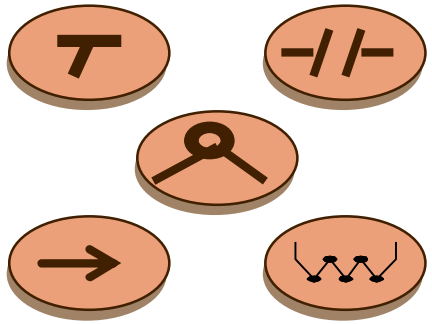
Cours 7 : Programmation d ’une chaîne d’acquisition

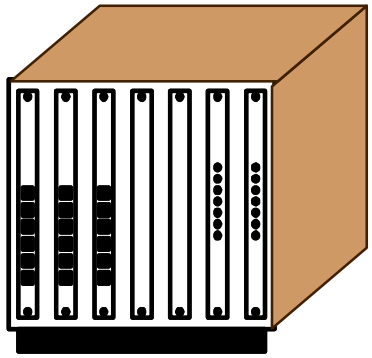
- Concepts
- Programmation
  - Cible Pentium : Langages de haut niveau
    - Langage graphique G sous LabView + librairies de VI ;
    - Langage C + librairies de fonctions C ;
    - Python …
  - Cible Micro contrôleur : Langages intermédiaires
    - Langage C + librairies de fonctions C ;
    - Langage assembleur.
  - Cible circuit programmable : Hardware Description Language
    - ABEL
- Logiciel intégré LabView
  - Instruments Virtuels
  - Contrôle de Cartes d ’acquisition multi-fonctions
  - Contrôle d ’instruments réels [traité en S3]

