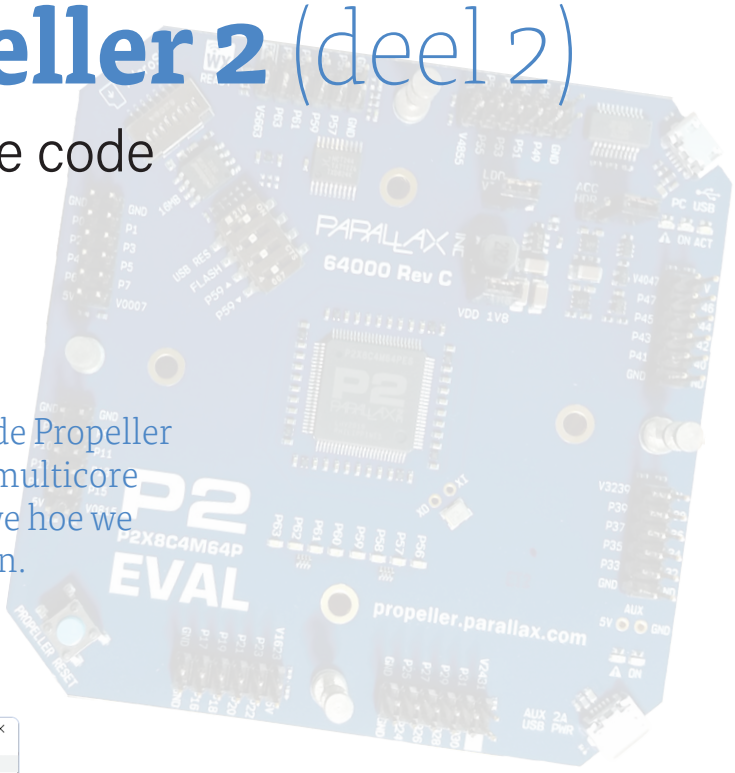
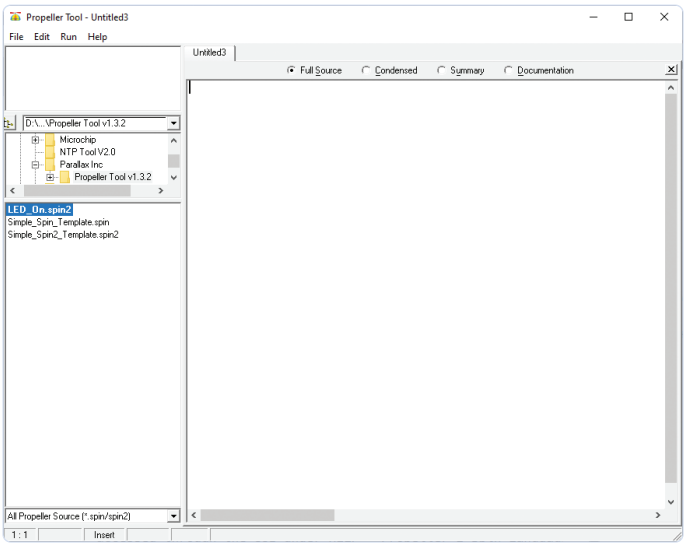
Kennismaking met de Parallax Propeller 2 (deel 2)

de ontwikkelomgeving en de code

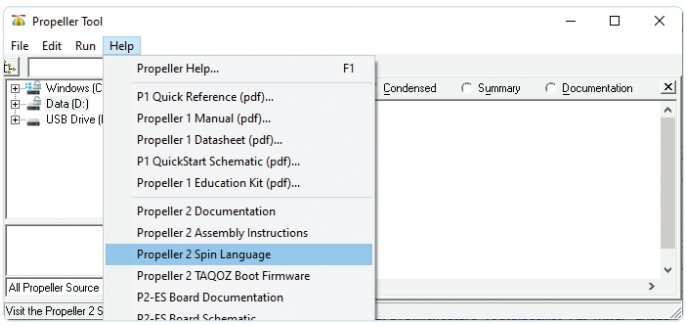


Mathias Claußen (Elektor)

In het eerste deel van deze serie hebben we de Propeller 2 geïntroduceerd, de geavanceerde nieuwe multicore microcontroller van Parallax. Nu bekijken we hoe we met de taal Spin2 een LED kunnen aansturen.



