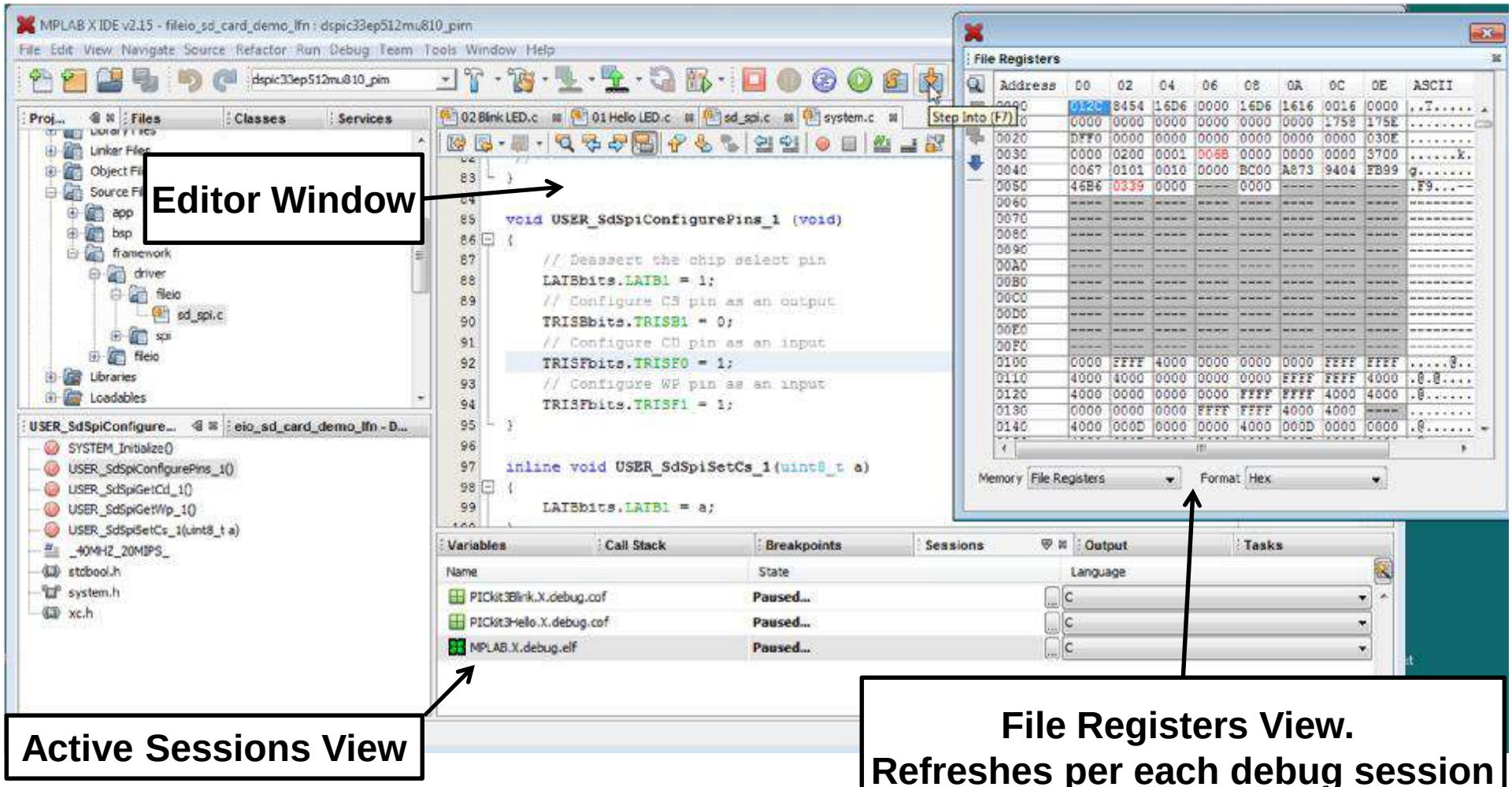
MPLAB® X IDE Multi-Session Debugging

- Позволяет несколько debug сессий на одной или нескольких отлаживаемых платах
- Через Sessions window управление контекстом
- Используемая память отображается динамически для текущей сессии

