

Управление программными средами с обменом данными

Цель работы: Освоение средств обмена данными с закрытыми программными средами.

Задача работы: Разработка сценариев обмена данными между MATLAB и программой, отлаживаемой в интегрированной среде MPLAB IDE.

Приборы и принадлежности: Персональный компьютер, MATLAB, демонстрационная версия MPLAB IDE 8.92, скриптовый автокликер Clickermann, и графический редактор Paint.

ВВЕДЕНИЕ

Важное место в разработке систем управления занимает тестирование этих систем и их составляющих. Результаты тестирования должны быть доступны для сравнения с "эталонными" результатами. В случаях, когда среда, в которой работает тестируемая система, только отображает результаты, но не накапливает их в доступных хранилищах (например, текстовых файлах), необходимо после каждого тестового прогона передавать результаты через буфер обмена средствами накопления и обработки результатов. Многократный запуск программ и копирование данных в ручном режиме может приводить к ошибкам тестирования.

Автоматизацию считывания данных, отображаемых на дисплеях устройств, можно обеспечить при помощи видеокамеры и системы распознавания символов [1]. Обмен данными с "закрытыми" компьютерными программами в автоматическом режиме можно выполнить при помощи программ автоматизации диалога пользователя с операционной средой, в которых рутинные действия пользователя заменяются работой по заранее созданному сценарию эмуляторов устройств ввода, таких как мышь и клавиатура с использованием буфера обмена и средств распознавания.

В этой работе приведены примеры автоматизации тестирования ассемблерной программы контроллера (Рис. 1), выполняемой под отладчиком в среде MPLAB IDE. Тестирование включает запуск генерации и передачу входных данных программе контроллера, запуск отладчика и приём результатов m-программой MATLAB. Циклический запуск программ в среде ОС Windows и обмен данными выполняется программой Clickermann [2].

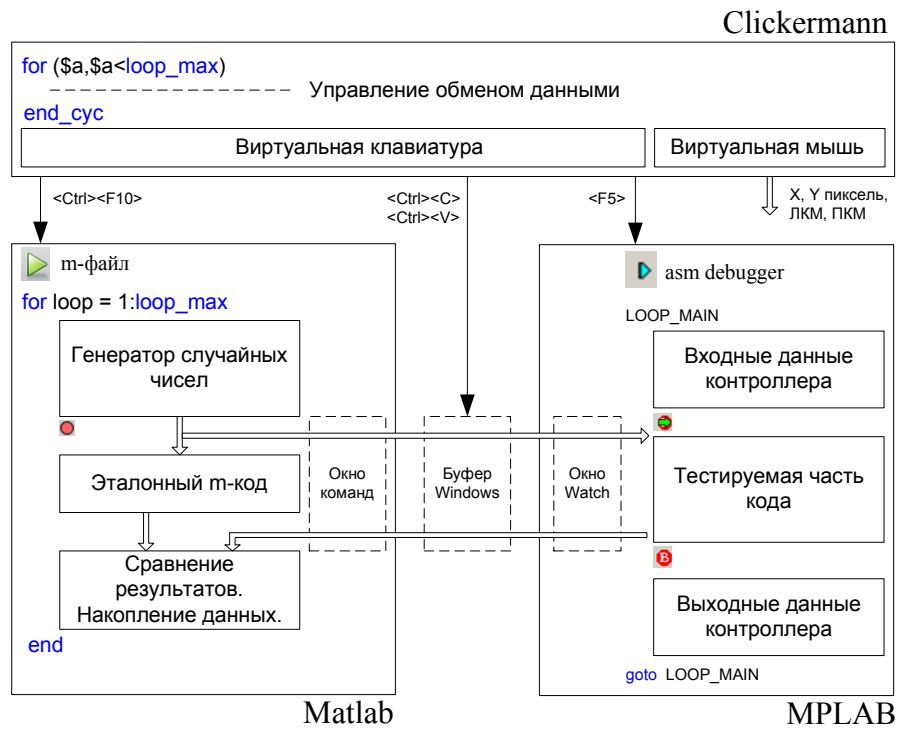


Рис. 1. Структура тестирования ассемблерной программы с обменом данными между Matlab и MPLAB под управлением Clickermann.

ОБЩИЕ СВЕДЕНИЯ

Отсчет координат экранов

При автоматизации диалога пользователя учитываются координаты изображения и указателя мыши в пикселях экрана. Порядок отсчета координат двух экранов компьютера показан на Рис. 2.

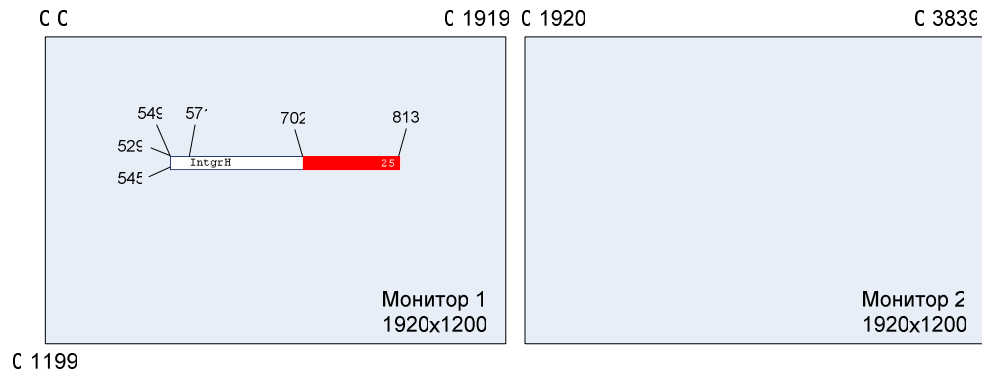


Рис. 2. Отсчет координат двух экранов и пример координат изображения.