Figuur 1. Gebruikersinterface van de Propeller Tool.



Figuur 2. De Spin2-documentatie is via het Help-menu toegankelijk.

Nadat we in deel 1 de mogelijkheden van de Parallax Propeller 2 hebben leren kennen, kijken we nu naar de ontwikkelomgeving en het schrijven van code. Lees hier hoe u de Spin2-taal kunt gebruiken om een LED aan te sturen.

Toegang tot de onboard-LED's

Laten we beginnen met de software-omgeving en het aansturen van een van de LED's op het board. We ontwikkelen onder Windows 10 met de door de leverancier geleverde tools. Het resultaat is een basis-softwaresetup om code te compileren en te uploaden naar de Propeller 2. Parallax heeft zijn Propeller Tool versie 2.3 Alpha ter beschikking gesteld, inclusief Propeller 2-ondersteuning voor Spin/Spin2. Als assembleertaal uw voorkeur heeft, kunt u PNut gebruiken als ontwikkelomgeving of van inline assembler gebruik maken. Als u net als ik graag in C/C++ codeert: de compiler en ontwikkelomgeving zijn momenteel helaas nog niet beschikbaar. Er wordt aan gewerkt om dit te verbeteren, zoals te zien op de videopresentatie [1] over C/C++ op de Propeller 2.

De basis-ontwikkelomgeving

U kunt de Propeller Tool 2.3 Alpha downloaden op [2]. Download eerst de Propeller Tool 1.3.2 en voeg als tweede stap in de programmamap de Propeller Tool 2.3 Alpha-executable toe. Nadat de installatie is voltooid, kunt u de editor starten en beginnen met coderen. De gebruikersinterface van de tool is te zien in **figuur 1**.

Coderen in assembleertaal of in Spin2?

De vraag is: Spin2 of assembleertaal voor de eerste code? Om het eenvoudig te houden, gaan we voorlopig aan de slag met Spin2. Spin2 is een interpreter zoals BASIC, maar dan anders. De commando's en taalreferenties voor Spin2 zijn in de Propeller Tool te vinden onder de menu-optie Help (**figuur 2**).

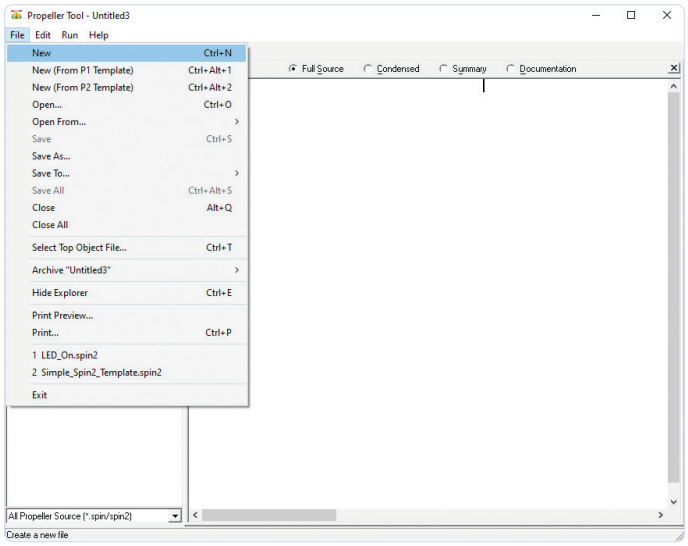


Figure 3. Een nieuw Spin2-project aanmaken.

Figure 4. Close-up van de LED's die zijn aangesloten op MCU-pinnen 56...63.

