

# Ứng dụng MFC (Visual C++) trong mô phỏng Robot và hệ Cơ điện tử

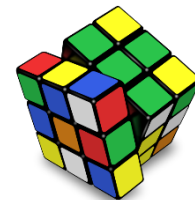


## Bài 1: Giới thiệu Visual C++ và MFC

PHẠM MINH QUÂN

[mquan.ph@gmail.com](mailto:mquan.ph@gmail.com)

# Nội dung

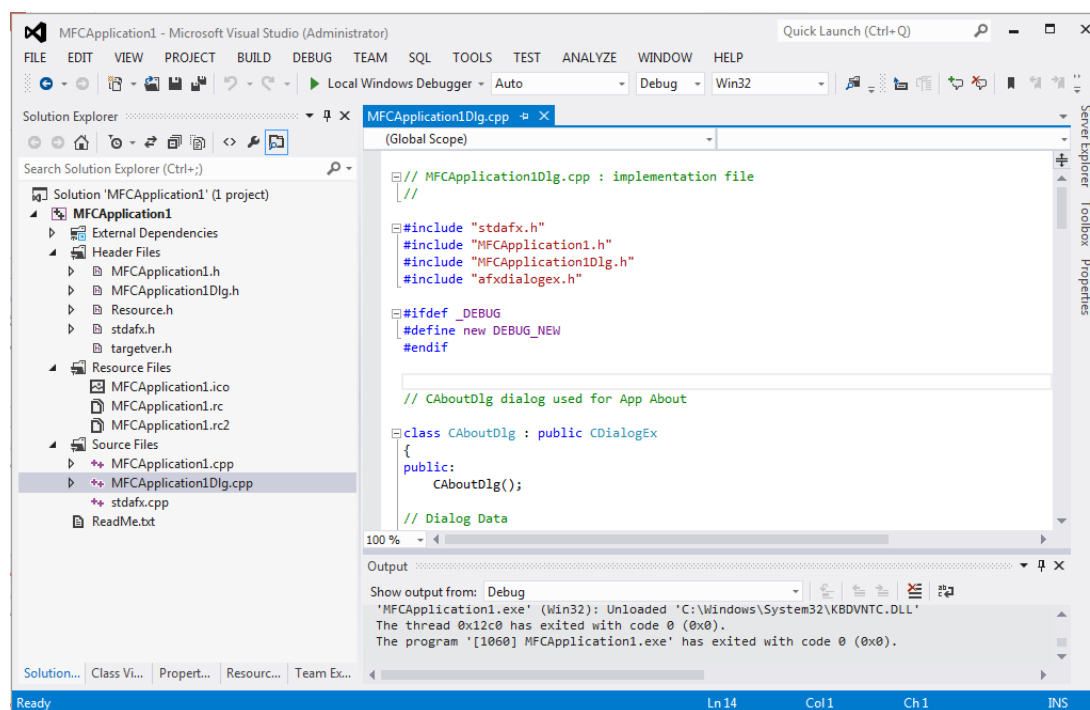


1. Giới thiệu Visual Studio và MFC
2. Hướng dẫn xây dựng một ứng dụng kiểu Dialog-based
  - 2.1. Cấu trúc của lớp CDialog, các biến và hàm cơ bản
  - 2.2. Cách sử dụng một số Control đơn giản: Button, EditControl
  - 2.3. Cập nhật dữ liệu liên tục với hàm OnTimer()
3. Mở rộng
  - 3.1. Thư viện hỗ trợ online MSDN của Microsoft
  - 3.2. Giới thiệu phần mềm e-Robot của phòng Cơ điện tử, Viện Cơ học

# 1. Giới thiệu Visual Studio và MFC

❑ Visual Studio là bộ công cụ của Microsoft:

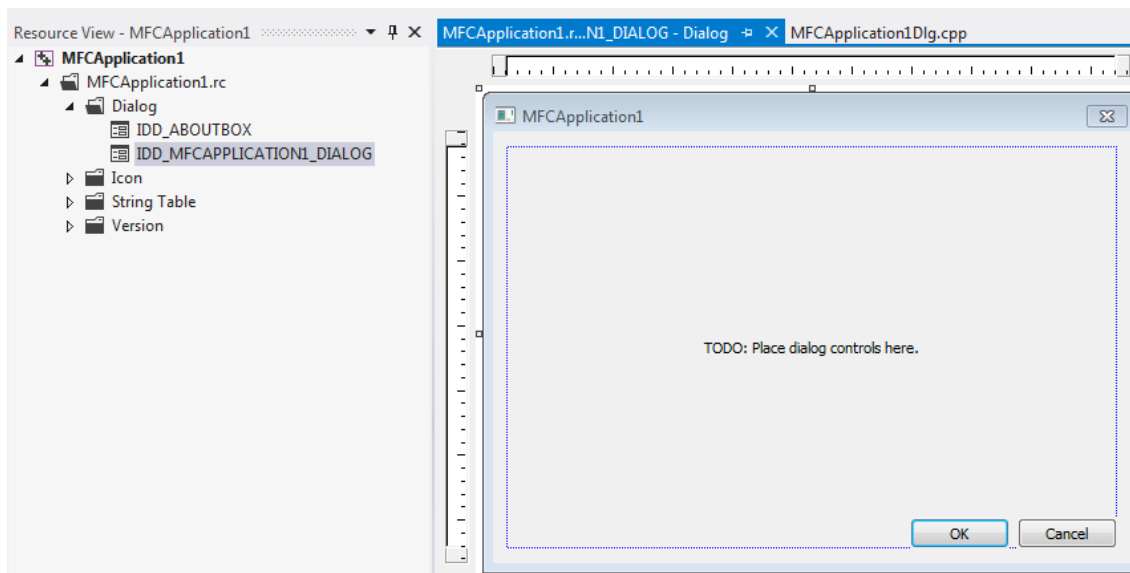
- Cung cấp môi trường lập trình (IDE) và các thư viện để phục vụ cho việc lập trình các ứng dụng Windows.
- Hỗ trợ các ngôn ngữ như: C++, C#, F#, Visual Basic...