Программа Clickermann

Программа Clickermann [2] позволяет автоматизировать диалог пользователя с операционной средой персонального компьютера. Этот скриптовый автокликер поддерживает следующие функции.

- Имитация устройств ввода: нажатия, удержания и освобождения клавиш мыши, движения мыши, нажатия, удержания и освобождения кнопок клавиатуры;
- Запись действий пользователя с мышью и клавиатурой с последующим их воспроизведением;
- Определение координат пикселей и отображаемых на экране окон и других объектов;
- Чтение и запись данных в текстовые файлы;
- Запуск сторонних программ и их "горячих" клавиатурных команд;
- Озвучивание действий автокликера;
- Создание и редактирование сценариев работы компьютера в собственном редакторе. Скриптовый язык включает переменные, вычисления, функции работы с числами и строками, циклы, условия и подпрограммы.

Главное окно

Главное окно кликера показано на Рис. 3. Вверху окна располагаются функциональные кнопки загрузки сценария, вызова окон, настройки выполнения сценария, лог данных, настройки работы кликера, справки и др.

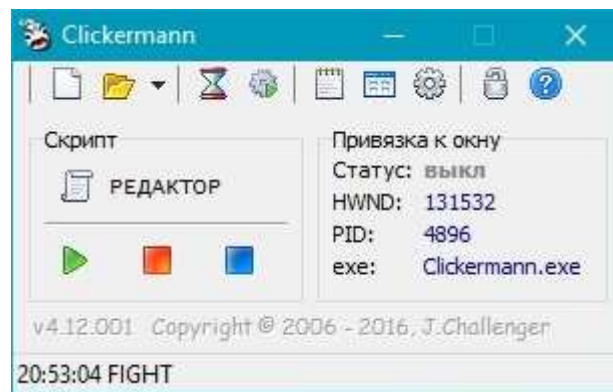







Рис. 3. Главное окно программы.

Кнопка "Запуск"  запускает выполнение сценария, находящегося в редакторе ( РЕДАКТОР) кликера. Повторное нажатие на эту кнопку останавливает выполнение и включает паузу . После снятия паузы, сценарий запускается с того места, где его остановили.

ВНИМАНИЕ! Нажатие клавиш **Alt+S** клавиатуры (по умолчанию) останавливает выполнение сценария. Такое прерывание сценария может пригодиться, когда остановка сценария при помощи реальной компьютерной мыши блокируется командами выполняемого сценария.

Кнопка "Запись"  предназначена для записи в сценарий всех действий пользователя с мышью и клавиатурой до момента нажатия кнопки останова  кликера или клавиш **Alt+S**.


Кнопка "Остановить"  предназначена для прерывания процесса выполнения или записи сценария.

Информационная панель

HWND:	131872
PID:	2288

 отображает PID - уникальный номер (идентификатор) процесса в многозадачной операционной системе (ОС) и уникальный номер объекта (окна) HWND принадлежащий процессу. Эту информацию можно использовать в сценарии для обращения, например, к текущему процессу MATLAB с уникальным номером 2288 в котором открыто несколько окон (131872 и др.). Однако, следует отметить, что при повторном запуске процессов ОС присвоит им и открывающимся окнам процессов новые номера.

Редактор

Кнопка "  РЕДАКТОР " (**Error! Reference source not found.**) открывает окно редактора сценария (**Error! Reference source not found.**).

