Kennismaking met de **Parallax Propeller 2** (Deel 5)

Smart Pin-functie

Mathias Claussen

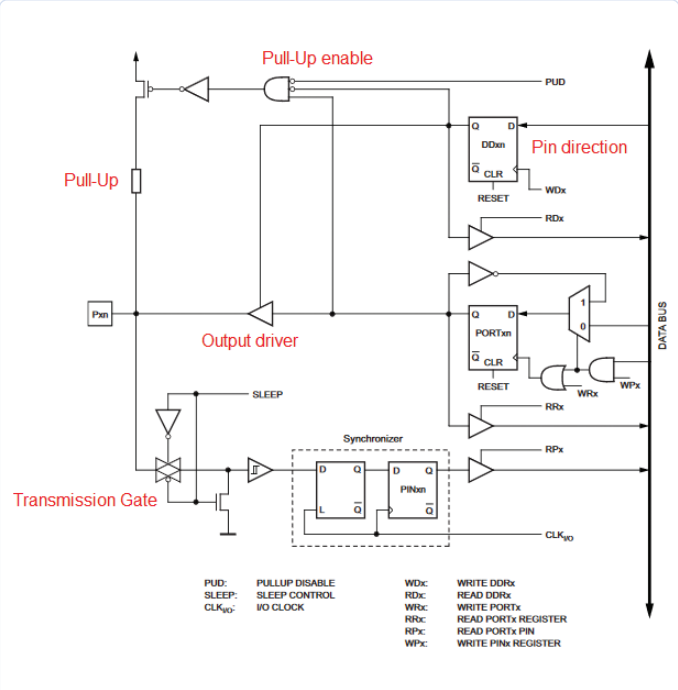
Ter afronding van deze serie over de Parallax Propeller 2 , introduceren we de “smart pin”-functie, die universele en flexibele configuraties voor de I/O-pinnen mogelijk maakt. We zullen ook kijken naar de pull-up en pull-down weerstanden van de I/O pinnen. Maar er is meer te ontdekken.

Aangezien de pinnen van de Propeller 2 meer functies hebben dan gebruikelijk in andere MCU’s, kan een nadere beschouwing interessante inzichten opleveren. Om een soort referentie te krijgen, kijken we eerst naar de GPIO-structuur van een Microchip Technology ATmega328P. Daarna gaan we naar de Propeller 2 om te zien of er verschillen zijn en hoe die later van pas kunnen komen. Daarna is het eenvoudig om toestanden van druktoetsen in te lezen.

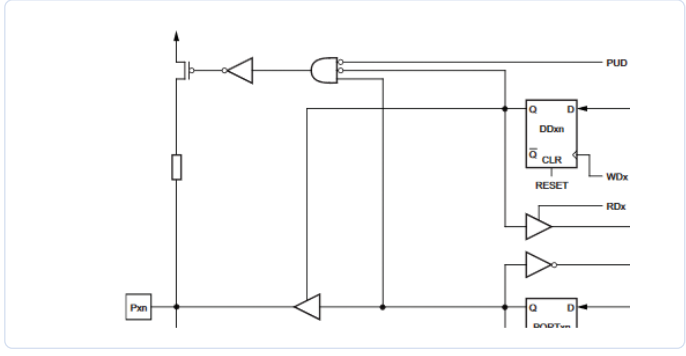
Hoe I/O-Pinnen Werken in Andere MCU’s

Wie bekend is met de ATmega328P weet dat er in principe vier I/O-pintoestanden zijn: ingang, ingang met pull-up, uitgang laag en uitgang hoog. Analoge functies zijn voorbehouden aan een paar pinnen op de ATmega328P en zullen hier niet worden behandeld. Uit de datasheet kunnen we in **figuur 1** het principe van een I/O-pin zien. In rood zijn enkele interessante secties toegevoegd. We zien een mechanisme voor een pull-up weerstand, die tussen 20 kΩ tot 50 kΩ is gespecificeerd, bestaande uit een FET, de weerstand zelf en besturingslogica om de FET in te schakelen (zoals het onderdeel in **figuur 2** laat zien). In het onderste deel van **figuur 1** zien we een poort (**figuur 3**) die fungeert als een digitaal gestuurde schakelaar die analoge spanningen doorlaat.