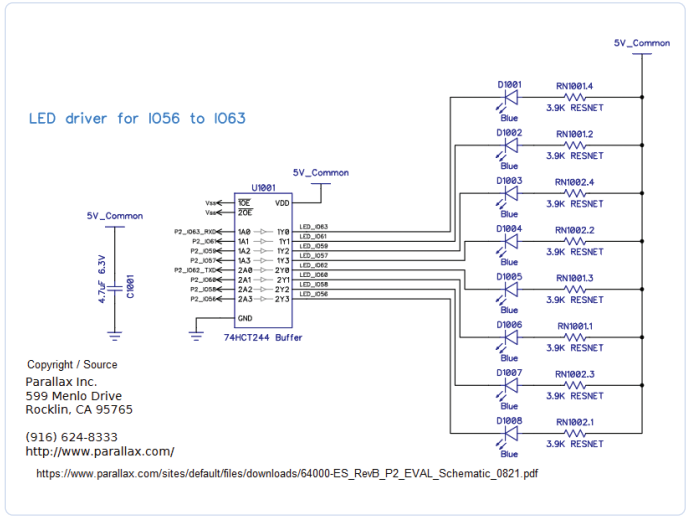


Figure 5. De LED's worden aangestuurd via een buffer van het type 74HCT244.

Om met Spin2 aan de slag te gaan, kunt u gebruik maken van een handleiding, maar bedenk wel dat deze nog niet af is. Via het menu krijgen we dus de voorlopige documentatie voor de geïmplementeerde Spin2-taal op de Propeller 2 MCU.
Aangezien dit uiteindelijk geïnterpreteerde code is, zijn ongeveer zes instructie-cycli, dus 12 klok-cycli, nodig om één commando uit te voeren. Dit is – in termen van klokcycli – niet de snelste manier, maar met Spin2 beginnen is gemakkelijker dan met assembleertaal. Omdat Spin2 meer een high-level benadering heeft, blijven van commando's de meeste assembleertaal-instructies verborgen en zijn ze makkelijker te gebruiken. U kunt een nieuw Spin2-project maken zoals in **figuur 3** getoond.

I/O-pinnen

Voor we aan het coderen slaan, kijken we kort hoe de I/O-pinnen werken. We gebruiken ze hier als gewone I/O-pinnen; we zullen later een uitgebreid overzicht geven over al hun mogelijkheden. In het algemeen moeten we de pin van onze keuze instellen als een uitgang om iets met een laag of hoog niveau aan te sturen. Op het evaluatieboard zijn de LED's (gemarkeerd met P56...P63) duidelijk zichtbaar (**figuur 4**). Dit is een groep LED's die verbonden is met pinnen 56...63 op de Propeller 2; we gaan hier de eerste van die LED's (nummer 56) aansturen. Uit het schema (**figuur 5**) blijkt dat ze niet rechtstreeks op de chip zijn aangesloten, maar via een achtvoudige buffer van het type 74HCT244. De anodes van de LED's hangen aan V_{CC}; de I/O-pin kan met behulp van de buffers de kathodes aan massa leggen om de betreffende LED' te doen oplichten.

Geïnverteerde LED

Voor de code die we gaan schrijven, betekent dit dat een hoog niveau op de MCU-pin de LED zal uitschakelen en een laag niveau hem zal laten oplichten. Met deze omgekeerde logica in gedachten, moeten we in onze code pin 56 hoog maken om de LED uit te schakelen. Als we niets doen, is de pin standaard als ingang geconfigureerd en zal de achtvoudige buffer denken dat de pin logisch hoog is, zodat de aangesloten LED wordt uitgeschakeld. Om in onze Spin2-code de LED te laten oplichten, moetn we dus het volgende doen:

- de MCU initialiseren, plus tenminste één core (cog);
- pin 56 als uitgang configureren;
- pin 56 laag maken;
- niets doen.