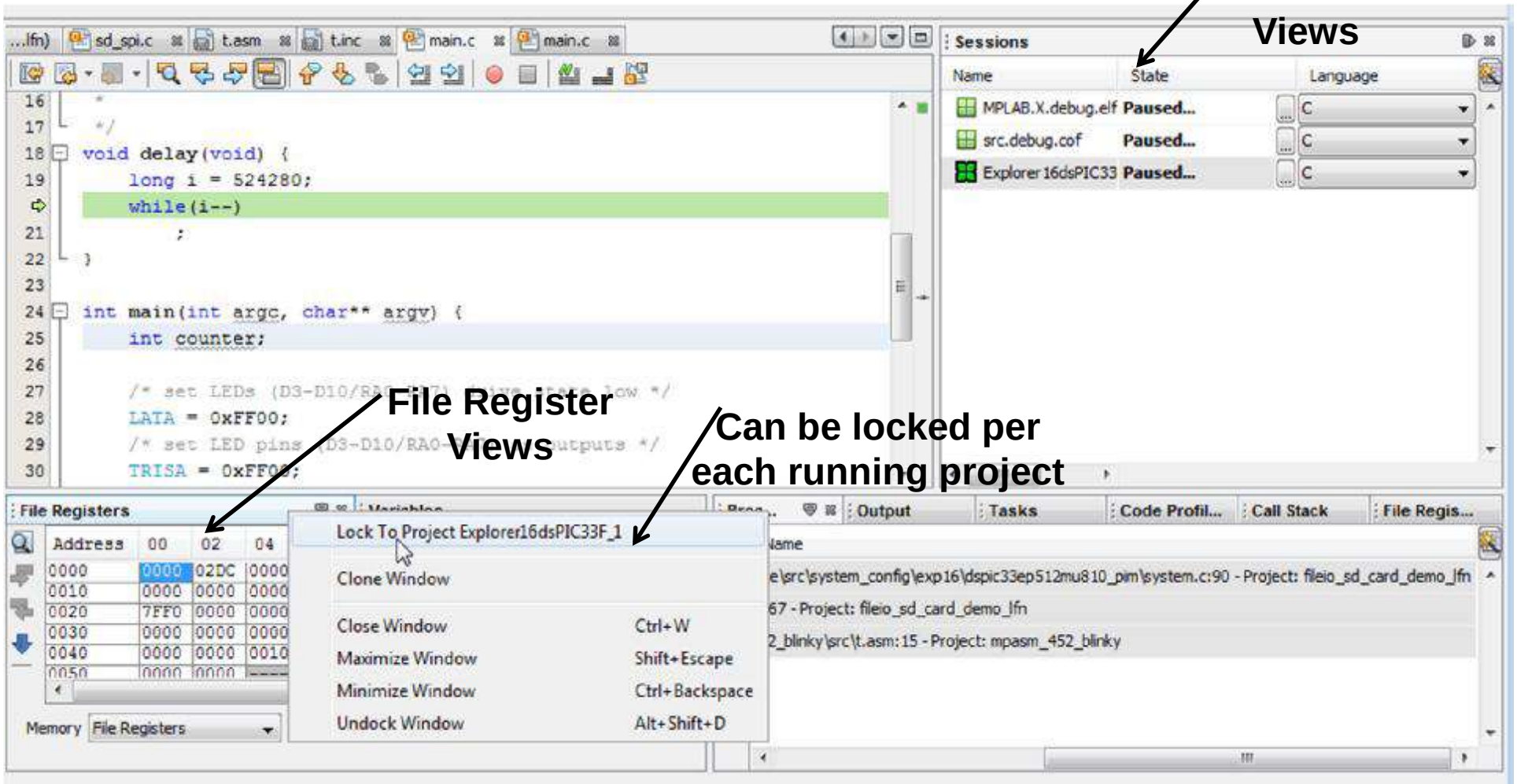
MPLAB® X IDE Multi-Session Debugging



MPLAB® X IDE Multi-Session Debugging



*MPLAB® X IDE Multi-Session Debugging



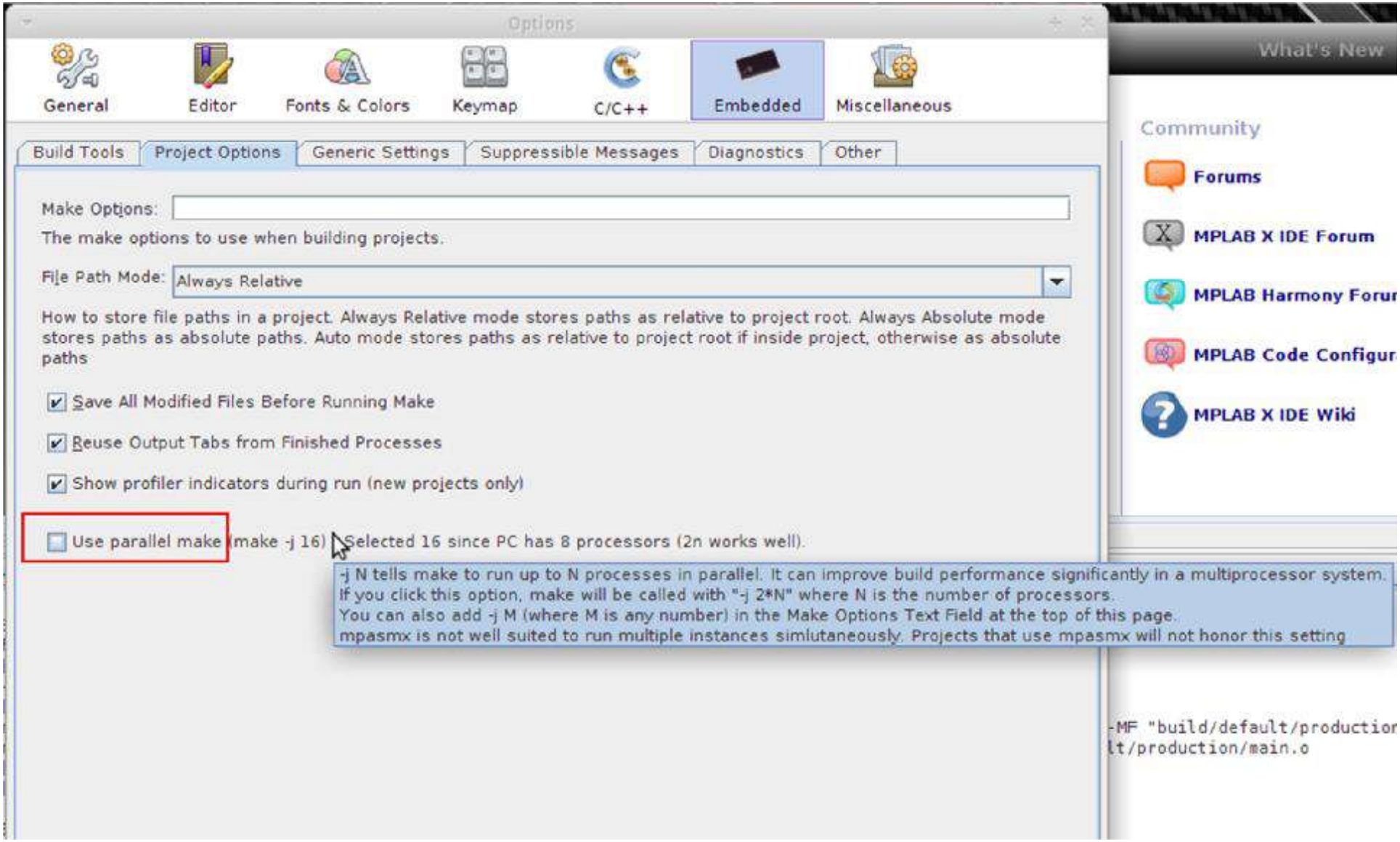
MPLAB® X IDE улучшения

- **Скорость:**
  - Открытие/Закрытие проекта – увеличена скорость и производительность
    - **За последний год проведено оптимизации загрузки проекта и обработка его конфигурации**
  - Скорость симуляции так же увеличена. Позволяет производить более быстрое тестирование.
  - Parallel make из настроек среды
    - **Увеличение сборки проекта на многоядерных процессорах**

MICROCHIP
MASTERs 2014

*MPLAB® X IDE

Parallel Make From Generic Settings



Command line tools

MPLAB® IPE Command Line Interface (IPECMD)