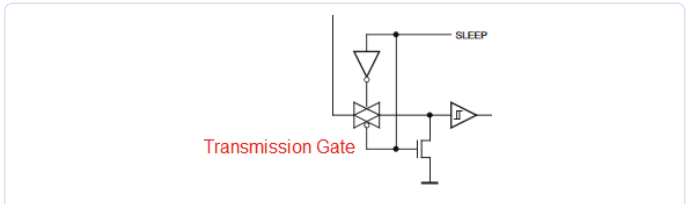
Interessant is het *SLEEP*-signaal, omdat dit de poort uitschakelt en tegelijkertijd met een speciale FET zijn uitgang met massa verbindt. De reden hiervoor is de Schmitt-trigger, die zich aan de uitgang van



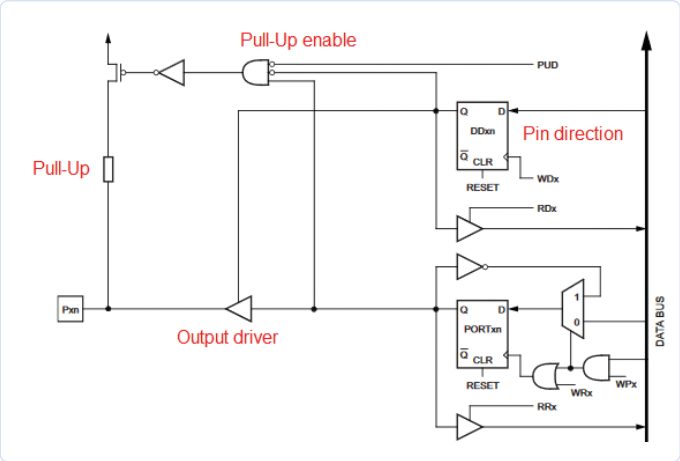
Figuur 1: I/O-block van de ATmega328 in de datasheet.



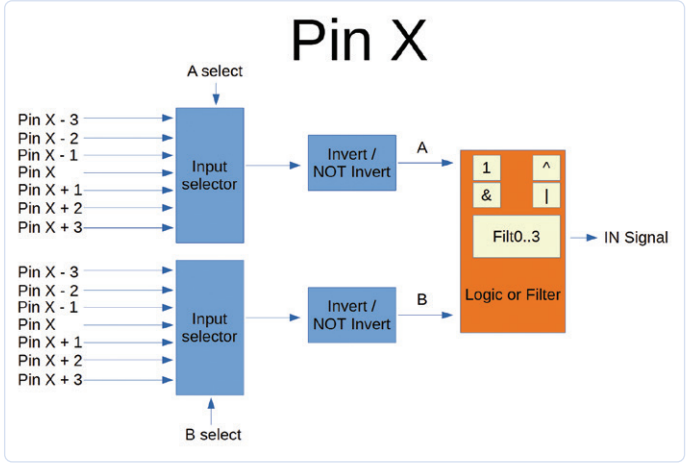
Figuur 2: Uitgangstrap van de ATmega328.



Figuur 3: Poort die ingang doorschakelt.



Figuur 4: ATmega328 FET voor pull-up.



Figuur 5: Input-paden voor een Propeller 2 I/O-pin.

de transmissiepoort bevindt. Deze wordt gebruikt om een analoog spanningsniveau om te zetten in een binaire waarde van nul of één. In het midden van **figuur 4** staat de eindtrap met enable.
Als enable niet actief is, wordt de eindtrap uitgeschakeld, want anders zal hij -afhankelijk van de ingang- een laag of hoog uitsturen. Een hoop tekst voor alleen aan en uit, maar dit laat de basis zien en waarom de pullup handig is. We kunnen alleen alle pinnen tegelijk in slaapstand zetten (zoals andere onderdelen van de ATmega328). Als we één pin ongebruikt willen laten, zal zijn ingang zweven. Dit betekent dat het willekeurige ruis oppikt die de Schmitt-trigger zal omzetten in een binaire nul of één. Omdat de ingang willekeurig is, zullen de Schmitt-trigger en de interne logica snel tussen nul en één schakelen, waardoor elke overgang een beetje energie gebruikt. Voor een ongebruikte pin is dit niet wenselijk en verspilt het energie. Vooral als we het apparaat met batterijvoeding laten werken, is dit iets dat we moeten vermijden. Daarom zal het gebruik van de pull-up de ingangsspanning op VCC zetten en zal een willekeurig schakelen van de ingang niet plaatsvinden. Dit is een lange inleiding tot iets eenvoudigs dat alleen aan en uit schakelt, maar nu zullen we ons richten op de Propeller 2 implementatie.

