Propeller 2 de Parallax (5)

La fonction de « broche intelligente »

Mathias Claussen (Elektor)

Pour conclure cette série sur le microcontrôleur Propeller 2 de Parallax, nous présentons la fonction « smart pin » (broche intelligente), qui permet des configurations universelles et flexibles pour les broches d'E/S. Nous nous intéresserons également aux résistances de rappel vers le haut et le bas des broches d'E/S. Et ce n'est pas tout…

Comme les broches du Propeller 2 possèdent davantage de fonctions que celles que l'on trouve habituellement dans d'autres microcontrôleurs, un examen plus approfondi pourrait révéler des informations intéressantes. Pour établir une sorte de référence, nous jetterons d'abord un coup d'œil à la structure GPIO d'un ATmega328P de Microchip Technology. Ensuite, nous passerons au Propeller 2 pour voir s'il y a des différences et comment elles pourraient être utiles ultérieurement. Enfin, la lecture de l'état de quelques boutons devrait être assez facile.

Fonctionnement des broches d'E/S sur les autres microcontrôleurs

Les lecteurs familiers avec l'ATmega328P savent que les broches d'E/S ont pour l'essentiel quatre états : entrée, entrée avec rappel au niveau haut, sortie à niveau bas et sortie à niveau haut. Les fonctions analogiques occupent spécifiquement quelques broches sur l'ATmega328P et ne seront pas abordées ici. D'après la fiche technique, la **figure 1** montre comment est construite une broche d'E/S.

Nous avons essentiellement un mécanisme de commande pour une résistance de rappel au niveau haut, d'une valeur comprise entre 20 kΩ et 50 kΩ, composée d'un FET et de la résistance elle-même, et d'une logique de commande pour activer le FET (comme le montre la **figure 2**).
Dans la partie inférieure de la figure 1, vous pouvez voir une porte de transmission (**figure 3**) qui joue le rôle de commutateur à commande numérique permettant le passage de tensions analogiques.

Ce qui est intéressant, c'est le signal *SLEEP*, car il désactive la porte de transmission et ramène en même temps sa sortie à la masse avec un FET qui lui est propre. Ceci est dû au trigger de Schmitt, situé à la sortie de la porte de transmission. Il sert à transformer un niveau de tension analogique en valeur binaire zéro ou un. Au centre de la **figure 4**, se trouve l'étage de sortie avec activation.
Si le signal *enable* n'est pas actif, l'étage de sortie sera déconnecté ou il produira un niveau bas ou haut, selon l'entrée appliquée. Nous donnons beaucoup d'explications pour une simple activation ou désactivation,

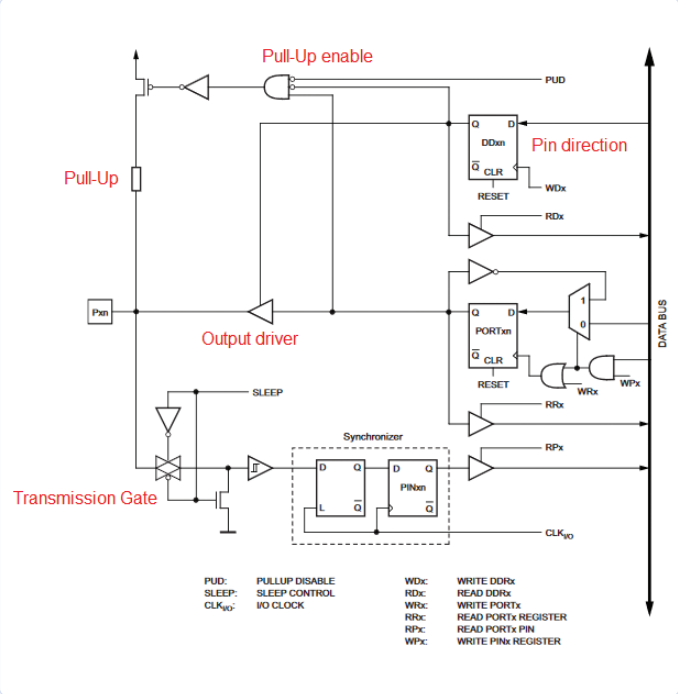

Figure 1. Partie de la fiche technique consacrée au bloc d'E/S ATmega328.