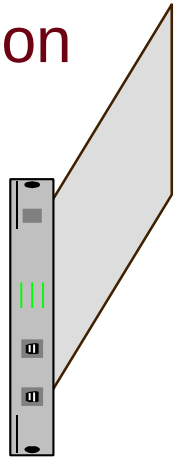
Instrumentation et capteurs

1. Imaginer et réaliser un capteur.
2. Tester un capteur instrumenté.
3. Instrumenter un capteur.

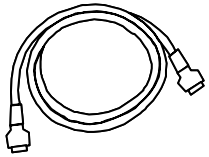
Concept 1 : Chaîne d 'instrumentation



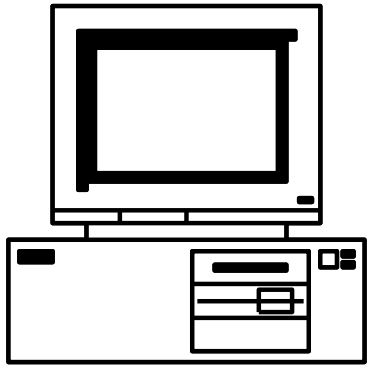




Capteurs Conditionneurs Cartes

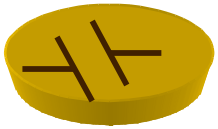


Bus

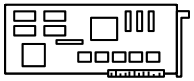


Logiciels de traitement

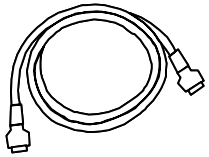
Concept 2 : Capteur + micro Contrôleur



+

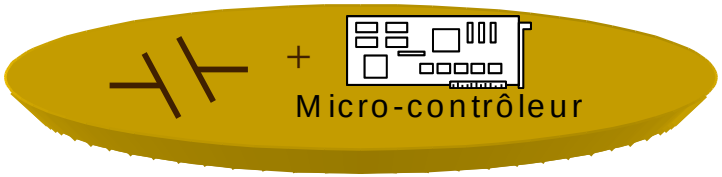


Micro-contrôleur



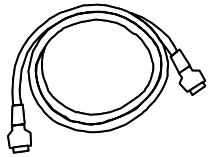
Bus

Concept 3 : System On Chip



+

Micro-contrôleur



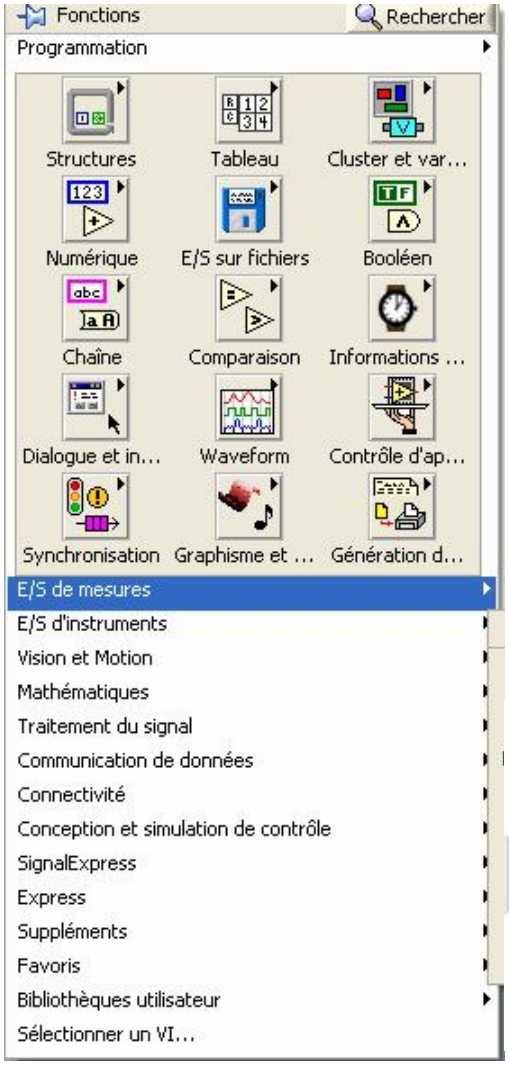
Bus

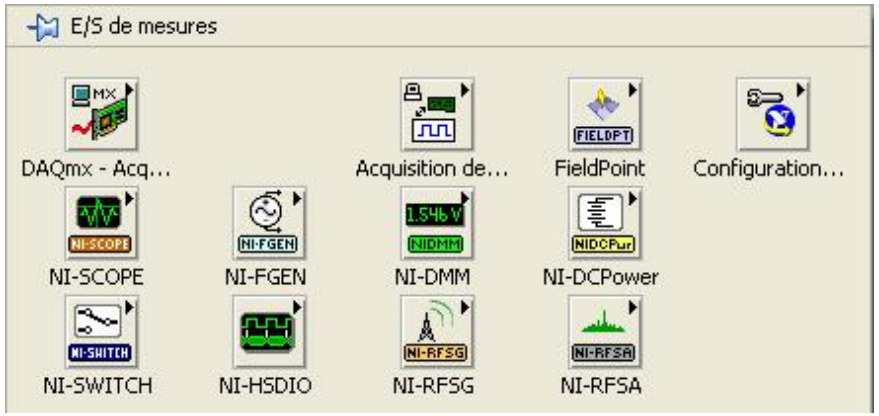
Programmation

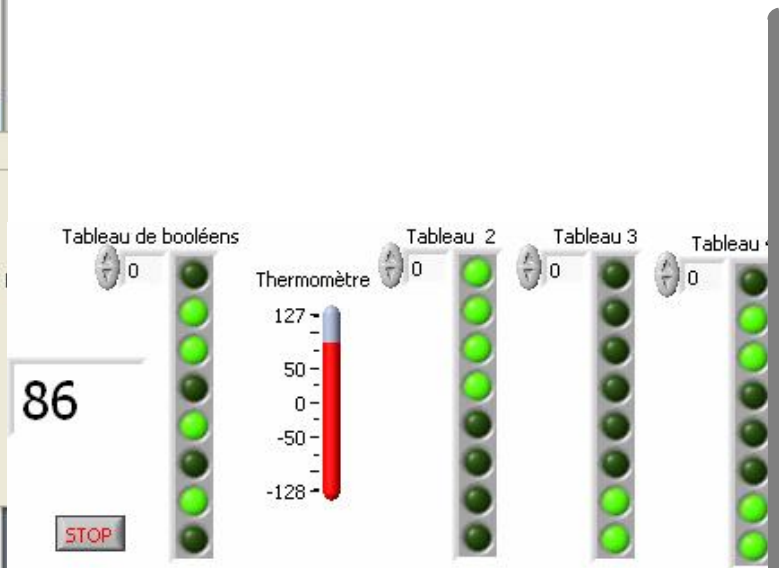
4 Cible Pentium : Langages de haut niveau

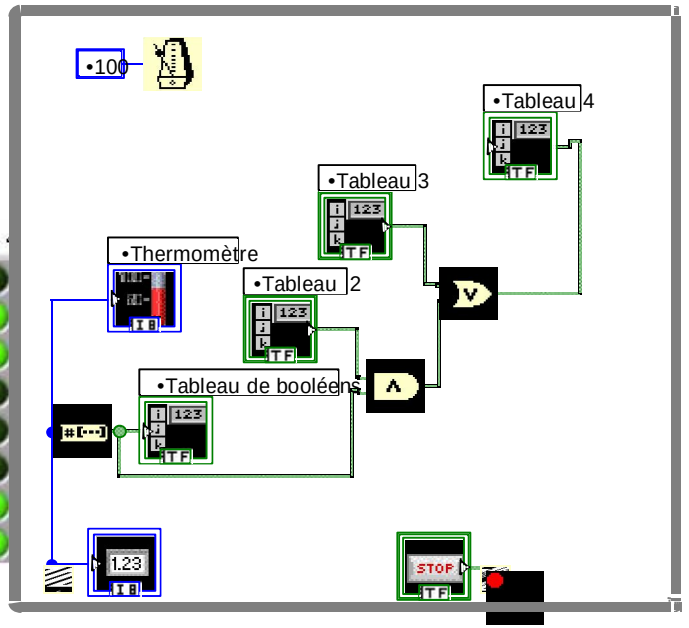
- Langage graphique G sous LabView + librairies de VI ;
- Langage C + librairies de fonctions C ;
- Python