Smart Pins op de Propeller 2

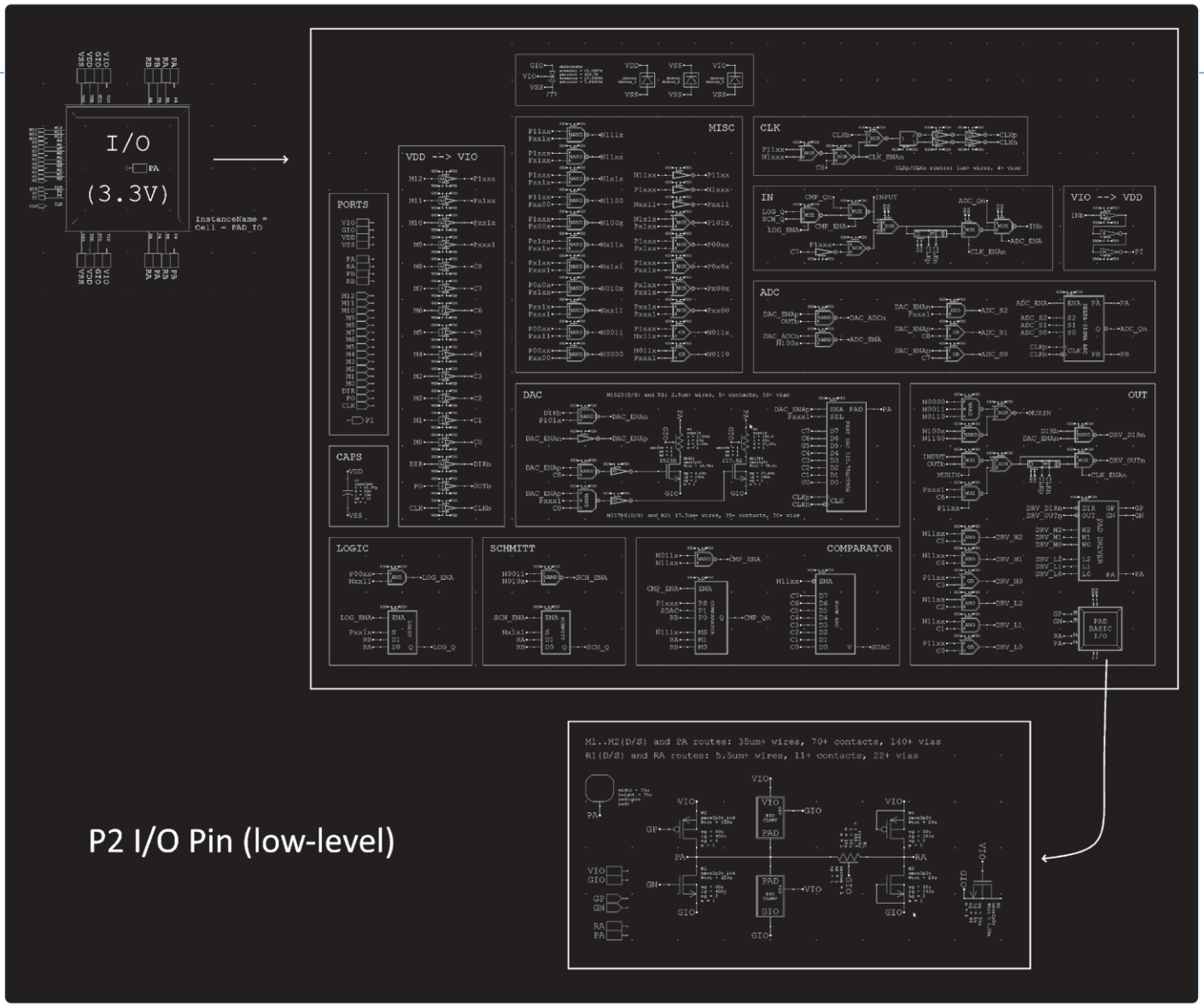
Zoals ik al eerder zei, zijn er geen eenvoudige I/O-pinnen op de Propeller 2. Zelfs voor de basis in- en uitgangsfuncties blijkt dat we meer dan vier keuzes hebben. Alle pinnen kunnen in digitale of analoge in- of uitgangsmodus werken. We beginnen met de digitale. Beginnend met de ingang, zoals u zich wellicht herinnert van de ATmega328, was alles rechttoe rechtaan, een transmissiepoort en een Schmitt-trigger om het signaal te lezen. Deze keer hebben we wat meer slimmigheidjes in de pin. **Figuur 5** toont het digitale ingangspad, of beter paden, voor een enkele pin.
Het eerste vreemde is dat we twee ingangsselectors hebben om uit te kiezen. Dat maakt het mogelijk om voor elk van deze ingangen de huidige pin en +/- drie pinnen ernaast te selecteren. Na deze selectie kunnen we het geïnverteerde of niet-geïnverteerde signaal gebruiken, wat resulteert in een A of B term. Onze A en B ondergaan vervolgens in een tweede deel logische bewerkingen of filtering. Het uiteindelijk gekozen resultaat wordt als IN-signaal aan het systeem geleverd. Dit maakt de standaardpinnen wat ingewikkelder maar ook veelzijdiger dan die op een ATmega328P. Voor de digitale uitgang zijn de zaken eenvoudig: we hebben gewoon een lage en hoge uitgang, niets bijzonders hier op het eerste gezicht.
Herinnert u zich de pull-up weerstand waar we het in het begin over hadden? De Propeller 2 heeft er meer dan één en kent ook een pull-down functie. De combinaties zijn vrij eenvoudig:

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

Aangezien we kunnen kiezen tussen pull-up- of pull-down-weerstanden of stroombronnen, maakt dit de pennen zeer flexibel, zelfs voor de verschillende bussen die we nodig hebben of waarmee we willen interageren.
De pull-ups of pull-downs zijn niet zoals in de ATmega328 uitgevoerd met een afzonderlijke FET die ze in- of uitschakelt, maar bepalen ze de *drive strength*, alsof ze tussen de stuurtrap en de pin-uitgang staan. Om de pull-up of pull-down te gebruiken schakel je de pin om als uitgang met de gegeven weerstand en gebruik je laag of hoog voor pull-up of pull-down. Maar er is meer aan de hand, zoals je kunt zien in **figuur 6**, die inzicht geeft in het low-level ontwerp van een pin. We hebben ook een DAC en ADC voor elke pin die ook in analoge mode kan worden gebruikt. Zoals ik al eerder in deze serie schreef, is de documentatie nog niet compleet, dus enkele I/O-configuraties ontbreken nog, vooral als het gaat om de low-level configuraties. Voor een eerste hands-on ervaring met pre-productie silicium is dit te verwachten. **Figuur 7** zou u een overzicht moeten geven van het slimme pin-configuratieregister. Maar voor nu is dit genoeg theorie.

Hands On

Voor onze eerste echte test, gebruiken we onze vier input-toetsen. We kunnen een LED laten oplichten met een druk op de knop. Hiervoor



Figuur 6: Low-level ontwerp van een Propeller 2 pin.

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

Figuur 7: Smart-pin configuratieregister.

zullen we onze bestaande SPIN2-code uitbreiden en alle leuke mogelijkheden gebruiken die we hebben, zoals het aansturen van een I/O-pin voor een LED en onze `prints()`-functie om strings naar een UART te schrijven. Wat we willen bereiken is simpel. We lezen vier keer de toestand van de pin en repliceren die naar de LED, één voor elke aanwezige LED en knop. Bovendien, als een knop wordt ingedrukt of losgelaten, voeren we een overeenkomstige string uit. Laten we beginnen met de ingangsconfiguratie. Omdat de hardware-knoppen geen eigen pull-up of pull-down weerstanden hebben, moeten we de interne weerstanden van de chip gebruiken. Uit **figuur 8** en **figuur 7** moeten we de waarde voor het config-register berekenen. Met **figuur 5** in gedachten hebben we alleen het logische niveau dat door onze pen wordt aangeboden, dus kiezen we ingang A en B hetzelfde, niet geïnverteerd en naar uitgang A zonder enige logica of

WEBLINK

[1] **Elektor GitHub**: 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

Figuur 8: Overzicht van de Propeller 2 pin-modes.

filtering toe te passen. Volgens de Propeller 2 datasheet komt dit neer op 0000 0000 000 000 010 00 00000 in binair of 0x900 in hex als we een 15-kΩ pull-up willen gebruiken.
Voor onze code betekent dit dat we nu vier ingangen toevoegen en vier uitgangen gebruiken door een klein addon-board (**figuur 9**) aan te sluiten op het Propeller 2 evaluatiebord. In **listing 1** ziet u de aangepaste SPIN2-code om onze vier ingangspinnen in te stellen met pull-ups en de uitgangsconfiguratie voor onze LED-uitgangspinnen. We gebruiken vier globale variabelen om het laatst uitgelezen pin-niveau op te slaan en alleen een serieel bericht te sturen als een pin veranderd is. We breiden de code die we gebruikten uit met twee functies, de eerste om de LEDs te initialiseren en de tweede om onze ingangen te initialiseren. In **listing 2** hebben we vier bytes gedeclareerd om de toestand op te slaan van de vier schakelaars die op het bord zijn aangesloten. Ook kunt u zien dat de code is veranderd met de twee functies voor initialisatie (**listing 3**). De magie om de ingangen te laten werken met actieve pullup-capaciteit zit verborgen in de bits in **listing 4**. We stellen de in- en uitgang in op een weerstand van 15 kΩ en met de laatste regel zetten we onze vier ingangspinnen laag, omdat wij ze eigenlijk als uitgang gebruiken.