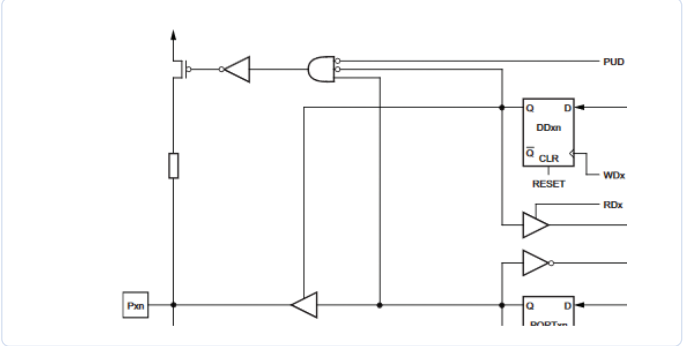

Figure 2. Transistor FET de l'ATmega328 pour le rappel au niveau haut.

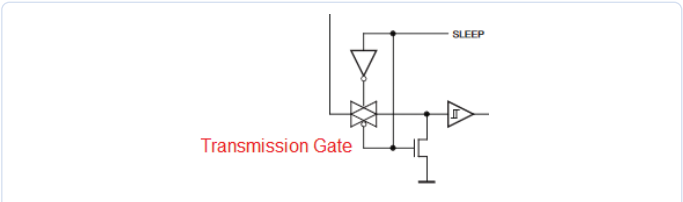

Figure 3. Porte de transmission.

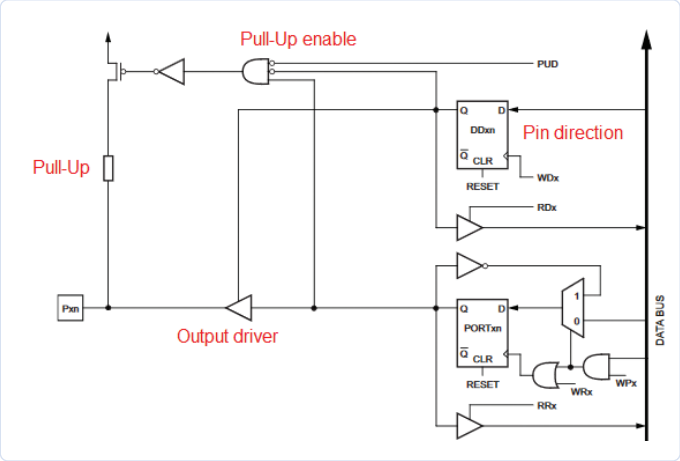

Figure 4. FET de l'ATmega328 pour le rappel au niveau haut.

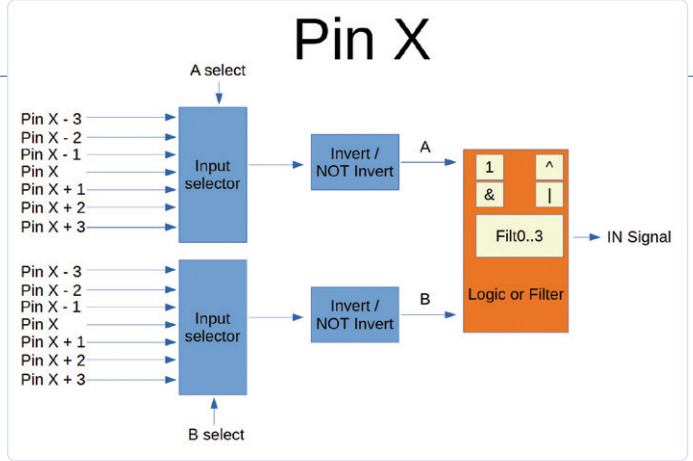

Figure 5. Voies d'entrée pour les broches d'E/S du Propeller 2.