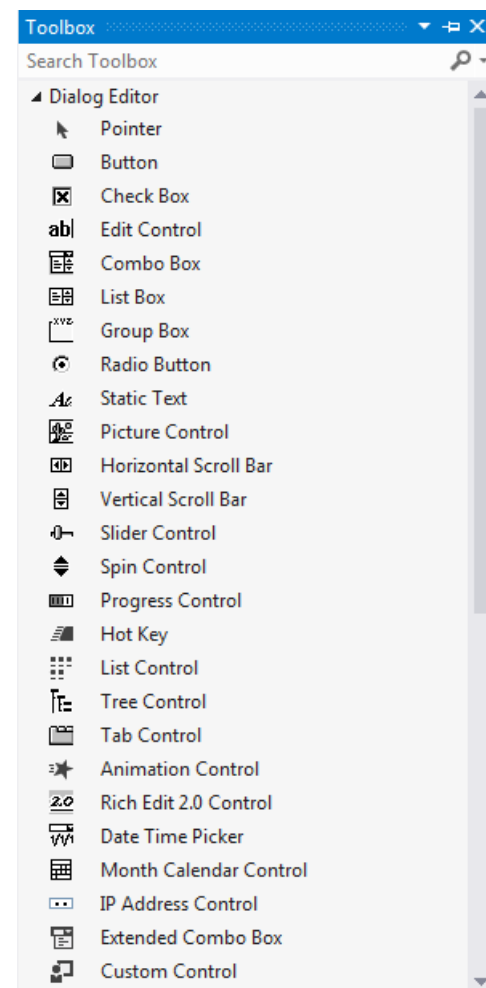
# 1. Giới thiệu Visual Studio và MFC

❑ MFC là thư viện cơ sở chứa các lớp C++ do Microsoft cung cấp:

- Tạo và quản lý các control của windows dễ dàng và nhanh chóng.
- Xây dựng sẵn khung ứng dụng với cấu trúc chương trình đơn giản, uyển chuyển và dễ phát triển.



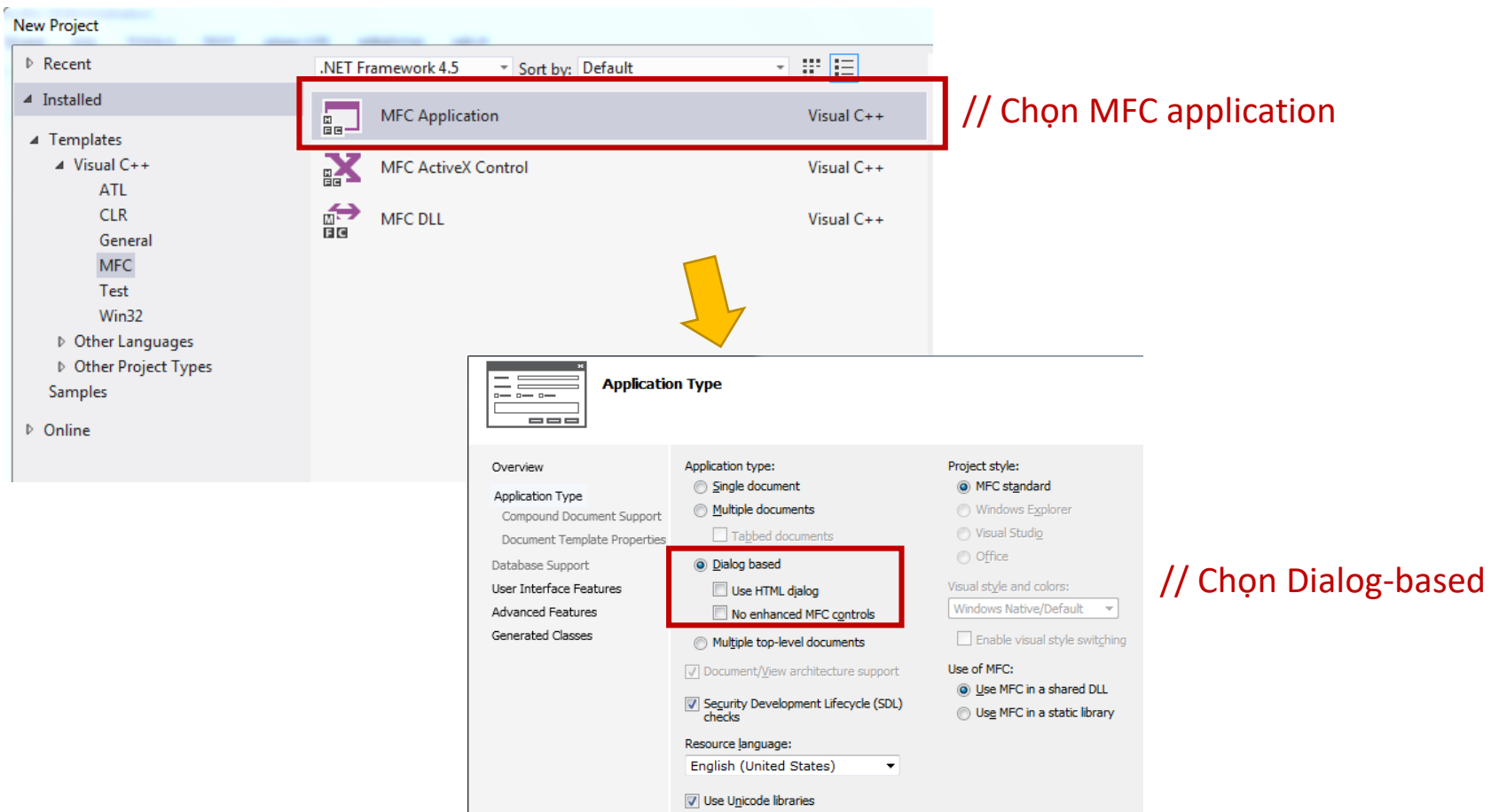
Thiết kế ứng dụng kiểu hộp thoại (dialog-based)