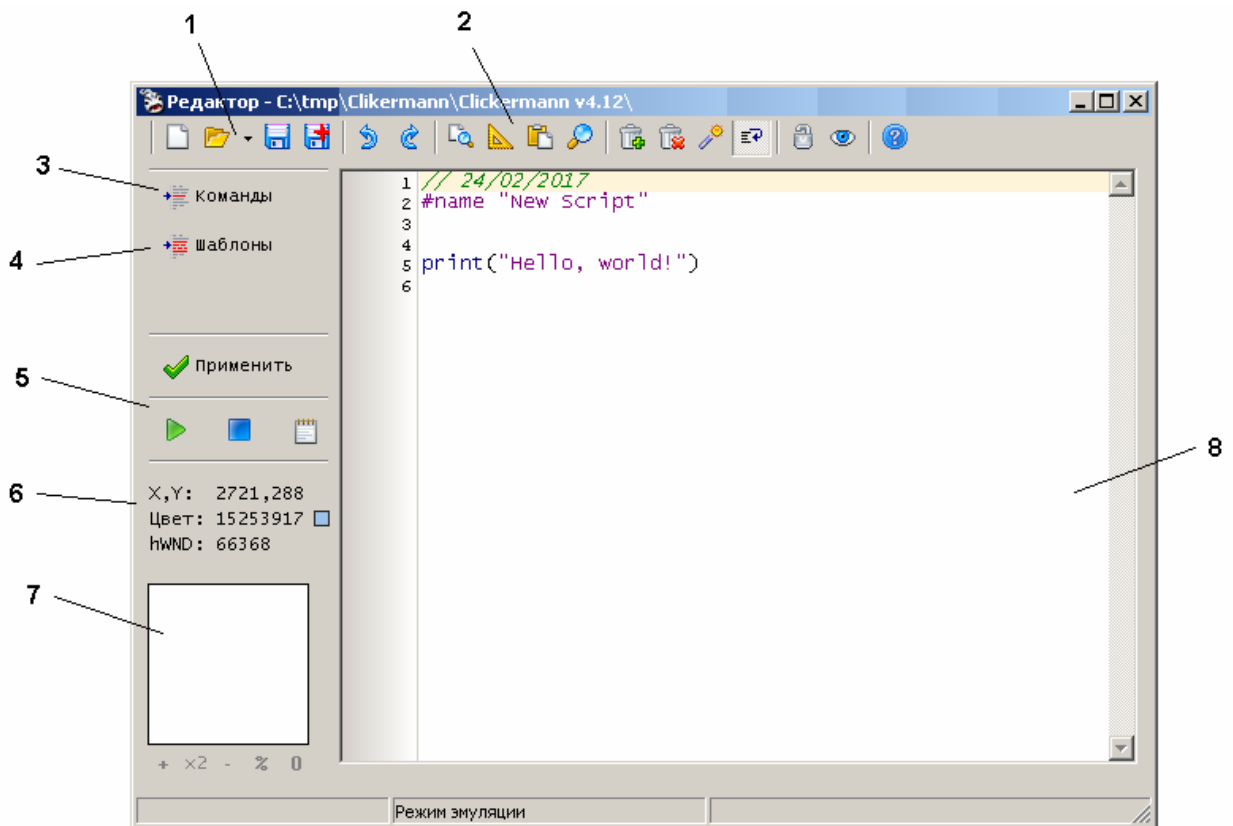


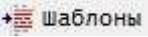
Рис. 4. Окно редактора кода сценария.

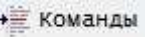
Редактор сценария позволяет загрузить, создать, отредактировать и сохранить сценарий. Редактор имеет подсветку синтаксиса, функции форматирования кода и быстрой вставки шаблонов, различные вспомогательные инструменты.

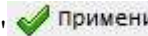
Окно редактора кода сценария содержит следующие элементы.




1. Кнопки управления файлом скрипта: новый, открыть, открыть историю, сохранить, сохранить как;
2. Вспомогательные кнопки. Включают поиск и замену текста, активируют экранную лупу, применяют автоформатирование кода и прочее;
3. Кнопка выбора команд из списка и вставки команд в сценарий;
4. Кнопка быстрой вставки готовых фрагментов кода из существующих шаблонов;

5. Кнопки управления скриптом: запуск, стоп, лог;
6. Параметры окна, на которое указывает курсор: координаты, цвет, hWnd (уникальный номер окна);
7. Поле отображения зоны экрана рассматриваемой через лупу.
8. Поле ввода и редактирования кода сценария.

Кнопка  используется для выбора из списка и вставки в сценарий готового шаблона. Текстовые файлы с шаблонами находятся в директории “%Clickermann%/data/lang/ru/templates/”. Можно редактировать и создавать новые шаблоны.


Кнопка  позволяет выбрать команды сценария из структурированных списков и вставить их в сценарий (зона 6, Рис. 4). Текстовый файл иерархического меню вставки команд находится по адресу “%Clickermann%/data/lang/ru/qinsert_menu.txt”. Его также можно редактировать с соблюдением формата.

Кнопка  обновляет рабочий сценарий кликера на вариант, находящийся в редакторе.

Кнопки "пуск , "стоп  " и "лог  " дублируют аналогичные кнопки главного окна (Рис. 3).

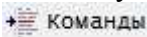

X, Y: 700, 366
Цвет: 16777215
hWnd: 31590448

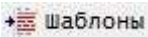
В зоне 6 (Рис. 4) отображаются текущие координаты курсора мыши в пикселях экрана, цвет пикселя на который указывает курсор и уникальный номер окна hWnd присваиваемый ему ОС Windows.

Зона 7 (Рис. 4) отображает часть экрана, наблюдаемую через лупу, которая включается и выключается кнопкой . Положение лупы связано с указателем мыши. Под зоной 6 находятся кнопки увеличения, уменьшения и изменения цвета изображения.


Сценарий

Сценарий представляет собой текстовый документ включающий комментарии и последовательность команд, которые исполняет кликер в режиме интерпретации.

Команды сценария можно ввести в документ (зона 8, Рис. 4) напечатав или выбрав из списков, открываемых кнопкой  (зона 3, Рис. 4). Описание команд и примеры можно найти в разделе  - справка по языку сценария.

Кнопка  (зона 4, Рис. 4) используется для выбора из списка и вставки в сценарий готового шаблона.

Текущие координаты курсора можно мгновенно вставить в сценарий в виде инструкции **LCLICK** по команде **Alt+Q** (по умолчанию) с клавиатуры.

Количество повторов выполнения сценария устанавливается в окне (Рис. 5), которое открывается кнопкой  главного окна кликера Рис. 3.

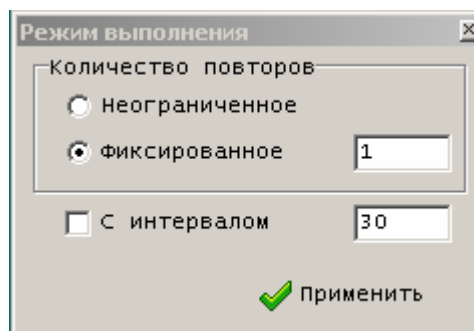




Рис. 5. Окно установки режима выполнения сценария.

Запуск и останов сценария выполняется кнопками "пуск"  и "стоп"  (Рис. 3 и Рис. 4).
Экстренная остановка сценария вызывается клавишами **Alt+S**.

Установка программы Clickermann




Бесплатную (FREEWARE) программу Clickermann можно скачать с сайта разработчика по ссылке <http://crapware.aidf.org/page/clickermann>

Программа **не требует установки**.


Исполняемый файл программы: **Clickermann.exe**.

ПРИМЕРЫ ПОЛУЧЕНИЯ ПРОВЕРЕННЫХ РЕЗУЛЬТАТОВ И ВАРИАНТЫ ДЛЯ САМОКОНТРОЛЯ

Задание 1. Автоматическая запись и воспроизведение действий пользователя.