même s'il s'agit surtout de décrire l'essentiel. La fonction de rappel au niveau haut va être très pratique pour une bonne raison. Nous ne pouvons mettre toutes les broches en sommeil que simultanément (comme d'autres parties de l'ATmega328). Si nous voulons qu'une broche soit inutilisée et ne fasse rien, son entrée sera flottante. Dans ce cas, elle capte du bruit aléatoire que le trigger de Schmitt convertit en zéro ou un binaire. Comme l'entrée est aléatoire, le trigger de Schmitt et la logique interne vont basculer rapidement entre zéro et un. Ainsi, à chaque transition, nous avons besoin d'un peu d'énergie. La broche étant inutilisée, ce n'est pas souhaitable et cela gaspille de l'énergie. Cette approche est à éviter, surtout si le montage fonctionne sur batterie. L'utilisation du rappel au niveau haut va donc fixer la tension à l'entrée à VCC et la commutation aléatoire de l'entrée ne se produira pas. Cette introduction d'une fonction simple était un peu longue, mais nous allons maintenant nous concentrer sur la mise en œuvre du microcontrôleur Propeller 2.

Broches intelligentes du Propeller 2

Comme je l'ai déjà mentionné, il n'y a pas de broches d'E/S simples sur le Propeller 2. En ce qui concerne les fonctions d'E/S de base, un coup d'œil à la fiche technique préliminaire du microcontrôleur révèle que nous avons plus de quatre choix, même pour une configuration simple d'entrée et de sortie. Toutes les broches peuvent fonctionner en mode entrée ou sortie numérique ou analogique. Commençons par le mode numérique. Pour l'entrée, si vous vous souvenez de l'ATmega328, tout était simple avec une porte de transmission et un trigger de Schmitt pour lire le signal. Dans le cas présent, nous avons un peu plus d'intelligence à l'intérieur de la broche. La **figure 5** montre la voie d'entrée numérique, ou les autres voies possibles, pour une seule broche.

La première chose étrange est que nous avons deux sélecteurs d'entrée parmi lesquels choisir. Vous pouvez ainsi sélectionner la broche actuelle et ± trois broches proches pour chacune de ces entrées. Une fois cette sélection effectuée, nous pouvons utiliser le signal inversé ou non inversé, ce qui donne un terme A ou B. Ces termes A et B sont ensuite introduits dans une deuxième partie où il est possible d'effectuer des opérations logiques ou des filtrages. Le résultat final sélectionné est ensuite présenté comme signal d'entrée (IN) du système. Ainsi, les broches standard sont un peu plus compliquées, mais aussi un peu plus polyvalentes que celles d'un ATmega328P. Pour la sortie numérique, les choses sont simples : nous avons une sortie au niveau bas et au niveau haut. Rien d'extraordinaire à première vue.

Vous vous souvenez de la résistance de rappel au niveau haut dont nous avons parlé au début ? Le Propeller 2 en a plusieurs et peut aussi appliquer un rappel vers le haut et vers le bas. Les combinaisons possibles sont assez simples :

- 1,5 kΩ
- 15 kΩ
- 150 kΩ
- 19 Ω
- 1 mA
- 100 µA
- 10 µA
- État flottant

Comme nous pouvons choisir d'avoir des résistances de rappel vers le haut ou le bas ou bien des sources de courant, les broches sont très flexibles même pour les différents bus dont nous pouvons avoir besoin ou avec lesquels nous voulons interagir.

Les rappels vers le haut ou le bas ne sont pas réalisés comme dans l'ATmega328 avec un FET spécial qui les active ou les désactive. Ces résistances agissent plutôt comme une *force de commande*, comme si elles se situaient entre l'étage de commande et la sortie de la broche. Pour utiliser les rappels vers le haut ou vers le bas, vous basculez la broche en mode sortie, avec la résistance donnée, et vous appliquez le niveau haut ou bas pour un rappel correspondant. Mais l'histoire ne s'arrête pas là, comme le montre la **figure 6**, qui donne un aperçu de la structure détaillée d'une broche.
Nous avons également un CN/A (convertisseur numérique-analogique) et un CA/N (convertisseur analogique-numérique) pour chaque broche qui peut également être utilisée en mode analogique. Comme je l'ai écrit dans un article précédent, la documentation n'est pas encore complète. Il manque notamment la description de certains domaines pour quelques configurations d'E/S, surtout lorsqu'il s'agit de configurations détaillées. Pour une première expérience pratique avec un composant électronique de pré-production, c'est assez courant. La **figure 7** propose un aperçu du registre de configuration des broches intelligentes. Et nous en avons fini avec la théorie pour le moment.