Nadat de initialisatie is gedaan, wordt de functie `read_switch()` herhaaldelijk aangeroepen, laten we bekijken wat er in deze functie gebeurt. Zoals te zien is in **listing 5**, gebruikt de code een paar *IF* en *IF-ELSE* constructies na het inlezen van de pin-toestanden in de variabelen `a` tot en met `d`. Eerst moet worden nagegaan of een pin veranderd is, door de oude opgeslagen toestand te vergelijken met de zojuist gelezen nieuwe toestand. Als deze waarden verschillen, kunnen we aannemen dat de toestand op de pin veranderd is en bepalen of dit komt doordat de overeenkomstige toets werd ingedrukt

Listing 1: Pin setting.

```
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)
```

Listing 2: Byte declaration.

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

Listing 3: Added initialization functions.

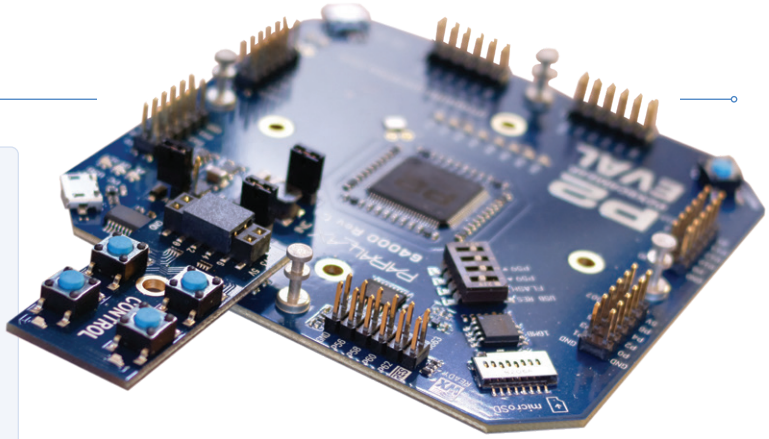
```
PUB main()

  LED_init()
   Input_init()

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

Listing 4: Input initialization.

```
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
```

Listing 5: read_switch( ) function.

```
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)
```



Figuur 9: Propeller 2 met druktoetsen.

of losgelaten. Afhankelijk van de huidige pintoestand, gebruiken we de functie `prints()` om "*Toets ingedrukt*" of "*Toets losgelaten*" uit te voeren. In **listing 5** kun je zien hoe onze `read_switch()` eruit ziet en werkt, met eenvoudige `IF-THEN-ELSE` statements voor elke schakelaar die we gebruiken. Ook kun je hier zien dat de toestand van de LED wordt aangepast aan de nieuwe status die we hebben gedetecteerd. Voorlopig hebben we genoeg gedaan met de pinnen en hopen dat dit u een beetje inzicht geeft. Dit is voorlopig ook het einde van deze serie. Dit zal vast niet de laatste keer zijn dat we over de Propeller 2 schrijven of een korte video maken. Naarmate de software en IDE evolueren, zullen we hem wat tijd geven om volwassen te worden. In de tussentijd kunt u toegang krijgen tot de code die in deze serie is ontwikkeld in onze GitHub repository [1].

Algemene Conclusies

Na het werken met de Propeller 2 kan ik zeggen dat het een krachtige MCU is, waarvan de details zeker meer aandacht vragen dan bijvoorbeeld bij een Raspberry Pi Pico. Het gebruik van SPIN2- en assembly is bijzonder, en het maakt hergebruik van bestaande C-code onmogelijk. Persoonlijk denk ik dat ik voor de meeste van mijn projecten niet voor deze MCU zal kiezen, maar er zijn zeker niches waar deze chip van pas zal komen.

Hoe zit het met de in het begin genoemde SPI, I²C en HDMI? Aangezien er enkele verbeteringen in de toolchains plaatsvinden, zou het interessant zijn om dat af te ronden en verder te gaan met verbeterde tools. Als u niet wilt wachten op het volgende hoofdstuk over de Propeller 2, stuur ons dan een e-mail of gebruik een van uw favoriete sociale mediakanalen om ons een bericht te sturen.

200479-E-02

Een bijdrage van

Idee en illustraties: **Mathias Claußen**
Tekst en redactie: **Jens Nickel, C. J. Abate**
Layout: **Giel Dols**
Vertaling: **Jelle Aarnoudse**

Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur dan een e-mail naar de auteur mathias.claussen@elektor.com of naar de redactie van Elektor via redactie@elektor.com.