1. Продумайте последовательность собственных действий с клавиатурой и мышью которую должен повторить автокликер. Пример тестовой последовательности:
 - Открывается программа Paint
 - Создаётся небольшой произвольный рисунок
 - Сохраняется рисунок.
 - Закрывается программа Paint.
2. Запустите исполняемый файл **Clickermann.exe**.
3. Откройте редактор (кнопка  РЕДАКТОР, Рис. 3). Обратите внимание на содержание сценария (зона 8, Рис. 4).
4. Закройте окно редактора.
5. Для записи ваших последующих действий нажмите кнопку  (Рис. 3).
6. Выполните свою последовательность.
7. Остановите запись командой **ALT+S** вводимой с клавиатуры.
8. Установите количество повторов воспроизведения сценария (см. Рис. 5).
9. Для воспроизведения записанных действий запустите сценарий кнопкой .

ВНИМАНИЕ! Для экстренного останова воспроизведения используйте команду **ALT+S**

10. После остановки воспроизведения откройте записанный сценарий (кнопка  РЕДАКТОР, Рис. 3) и рассмотрите его структуру и используемые команды.

Справочные данные ( > Clikermann Help > Index):

Команда **move** (x, y) перемещает курсор мышки в координаты x, y экрана.
 Команда **waitms(8 + \$\$CORP)** приостанавливает выполнение сценария на 8мс.

11. Команда **HALT** сценария выполняет его остановку. Проверьте это.
12. Запись движений мыши можно отключить в настройках программы (Рис.1 >  > Интерпретатор > Выполнение). Проверьте это.
13. Сохраните сценарий  в рабочей директории

Задание 2. Сценарий поиска координат фрагментов изображения.

1. Сделайте снимок экрана (например, кнопкой “Prt Scr”)
2. Вставьте снимок экрана в графический редактор **Paint**.
3. Оконтурируйте и сохраните фрагмент окна Matlab (Рис. 6) в “**bmp**” формате в каталоге **Clikermann > Project** с названием файла “**PIC.bmp**”.



Рис. 6. Фрагмент окна Matlab, выбранный для поиска координат изображения.

4. Наблюдая за положением мыши в зоне 6 (Рис. 4) вычислите координаты окна Matlab относительно левого верхнего угла фрагмента (Рис. 6).
5. Создайте сценарий поиска координат **bmp** изображения (Рис. 6), установки указателя мыши в зоне окна команд Matlab и ввода переменной в окно команд.

Код	Код с комментариями
GETSCREEN	GETSCREEN
IF_PICTURE_IN (0,0, \$_xmax,\$_ymax, "PIC.bmp")	IF_PICTURE_IN (0,0, \$_xmax,\$_ymax, "PIC.bmp") // поиск bmp фрагмента
MOVE(\$_return1,\$_return2)	MOVE(\$_return1,\$_return2) // перемещение мыши к фрагменту
END_IF	END_IF
DEFINE(\$_cmd,\$_return1+167)	DEFINE(\$_cmd,\$_return1+167) // Координаты точки внутри окна команд
DEFINE(\$_cmd,\$_retn2+344)	DEFINE(\$_cmd,\$_retn2+344)
MOVE(\$_cmd,\$_cmd)	MOVE(\$_cmd,\$_cmd) // Перемещение указателя в окно команд
LDOWN(\$_cmd,\$_cmd)	LDOWN(\$_cmd,\$_cmd) // Нажатие на ЛКМ
WAITMS(50)	WAITMS(50) // Ожидание 50 мс
LUP(\$_cmd,\$_cmd)	LUP(\$_cmd,\$_cmd) // Отпускание ЛКМ
KEYSTRING("Hello = 10")	KEYSTRING("Hello = 10") // Вывод текста в окно команд
KEYPRESS(#ENTER)	KEYPRESS(#ENTER) // Возврат каретки
HALT	HALT // Конец сценария

6. Перемещая мышь и наблюдая за её координатами в зоне 6 (Рис. 4) найдите расстояние от фрагмента (Рис. 6) до окна команд и обновите значение расстояния в следующих строках сценария.

```
DEFINE($_cmd,$_return1+167)
DEFINE($_cmd,$_retn2+344)
```

7. Установите раскладку клавиатуры «ENG» и отключите «Caps Lock».
8. Запустите сценарий.
9. Наблюдайте выполнение сценария.

Задание 3. Обмен данными сред MPLAB и Matlab при автоматическом тестировании ассемблерной программы.

Необходимо автоматизировать выполнение следующей последовательности

- Генерация случайного числа в Matlab.
 - Передача полученного числа в окно Watch MPLAB.
 - Запуск ассемблерной программы под отладчиком MPLAB.
 - Передачу программе Matlab результата обработки случайного числа ассемблерной программой.
 - Накопление результатов в Matlab.
 - Повторное выполнение последовательности.
1. Подготовка к тестированию ассемблерной программы в среде MPLAB. Проверка работоспособности в ручном режиме.
 - 1.1. Запустите MPLAB (см. раздел [Загрузка MPLAB IDE](#) [3]).
 - 1.2. В рабочем каталоге создайте проект (см. раздел [Создание проекта](#) [3])
 - 1.3. Откройте окно редактора новой программы (Рис. 7).



Рис. 7. Создание нового окна редактора.