Mise en pratique

Pour notre premier test dans le monde réel, nous utiliserons nos quatre boutons d'entrée. Nous pouvons par ex. allumer une LED en appuyant sur un bouton. Pour cela, nous étendons notre code Spin2 existant et utilisons toutes les capacités que nous avons, comme la commande d'une LED à l'aide d'une broche d'E/S et la fonction `prints()` pour écrire des chaînes de caractères dans un UART. Notre objectif est

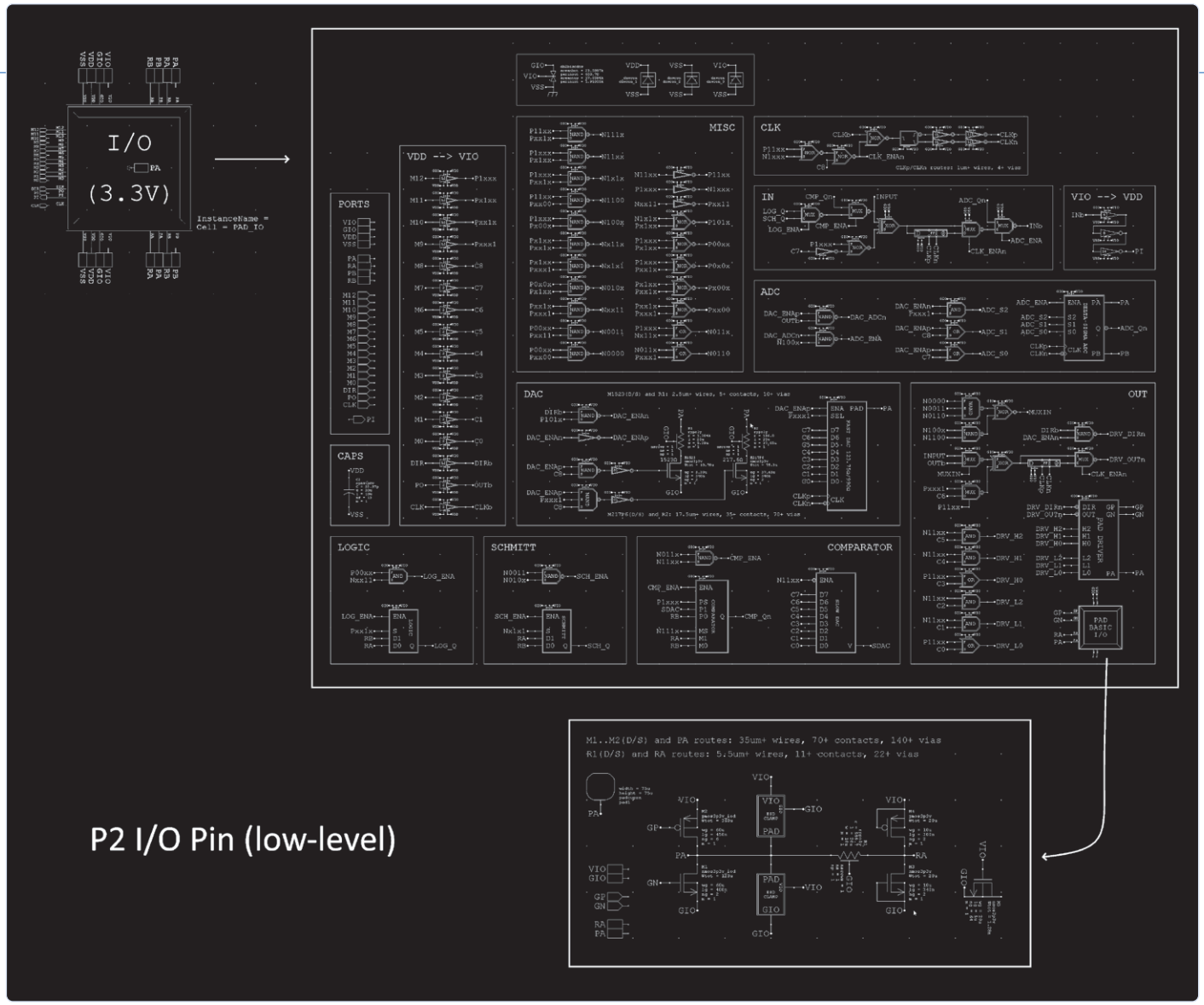

Figure 6. Description détaillée d'une broche du Propeller 2.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Inv. A	Input A			Inv. B	Input B			Logic / Filter			Low level pin control														Smartpin mode					0

