Introduction

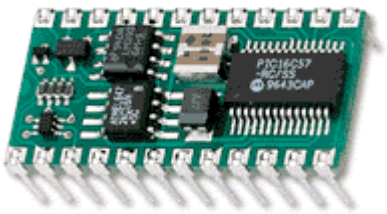
Nous allons voir au cours de cette note technique une utilisation moins courante de 4D puisque nous allons établir un dialogue avec un microcontrôleur afin de faire de l'acquisition de données. En l'occurrence, il s'agira de la lecture d'un capteur de température.
Dans cette optique, 4D se chargera d'interroger le microcontrôleur et de récupérer les données pour les stocker et de les traiter pour en extraire des statistiques ou des graphiques et le cas échéant donner des ordres en retour au microcontrôleur.

Le choix du microcontrôleur

Afin de rester dans l'esprit 4D, le choix s'est porté sur un microcontrôleur dont l'utilisation et la mise en œuvre est d'une simplicité extrême puisqu'il se programme en Basic, qu'il ne nécessite quasiment aucun composant électronique externe et que sa taille est proche d'un timbre poste. Son nom est d'ailleurs le Basic Stamp II.

Voici brièvement quelques unes de ses caractéristiques principales :
- Programmable en Basic
- 16 entrées sorties (configurables par programmation)
- 1 eeprom de 2 Ko
- 1 interface RS232

De plus, un environnement de développement permettant le chargement des programmes ainsi que la visualisation de la mémoire est fourni en standard.
Par ailleurs, et ce n'est pas le moindre de ses avantages, son prix est des plus attractif puisqu'il est à ce jour d'environ 50$



L' objectif de cette note n'étant pas d'expliquer en détail le fonctionnement de ce microcontrôleur, nous vous invitons à vous rendre sur le site du fournisseurs où vous pourrez trouver toutes les informations détaillées concernant ce produit : http://www.parallaxinc.com/

Le capteur de température

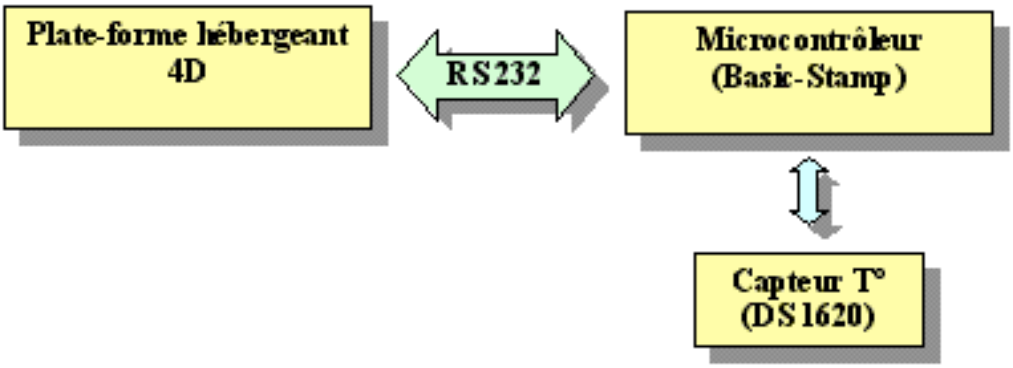
Le capteur de température utilisé ici n'est bien entendu qu'un exemple car il est possible de connecter n'importe quel capteur sur le microcontrôleur mais là encore, le choix a été dicté par la simplicité de mise en

œuvre. Nous utiliserons ici un produit ne nécessitant aucun autre composant externe pour son fonctionnement puisqu'il peut se connecter directement sur le Basic Stamp (BS II). Le composant utilisé est le DS 1620 de chez Dallas Semiconducteur.

