TinyCLR OS - GPIO

Rédacteur : Philippe Mariano

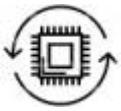
[Mise à jour le 8/4/2020]

- **Sources**
  - Site GHI : General Purpose Input Output (GPIO)

1. Généralités

« *Dans un système à base de **microcontrôleur**, on appelle **entrées-sorties** les échanges d'informations entre le processeur et les périphériques qui lui sont associés. De la sorte, le système peut réagir à des modifications de son environnement, voire le contrôler. Elles sont parfois désignées par l'acronyme **I**/**O**, issu de l'anglais **I**nput/**O**utput ou encore **E**/**S** pour **e**ntrées/**s**orties.* » Wikipédia

Pour éviter de faire référence à des valeurs électriques (tension ou intensité), on définit souvent l'état d'un signal numérique en utilisant la logique booléenne.
- **true** (« 1 » logique) correspondra par exemple à 5V ou 3,3V
- **false** (« 0 » logique) correspondra à 0V.



Un **microcontrôleur** dispose de broches pouvant être contrôlées par un logiciel. Elles peuvent se comporter comme des entrées ou des sorties, d'où le nom "entrée / sortie à usage général", ou **GPIO** (**G**eneral **P**urpose **I**nput **O**utput.

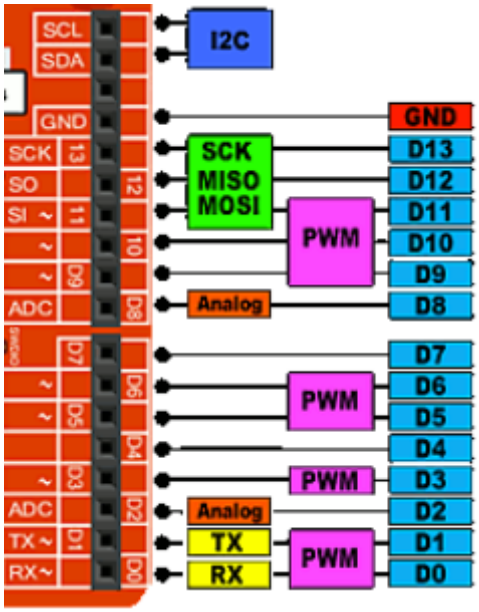
Le nombre de broches d'un microcontrôleur étant limité, il est fréquent d'avoir plusieurs fonctionnalités sur une même broche.

---

2. Les entrées, sorties numériques des cartes FEZ

2.1 GPIO accessibles sur la carte FEZ

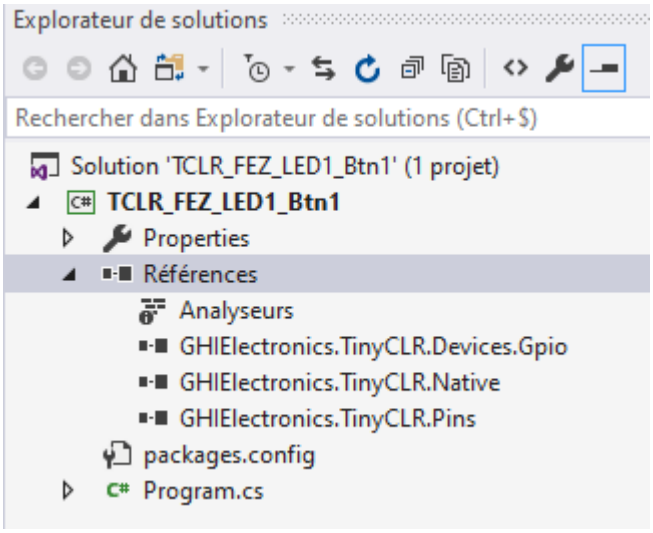
Les connexions **D13** à **D0** peuvent être configurées en **entrée** ou en **sortie numérique**. On voit sur la copie d'écran ci-dessous qu'elles partagent la connectique avec d'autres fonctionnalités (PWM, SPI etc.).



