Intro IHM suite Intro Mikro C

Objectif: Piloter le même type d’action que celles obtenues avec MikroC à partir de l’interface C# sous Visual Studio.

# Gestion d’entrées supplémentaires sur la carte Velleman

Nous allons utiliser les switchs SK5 et SK6, qui apparaitront comme des visualisations supplémentaires sur l’IHM.

1. En ouvrant le document introduction-au-langage-mikroc en dernière page le schéma structurelle de la carte, donner l’adresse complète du PORT auxquelles sont reliées les Pattes de SK5 et SK6, donner l’ etat de ces PORTS si le cavalier est installé ou pas.

La ligne du programme qui lit les entrées est la suivantes :

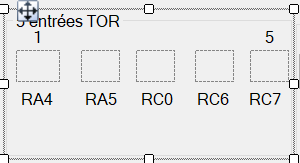
writebuff[2]= PORTC.RC7\*16+PORTC.RC6\*8+PORTC.RC0\*4 +PORTA.RA5\*2+PORTA.RA4;

1. Modifier la pour intégrer SK5 et SK6

Objectif :

Si SK5 est «  en l’air », OUT8 est fausse, sinon elle est vrai. Il faudra créer les emplacements pour 2 visualisations supplémentaires dans l’IHM et modifier le code dans MikroC et l’ihm pour cette modification.

* Penser à faire une copie du répertoire origininal ci-dessous.
* Ouvrir IHM\_usb\_2\USB Generic HID reference application.sln
* Modifier l’interface le groupe suivant Entrée TOR pour intégrer les switchs SK5 et SK6. Ils intégreront les positions respectives 6 et 7 avec les noms pictureBox6 et pictureBox7.



La mise à jour de ce groupes Entrées TOR ( Tout Ou Rien ) est réalisé par le timer deviceStatusPollTimer et son évennement Tick

1. Expliquer le principe de fonctionnement de cet évennement et la signification de sa propriété Interval.

Double-cliquer sur ce timer.

Le mécanisme de mise à jour des PictureBox est présenté ci-dessous :

byte input = Inputs[3];// mise à jour entrées

foreach (Control im in this.groupBox2.Controls)

{

if (im is PictureBox)

{

string s = im.Name;

byte val = byte.Parse(s.Substring(s.Length - 1));

if ((input & (1 << (val - 1))) > 0)

{

im.BackColor = System.Drawing.Color.LimeGreen;

}

else

{

im.BackColor = System.Drawing.Color.Gray;

}

}

}

1. Expliquer le fonctionnement ci-dessus en faisant le lien avec l’expression de writebuff[2] donnée plus haut.

☞ Tester vos modifications ( IHM et MikroC)

1. Coder le pilotage de Out8 par SK5.

# Insertion d’une action supplémentaire

On souhaite ajouter un bouton (Name bouton9) au groupe « 8 sorties TOR » qui alternativement allume ou éteint toutes les sorties. Vous devrez obtenir la figure suivante.



1. Insérer le nouvel élément dans le groupe « 8 sorties TOR » qui sera copié à partir à du bouton 8, mettre à jour la propriété Tag avec le numéro 9.

## Modification logicielle C#

private void Form1\_Load(object sender, EventArgs e)

{

foreach( Control c in this.groupBox1.Controls)

{

c.Click += new EventHandler(Shiva);

}

}

1. Définir la condition d’exécution de la procédure précédente et son rôle.

private void Shiva(object sender, EventArgs e)

{

Control c = (Control)sender;

textBox1.Text = c.Text;

int etat;

if (c.BackColor == Color.Red)

{

etat = 0;

c.BackColor = Color.Silver;

}

else

{

etat = 1;

c.BackColor = Color.Red;

}

theReferenceUsbDevice.Test1(etat, *byte.Parse(\*)*); // \* à modifier en //fonction du Tag du contrôle cliqué qui correspond à son numéro

}

Le rôle de la variable etat est de donner la valeur de la sortie et l’argument suivant en italique dans la fonction theReferenceUsbDevice.Test1 indique le numéro port compris entre 1 et 9. (de 1 à 8 pour les voies du portB et 9 correspond au bouton Set/Reset All)

1. Expliquer comment fonctionne le code précédent et faire la modification sur la ligne indiquée.

Le code suivant est extrait de la procédure timer

private void deviceStatusPollTimer\_Tick(object sender, EventArgs e)

byte port = Inputs[2];// récupération de l’état du port

foreach (Control bt in this.groupBox1.Controls)

{

string s =bt.Name;

byte val = byte.Parse(s.Substring(\*));// A modifier

if (\*) // A modifierSi le contrôle correspond à une voie particulière //du PORTB

{

if ((port & (1 << (val - 1))) > 0)

{

bt.BackColor = Color.Red;

}

else

{

bt.BackColor = Color.Silver;

}

}

}

1. Modifier la procédure précédente.

La procédure suivante est appelée par la procédure Shiva :

public void Test1(int etat, byte port)

{

// Test 1 - Send a single write packet to the USB device

// Declare our output buffer

Byte[] outputBuffer = new Byte[65];

Byte[] inputBuffer = new Byte[65];

bool success;

// Byte 0 must be set to 0

outputBuffer[0] = 0;

if (port < 9) // il s’agit d’une voie du portB

{

outputBuffer[1] = 1;

success = readSingleReportFromDevice(ref inputBuffer);

// Byte 1 must be set to our command

outputBuffer[2] = inputBuffer[2]; // valeur actuelle de PortB

*if (etat == 1)*

*{*

*outputBuffer[2] |= (byte)Math.Pow(2, port - 1);*

*}*

*else*

*{*

*outputBuffer[2] &= (byte)(0xFF - Math.Pow(2, port - 1));*

*}*

}

Else // il s’agit de Set/Reset All

{

outputBuffer[1] = 10;

outputBuffer[2] =(\*) // A modifier

}

success = writeRawReportToDevice(outputBuffer);

}

1. Expliquer les lignes en italique.
2. Modifier la ligne repérée avec \* pour que output[2] qui sera l’image du portB dépendant de la variable etat

outputBuffer[1] = 10 correspond à readbuff[0] dans le logiciel MikroC.

1. Compléter le programme MikroC sur la ligne indiquée ci-dessus

case 10:

// PORTB=fonction( readbuff[1]);

}

1. Tester toutes vos modifications