Toolbox chứa các điều khiển (control) xây dựng sẵn

## 2. Xây dựng một ứng dụng MFC dialog-based

❑ Tạo một project mới, chọn MFC Application, kiểu Dialog based



The image shows the 'New Project' dialog in Visual Studio. The 'MFC Application' template is selected under the 'Visual C++' category. A red box highlights this selection, with a red arrow pointing to it and the text '// Chọn MFC application'. Below this, the 'Application Type' settings are shown. The 'Dialog based' option is selected under 'Application type', highlighted with a red box and a red arrow, with the text '// Chọn Dialog-based'. Other options include 'Single document', 'Multiple documents', 'Tabbed documents', 'Use HTML dialog', 'No enhanced MFC controls', 'Multiple top-level documents', 'Document/view architecture support', 'Security Development Lifecycle (SDL) checks', 'Project style' (MFC standard, Windows Explorer, Visual Studio, Office), 'Visual style and colors' (Windows Native/Default), 'Use of MFC' (Use MFC in a shared DLL, Use MFC in a static library), 'Resource language' (English (United States)), and 'Use Unicode libraries'.

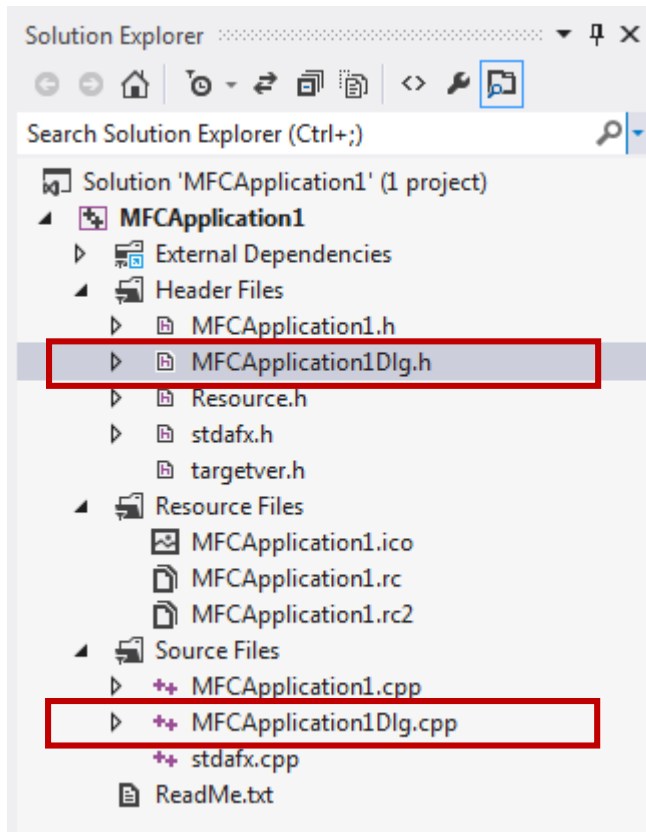
// Chọn MFC application

// Chọn Dialog-based

## 2. Xây dựng một ứng dụng MFC dialog-based

### 2.1. Lớp *CDialog*, các biến và hàm cơ bản

❑ Quan sát các file được tạo sẵn trong cửa sổ Solution Explorer



// File ...Dlg.h khai báo lớp quản lý dialog

// File ...Dlg.cpp định nghĩa lớp quản lý dialog

## 2. Xây dựng một ứng dụng MFC dialog-based

### 2.1. Lớp *CDialog*, các biến và hàm cơ bản

❑ Mở và xem xét nội dung file ...Dlg.h

```
// CMFCApplication1Dlg dialog
class CMFCApplication1Dlg : public CDialogEx
{
// Construction
public:
    CMFCApplication1Dlg(CWnd* pParent = NULL); // standard constructor

// Dialog Data
    enum { IDD = IDD_MFCAPPLICATION1_DIALOG };

protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support

// Implementation
protected:
    HICON m_hIcon;

// Generated message map functions
    virtual BOOL OnInitDialog();
    afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
    afx_msg void OnPaint();
    afx_msg HCURSOR OnQueryDragIcon();
    DECLARE_MESSAGE_MAP()
};
```

// Hàm thực thi khi dialog bắt đầu được mở ra, mang ý nghĩa khởi tạo dialog

// Hàm thực hiện vẽ giao diện dialog, thực thi mỗi lần giao diện dialog cần vẽ lại