- Общий инструмент для Microchip программаторов: PICkit™ 3, MPLAB ICD 3, MPLAB REAL ICE™ и PM3
- Кросс-платформенный
- Можно использовать для конфигурирования MPLAB PM3 в автономном режиме
- Поддержка SQTP℠
- Программирование в составах производственных линий из командной строки

Command line tools

MPLAB® IPE Command Line Interface (IPECMD)

- IPECMD это JAR файл
- Пример программирования из командной строки с помощью MPLAB ICD:

```
>java -jar ipecmd.jar -P24FJ256GB106 -TPICD3 -M –F”c:/pic24.hex”
```

- Полный список команд:

```
>java -jar ipecmd.jar -?
```

Command line tools

Старые инструменты

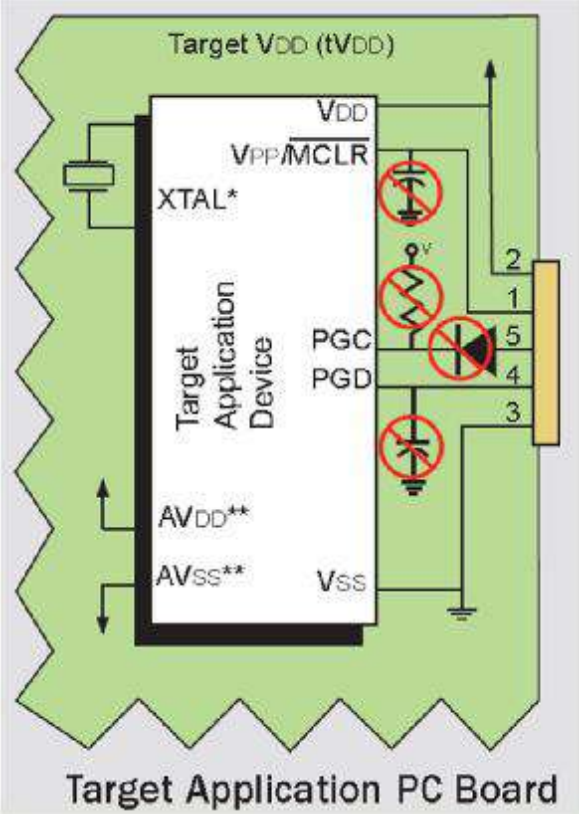
- RealICECMD.exe, PM3CMD.exe, PK3CMD.exe and ICD3CMD.exe так же включены в MPLAB® X IDE
- Предоставляют совместимость для платформы Windows