L'avantage de ce capteur est qu'il est paramétrable et interrogeable directement sans interface particulière. C'est d'ailleurs presque un microcontrôleur à lui tout seul puisqu'il possède son propre jeu d'instructions permettant de l'utiliser en capteur ou thermostat, d'effectuer des conversions à la volée, etc. Pour plus de détails, là aussi nous vous invitons à visiter le site du constructeur où vous trouverez la documentation technique de ce composant : http://www.dalsemi.com/.

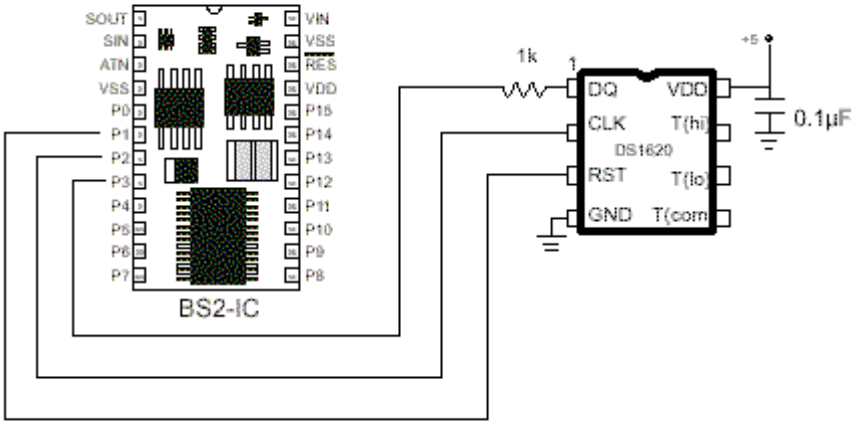
Synoptique de la configuration

Cette section va décrire l'interconnexion des différentes briques de cette solution. Pour mémoire, nous avons trois briques principales : le PC hébergeant la base 4D, le Basic Stamp (notre microcontrôleur) et le capteur de température.



Le schéma de principe électronique

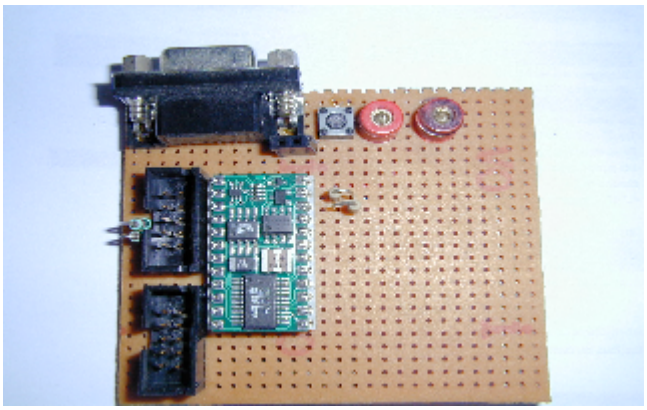
Il se résumera à la connexion du capteur au microcontrôleur. En voici le détail :



Le hardware

La partie matérielle de ce montage sera réduite à sa plus simple expression. Il faudra seulement câbler le microcontrôleur et le capteur sur une platine. Deux possibilités s'offrent à vous : acheter la platine chez

Parallax à l'url citée plus haut ou bien câbler vous même une platine ce qui dans le cas du Basic Stamp n'est qu'une formalité.