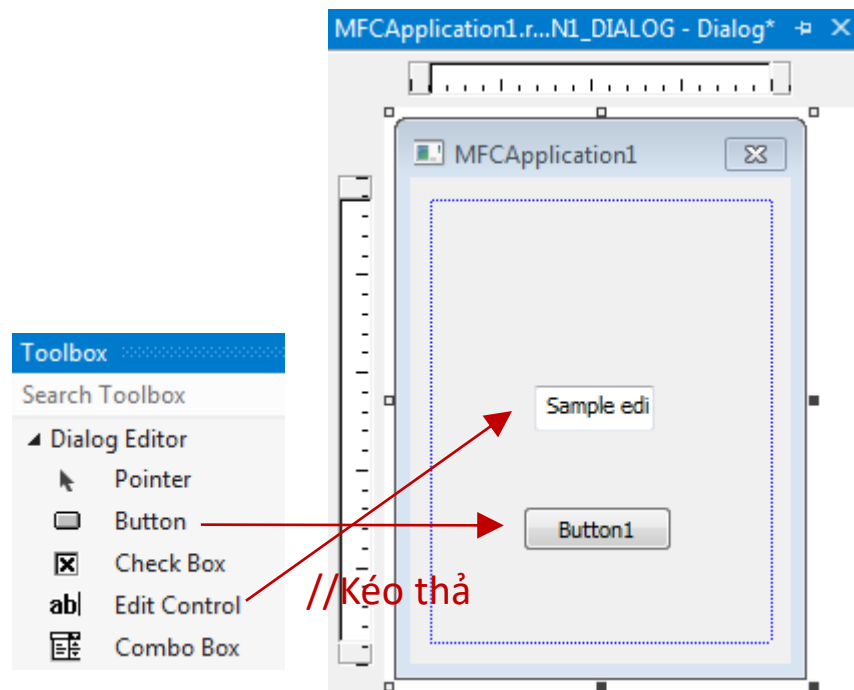
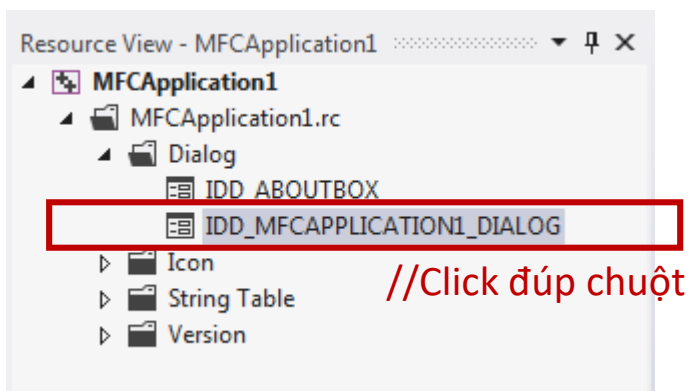
## 2. Xây dựng một ứng dụng MFC dialog-based

### 2.2. Cách sử dụng một số control cơ bản

❑ Mở giao diện Dialog trong cửa sổ Resource View



❑ Kéo thả các control từ Toolbox vào Dialog





## 2. Xây dựng một ứng dụng MFC dialog-based

### 2.2. Cách sử dụng một số control cơ bản

#### ❑ Cách sử dụng Button:

- Click đúp vào nút Button trên giao diện thiết kế Dialog, hàm OnBnClicked...() được tự động tạo ra, thuộc lớp C...Dlg.

➤ Quan sát code được chương trình tạo tự động

File "...Dlg.h"

```
public:
    afx_msg void OnBnClickedButton1();
}; // Khai báo hàm
```

File "...Dlg.cpp"

```
BEGIN_MESSAGE_MAP(CMFCApplication1Dlg, CDialogEx)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_BN_CLICKED(IDC_BUTTON1, &CMFCApplication1Dlg::OnBnClickedButton1) // Gán hàm với sự kiện Click chuột
END_MESSAGE_MAP()

void CMFCApplication1Dlg::OnBnClickedButton1() // Định nghĩa hàm
{
    // TODO: Add your control notification handler code here
}
```

- Viết code bên trong hàm OnBnClicked...() này, đoạn code sẽ được thực thi khi người dùng Click vào button trong lúc chạy chương trình.

```
void CMFCApplication1Dlg::OnBnClickedButton1()
{
    // TODO: Add your control notification handler code here
    OnOK(); // Hàm đóng dialog
}
```

➡ Kết quả chạy: Khi click vào Button, dialog được đóng lại

## 2. Xây dựng một ứng dụng MFC dialog-based

### 2.2. Cách sử dụng một số control cơ bản

#### ❑ Cách sử dụng EditControl:

- Tạo biến để quản lý EditControl:
  - Nhấn chuột phải vào editcontrol trên giao diện dialog, chọn Add Variable.
  - Khi hộp thoại Add Member Variable hiện ra, đặt tên và chọn kiểu biến mong muốn.
- Trường hợp chọn kiểu biến ***“Value”***

Access: public

☒ Control variable

Variable type: double

variable name: edit1\_value

Control ID: IDC\_EDIT1

Control type: EDIT

Min value:

Max value:

Category: Value

Max chars:

//2. Chọn kiểu giá trị *double|float|...*

//3. Đặt tên biến tùy ý

//1. Chọn kiểu biến *Value*

## 2. Xây dựng một ứng dụng MFC dialog-based

### 2.2. Cách sử dụng một số control cơ bản

#### ❑ Cách sử dụng EditControl:

- Trường hợp chọn kiểu biến “**Value**”

➤ Quan sát các đoạn mã khai báo biến được tạo tự động

File “...Dlg.h”

```
protected:
    HICON m_hIcon;

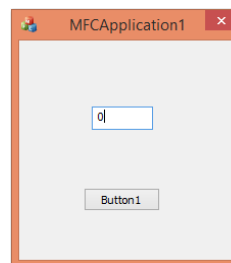
    // Generated message map functions
    virtual BOOL OnInitDialog();
    afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
    afx_msg void OnPaint();
    afx_msg HCURSOR OnQueryDragIcon();
    DECLARE_MESSAGE_MAP()
public:
    afx_msg void OnBnClickedButton1();
    double edit1_value;
};
```