- 1.4. Включите в окно редактора следующую ассемблерную программу PIC контроллера.

```

;=====
; Программа добавления OFFSET к переменной RSLT
;=====
          LIST      p=12F629      ; Тип микроконтроллера
          __CONFIG  03E5CH      ; Слово конфигурации (2007h)
W        EQU      0              ; Бит направления результата в аккумулятор
F        EQU      1              ; Бит направления результата в регистр
OFFSET   EQU      20H           ; Присвоение имени OFFSET ячейке памяти 20H
RSLT     EQU      30H           ; Присвоение имени ячейке памяти

;=====
; Начало программы
          ORG      0              ; Начало выполнения программы
          GOTO    START         ; с метки START

START
          MOVLW   4              ; Запись десятичного числа в регистр W
          MOVWF   OFFSET        ; Копирование содержимого W в ячейку OFFSET
          MOVLW   21H           ; Запись шестнадцатеричного числа в регистр W
          MOVWF   RSLT         ; Копирование регистра W в ячейку RSLT

          METKA

          MOVF    OFFSET,W      ; Копирование Offset в регистр W.
          ADDWF   RSLT,F        ; Сложение CPL = W + CPL

          GOTO    METKA        ; Переход на метку

          END                  ; конец программы

```

- 1.5. Включите программу в проект (Рис. 8).

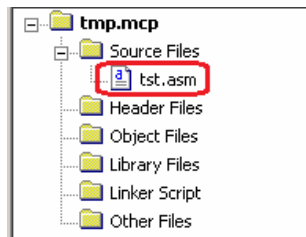


Рис. 8. Положение ассемблерной программы в структуре проекта.

1.6. Откройте отладчик (Рис. 9).

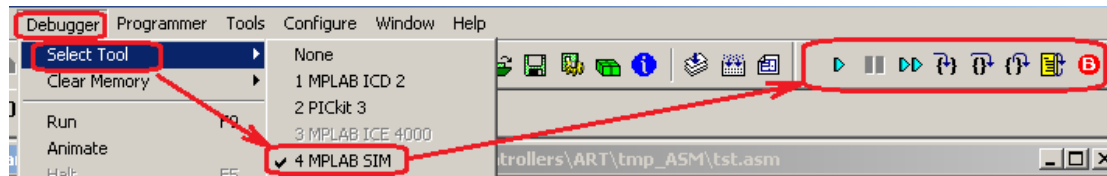


Рис. 9. Вызов отладчика и его компонентов.

1.7. Откройте окна и расставьте их как показано на Рис. 10.

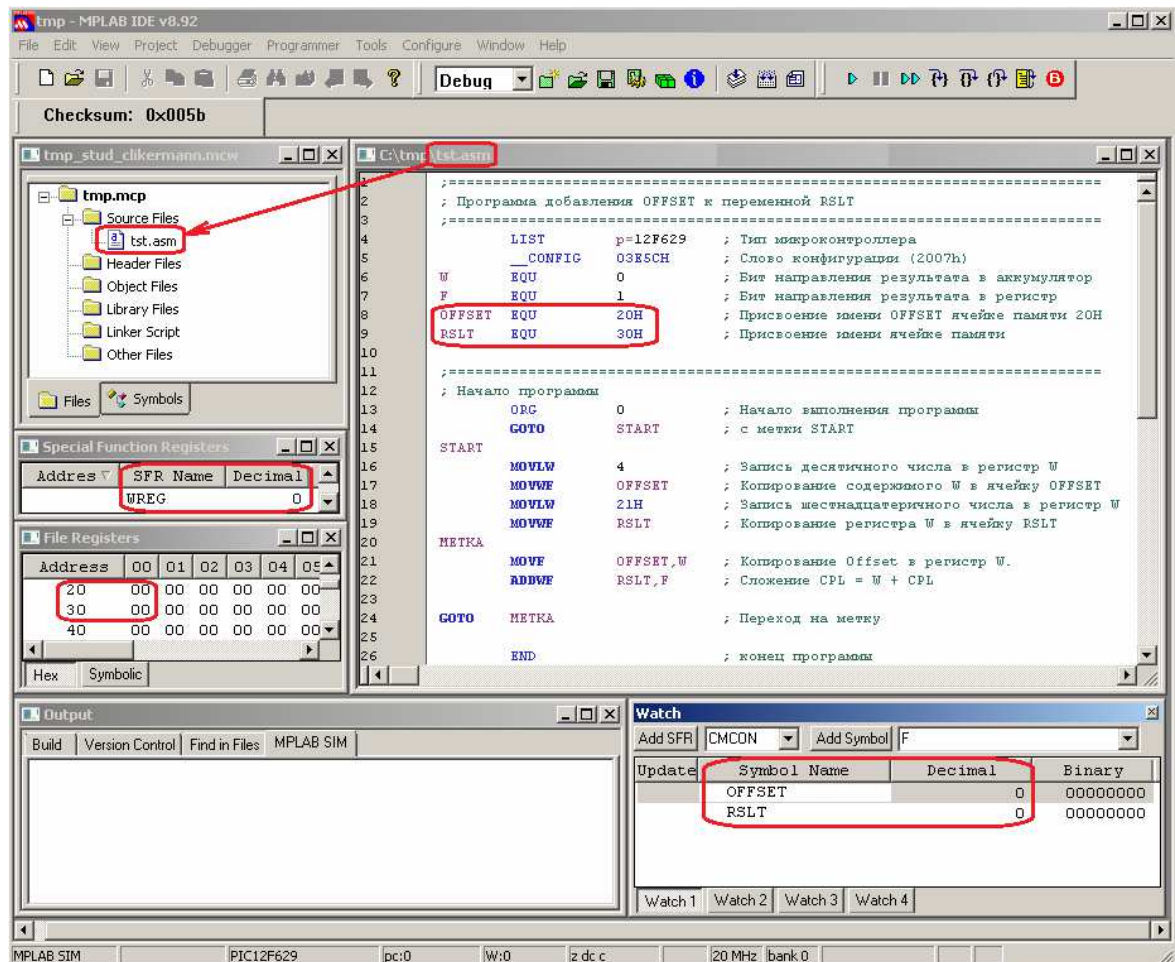


Рис. 10. Размещение окон MPLAB.