Программирование МК REALICECMD.exe:

```
REALICECMD -P18F67J50 -FC:\DemoCode.Hex -M
```

Emulator Extension Pak

- **МК с встроенной поддержкой отладки (ICD):**
  - SW Breakpoints
  - 3 Hardware Breakpoints
  - Нет трассировки
  - Нет Real time Watch
  - Используют выводы МК
  - Некоторый ограничения по схемотехнике (подключению)



Emulator Extension Pak

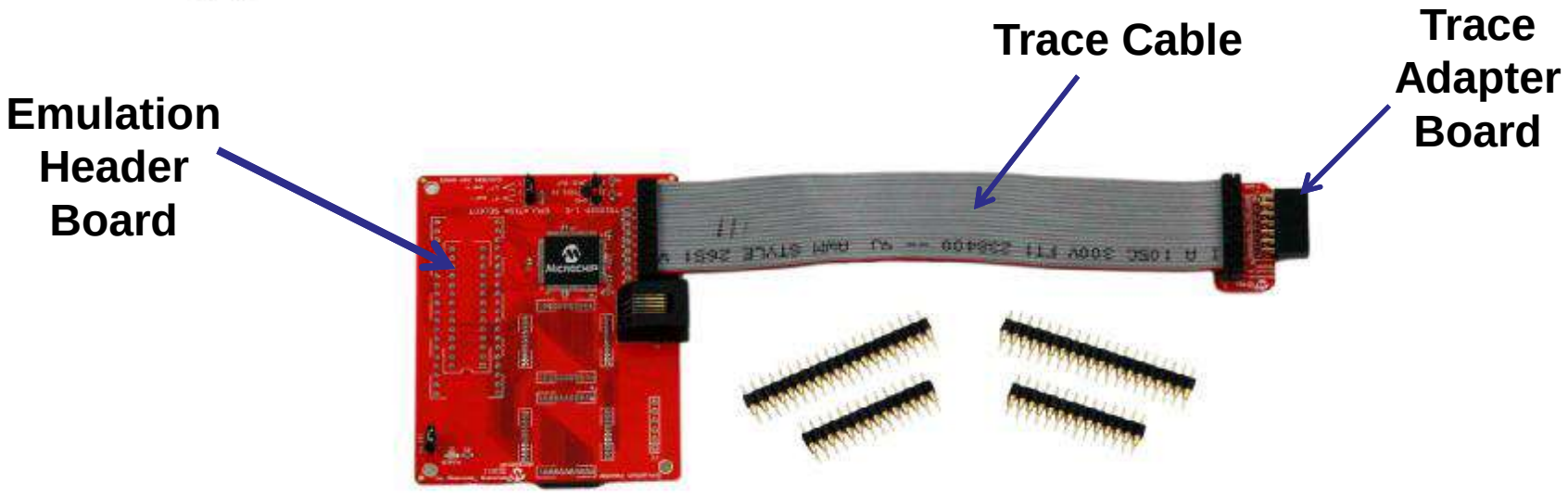
- Дешевая и мощная система для эмуляции
- Разработан для поддержки сложных отладочных работ
- Новое поколение – ME2 устройств:

Emulator Extension Pak

- **Может работать с:**
  - MPLAB® REAL ICE™ in-circuit emulator
  - MPLAB ICD 3
  - PICkit™ 3

PIC16F1xxx Emulator Extension Pak

- Требуется MPLAB® X IDE v2.0 или старше
- Эмулирует несколько МК в пределах семейства



Emulator Extension Pak

- **AC244055 (PIC16F1939-ME2) эмулирует:**
  - PIC16(L)F1933,
  - PIC16(L)F1934
  - PIC16(L)F1936
  - PIC16(L)F1937
  - PIC16(L)F1938
  - PIC16(L)F1939