File “...Dlg.cpp”

```
CMFCApplication1Dlg::CMFCApplication1Dlg(CWnd* pParent /*=NULL*/)
: CDialogEx(CMFCApplication1Dlg::IDD, pParent)
, edit1_value(0)
{
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CMFCApplication1Dlg::DoDataExchange(CDataExchange* pDX)
{
    CDialogEx::DoDataExchange(pDX);
    DDX_Text(pDX, IDC_EDIT1, edit1_value);
}
```

➡ Kết quả khi chạy chương trình:



## 2. Xây dựng một ứng dụng MFC Dialog-based

### 2.2. Cách sử dụng một số control cơ bản

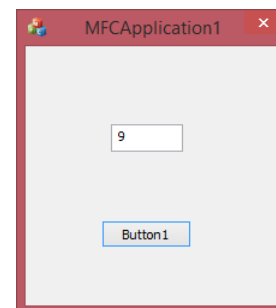
#### ❑ Cách sử dụng EditControl:

- Trường hợp chọn kiểu biến “**Value**”

#### ➤ Sửa hàm OnBnClicked...():

```
void CMFCApplication1Dlg::OnBnClickedButton1()
{
    // TODO: Add your control notification handler code here
    edit1_value = 9;
    UpdateData(FALSE); //Hàm UpdateData cập nhật giá trị các biến của dialog
                       UpdateData(FALSE): cập nhật từ bộ nhớ ra giao diện
                       UpdateData(TRUE): cập nhật từ giao diện vào bộ nhớ
}
```

➡ *Kết quả chạy: Khi nhấn Button, giá trị ô editbox được cập nhật*



## 2. Xây dựng một ứng dụng MFC Dialog-based

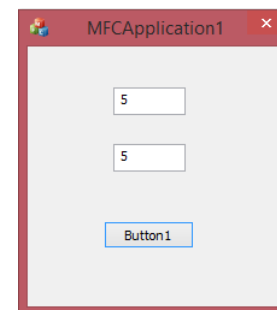
### 2.2. Cách sử dụng một số control cơ bản

#### ❑ Cách sử dụng EditControl:

- Trường hợp chọn kiểu biến “**Value**”
  - Thêm EditControl thứ 2 vào giao diện và tạo biến kiểu value thứ 2 gắn với EditControl mới này. Chỉnh sửa hàm OnBnClicked...():

```
void CMFCApplication1Dlg::OnBnClickedButton1()
{
    // TODO: Add your control notification handler code here
    UpdateData(TRUE);
    edit2_value = edit1_value;
    UpdateData(FALSE);
}
```

➡ *Kết quả chạy: Nhập giá trị cho ô Editbox thứ nhất, nhấn Button, giá trị sẽ được copy sang ô Editbox thứ 2*

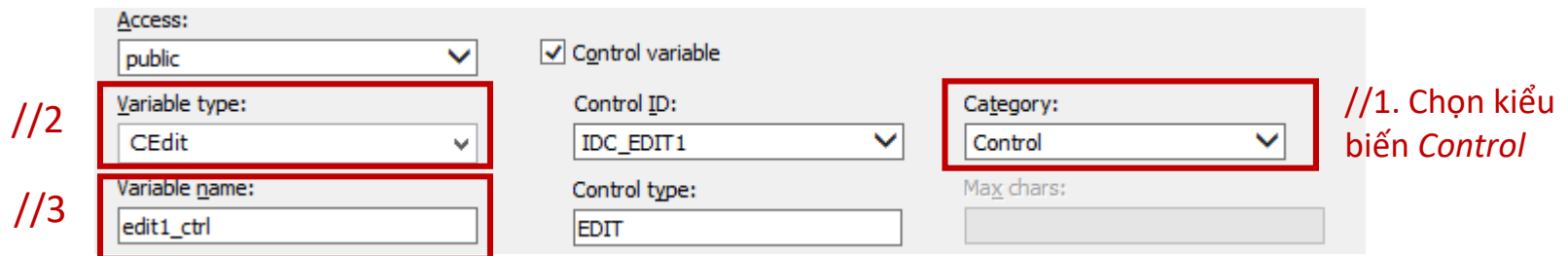


## 2. Xây dựng một ứng dụng MFC Dialog-based

### 2.2. Cách sử dụng một số control cơ bản

#### ❑ Cách sử dụng EditControl:

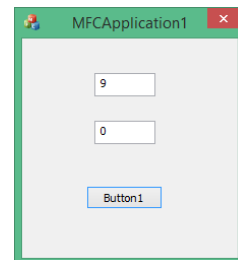
- Trường hợp chọn kiểu biến “**Control**”



- Sửa hàm OnBnClicked...() để cập nhật giá trị editbox

```
void CMFCApplication1Dlg::OnBnClickedButton1()
{
    // TODO: Add your control notification handler code here
    edit1_ctrl.SetWindowTextW(_T("9"));
}
```

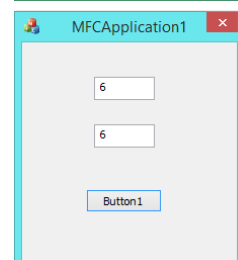
➡ Kết quả chạy:



- Sửa hàm OnBnClicked...() để copy giá trị từ ô thứ nhất sang ô thứ 2

```
void CMFCApplication1Dlg::OnBnClickedButton1()
{
    // TODO: Add your control notification handler code here
    CString sWindowText;
    edit1_ctrl.GetWindowText(sWindowText);
    edit2_ctrl.SetWindowTextW(sWindowText);
}
```