1.8. Откомпилируйте программу (см. раздел **Компиляция** [3]). По сообщению **BUILD SUCCEEDED** в окне **Output > Build** (Рис. 11) убедитесь, что компиляция прошла успешно, в противном случае, исправьте ошибки.

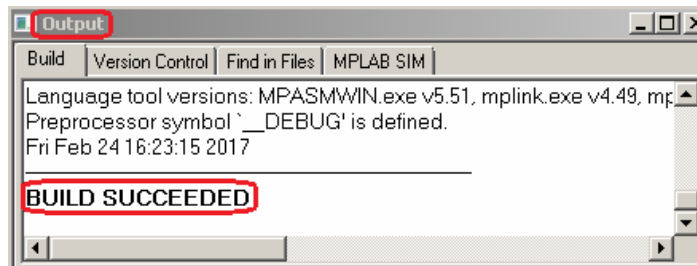



Рис. 11. Сообщение об успешной компиляции.

1.9. Поставьте точку останова в конце основного цикла программы (Рис. 12).



Рис. 12. Точка останова в строке GOTO.


1.10. Наблюдая за состоянием регистров и переменных в окнах выполните программу по шагам до точки останова ( - команда выполнения кода по шагам).

1.11. Запишите результат выполнения программы - переменная RSLT.

1.12. В окне Watch установите новое значение входной переменной OFFSET (Рис. 13).

Symbol Name	Decimal
OFFSET	25
RSLT	37


Рис. 13. Переменные ассемблерной программы в окне Watch.

1.13. Запустите программу в режиме непрерывного выполнения (команда ).

1.14. Запишите результат выполнения программы - значение переменной RSLT в окне Watch (Рис. 14).

Update	Symbol Name	Decimal
	OFFSET	25
	RSLT	62

Рис. 14. Входная переменная OFFSET и переменная результата RSLT.

1.15. Переведите программу в исходное состояние командой .

2. Подготовка к тестированию программы Matlab, включающую генератор случайных чисел и накопление результатов тестирования ассемблерной программы среды MPLAB.

2.1. Запустите Matlab



2.2. В рабочем каталоге создайте m-файл.

2.3. Включите в m-файл следующие команды.

```

clear all % очистить Workspace
rslt = 0;
for i = 1:10 % начало цикла
% Вывод целого случайного числа от 0 до 15 в Command Window
offset(i) = round(16*rand);
disp(sprintf('offset = %d', round(offset(i))));
% Запись переменной rslt в массив
tst(i) = rslt;
end
% Сохранение массива в файле tst_result.mat
save('tst_result','offset','tst');

```

2.4. Запустите программу  и рассмотрите массив результатов (вектор  `tst` $[0,0,0,0,0,0,0,0,0,0]$)

2.5. Поставьте точку останова в строке `tst(i) = rslt;`

3. Подготовка к тестированию программы Clickermann.


3.1. Запустите программу `Clickermann.exe`.

3.2. Откройте редактор (кнопка  РЕДАКТОР)

3.3. Разработайте сценарий тестирования функции ассемблерной программы в MPLAB с использованием случайных чисел Matlab, накопления и сохранения результатов тестирования. Пример сценария приведен ниже.

Код	Код с комментариями
<pre> SUB(CTRL-C) WAITMS(50) KEYDOWN(#CTRL) WAITMS(50) KEYPRESS(#C) WAITMS(50) KEYUP(#CTRL) WAITMS(50) END_SUB SUB(CTRL-V) WAITMS(50) KEYDOWN(#CTRL) WAITMS(50) KEYPRESS(#V) WAITMS(50) KEYUP(#CTRL) WAITMS(50) END_SUB SUB(MPLAB-CTRL-C, \$par1, \$par2) WAITMS(50) LDOWN(\$par1, \$par2) WAITMS(50) LUP(\$par1, \$par2) WAITMS(50) CTRL-C() // <CTRL>+<C> WAITMS(50) END_SUB LOGCLEAR \$MP = WNDFIND("tmp - MPLAB IDE v8.92") \$ML = WNDFIND("MATLAB R2015a") \$SL = WNDFIND("Editor - C:\tmp\tmp.m") LOGWRITE("MPLAB=",\$MP, ", MATLAB=",\$ML) LOGWRITE("Matlab m-file:",\$SL) If (\$MP!0) else HINTPOPUP("MPLAB IS NOT FOUND", "CLICKERMANN") END_IF If (\$ML!0) If (\$SL!0) \$хMP = 761 \$уMP = 628 \$хML = 1155 \$уML = 336 </pre>	<pre> //===== // Подпрограммы. // Подпрограммы должны располагаться в сценарии до основной программы //===== // Подпрограмма имитации команды Ctrl+C SUB(CTRL-C) // CTRL-C - имя подпрограммы WAITMS(50) // Задержка на 50 мсек. KEYDOWN(#CTRL) // Нажатие и удержание клавиши WAITMS(50) KEYPRESS(#C) // Кратковременное нажатие клавиши CTRL WAITMS(50) KEYUP(#CTRL) // Отпускание клавиши WAITMS(50) END_SUB // Конец подпрограммы //===== // Подпрограмма имитации команды Ctrl+V SUB(CTRL-V) WAITMS(50) KEYDOWN(#CTRL) // Нажатие и удержание клавиши WAITMS(50) KEYPRESS(#V) // Кратковременное нажатие клавиши WAITMS(50) KEYUP(#CTRL) // Отпускание клавиши WAITMS(50) END_SUB //===== // Подпрограмма с параметрами // Копирование переменных с координатами \$par1, \$par2 в буфер SUB(MPLAB-CTRL-C, \$par1, \$par2) // MPLAB-CTRL-C - имя подпрограммы WAITMS(50) LDOWN(\$par1, \$par2) // Нажатие на ЛКМ WAITMS(50) LUP(\$par1, \$par2) // Отпускание ЛКМ V CTRL-C() // Вызов подпрограммы CTRL-C WAITMS(50) END_ //===== // Основная программа //===== LOGCLEAR Очистка Лога кликера //Присвоение переменным уникальных кодов заголовков программ \$MP = WNDFIND("tmp - MPLAB IDE v8.92") \$ML = WNDFIND("MATLAB R2015a") \$SL = WNDFIND("Editor - C:\tmp\tmp.m") // Отображение в окне Лог кодов заголовков найденных программ LOGWRITE("MPLAB=",\$MP, ", MATLAB=",\$ML) LOGWRITE("Matlab m-file:",\$SL) // Если программы найдены (код не равен нулю) начинается выполнение If (\$MP!0) else // Вывод сообщения если программа не найдена HINTPOPUP("MPLAB IS NOT FOUND", "CLICKERMANN") END_IF If (\$ML!0) If (\$SL!0) //Присвоение абсолютных координат центра строки с десятичным значением переменной OFFSET MPLAB \$хMP = 265 \$уMP = 1016 // Абсолютные координаты начала переменной offset в командном окне Matlab \$хML = 2229 \$уML = 1144 </pre>