2.2 Bibliothèques à installer

Un programme implanté dans une carte FEZ accède à ses entrées, sorties numériques par l'intermédiaire des **bibliothèques** (assemblys) suivantes :

- **GHIElectronics.TinyCLR.Devices.Gpio** (les entrées, sorties)
- **GHIElectronics.TinyCLR.Pins** (les définitions des broches)



Ces bibliothèques sont installées dans le projet sous la forme de paquets **NuGet** comme dans l'exemple ci-contre.

L'**installation** de ces bibliothèques est détaillée dans :

- ***Premiers programmes en C# "étape par étape" avec une carte BrainPad 2***
  - **Etape 2 : Installer des bibliothèques dans le projet**

2.3 Espaces de noms

L'accès aux entrées, sorties s'écrira plus simplement en entrant les espaces de noms suivants dans le programme.

*.cs

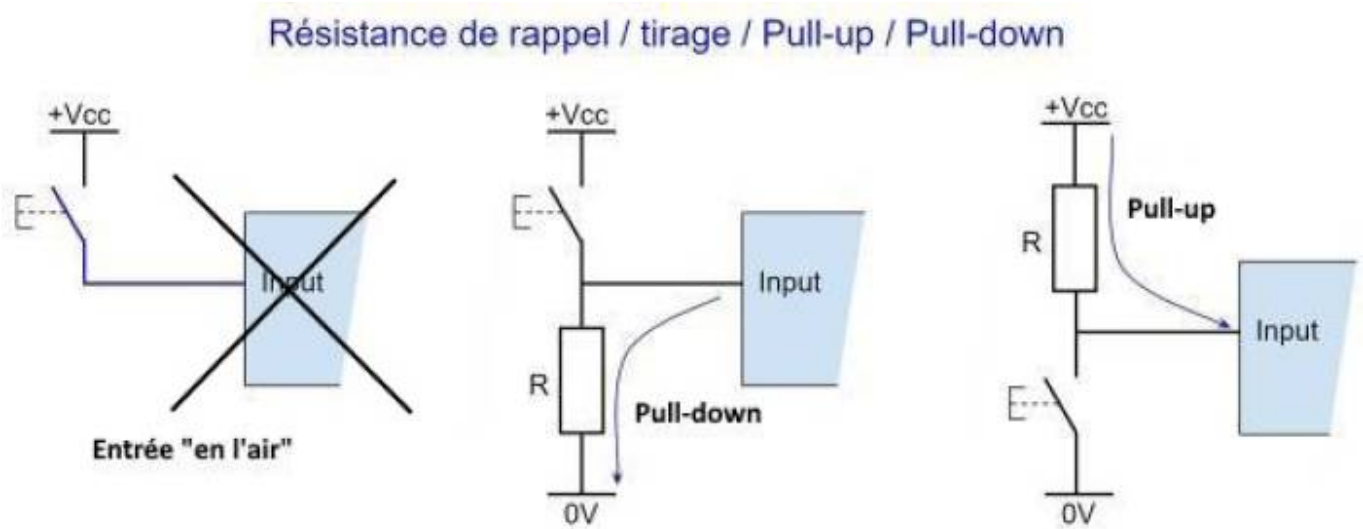
```
using GHIElectronics.TinyCLR.Pins;
using GHIElectronics.TinyCLR.Devices.Gpio;
```

3. Les entrées numériques

Les entrées numériques permettent de détecter un état logique « 0 » ou « 1 ».

3.1 Précautions d'utilisation

> ⚠ Les entrées numériques sont **fragiles**. Elles ne supportent ni les **décharges électrostatiques** ni les **surtensions**. Il ne faut ni les toucher ni leur appliquer une tension supérieure à 5V ou inférieure à 0V.
> Une entrée numérique utilisée dans un programme **ne doit pas être laissée “en l’air”** (non connectée) car elle prendra alors un état logique aléatoirement et le comportement du programme deviendra imprévisible.



Remarque : le processeur de la carte FEZ dispose de **résistances de rappel internes** pouvant être connectées par le logiciel.