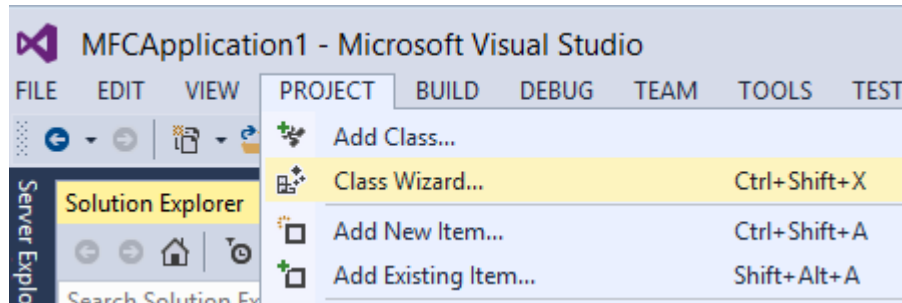
➡ Kết quả chạy:



## 2. Xây dựng một ứng dụng MFC Dialog-based

### 2.3. Cập nhật dữ liệu liên tục với hàm OnTimer()

- ❑ Thêm hàm OnTimer() vào lớp quản lý hộp thoại bằng Class Wizard



// Chọn menu PROJECT -> Class Wizard...

## 2. Xây dựng một ứng dụng MFC Dialog-based

### 2.3. Cập nhật dữ liệu liên tục với hàm *OnTimer()*

❑ Thêm hàm *OnTimer()* vào lớp quản lý hộp thoại bằng Class Wizard

The screenshot shows the MFC Class Wizard dialog box with the following settings:

- Project: MFCApplication1
- Class name: CMFCApplication1Dlg
- Base class: CDialogEx
- Class declaration: mfcapplication1dlg.h
- Class implementation: mfcapplication1dlg.cpp
- Resource: IDD\_MFCAPPLICATION1\_DIALOG

The **Messages** tab is selected. The **Messages** list on the left contains the following messages:

- WM\_SYSCOLORCHANGE
- WM\_SYSCOMMAND**
- WM\_SYSDEADCHAR
- WM\_SYSKEYDOWN
- WM\_SYSKEYUP
- WM\_TCARD
- WM\_THEMECHANGED
- WM\_TIMECHANGE
- WM\_TIMER**
- WM\_UNICHAR
- WM\_UNINITMENUPOPUP
- WM\_UPDATEUISTATE
- WM\_USERCHANGED

The **Existing handlers** list on the right contains the following handlers:

Function name	Message
OnPaint	WM_PAINT
OnQueryDragIcon	WM_QUERYDR...
OnSysCommand	WM_SYSCOM...

The **Add Handler** button is highlighted. The **WM\_TIMER** message is highlighted in the Messages list.

Annotations:

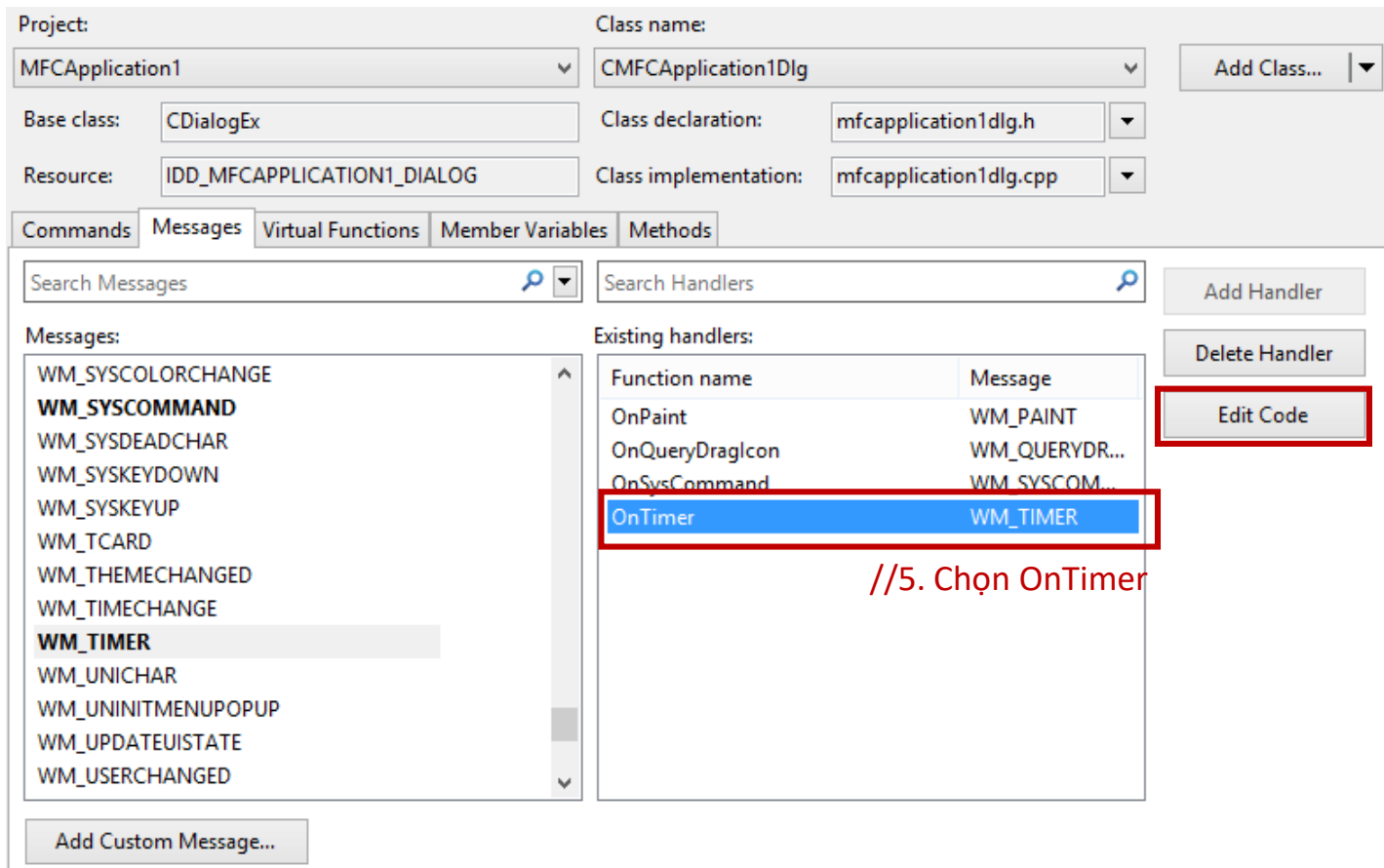
- //1. Chọn lớp cần thêm hàm (Select the class to add the function)
- //2. Mở thẻ Messages (Open the Messages tab)
- //3. Chọn WM\_TIMER (Select WM\_TIMER)
- //4. Nhấn Add Handler (Press Add Handler)