<pre> WINDUMP(\$SSL) KEYPRESS(#F5) WAITMS(2000) FOR(\$a,\$a<10) LDOWN(\$xML,\$yML) MOVE(\$xML+31,\$yML) LUP(\$xML+31,\$yML) CTRL-C() LCLICK(\$xMP,\$yMP) WAITMS(50) CTRL-V() WINDUMP(\$MP) KEYPRESS(#F9) WAITMS(250) MPLAB-CTRL-C(\$xMP,\$yMP+16) WINDUMP(\$SML) KEYSTRING("rslt=") CTRL-V() KEYSTRING(";") KEYPRESS(#ENTER) WINDUMP(\$SSL) KEYPRESS(#F5) WAITMS(500) END_CYC HALT </pre>	<pre> // Активация и запуск m-программы Matlab до точки останова WINDUMP(\$SSL) // Активация Matlab KEYPRESS(#F5) // Запуск, клавиша F5 WAITMS(2000) // Копирование результата // Начало основного цикла // ===== FOR(\$a,\$a<10) // 10 циклов, переменная \$a начинается с нуля // Копирование переменной Matlab в MPLAB LDOWN(\$xML+31*3,\$yML) // Нажатие на ЛКМ в начале строки с переменной offset MOVE(\$xML+31*4,\$yML) // Перемещение курсора в конец строки LUP(\$xML+31*4,\$yML) // Отказание ЛКМ CTRL-C() // Копирование выделенной переменной в буфер LCLICK(\$xMP,\$yMP+16) // Активация зоны переменной OFFSET окна Watch среды MPLAB WAITMS(50) CTRL-V() // Вставка содержимого буфера со значением переменной offset Matlab // Запуск отладчика с обновлённой переменной OFFSET до точки останова WINDUMP(\$MP) // Активация MPLAB KEYPRESS(#F9) // Запуск отладчика WAITMS(250) // Копирование результата - переменной RSLT из MPLAB в Matlab MPLAB-CTRL-C(\$xMP,\$yMP+16) // Копирование в буфер переменной из указанной позиции WINDUMP(\$SML) // Активация Matlab KEYSTRING("rslt=") // Вывод строки rslt= в окне команд CTRL-V() // Вывод переменной из буфера KEYSTRING(";") // Добавление символа завершения строки ";" KEYPRESS(#ENTER) // Возврат каретки // Запуск m-программы до точки останова. // Генерация нового случайного значения переменной offset WINDUMP(\$SSL) // Активация Matlab KEYPRESS(#F5) // Запуск m-программы, клавиша F5 WAITMS(500) END_CYC // Конец цикла HALT // Конец выполнения сценария </pre>
--	---

3.4. Откройте менеджер окон (кнопка  Рис. 3), найдите заголовки программ MPLAB (Рис. 15), Matlab и редактора m-файла и обновите в сценарии переменным \$MP, \$ML и \$SSL - присвойте им соответствующие названия заголовков.

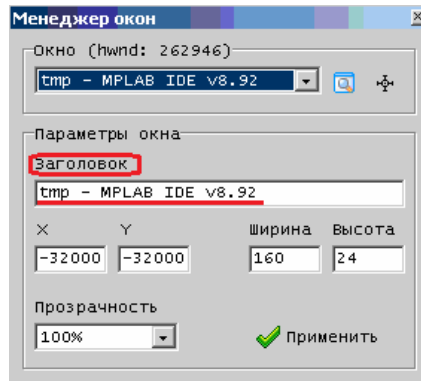


Рис. 15. Отображение заголовка программы в окне Менеджера.

```

$MP = WNDFFIND("tmp - MPLAB IDE v8.92")
$ML = WNDFFIND("MATLAB R2015a")
$SSL = WNDFFIND("Editor - C:\tmp\tmp.m")

```

3.5. Обновите в сценарии абсолютные координаты вывода переменной OFFSET окна Watch MPLAB и координаты переменной offset выводимые в окно команд Matlab. Позиции переменных отмечены на (Рис. 16).

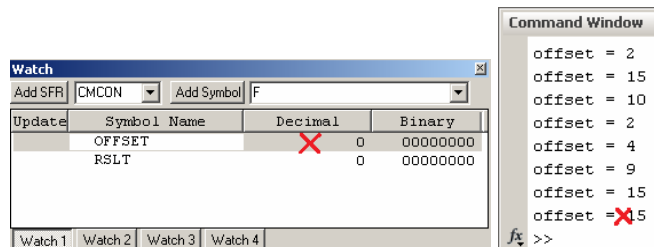


Рис. 16. Перекрестия координаты которых используются в сценарии.

```

$xMP = 766
$yMP = 628
$xML = 1193

```

\$yML = 336

4. Проверьте исходное состояние окон Matlab (Рис. 17).