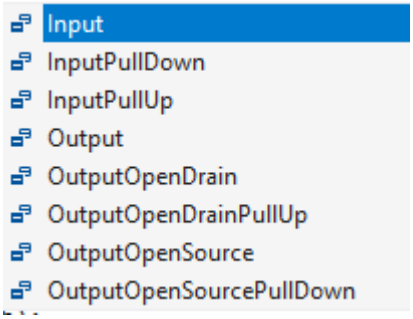
3.2 Configuration d'une broche en entrée

L’accès à une entrée numérique “physique” se fait à l'aide d’un objet logiciel **GPIOpin**. Pour cela, il faut :

1. Créer un objet de la classe **GpioController** et lui affecter le contrôleur GPIO de la carte ciblée

avec la méthode **GetDefault**.
2. Créer un objet de la classe **GpioPin** et lui affecter la broche à contrôler; ouvrir cette broche avec la méthode **OpenPin**.
3. Configurer la broche en entrée ou en sortie (avec résistance de pull-up, etc.) en appliquant la méthode **SetDriveMode** à l'objet GpioPin.

Les différentes configurations possibles pour une broche sont énumérées dans ***GpioPinDriveMode***.



Exemple : configuration de la broche connectée au bouton-poussoir BTN1 de la carte FEZ.

*.css

```
// 1. Création de l'objet gpio (classe GpioController) auquel on
affecte le
// contrôleur GPIO de la carte ciblée (à déclarer une seule fois).
var gpio = GpioController.GetDefault();
// 2. Création de l'objet btn1 (classe GpioPin) auquel on affecte
// la broche associée au bouton-poussoir BTN1,
// broche ouverte par OpenPin,
var btn1 = gpio.OpenPin(FEZ.GpioPin.Btn1);
// 3. Configuration de la broche reliée à BTN1 en entrée avec
// une résistance de rappel
btn1.SetDriveMode(GpioPinDriveMode.InputPullUp);
```

3.3 Lecture d'une broche

La lecture d'une entrée numérique se fait avec une méthode **Read**() de la classe GpioPin.

Exemple : Le bouton-poussoir BTN1 est utilisé pour commander la led LED1 de la carte FEZ.

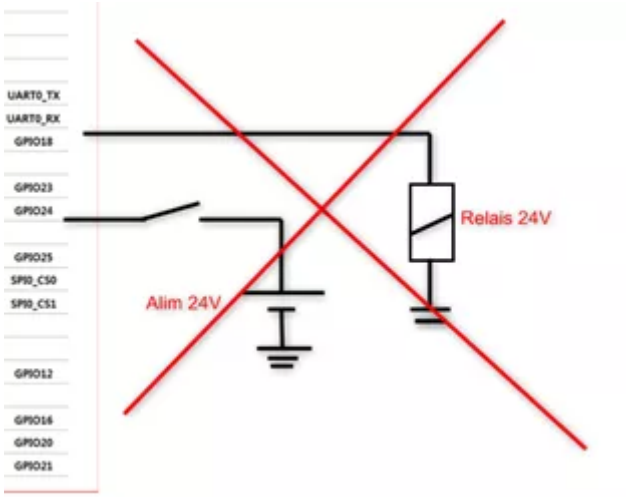
*.css

```
if (btn1.Read()== GpioPinValue.High)
{
  LED1.Write(GpioPinValue.Low);
}
else
{
  LED1.Write(GpioPinValue.High);
}
```

4. Les sorties numériques

4.1 Précautions d'utilisation

⚠ Une sortie numériqque est fragile. Ne **JAMAIS** la relier à un générateur
Une sortie numérique délivre **très peu de puissance** (quelques centaines de mW). Il n'est donc pas possible de la relier directement à un actionneur (moteur). Il est nécessaire de placer une interface de puissance (hacheur, relais) entre elle et l'actionneur à commander.