Stap één, de initialisatie, wordt in dit geval voor ons gedaan; daar hoeven we ons nu geen zorgen over te maken. De Spin2-code zelf beslaat slechts een paar regels. De code begint met de functie `pub main()` gevolgd door de methode `pinwrite()` (**figuur 6**). Die `pinwrite()` is ingebouwd in de Spin2-taal en maakt het mogelijk om een of meer pinnen met een gegeven waarde in te stellen. Hier gebruiken we pin 56, waar onze LED op is aangesloten, en geven voor het uitgangsniveau een '0' door om de pin laag te maken en de LED in te schakelen. De laatste regel `repeat` forceert de cog in een eindeloze lus aangezien de code onder de `repeat` zal worden uitgevoerd. In dit geval staat daar verder geen code, waardoor de cog gedwongen wordt niets te doen en te blijven doorgaan. Zonder `repeat` zou de cog het einde van de code bereiken en ophouden de I/O-pinnen aan te sturen.

De code naar de Propeller 2 overbrengen

Als de code klaar is, kunt u deze uploaden naar het Propeller 2-evalu-

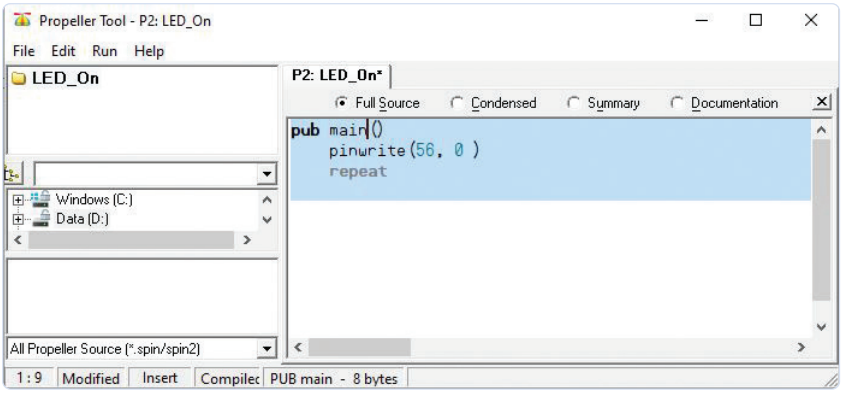


Figure 6. De code om LED56 te laten oplichten.

atieboard en daar doen uitvoeren. Hiervoor sluit u het board aan op een van de USB-poorten van het systeem en kiest u in het menu *Run->Compile Current->Load RAM* om de code direct in het RAM van de MCU te uploaden en te laten uitvoeren. Aangezien de LED op pin 56 nu oplicht, is deze eerste soort "Hello world" nu klaar. Nu kunt u zich afvragen of we de LED ook kunnen laten knipperen – ja natuurlijk kan dat. In Spin2 hebben we een `WAITMS`-instructie die bijvoorbeeld met `WAITMS(500)` de uitvoering van code 500 ms zal vertragen. Aangezien we hebben gezien dat de code na `repeat` steeds opnieuw wordt uitgevoerd, is de volgende stap het laten knipperen van de LED. Gebruik de informatie die u tot nu toe hebt verzameld om de code te ontwikkelen en aan te passen zodat de LED gaat knipperen. Hiervoor zullen we later een voorbeeldoplossing geven.

Nu we een I/O pin kunnen aansturen, is de volgende stap om uit te zoeken hoe we seriële data kunnen versturen. De meesten van ons zijn gewend aan deze manier van debuggen, vooral als eerste of tweede stap op een apparaat met een AVR. Aangezien dit de Smart-Pin functies zal betreffen, zullen we ons in de volgende aflevering daarmee vertrouwd maken.

200479-B-04

Een bijdrage van

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

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

WEBLINKS

[1] **Parallax, "Propeller 2 Live Forum Early Adopter Series – C Programming with Eric Smith," 6 juli 2020**: https://bit.ly/propeller2-c

[2] **Download van de Propeller Tool 2.3 Alpha**: https://propeller.parallax.com/p2.html#software