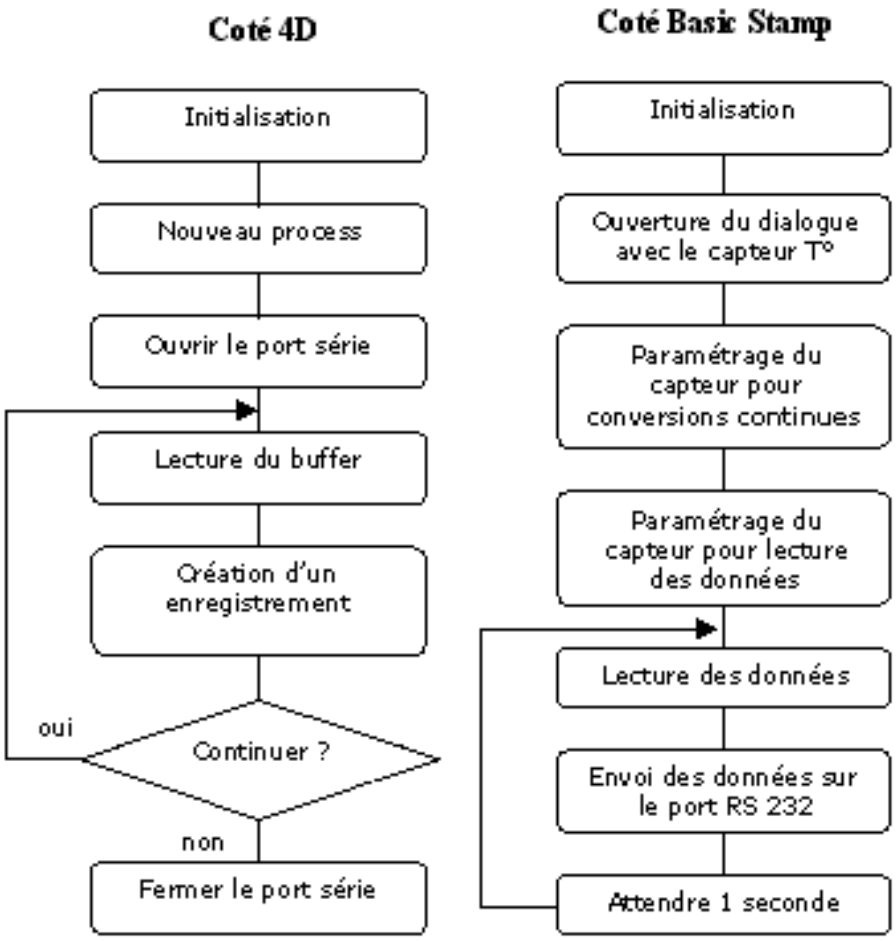
L'organisation logicielle

Il y a deux façons de gérer l'acquisition des données avec un microcontrôleur : récupérer les données en temps réel ou bien stocker les données dans la mémoire du microcontrôleur et faire un relevé périodique (1 fois par mois par exemple). Le choix sera fait en fonction de l'utilisation que l'on souhaite faire de ces données. Cette note décrira les deux aspects puisque notre microcontrôleur est équipé d'une eeprom capable de stocker durablement les données.

Gestion des données en temps réel

Dans ce cas de figure, il s'agit d'effectuer les mesures toutes les secondes et de les envoyer à la base de données pour un traitement immédiat.

Les organigrammes sont les suivants :



Coté 4D, la programmation est des plus standard. Nous commençons par ouvrir un process dans lequel sera exécuté le dialogue avec le microcontrôleur.
Le Basic Stamp communique avec le protocole suivant : 9600 bauds, 8 bits de données, 1 bit de stop et pas de parité. La commande REGLER SERIE permettra d'effectuer le réglage correspondant au niveau de 4D.