Emulator Extension Pak

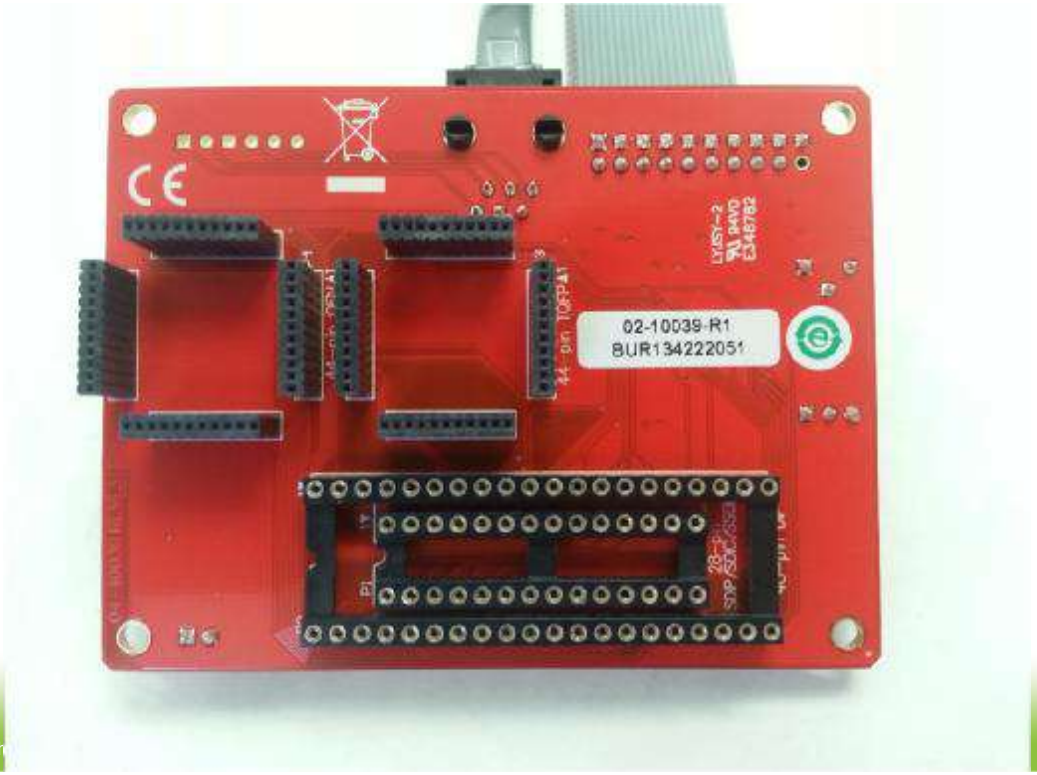
- **AC244063 (PIC16F1829-ME2) эмулирует:**
  - PIC12(L)F1822, PIC12(L)F1840,
  - PIC16(L)F1823, PIC16(L)F1824,
  - PIC16(L)F1825, PIC16(L)F1826,
  - PIC16(L)F1827, PIC16(L)F1828,
  - PIC16(L)F1829, PIC16(L)F1847

Emulator Extension Pak

- **AC244064 (PIC16F1789-ME2) эмулирует:**
  - PIC16(L)F1782, PIC16(L)F1783,
  - PIC16(L)F1784, PIC16(L)F1785,
  - PIC16(L)F1786, PIC16(L)F1787,
  - PIC16(L)F1788, PIC16(L)F1789

Emulator Extension Pak

- **Содержит несколько коннекторов для подключения к отлаживаемым платам**



Возможности

- Real-time аппаратная трассировка программы
- 32 Runtime Data Watch Points
- 32 Hardware Address/Data trigger events
- Stopwatch счетчик циклов
- Предыдущее состояние PC

Возможности

- Четыре комбинации событий (Event) – управление трассировкой, триггеры
- Breakpoints без остановки halting*
- Trigger out on event
- Теневая отладка

(Background Debug*)

Event Sources

- RESET instruction
- External Halt
- SLEEP
- Wake-up
- WDT Reset
- Interrupts
- MCLR Reset
- SW Breakpoints
- HW Breakpoints
- Event Combiners

Hardware Instruction Trace

- **Real-time дамп исполнения программы**
  - Можно записать и анализировать
  - Скорость выполнения команд до 32МГц
  - События (event) могут запустить/остановить трассировку

Hardware Instruction Trace

- **MPLAB® X IDE Trace Display Window**

Line	Address	Op	Label	Instruction
1	000	0x3180		MOVLP 0x0
2	001	0x2802		GOTO 0x2
3	002	0x0064	MainProgram	CLRWDT
4	003	0x30FE	Loop_202	MOVLW 0xFE
5	004	0x0023		MOVLB 0x3
6	005	0x008E		MOVWF PORTC
7	006	0x0022		MOVLB 0x2
8	007	0x100E		BCF PORTC, 0x0
9	008	0x0021		MOVLB 0x1
10	009	0x100E		BCF PORTC, 0x0
11	00A	0x0022		MOVLB 0x2
12	00B	0x140E		BSF PORTC, 0x0
13	000	0x3180		MOVLP 0x0
14	001	0x2802		GOTO 0x2

