Programming with Windows Visual C++

April 25, 2012

1 Add New Motion

- Create a new motion activated through a GUI button press

1.1 Add New Motion Button

- Open the Rainbow.dsw file in the C://Rainbow folder using Visual C++
- Dialog boxes are under Resource tab below the left panel in the Dialog folder
- Open IDD_USER_DIALOG (or a dialog window you created)
- Add a new button via the tool panel
  - Click the button icon (When the mouse is placed over it, it will say "Button").
  - Then click and drag on dialog window
- Right click on the new button, then select **Properties**
  - Set the Button ID to its corresponding variable name in the code (e.g. IDC_RAISE_ARMS)
  - Caption: What is displayed on the GUI (e.g. Raise Both Arms)
- Set the function to run on button click:
  - Double Click on button, change the function name or accept the suggested name, click OK (e.g. OnRaiseArms)
  - On OK, the program will bring up file UserDlg.cpp with your new function

1.2 Modify Button Click Function in UserDlg.cpp

- This code goes in the new function created to run on button click (OnRaiseArms)
- Make sure no other commands are running

```
if (pSharedMemory -> CommandFlag ==NO_ACT)
   {...}
```

Set the command flag to be the desired action (This flag tells the RTX system what code in core.cpp to run)

```
PSharedMemory-> CommandFlag = NewCommandName;
```

Example Code:

```
void CUserDlg::OnRaiseArms()
{
    /* OnRaiseArms()
    **
    ** If no command is currently running, set the appropriate
    ** command flag so that the robot will raise both of its
    ** arms together.
    **
    */


    if(pSharedMemory->CommandFlag == NO_ACT)     // Make sure no other command is running
    {
       pSharedMemory->CommandFlag = RAISE_ARMS; // Run the case statement (in Core.cpp)
                                                //corresponding to RAISE_ARMS}
    }

}
```

1.3 Add Command Name to CommonDefinition.h

- Open CommonDefinition.h (under header files for Rainbow)
- Add CommandName to Commands from Win32 to RTX

  ```
  typdef enum
  {
  OLDcommandNames,RAISE_ARMS
  }_COMMAND_FLAG;
  ```

1.4 Add command to Core.cpp

- Open the Core.cpp file
  - Click File View tab, expand Rainbow files folder and Source Files subfolder
- Add case for new command

  ```
  Switch (PSharedMemory -> CommandFlag)
  ```

  - This code will be run when the CommandFlag is set to the command name
- Set the joint angles and the duration
- Set MoveJointAngle (joint,float angle, float ms, mode)
  - joint = joint # to move
  - angle = degrees to set for joint
  - ms = move time in milliseconds
  - mode = absolute (0x01) or relative (0x00) motion
  - Absolute moves to the specified angle, relative moves the specified # of degrees from the current position
- Set the control mode

```
PSharedMemory -> MotorControlMode = CTRLMODE_POSITION_CONTROL_WIN;
```

- When finished, reset command Flag to NO_ACT

  ```
  PSharedMemory -> CommandFlag = NO_ACT;
  ```

Example Code:

```
case RAISE_ARMS:
  /* case RAISE_ARMS:
  **
  ** Have the robot raise both arms from the shoulder using absolute joint references.
  ** This motion will take 2 seconds to complete.
  **
  */

  // Make sure the joints are not already moving
  if((Joint[RSP].MoveFlag == false) && (Joint[LSP].MoveFlag == false))
  {


     SetMoveJointAngle(RSP,-40.0f,2000.0f,0x01); // Raise the right arm so the
                                                 //shoulder is at -40 degrees
     SetMoveJointAngle(LSP,-40.0f,2000.0f,0x01); // Raise the left arm so the
                                                 //shoulder is at -40 degrees


     pSharedMemory->MotorControlMode = CTRLMODE_POSITION_CONTROL_WIN; // Tell the program
                                                                      // to move the arm
     pSharedMemory->CommandFlag = NO_ACT;   // Set the command flag so the system knows
                                            // the motion has ended
  }
  break;
```

1.5 Other Useful Code

1.5.1 Reading & Setting Text

- (button,textbox,etc.)
- To read text off a GUI element:

  ```
  CString strTemp;                                    // C string used to store
                                                      // read strings
  GetDlgItem(IDC_RAISE_ARMS)->GetWindowText(strTemp);// Read the display text
                                                      // written on the element
                                                      // (in this case a button)
                                                      // IDC_RAISE_ARMS
  ```

- To set text on a GUI element:

  ```
  GetDlgItem(IDC_RAISE_ARMS)->SetWindowText("New Display Text");
  ```

1.5.2 Optional: Sequence Variable

If you need to track location in a sequence, add an unisgned char at the top of core.cpp. (e.g. unsigned char arm_motion_sequence;) It will be considered 0x00 if not set, but best practice is to run an initialization command that sets it to 0x00, then run your commands as usual. Use the value stored in the variable to advance through code sections , as shown below:

```
case RAISE_ARMS:
  /* case RAISE_ARMS:
  **
  ** Have the robot raise both arms from the shoulder using absolute joint references.
  ** Then lower the arms back to the sides. Each half of this motion will take 2
  ** seconds to complete.
  **
  */

  // Make sure the joints are not already moving before running the initial sequence
  if((arm_motion_sequence == 0x00) && (Joint[RSP].MoveFlag == false) &&
  (Joint[LSP].MoveFlag == false))
  {
    SetMoveJointAngle(RSP,-40.0f,2000.0f,0x01); // Raise the right arm so the shoulder
                                                // is at -40 degrees
    SetMoveJointAngle(LSP,-40.0f,2000.0f,0x01); // Raise the left arm so the shoulder
                                                // is at -40 degrees

    arm_motion_sequence = 0x01; // Advance to the next part of the sequence

    pSharedMemory->MotorControlMode = CTRLMODE_POSITION_CONTROL_WIN;     // Tell the
                                                                         // program to
                                                                         // move the arm
  }

  // Wait for the joints to finish moving, then run the second step in the sequence
  else if((arm_motion_sequence == 0x01) && (Joint[RSP].MoveFlag == false) &&
  (Joint[LSP].MoveFlag == false))
  {
      SetMoveJointAngle(RSP,0.0f,2000.0f,0x01); // Lower the right arm so the shoulder
                                                //  is at 0 degrees
      SetMoveJointAngle(LSP,0.0f,2000.0f,0x01); // Lower the left arm so the shoulder
                                                // is at 0 degrees

      arm_motion_sequence = 0x00; // Reset to the beginning of the sequence

      pSharedMemory->MotorControlMode = CTRLMODE_POSITION_CONTROL_WIN;   // Tell the
                                                                         // program to
                                                                         // move the arm
      pSharedMemory->CommandFlag = NO_ACT;                // Set the command flag
                                                          // so the system knows
                                                          // the motion has ended
  }

  break;
```