Langage graphique G sous LabView









Programmation

- Cible Pentium : Langages de haut niveau
  - Langage graphique G sous LabView + librairies de VI ;
  - Langage C + librairies de fonctions C ;
  - Python

Développement des programmes sous Labview

Lancer l ’application Labview

1. Créer un instrument virtuel VI
- Editer la face avant
  - Les commandes
  - Les indicateurs
  - Modifier le graphisme
- Editer le diagramme
  - Variables
  - Opérateurs
  - Structures
  - Nœuds

2. Exécuter le programme V.I.
- Entrer des données test
- Déboguer en mode Hilighting
- Accéder à des cartes
  - de communication
  - d ’acquisition
- Tracer des courbes
- Faire de la métrologie

Programmation

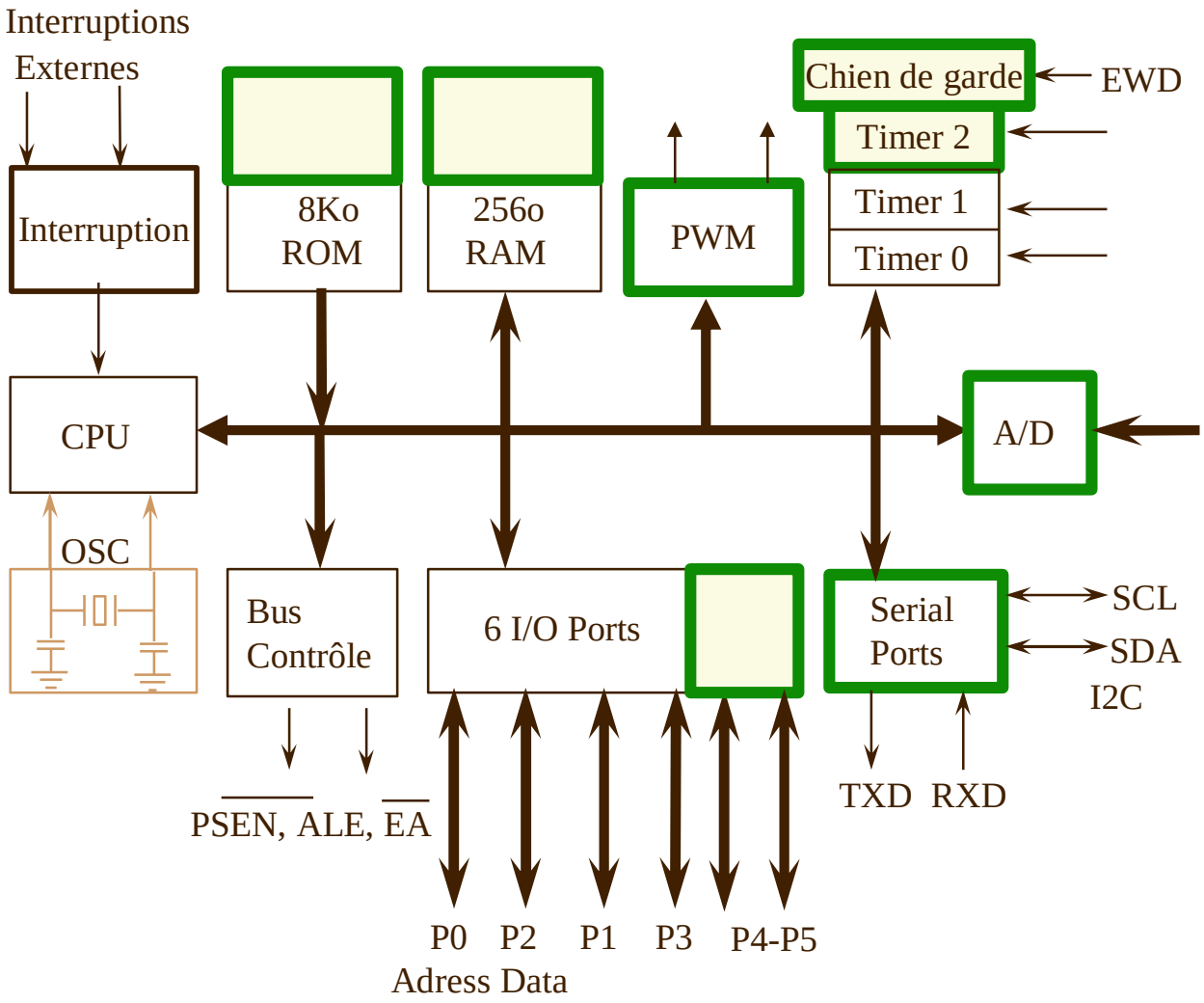
- Chaînes de développement pour l'acquisition et le contrôle de données

