Using DMX-Protocol over SPI/UART

Author: Sam Knoors/ Rein van der Linden  
Intern:  
Extern  
Company: PXL

Het doel van dit onderzoek is het begrijpen en het creëren van een DMX-signaal met de PSOC 4. Als dit verkregen is zal er een DMX RGB Led par worden aangesloten op de PSOC en zal de data verstaan moeten worden door die led par. De verschillende communicatieprotocollen die worden gebruikt zijn UART en SPI. Het is de bedoeling de UART en SPI blokken zo te configureren dat er een bruikbaar DMX-signaal verkregen wordt. De uiteindelijke methode zal SPI zijn omdat tijdens het onderzoeken van de UART-methode onvoorziene complicaties aan het licht kwamen. Het uiteindelijke resultaat is dat de LED par luistert naar de data die gestuurd wordt en dat die de juiste kleur weer geeft. Alleen zit er een flikkering van andere kleuren doorheen. De oorzaak wordt verder beschreven in deze paper. Ook zullen de onderzoeksmethode, resultaten en adviezen besproken worden.

Inhoud

[2. Introductie 3](#_Toc29040454)

[3. Materiaal en methode 3](#_Toc29040455)

[DMX-protocol 4](#_Toc29040456)

[RS-485 in het kort 4](#_Toc29040457)

[Seriële communicatie 4](#_Toc29040458)

[DMX in detail 5](#_Toc29040459)

[Methode met UART 6](#_Toc29040460)

[Methode met SPI 6](#_Toc29040461)

[4. Resultaten 7](#_Toc29040462)

[5. Discussie 7](#_Toc29040463)

[6. Conclusie 8](#_Toc29040464)

[7. Referentielijst 9](#_Toc29040465)

[8. Bijlagens 10](#_Toc29040466)

# Introductie

Naar aanleiding voor het onderzoek was er de uitdaging om lichttechniek apparatuur zoals een laser of LED par aan te sturen met een PSOC 4 microprocessor. Hiervoor moest er een signaal over SPI gestuurd worden zodoende dat het leesbaar werd voor die apparatuur, deze maken namelijk gebruik van het DMX-protocol.

Hierdoor is er ook een onderzoek naar de werking van zowel UART als SPI gedaan, gezien beide gebruikt werden in het project opzoek naar een werkend eindproduct. Op een blogpost op de element 14 community (onderaan gelinkt) staat een uitgebreide uitleg die gebruikt is om het DMX-protocol te begrijpen.

Als eerste is er een reeds bestaand klein programma van de “Cyrpess community” gebruikt om een idee te krijgen over hoe dit project aangepakt moest worden, namelijk applicatie node AN49187. Vervolgens is er verder gegaan met het uitwerken van het programma tot er een realistisch werkend signaal ontstond dat vergeleken kon worden met een werkend signaal. Om het programma te tweaken en een eindresultaat te bereiken.

# Materiaal en methode

Als primaire bron van het signaal wordt er gebruik gemaakt van een Cypress PSOC 4 kit (CY8CKIT-042). Deze is vervolgens aangesloten aan een SN75176 dit is een “differential bus tranceiver”, deze produceert twee gebalanceerde signalen van het signaal dat door de psoc wordt uit gestuurd. Deze worden vervolgens met een 3-pins XLR-connector verbonden zodoende dat deze aangesloten zou kunnen worden aan een LED par etc. Er werd ook nog gebruik gemaakt van een DMX-controller om een perfecte versie van een signaal te simuleren en van een LED par om het werk te controleren.

Om een duidelijk beeld te krijgen van het DMX-protocol zal dit uitgebreid uitgelegd worden in deze paper. Ook zal de methode waarop het signaal gegeneerd tracht te worden besproken worden. Zowel met UART als met SPI.