Figure 7. Registre de configuration des broches intelligentes.

très simple. Nous lisons l'état de la broche, et nous l'appliquons à la LED, quatre fois seulement, une fois pour chaque LED et bouton présent. De plus, si un bouton est actionné ou relâché, nous émettons une chaîne correspondante. Commençons par la configuration de l'entrée. Comme les boutons matériels ne possèdent pas leurs propres résistances de rappel au niveau haut ou bas, nous devons utiliser les résistances internes de la puce. À partir de la **figure 8** et de la figure 7, nous calculons la valeur du registre de configuration.

En gardant la figure 5 à l'esprit, nous n'aurons à la fin que le niveau logique présenté par notre broche. Nous sélectionnons donc les entrées A et B pour qu'elles soient identiques, non inversées et transférées vers la sortie A, sans aucune logique ou filtrage appliqués. D'après la fiche technique du Propeller 2, cela donne 0000 0000 000 000 010 010 00 00000 en binaire ou 0x900 en hexadécimal si nous voulons une résistance de rappel au niveau haut de 15 kΩ.
Pour notre code, cela signifie que nous ajoutons maintenant quatre entrées et utilisons quatre sorties en connectant notre petite carte d'extension (**fig. 9**) à la carte d'évaluation Propeller 2.

LIEN

[1] **Dépôt GitHub de cette série d'articles** : https://github.com/ElektorLabs/200479-propeller2-hands-on

Mode Groups	%PPPPPPPPPPPPP	Input	Output	Notes	CIOHHHLLL Bits Explained
Non-ADC, Non-DAC Modes	0000_CIOHHHLLL	PinA Logic	OUT	Output enabled by DIR = 1 (otherwise floating)	Clocked mode I/O available with C:
	0001_CIOHHHLLL	PinA Logic	IN		
	0010_CIOHHHLLL	PinB Logic	IN		
	0011_CIOHHHLLL	PinA Schmitt	OUT		
	0100_CIOHHHLLL	PinA Schmitt	IN		
	0101_CIOHHHLLL	PinB Schmitt	IN		
	0110_CIOHHHLLL	PinA > PinB	OUT		
	0111_CIOHHHLLL	PinA > PinB	IN		
ADC Modes without DAC	100000_OHHHLLL	ADC, GIO 1x	OUT	Output enabled by DIR = 1 (otherwise floating)	
	100001_OHHHLLL	ADC, VIO 1x	OUT		
	100010_OHHHLLL	ADC, PinB 1x	OUT		
	100011_OHHHLLL	ADC, PinA 1x	OUT		
	100100_OHHHLLL	ADC, PinA 3.16x	OUT		
	100101_OHHHLLL	ADC, PinA 10x	OUT		
	100110_OHHHLLL	ADC, PinA 31.6x	OUT		
	100111_OHHHLLL	ADC, PinA 100x	OUT		
ADC + DAC Modes	10100_DDDDDDDD	ADC, PINA 1x	DAC 990 Ω, 3.3 V	ADC activated when OUT = 1 DDDDDDDD = DAC Level	
	10101_DDDDDDDD	ADC, PINA 1x	DAC 600 Ω, 2.0 V		
	10110_DDDDDDDD	ADC, PINA 1x	DAC 123.75 Ω, 3.3 V		
	10111_DDDDDDDD	ADC, PINA 1x	DAC 75 Ω, 2.0 V		
DAC Comparison Modes (Non-ADC)	1100_CDDDDDDDD	PinA > D	OUT, 1.5 kΩ	HHH=001, LLL=001 DDDDDDDD = DAC Level Output enabled by DIR = 1 (otherwise floating)	
	1101_CDDDDDDDD	PinA > D	!IN, 1.5 kΩ		
	1110_CDDDDDDDD	PinB > D	IN, 1.5 kΩ		
	1111_CDDDDDDDD	PinB > D	!IN, 1.5 kΩ		