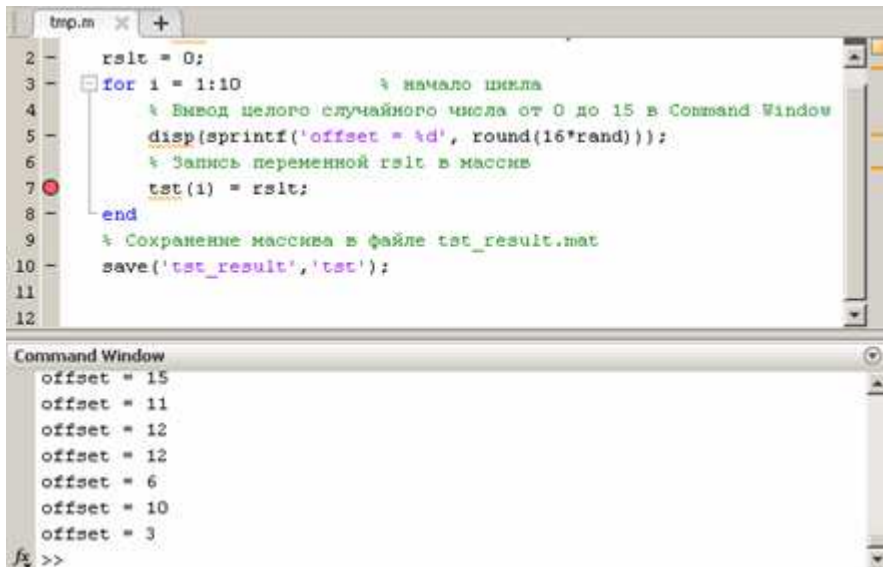



Рис. 17. Исходное состояние редактора и окна команд Matlab.

5. Запустите сценарий Clikermann (кнопка ). Проверьте правильность последовательности запуска программ и передачи данных (Рис. 18).

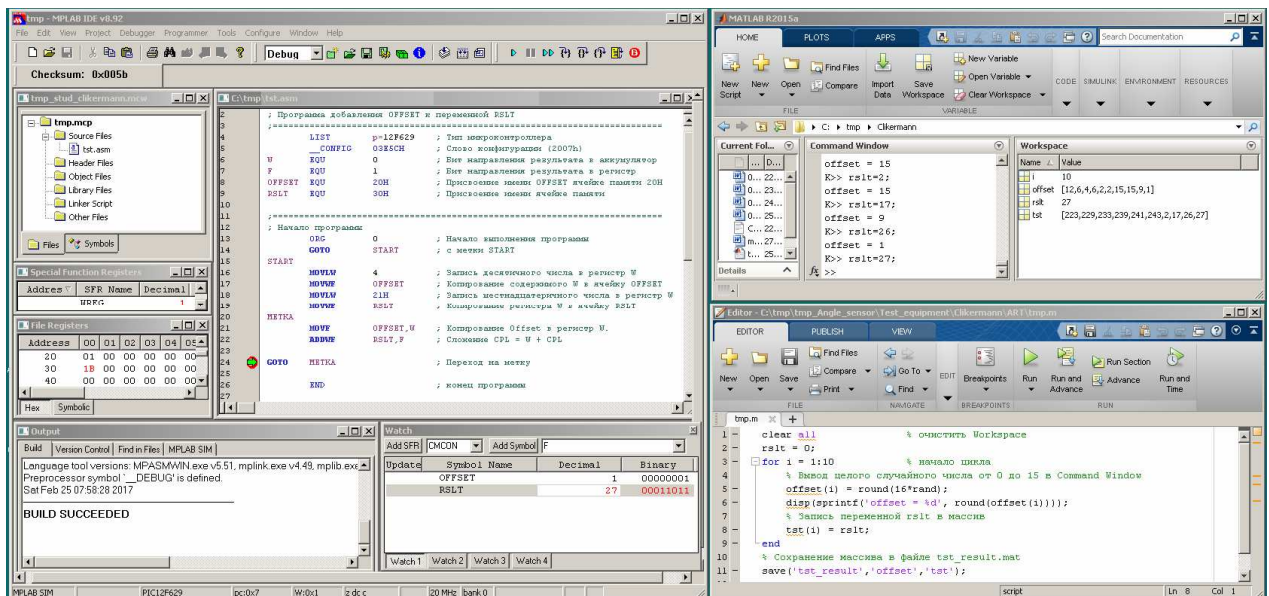


Рис. 18. Пример состояния окон MPLAB и Matlab после выполнения сценария.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какова примерная частота отслеживания положения указателя мыши автокликером в зоне б главного окна (**Error! Reference source not found.**) ?
2. Насколько быстро автокликер вычисляет координаты заданного фрагмента изображения на экране компьютера?
3. Как быстро автокликер передаёт данные через буфер обмена?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Dr. Bob Davidov. Распознавание объектов и определение координат в среде Matlab http://portalnp.ru/wp-content/uploads/2016/06/21.01_MATLAB_Recognition_2a.pdf.
2. CLICKERMANN <http://crapware.aidf.org>.
3. Dr. Bob Davidov. Средства разработки программного кода PIC контроллеров http://portalnp.ru/wp-content/uploads/2017/02/06.05_MPLAB_IDE_for-PIC_controllers_1b.pdf.
4. Dr. Bob Davidov. Портал научно-практических публикаций <http://portalnp.ru/author/bobdavidov>.