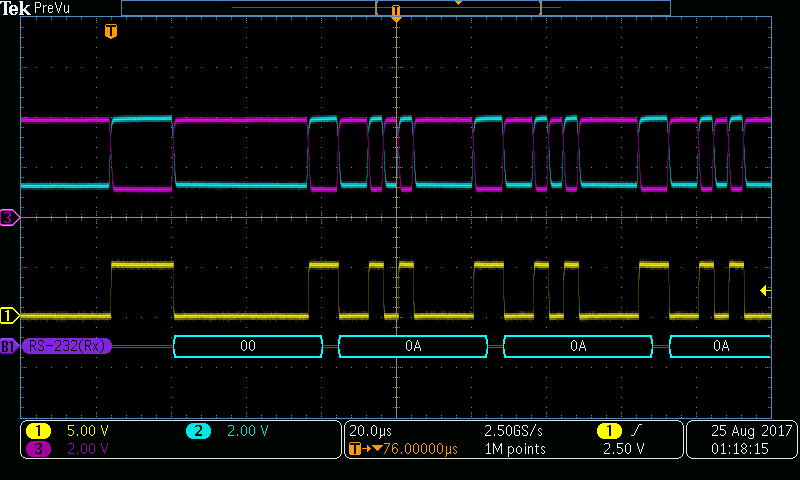
## DMX-protocol

Het DMX-protocol is een protocol dat veel gebruikt wordt in de podium/ studio technieken. DMX staat voor “Digital Multiplexing” en dat geeft meteen ook weer een idee van de werking van dit protocol. DMX berust op het principe van RS-485 omdat dit over grote afstanden gebruikt moet kunnen worden.

Dit protocol zendt frames van maximaal 513 bytes, elke byte is een kanaal. Dat wil zeggen dat bij een enkel DMX-universum 512 kanalen hebben om data over te sturen, 512 omdat de eerste byte niet gebruikt wordt. In elke byte kan logischerwijs een waarde opgeslagen worden van 0 tot 255. Die waarde geeft bijvoorbeeld aan op welke lichtsterkte een bepaalde kleur moet oplichten. Met een DMX kunnen dus 512 kleuren lampen dimmen in stappen van 0 tot 255, of 170 RGB-lampen waarbij elke lamp 3 kanalen nodig heeft om elke kleur apart aan te sturen.

## RS-485 in het kort

RS-485 maakt gebruik van 3 draden, twee waarop een gebalanceerd signaal wordt gezonden en een referentie punt (0V). De twee data draden worden vaak aangeduid met A en B. Op A wordt het geïnverteerde signaal gezonden en op B het niet geïnverteerde signaal. Omdat de meeste microcontroller single-ended logic levels hebben wordt er gebruik gemaakt van een RS-485 tranceiver. Bij dit project is dat de SN75176 differential bus tranceiver. In Figuur 1 is te zien hoe het DMX-signaal eruit moet zien. De gele lijn geeft het signaal van de controller weer, de blauwe lijn het signaal van de A-lijn en de paarse het signaal van de B-lijn.



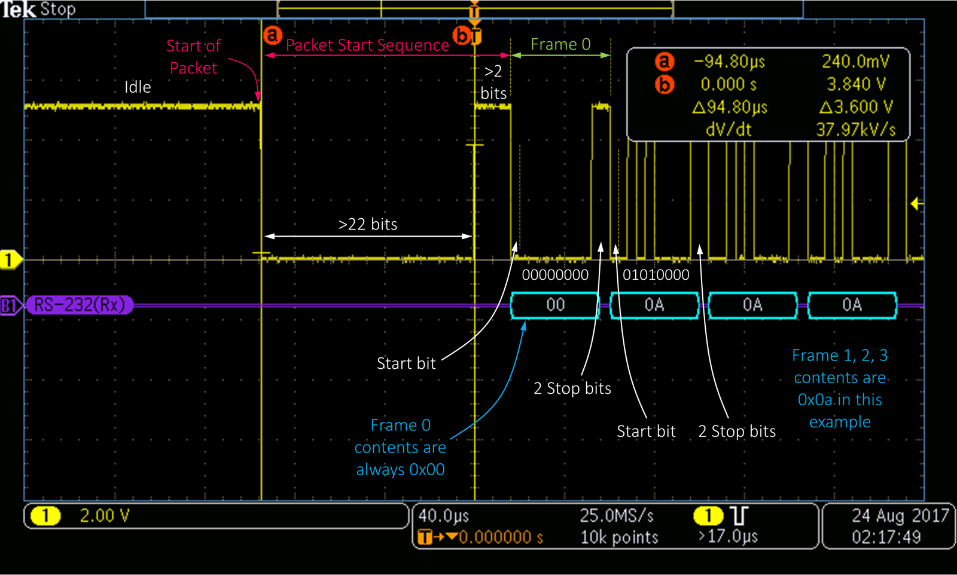
Figuur : DMX-signaal op scoop beeld.