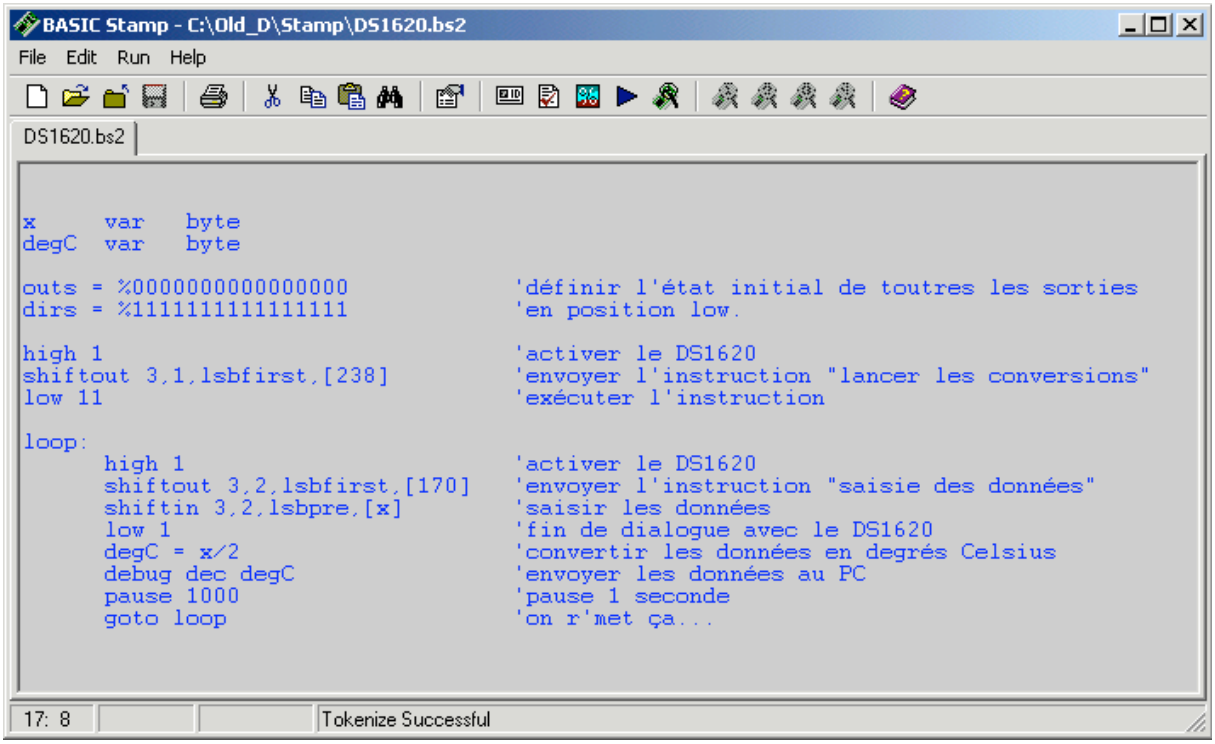
Voici le détail de la méthode 4D :

```
Si (Nombre de parametres=0)
   <>ProcessRecept:=Nouveau process("recept";32000;"Recept";"vide")
Sinon

  C_TEXTE($vtBuffer)
   $vtBuffer:=""

  REGLER SERIE(1;Vitesse 9600 +Bits de données 8 +Bit de stop un +Pas de parité )
   <>IP_Ecouter_Port_Série:=Vrai

  Tant que (<>IP_Ecouter_Port_Série)
    RECEVOIR BUFFER($vtBuffer)

    Si ($vtBuffer#"")
      CREER ENREGISTREMENT([Table 1])
      [Table 1]Temp:=Num($vtBuffer)
      STOCKER ENREGISTREMENT([Table 1])
      $vtBuffer:=""
```

```
    Fin de si
  Fin tant que
  REGLER SERIE(11)
Fin de si
```

Lors de la réception des données, 4D crée un enregistrement contenant la température mesurée. Voilà, il ne faut rien d'autre à 4D pour dialoguer avec un microcontrôleur.

Coté Basic Stamp, le programme n'est guère plus gros :



Détaillons un peu le contenu de la fenêtre :

Les variables 'outs' et 'dirs' sont des mots réservés du BS (comme 'document' ou 'ok' en 4D) et permettent de fixer l'état initial des 16 ports du BS. Dans notre exemple, la variable dir fixe les 16 ports en sortie et la variable outs fixe les 16 ports à l'état bas.
Par la suite, l'instruction High 1 passe le port spécifié (1 dans notre exemple) à l'état haut ce qui a pour effet d'activer la broche 'rst' (reset) du DS1620.
La commande Shiftout permet en une seule passe de configurer certaines broches en port RS232 et d'envoyer la valeur spécifiée. Dans notre cas, le port 3 sera le port de données et le port 1 correspondra au signal d'horloge (clock). Lsbfirst permet de spécifier l'ordre d'envoi des données (byte swapping). [238] est la valeur à envoyer. Cette valeur correspond à un ordre reconnu du DS1620 lui donnant l'ordre d'effectuer des conversions à la volée.