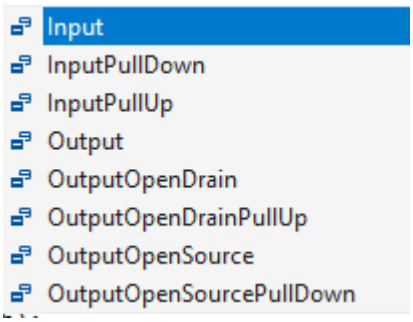


4.2 Configuration d'une broche en sortie

L'accès à une sortie numérique "physique" se fait à l'aide d'un objet logiciel **GPIOpin**. Pour cela, il faut :

1. Créer un objet de la classe **GpioController** et lui affecter le contrôleur GPIO de la carte ciblée avec la méthode **GetDefault**.
2. Créer un objet de la classe **GpioPin** et lui affecter la broche à contrôler; ouvrir cette broche avec la méthode **OpenPin**.
3. Configurer la broche sortie en appliquant la méthode **SetDriveMode** à l'objet GpioPin.

Les différentes configurations possibles pour une broche sont énumérées dans ***GpioPinDriveMode***.



Exemple : configuration de la broche connectée à la LED1 de la carte FEZ.

*.css

```
// 1. Création de l'objet gpio (classe GpioController) auquel on
affecte le
// contrôleur GPIO de la carte ciblée (à déclarer une seule fois).
var gpio = GpioController.GetDefault();
// 2. Création de l'objet led1 (classe GpioPin) auquel on affecte
// la broche associée à LED1.
// broche ouverte par OpenPin,
var btn1 = gpio.OpenPin(FEZ.GpioPin.LED1);
// 3. Configuration de la broche reliée à LED1 en sortie.
btn1.SetDriveMode(GpioPinDriveMode.OutPut);
```

4.3 Ecriture sur une broche

Une sortie numérique peut prendre l'état **true** ou **false**. Cet état est contrôlé avec la méthode Write.

Exemple : Le bouton-poussoir BTN1 est utilisé pour commander la led LED1 de la carte FEZ.

*.css

```
if (btn1.Read()== GpioPinValue.High)
{
  LED1.Write(GpioPinValue.Low);
}
else
{
  LED1.Write(GpioPinValue.High);
}
```

5. BTN1 contrôle Led1 !

*.cs

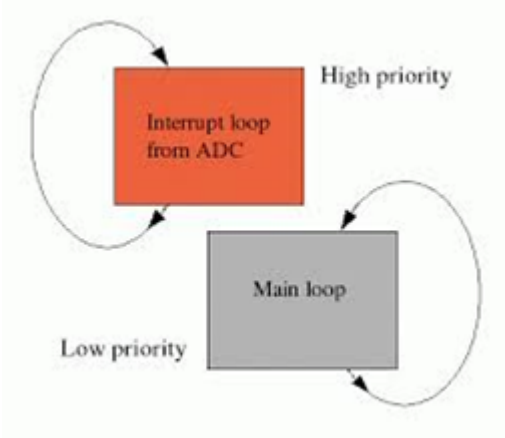
```
using GHIElectronics.TinyCLR.Devices.Gpio;
using GHIElectronics.TinyCLR.Pins;
using System.Threading;

namespace TCLR_FEZ_LED1_Btn1
{
    class Program
    {
        static void Main()
```

```
        {
            var gpio = GpioController.GetDefault();
            var LED1 = gpio.OpenPin(FEZ.GpioPin.Led1);
            LED1.SetDriveMode(GpioPinDriveMode.Output);
            var btn1 = gpio.OpenPin(FEZ.GpioPin.Btn1);
            btn1.SetDriveMode(GpioPinDriveMode.InputPullUp);

            while (true)
            {
                if (btn1.Read() == GpioPinValue.High)
                {
                    LED1.Write(GpioPinValue.Low);
                }
                else
                {
                    LED1.Write(GpioPinValue.High);
                }
                Thread.Sleep(10);
            }
        }
    }
}
```

Le projet **TCLR_FEZ_LED1_Btn1** pour **Visual Studio Community 2017** est téléchargeable ici



6. Les interruptions

Une **interruption** est un arrêt temporaire de l'exécution normale d'un programme par le processeur afin d'exécuter un autre programme (appelé service d'interruption).