Editeur	Produit	Programmation
Data Transalation	DT-VEE	Graphique Appel de routine C
Digimétrie	Digiview	Graphique
HEM	Snap-Master	Graphique
Hewlett-Packard	HP6VEE	Graphique Appel de routine C
Keithley	Testpoint Viewdac	Graphique Par menu
Labtech	Labtech notebook Labtech control	Graphique Graphique
National Instrument	Labview Labwindows/CVI	Graphique Bibliothèques de fonction C
Sysma	ATS	Graphique

Programmation

4 Cible Micro contrôleur : Langages intermédiaires
- Langage C + librairies de fonctions C ;
- Langage assembleur.

Micro Contrôleur 80C552



Programmation

- Cible Micro contrôleur : Langages intermédiaires
  - Langage C + librairies de fonctions C ;
  - Langage assembleur.

Langage C

```
#include <reg552.h>

sbit LCD_RS = 0x95;
sbit LCD_WR = 0x96;
sbit LCD_ENABLE = 0x97;
code char LcdTable[] = {0x30, 0x30, 0x30, 0x38, 0x08, 0x01, 0x0e, 0x00} ;

char LcdMsg[] = "IUT ORSAY - MP";

void InitLCD ();
void PrintLCD();
void delay (int t);

main ( ) {
  InitLCD ( );
  PrintLCD ( );

  while (1) {« Corps du  programme »}
}
```

```
void InitLCd () {
  int i;

    LCD_WR = 0;
    LCD_RS = 0;
  i = 0;
  while (LcdTable[i]!=0x00) {
    delay (100);
    P4 = LcdTable[i];
    LCD_ENABLE = 1;
    LCD_ENABLE = 0;
    i++;
  }
}
```

```
void PrintLCD() {
  int i;
    LCD_WR = 0;

    /* DD-RAM = 0 */
    LCD_RS = 0;
    delay (1);
    P4 = 0x80 + 0x00;
    LCD_ENABLE = 1;
    LCD_ENABLE = 0;

  i=0;
  while (LcdMsg[i] != 0x00) {
    if (i==8) {
        LCD_RS = 0;
        delay (1);
        P4 = 0x80 + 0x40;
        LCD_ENABLE = 1;
        LCD_ENABLE = 0;
    }
    LCD_RS = 1;
    delay (1);
    P4 = LcdMsg[i];
    LCD_ENABLE = 1;
    LCD_ENABLE = 0;
    i++;
  }
}
```

```
void delay (int t) {
   int i, k;
   for (i=0; i<t; i++)
      for (k=0;k<10;k++);
}
```

Programmation

- Cible Micro contrôleur : Langages intermédiaires
  - Langage C + librairies de fonctions C ;
  - Langage assembleur.

Développement en langage C

1. Créer un programme

- Créer un projet
  - Sources C
  - Machine cible
- Editer le source
  - Le programme
  - Les entrées -sorties
  - L'accès aux librairies
- Compiler le projet
  - Vérification de la syntaxe
  - Réservation de l 'espace mémoire
  - Edition des liens