En résumé, le programme fait appel principalement aux commandes Shiftin et Shiftout qui permettent de recevoir et d'envoyer des données sur telle ou telle sortie (de 1 à 16) du microcontrôleur. Les instructions Low et High permettent d'activer ou non telle ou telle sortie du microcontrôleur.
L'instruction Debug permet d'envoyer des données sur le port RS 232 par defaut du Basic Stamp. C'est par cette commande que les données seront renvoyées à 4D.
Les données seront renvoyées en flux continu toutes les secondes. Il importe donc, au niveau de 4D, d'être suffisamment rapide pour traiter ces données sous peine d'en perdre…
Dans le cas présent, il ne devrait pas y avoir de problème puisqu'il n'y a qu'un envoi par seconde.

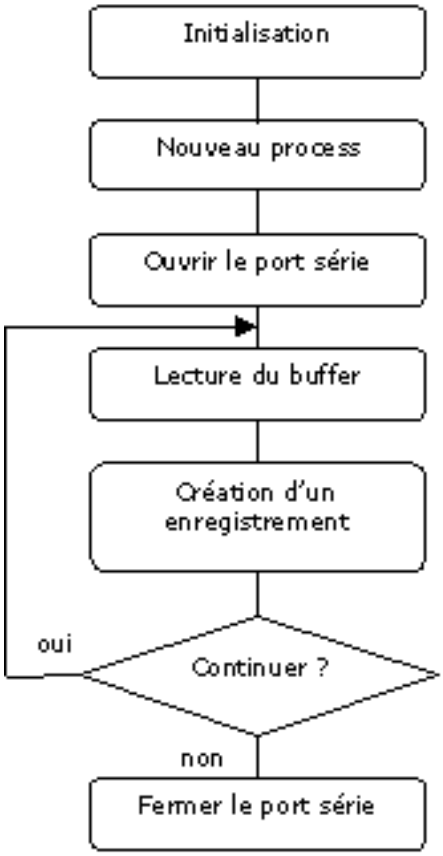
Bien entendu, une telle configuration révèle un certain nombre de faiblesses. On notera par exemple, le fait que 4D crée un enregistrement toutes les secondes. Il sera sans doute plus judicieux de stocker les différents

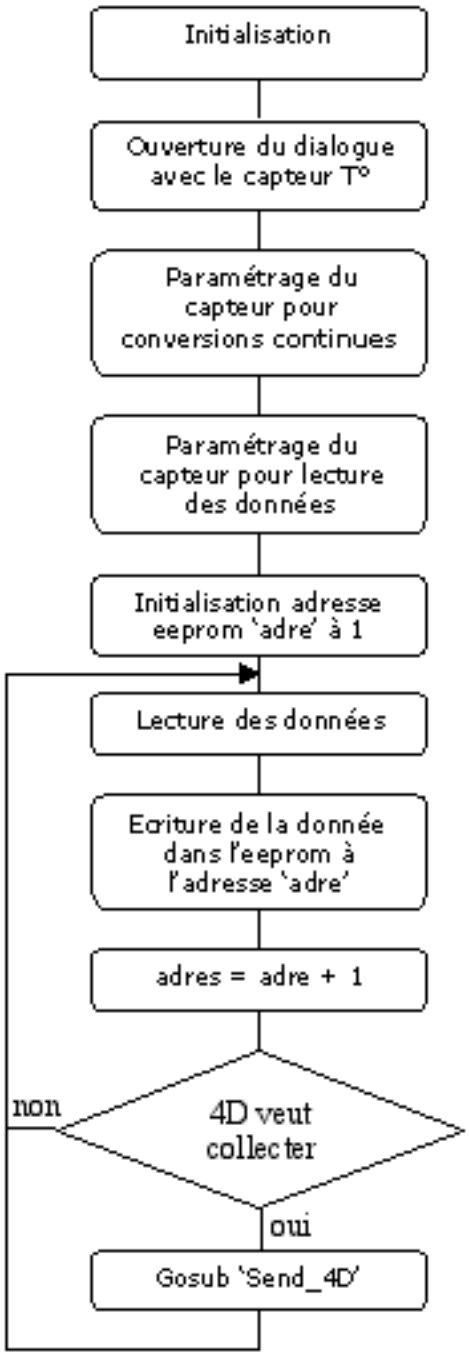
relevés dans un blob dont on limitera la taille et de créer un nouvel enregistrement dès que cette taille est atteinte. Il faudra aussi prévoir de supprimer les enregistrements dont les valeurs ne servent plus.