C	IN/OUT
0	Live
1	Clocked

Can invert polarity of I/O with I, O:

I	IN
0	Non-Inverted
1	Inverted

O	OUT
0	Non-Inverted
1	Inverted

Pull up and/or down with HHH/LLL:

HHH/LLL	Drive
000	Fast
001	1.5 kΩ
010	15 kΩ
011	150 kΩ
100	1000 µA
101	100 µA
110	10 µA
111	Float

Figure 8. Vue d'ensemble des modes de fonctionnement des broches du Propeller 2.

Le **listage 1** contient le code Spin2 modifié pour configurer nos quatre broches d'entrée avec des rappels au niveau haut ainsi que la configuration de sortie pour nos broches de sortie LED.
Nous utilisons quatre variables globales pour stocker le dernier niveau de broche lu et envoyer un message série, uniquement si une broche a changé. Nous étendons le code avec deux fonctions, la première pour initialiser les LED et la seconde pour initialiser nos entrées. Le **listage 2** montre les quatre octets déclarés pour stocker l'état des quatre interrupteurs connectés à la carte.
Vous pouvez également voir que le code a été modifié avec les deux fonctions d'initialisation (**listage 3**). La magie du fonctionnement des entrées avec la capacité de rappel au niveau haut est enfouie dans les bits décrits dans le **listage 4**.
Nous configurons l'entrée et la sortie pour qu'elles soient pilotées à l'aide d'une résistance de 15 kΩ. Avec la dernière ligne, nous configurons nos quatre broches d'entrée pour qu'elles soient fixées au niveau bas, car nous les utilisons effectivement comme sorties.

Une fois l'initialisation effectuée, la fonction `read_switch()` est appelée à plusieurs reprises afin que nous puissions examiner de plus près le code qui se déroule dans cette fonction. Comme on le voit dans le **listage 5**, le code utilise un certain nombre de structures IF et IF-ELSE après avoir lu les états des broches dans les variables `a` à `d`. La première chose à faire est de détecter si une broche a changé en comparant l'ancien état mémorisé avec le nouvel état lu. Si ces valeurs diffèrent, nous pouvons supposer que l'état de l'entrée de la broche a changé et déterminer si c'est parce que l'interrupteur correspondant a été actionné ou relâché. En fonction de l'état actuel de la broche, nous utilisons la fonction `prints()` pour afficher la mention « Switch pressed » (bouton pressé) ou « Switch released » (bouton relâché). Le **listage 5** montre comment notre fonction `read_switch()` se présente et fonctionne, avec de simples chemins IF-THEN-ELSE pour chaque

Listage 1. Configuration des quatre broches d'entrée.

```
WRPIN(52, %0000_0000_000_000_010_010_00_00000)
WRPIN(53, %0000_0000_000_000_010_010_00_00000)
WRPIN(54, %0000_0000_000_000_010_010_00_00000)
WRPIN(55, %0000_0000_000_000_010_010_00_00000)
```

Listage 2. Déclaration des octets de stockage des états des boutons.

```
VAR BYTE PinA, PinB, PinC, PinD
```

Listage 3. Fonctions d'initialisation ajoutées.

```
PUB main()

  LED_init()
   Input_init()

  pinwrite(56, 1 )
  serial_start()
  repeat
     read_switch()
```

Listage 4. Initialisation des entrées.

```
PUB Input_init( )
' A and B as input using A with 15k drive
' strength for pull-up and pull-down as output no
' smartpin function
   WRPIN(52, %0000_0000_000_000_010_010_00_00000)
   WRPIN(53, %0000_0000_000_000_010_010_00_00000)
   WRPIN(54, %0000_0000_000_000_010_010_00_00000)
   WRPIN(55, %0000_0000_000_000_010_010_00_00000)
   PINWRITE(55..52,0)
   'Drive pin low with 15k resistance
```