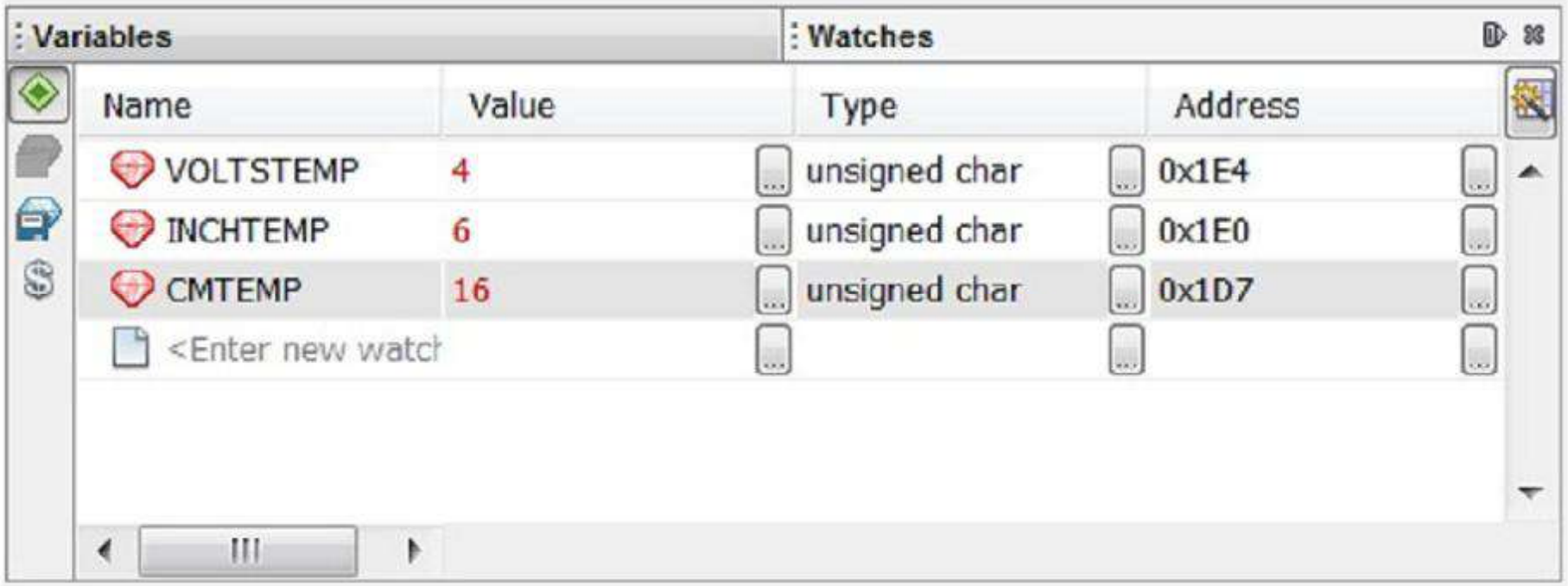
- **До 57K команд**

Data Watch Points

- Ширина 8 bit
- SFRs и переменные
- Глобальные переменные обновляются в реальном времени
- Программируемые интервалы для обновления состояния переменных
- Таймера не поддерживаются