2. Exécuter le programme

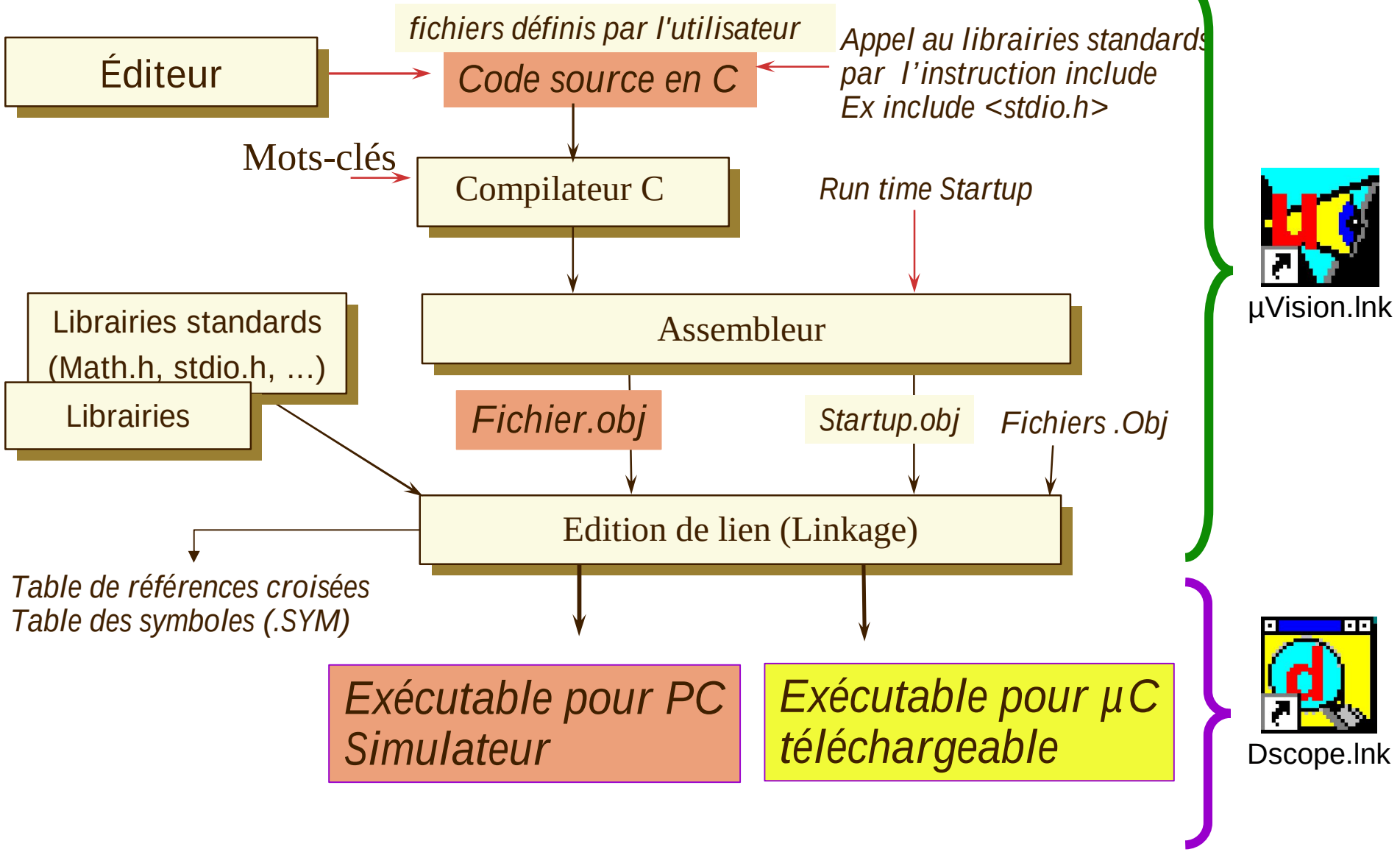
- Télécharger l'exécutable
- Lancer l'exécutable
- Entrer des données
- Analyser les sorties
- Déboguer
- Accéder à des cartes
  - de communication
  - d 'acquisition

grâce à des librairies

Programmation

4 Cible Micro contrôleur : Langages intermédiaires
- Langage C + librairies de fonctions C ;
- Langage assembleur.

µContrôleur : Générer l'exécutable à partir du C



Programmation

- Cible circuit programmable :
  Hardware Description Language
  - ABEL

- ABEL est un langage de programmation de circuits de logique programmable (PLD).
  - PAL,GAL FPGA et CPLD
- Langage HDL (Hardware Description Language)
- Applications de petites et moyennes importances à implémenter sur des circuits

Advanced Boolean Equation Language

Description du langage

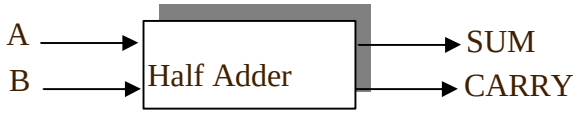
```
module mod_name [ ;]

[title string] [ ;]
[deviceID device deviceType ;]
[declaration] [ ;]
pin declarations ;
other declarations ;
equations [ ;]
equations          [" commentaires]
truth_table (entree ->sortie)[ ;]
table de vérité ;
state_diagram etat[ ;]
diagramme d'état ;
[Test_Vectors (entree->sortie)] [ ;]
vecteurs de tests ;

end mod_name [ ;]
```

Programmation

- Cible circuit programmable :
  Hardware Description Language
  - ABEL

Exemple ½ additionneur

Half Adder




```
module Addition;
title 'addition 2 bits'     // optionnel, information complémentaire

Addition device 'P16H8';    // optionnel, permet de spécifier le type de matériel
cible : PAL16P8

" input pins
A, B pin 3, 5;              // la spécification du numéro de broches est optionnelle
" output pins
SUM, Carry_out pin 15, 18 istype 'com';  // le compilateur considère les
                                         //broches de sortie par le mot-clé istype

equations
SUM = (A & !B) # (!A & B) ;          // génération des sorties en fonction des entrées
Carry_out = A & B;

end      $GGLWRQ
```

Programmation

- Cible circuit programmable :
  Hardware Description Language
  - ABEL

Développement d ’un circuit de logique programmable

1. Créer le programme

- Créer un projet
  - Un Code Source en entrée
    - un schéma logique
    - des équations logiques
    - des tables de vérité
    - des diagrammes d’état
  - une Cible
    - un circuit programmable du commerce
- Compiler le source
  - Syntaxe
  - Vérification que le code tient dans le circuit