## 2. Xây dựng một ứng dụng MFC Dialog-based

### 2.3. Cập nhật dữ liệu liên tục với hàm OnTimer()

- ❑ Thêm hàm OnTimer() vào lớp quản lý hộp thoại bằng Class Wizard



//6. Edit Code

//5. Chọn OnTimer

## 2. Xây dựng một ứng dụng MFC Dialog-based

### 2.3. Cập nhật dữ liệu liên tục với hàm OnTimer()

#### ❑ Sử dụng Timer

- Hàm OnTimer() được gọi đến mỗi khi bộ đếm thời gian (Timer) đạt tới giá trị đặt trước, sau đó Timer sẽ reset.
- Do đó, nội dung của hàm được thực thi sau mỗi bước thời gian cách đều nhau.

#### ➤ Chỉnh sửa nội dung hàm OnTimer()

```
void CMFCApplication1Dlg::OnTimer(UINT_PTR nIDEvent)
{
    // TODO: Add your message handler code here and/or call default
    edit1_value++;
    edit2_value--;
    UpdateData(FALSE);
    CDialogEx::OnTimer(nIDEvent);
}
```

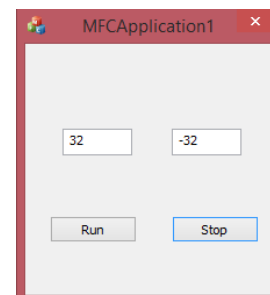
#### ➤ Tạo 2 nút Button trên giao diện một nút để thiết lập timer, một nút để dừng timer

```
void CMFCApplication1Dlg::OnBnClickedButton1()
{
    // TODO: Add your control notification handler code here
    SetTimer(1, 100, NULL); //Thiết lập timer số 1, bước thời gian là 100ms
}

void CMFCApplication1Dlg::OnBnClickedButton2()
{
    // TODO: Add your control notification handler code here
    KillTimer(1);
    //Hủy timer số 1
}
```



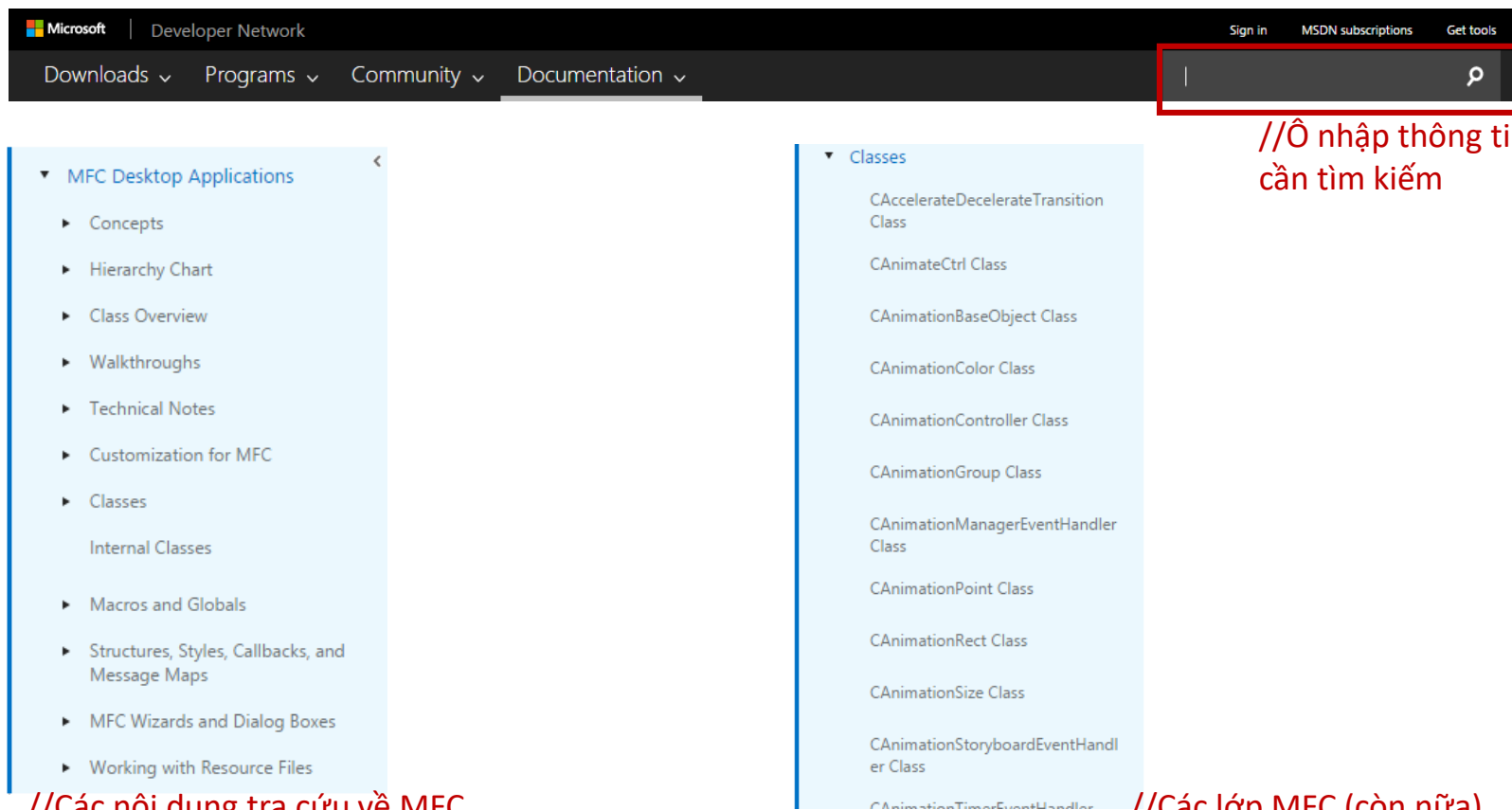
Kết  
quả  
chạy:



## 3. Mở rộng

### 3.1. Thư viện hỗ trợ online MSDN của Microsoft

<https://msdn.microsoft.com/en-us/library/d06h2x6e.aspx>

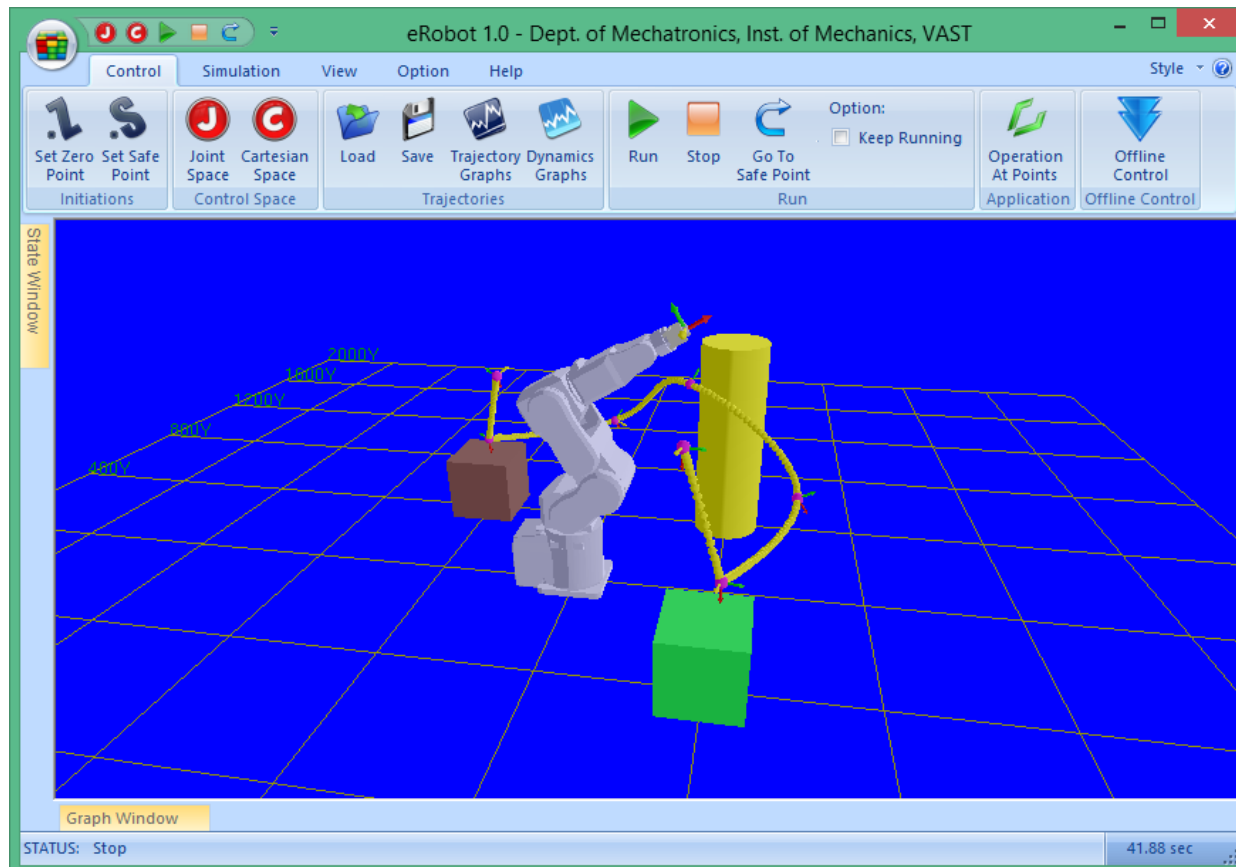


//Các nội dung tra cứu về MFC

//Các lớp MFC (còn nữa)

## 3. Mở rộng

### 3.2. Giới thiệu phần mềm e-Robot của phòng Cơ điện tử, Viện Cơ học (Xem tài liệu kèm theo)



Giao diện phần mềm e-Robot 1.0

# Hết Bài 1

---