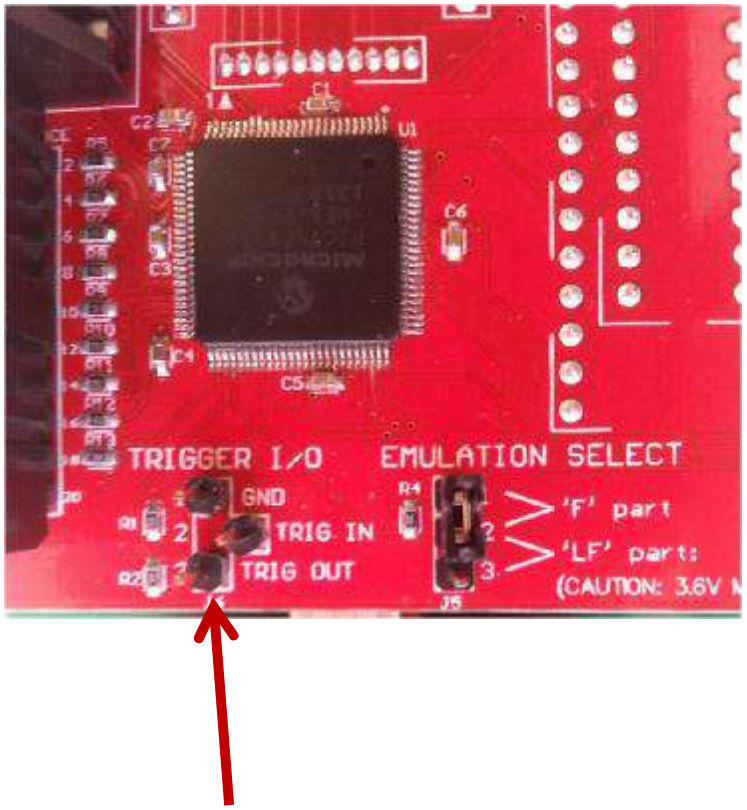
Data Watch Points

Data Watch Window



Breakpoints w/o Halting

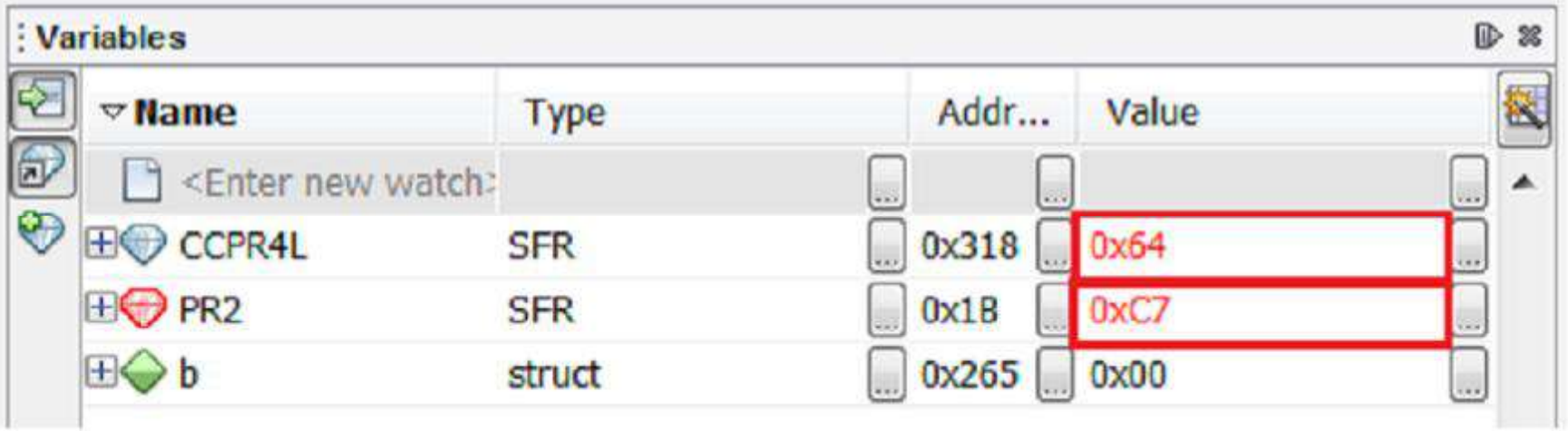
- **Вместе с:**
  - Trigger out
  - Stopwatch
- **Без остановки выполнения кода**



TRIGGER OUT

Runtime Debug

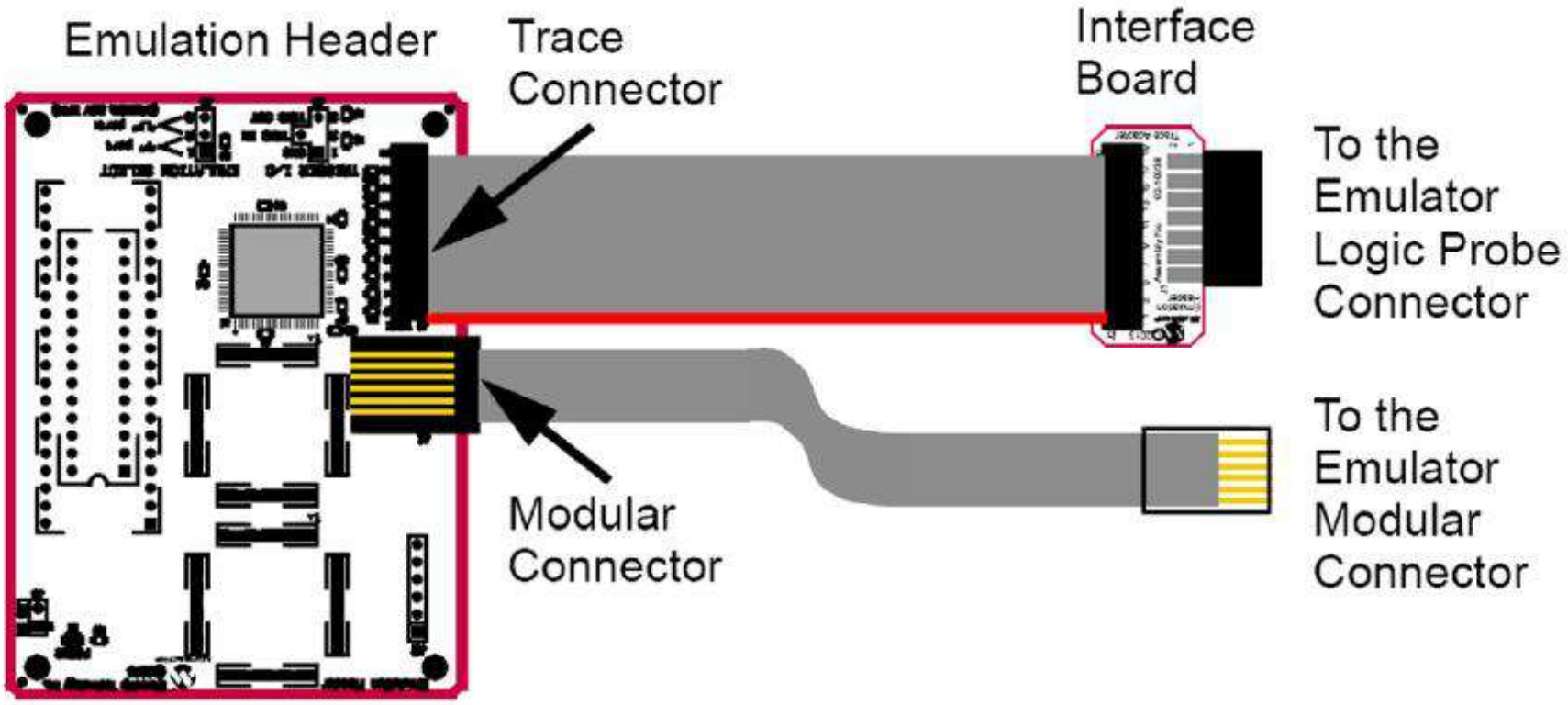
- **Позволяют читать/писать в RAM и SFR-ы без перепрограммирования МК**