2. Exécuter le programme

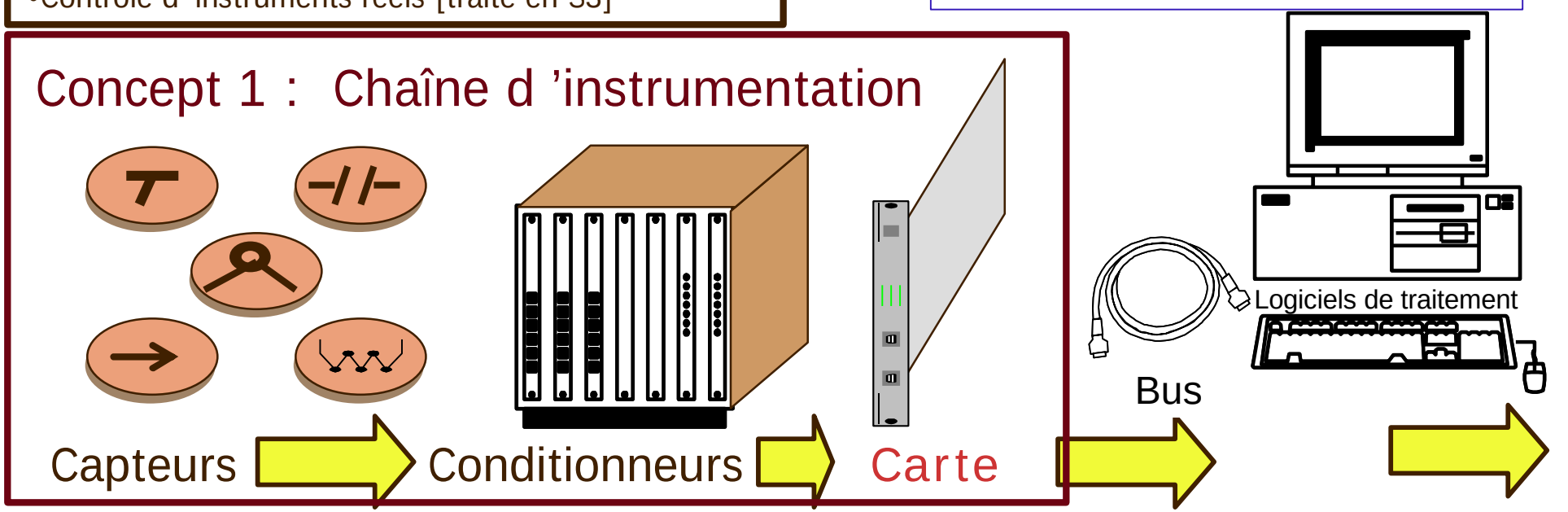
- Télécharger un fichier programmant le circuit
- Lancer le fonctionnement
- Tester le fonctionnement

Logiciel intégré LabView

- Contrôle de Cartes d 'acquisition multi-fonctions
- Instruments Virtuels
- Contrôle d 'instruments réels [traité en S3]

Carte d acquisition multi-fonctions

Concept 1 : Chaîne d 'instrumentation

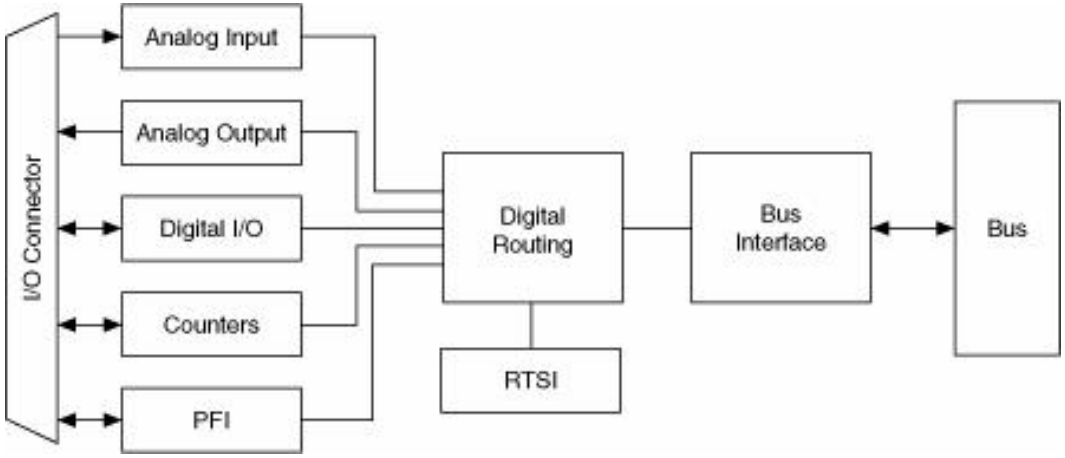


Contrôle de la carte d'acquisition multifonctions en 3 temps

1. Comprendre le fonctionnement de la carte
2. Programmer une tâche sous MAX
3. Acquérir les données sous labview en précisant la tâche et le type de donnée