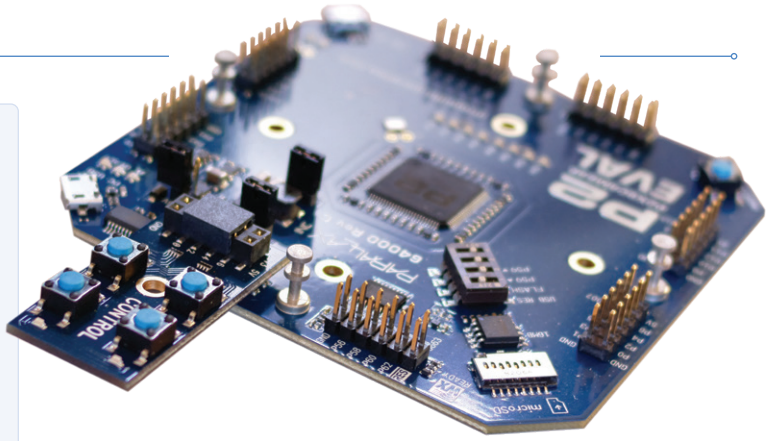

Figure 9. Propeller 2 avec la platine des boutons raccordée.

Listage 5. Fonction read_switch( ).

```
PUB read_switch( ) | a,b,c,d
 'We will read the pin, reflect the LED state
 'We send a change in state with the UART
  a := pinread(52)
  b := pinread(53)
  c := pinread(54)
  d := pinread(55)
  IF(PinA <> a )
     PinA := a
     prints(string("A:"))
     IF( a <> 0 )
       pinwrite(49,1)
       prints(@PinPressed)
     ELSE
       pinwrite(49,0)
       prints(@PinReleased)
  IF(PinB <> b )
     PinB := b
     prints(string("B:"))
     IF( b <> 0 )
       pinwrite(48,1)
       prints(@PinPressed)
     ELSE
       pinwrite(48,0)
       prints(@PinReleased)
  IF(PinC<> C )
     PinC := c
     prints(string("C:"))
     IF( c <> 0 )
       pinwrite(50,1)
      prints(@PinPressed)
     ELSE
       pinwrite(50,0)
       prints(@PinReleased)
  IF(PinD <> d )
     PinD := d
     prints(string("D:"))
     IF( d <> 0 )
       pinwrite(51,1)
       prints(@PinPressed)
     ELSE
       pinwrite(51,0)
       prints(@PinReleased)
```

bouton utilisé. Vous pouvez également y lire que l'état des LED est modifié en fonction du nouvel état détecté.
Pour l'instant, ce sera tout avec les broches, mais nous espérons que cela vous a donné un petit aperçu de leur fonctionnement. Cette série se termine avec cet article. Toutefois le Propeller 2 reste sur ma paillasse, j'écrirai encore à son sujet ou je ferai une courte vidéo. Comme le logiciel et l'EDI sont appelés à évoluer, nous laissons à ce microcontrôleur le temps de grandir. En attendant, vous pouvez accéder au code écrit pour cette série dans notre dépôt GitHub [1].

Première impression finale

Après avoir travaillé avec le Propeller 2, je peux dire qu'il s'agit d'un puissant microcontrôleur qui nécessite sans aucun doute d'accorder plus d'attention à ses caractéristiques qu'un Raspberry Pi Pico. L'utilisation du langage Spin2 et de l'assembleur est particulière, et rend impossible la réutilisation du code C existant. Personnellement, je pense que ce ne sera pas mon microcontrôleur de prédilection pour les projets courants, mais cette puce pourra certainement être utile pour un certain nombre d'applications de niche.
Qu'en est-il des interfaces SPI, I²C et HDMI mentionnées au début de cette série d'articles ? Au fur et à mesure des évolutions apportées aux chaînes d'outils, il serait intéressant de les finaliser et de passer à des outils améliorés. Impatient d'en savoir plus sur le Propeller 2 ? Envoyez-nous un courriel ou utilisez l'un de vos canaux de réseaux sociaux préférés pour nous envoyer un message.

200479-E-04

Contributeurs

Conception et texte : **Mathias Claußen**
Rédaction : **Jens Nickel, C. J. Abate**
Mise en page : **Giel Dols**
Traduction : **Pascal Godart**

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (mathias.claussen@elektor.com) ou contactez Elektor (redaction@elektor.fr).