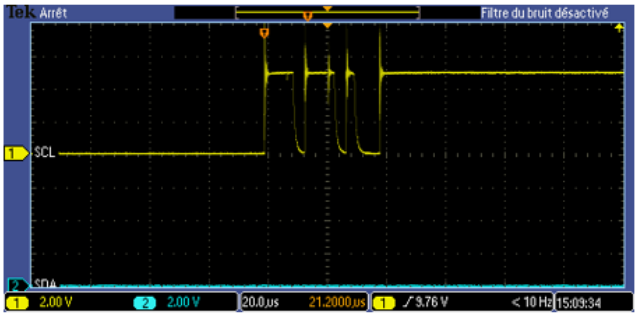
L'interruption est provoquée par une cause externe (action sur un bouton-poussoir, mesure réalisée par un capteur, horloge temps réel, etc.).

On utilise les interruptions afin de permettre des **communications non bloquantes** avec des périphériques externes.

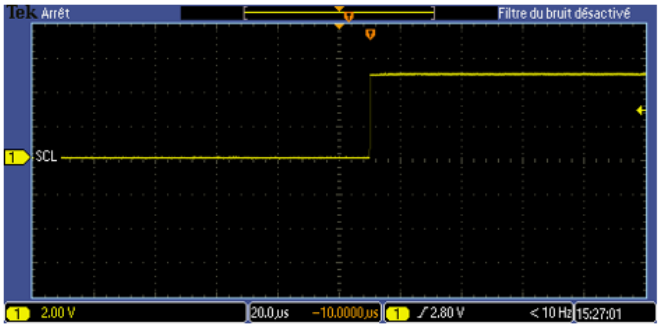
Une interruption tient compte de l'état logique présent sur une broche. Couramment, on la déclenchera sur **le front montant, le front descendant, ou chacun des fronts** d'un signal logique.

6.1 Précautions d'utilisation

Une interruption sera reconnue si le signal présente des fronts "propres". Il faudra donc s'assurer de la qualité du signal. Les figures ci-dessous représentent un signal transmis à la fermeture du contact d'un anémomètre. Le signal de gauche n'est pas utilisable, car, à cause du rebondissement du contact, il contient quatre fronts montants au lieu d'un seul comme dans le cas du signal de droite.



Rebondissement du contact => Signal inutilisable
(nécessite un filtrage)



Front montant « propre » =>
Signal utilisable

6.2 Configuration d'une entrée comme source d'interruption

La configuration en entrée de la broche destinée à recevoir un évènement est identique à celle du paragraphe 4.

6.3 Evènement et gestionnaire d'évènement

Un évènement est attaché à un gestionnaire (service d'interruption) comme dans l'exemple suivant où le Btn1 commande la LED1 de la carte FEZ lorsqu'un front montant apparaît sur la broche.

*.cs

```
using GHIElectronics.TinyCLR.Devices.Gpio;
using GHIElectronics.TinyCLR.Pins;
using System.Threading;


namespace TCLR_FEZ_LED1_Btn1_INT
{
    class Program
```

```
    {
        private static GpioPin LED1;

        private static void Main()
        {
            var gpio = GpioController.GetDefault();
            LED1 = gpio.OpenPin(FEZ.GpioPin.Led1);
            LED1.SetDriveMode(GpioPinDriveMode.Output);
            var btn1 = gpio.OpenPin(FEZ.GpioPin.Btn1);
            btn1.SetDriveMode(GpioPinDriveMode.InputPullUp);
            // Prise en compte de l'interruption
            btn1.ValueChanged += Btn1_ValueChanged;

            Thread.Sleep(-1); // Mise en Veille pour réduire la
consommation
            // d'énergie et permettre au système d'effectuer d'autres
tâches.

        }

        private static void Btn1_ValueChanged(GpioPin sender,
GpioPinValueChangedEventArgs e)
        {
            if (e.Edge == GpioPinEdge.RisingEdge)
                LED1.Write(GpioPinValue.Low);
            else
                LED1.Write(GpioPinValue.High);
        }
    }
}
```

Le projet **TCLR_FEZ_LED1_Btn1_INT** pour **Visual Studio Community 2017** est téléchargeable ici

7. Tous les exemples

- Exemples codés en C# pour la carte **Panda 3 (G80)** ici