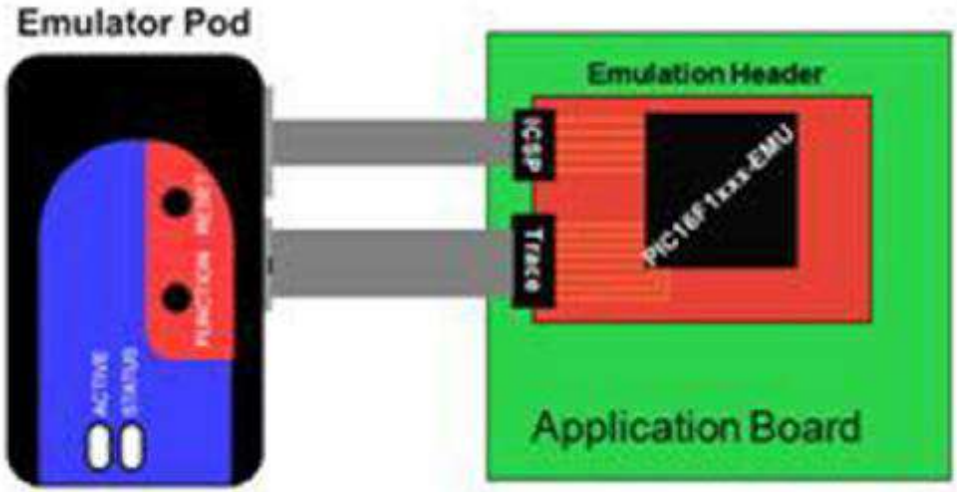
Future Features

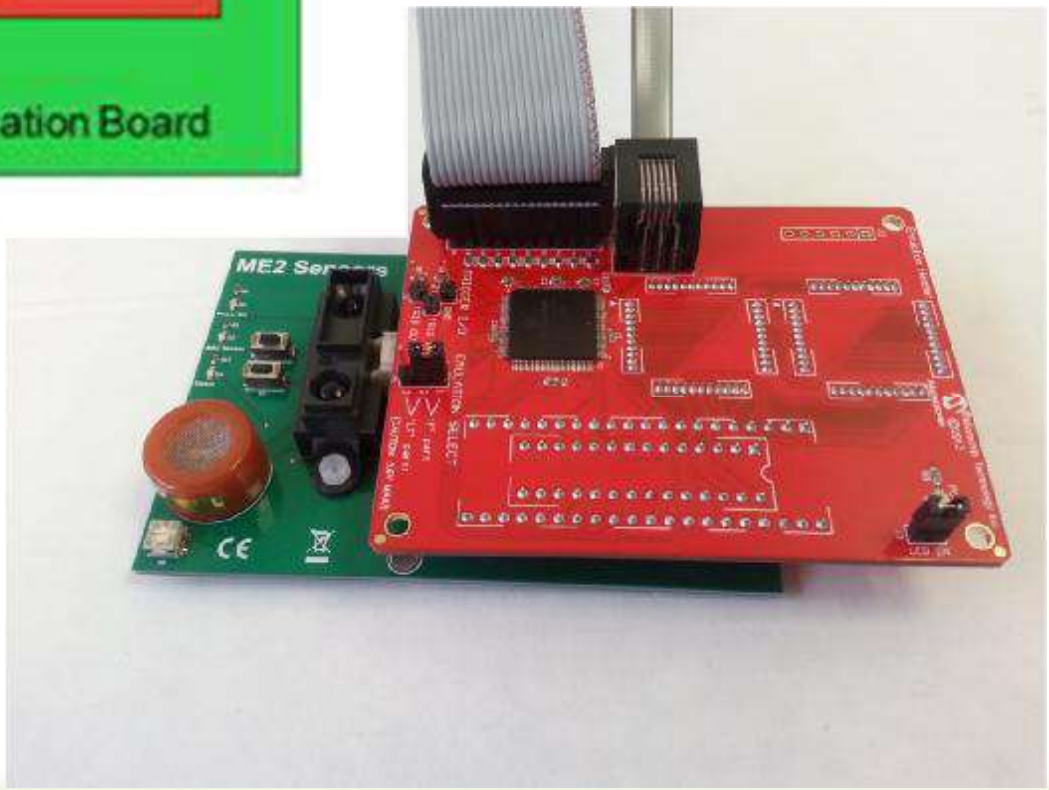
- **В следующих версиях MPLAB® X IDE планируется поддержка**
  - Background Debug
  - Stack Capture
  - 32 bit wide variable watch

CONNECTIONS



CONNECTIONS





MICROCHIP
MASTERs 2014

Спасибо!