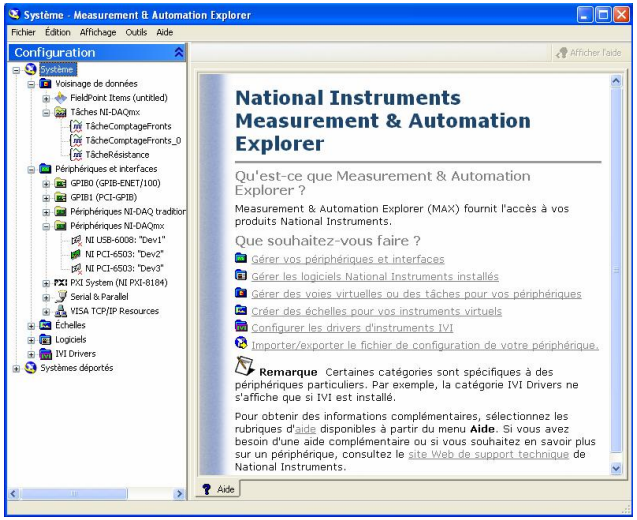
Fonctionnement de la carte NI 6040E

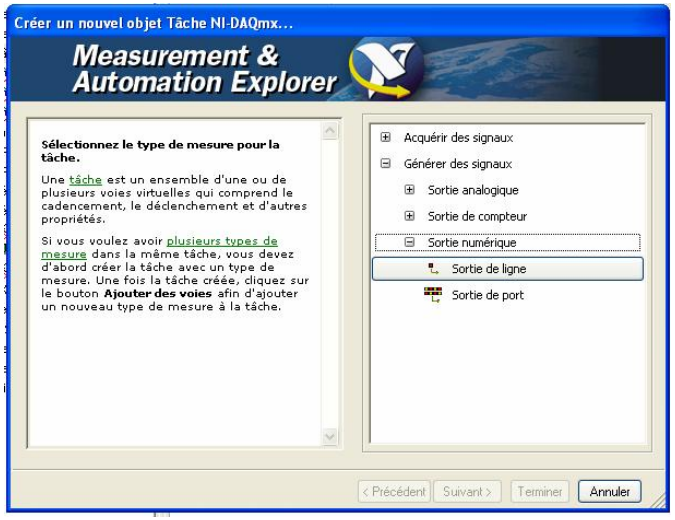
- Formerly known as PCI-MIO-16E-4
  - Included NI-DAQmx driver software
  - NIST-traceable calibration certificate
  - Two 12-bit analog outputs;
  - 8 digital I/O lines;
  - two 24-bit counters;
  - Analog triggering
- Analog Input Characteristics
  - Number of channels: 16 single-ended or 8 differ channel)
  - Type of A/D convert.: Successive approximatio
  - Resolution ..... 12 bits, 1 in 4,096
  - Maximum sampling rate
    - Single-channel scanning..... 500 kS/s
    - Multiple-channel scanning ..... 250 kS/s
- Input signal ranges
  - Input coupling .....DC
  - Maximum working voltage

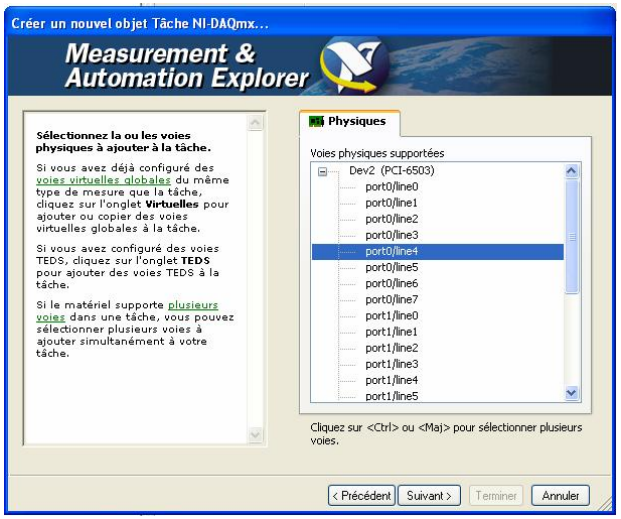


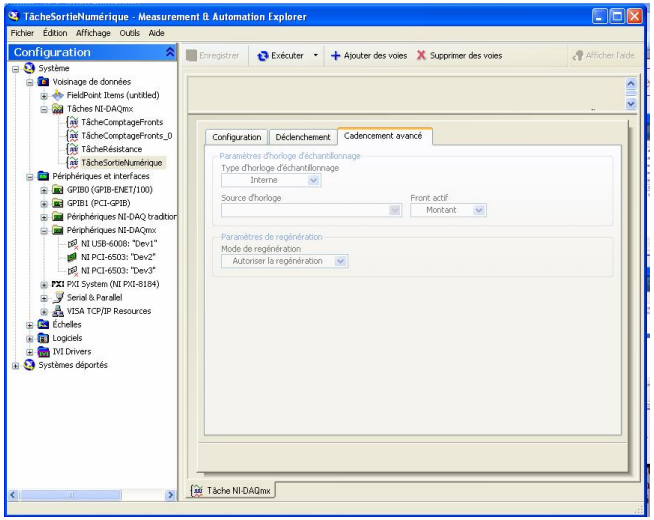
Range (Software-Selectable)	Input Range	
	Bipolar	Unipolar
20 V	±10 V	—
10 V	±5 V	0 to 10 V
5 V	±2.5 V	0 to 5 V
2 V	±1 V	0 to 2 V
1 V	±500 mV	0 to 1 V
500 mV	±250 mV	0 to 500 mV
200 mV	±100 mV	0 to 200 mV
100 mV	±50 mV	0 to 100 mV