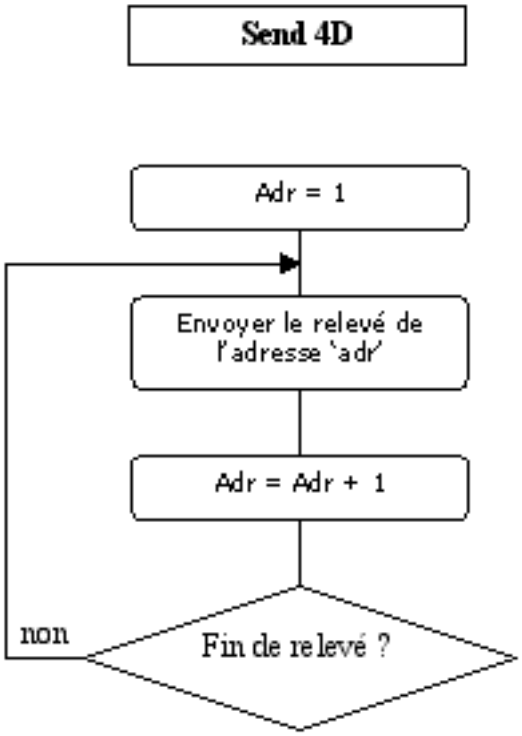
Gestion des données en différé

Dans ce cas, la stratégie est complètement différente. Il s'agira d'enregistrer les données du capteur au niveau du microcontrôleur (dans l'eeprom) et de récupérer ces données par paquets une fois par mois ou par semaine. La fréquence de récupération sera dictée par l'utilisation que l'on souhaite faire de ces données mais aussi par la taille de la mémoire du microcontrôleur (capacité de stockage) ainsi que par la fréquence des mesures.

L'organigramme de fonctionnement ne change pas vraiment coté 4D. La différence se trouve au niveau du microcontrôleur puisque l'écriture des données se fait au niveau de l'eeprom. Il suffira donc à 4D d'interrompre le microcontroleur qui du coup passera dans un sous programme lui envoyant le contenu de la mémoire.


Initialisation
Nouveau process
Ouvrir le port série
Lecture du buffer
Création d'un enregistrement
Continuer ?
oui
non
Fermer le port série


Initialisation
Ouverture du dialogue avec le capteur T°
Paramétrage du capteur pour conversions continues
Paramétrage du capteur pour lecture des données
Initialisation adresse eeprom 'adre' à 1
Lecture des données
Ecriture de la donnée dans l'eeprom à l'adresse 'adre'
adres = adre + 1
4D veut collecter
non
oui
Gosub 'Send_4D'



Conclusion

Nous avons pu voir au cours de cette note qu'il n'était réellement pas difficile d'utiliser 4D pour dialoguer avec un microcontrôleur. En revanche, la complexité peut intervenir au niveau du microcontrôleur lui-même si son protocole de communication est complexe.

Le BasicStamp est un modèle du genre puisqu'il se programme en Basic et qu'il n'est pas nécessaire d'être un mutant pour le mettre en oeuvre :-).