Measurement and Automation eXplorer : créer les tâches

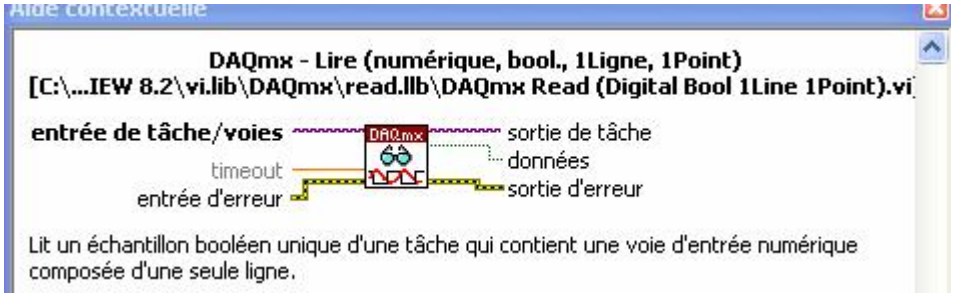








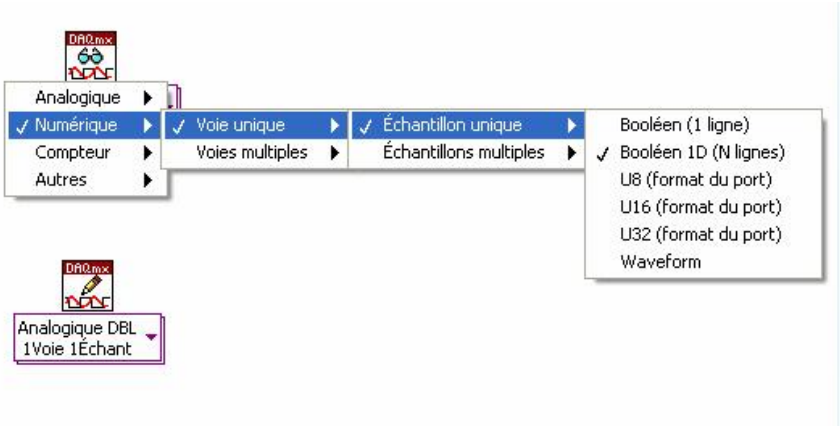
Utilisation sous labview en DAQmx



- entrée de tâche/voies est le nom de la tâche ou une liste de voies virtuelles auxquelles s'applique l'opération. Si vous fournissez une liste des voies virtuelles, NI-DAQmx crée automatiquement une tâche.
- timeout spécifie la durée en secondes pour écrire tous les échantillons. Si la durée allouée est écoulée, le VI renvoie une erreur. La valeur par défaut est 10 secondes. Si timeout à -1, le VI attend indéfiniment. Si timeout à 0, le VI essaie une fois de lire les échantillons demandés et renvoie une erreur s'il n'y arrive pas.
- entrée d'erreur décrit les conditions d'erreur qui ont lieu avant l'exécution de ce VI ou de cette fonction..
- sortie de tâche est une référence à la tâche une fois que ce VI ou cette fonction s'exécute. Si vous avez câblé une voie ou une liste de voies à entrée de tâche/voies, NI-DAQmx crée automatiquement cette tâche.
- données renvoie un échantillon. NI-DAQmx met les données à l'échelle des unités de mesure, en tenant compte, le cas échéant, de la mise à l'échelle personnalisée que vous appliquez aux voies. Utilisez le VI DAQmx - Créer une voie virtuelle ou l'Assistant DAQ pour spécifier ces unités.
- sortie d'erreur contient des informations concernant les erreurs. Si entrée d'erreur indique qu'une erreur s'est produite avant l'exécution de ce VI ou de cette fonction, sortie d'erreur contient les mêmes informations sur l'erreur. Sinon, sortie d'erreur décrit l'état d'erreur produit par ce VI ou cette fonction.

Composant polymorphique

Un composant :plusieurs types de sortie



1. Définir le type de sortie.
2. Sélectionner une tâche en entrée qui précise sur quelle carte et sur quelle voie sont le données et qui soit conforme au type de sortie. Le plus simple est de câbler une constante ou une commande et de sélectionner avec le doigt la tâche appropriée.