## Seriële communicatie

De data dat in een pakket gestuurd wordt bestaat uit 8 bits opgesloten in start en stop bits. Het start bit bestaat uit een enkele bit periode op 0, het stop bit bestaat ui twee bit periodes op een logische 1. De in totaal 11 bits worden samengevat als een frame. De bits worden verzonden met een snelheid van 250000 bit per seconden en dat is gelijk aan 250000 baud. Dit houdt dus in dat een bit 4 microseconden lang duurt.

## DMX in detail

Zoals eerder vermeld bestaat een DMX-pakket dus uit 513 frames. De periode tussen elk pakket is de pauze status en is standaard een logische 1 op de A-lijn, 0 op de B-lijn en het signaal uit de microcontroller een logische 1. De tijd hangt af van apparaat en toepassing van het protocol. Om een slave te laten weten wanneer een pakket start en eindigt wordt er gebruik gemaakt van een start en stop sequentie. De start procedure bij DMX gaat als volgt. Omdat het B-signaal uit de RS-485 tranceiver een inversie is van het A-signaal en het signaal van de uitgang van de microcontroller zal alleen het A-signaal besproken worden. Bij de start van het pakket zal het signaal van “pauze” naar actief gaan door het signaal logisch 0 te maken voor minsten 22 bit periodes   
(88 microseconden) gevolgd door een logische 1 van minstens 2 bit periodes   
(8 microseconden). Na deze start sequentie volgt altijd een leeg frame. De stop sequentie is niet meer dan dat na het laatste frame het signaal op de A-lijn teruggaat naar “pauze”. In Figuur 2 is dit duidelijk weergegeven.



Figuur : scoopbeeld met getailleerde beschrijving van DMX-protocol

De volgorde van het zenden van informatie is LSB naar MSB. Dat houdt in dat het laagste bit als eerst verzonden zal worden en als laatst het hoogste bit.

## Methode met UART

Bij de start van dit project was het de bedoeling om het UART-blok van de PSOC 4 te gebruiken om het DMX-signaal te generen. Dit omdat het sturen van data op bijna gelijkaardige manier gebeurt. Helaas werd erna lang zoeken geconstateerd dat dit uiteindelijk toch niet mogelijk was omdat de start procedure van DMX niet overeenkomt met die van UART. Uiteraard is er ook nog geprobeerd om deze na te bootsen maar dit werkte helaas niet.

Omdat de data op gelijkaardige manier verstuurd werd was het eigenlijk alleen zaak dat de start en stop procedure toegevoegd werd aan deze communicatiemethode. Dit was alleen mogelijk als de communicatie pin overschreven kon worden in het programma. Dit bleek niet te gaan bij het UART-blok. Daarom is er na overleg overgestapt op het SPI-protocol.

## Methode met SPI

De rede dat er als tweede optie voor SPI is gekozen, is dat het SPI-blok van PSOC bijna volledig naar eigen wil geconfigureerd kan worden. Omdat bij normaal gebruik van SPI geen start sequentie benodigd is zal deze buiten het blok om gecreëerd moeten worden. Dit wordt gedaan door de drivemode van de TX-pin in het programma zo aan te passen dat de start telkens gebeurt buiten het SPI-blok om. De start sequentie is verkregen door middel van delay en het hoog of laag schrijven van de TX pin. Na dat dat verkregen was moest de data in het juiste formaat door gestuurd worden om het voor de LED par leesbaar te maken. Zoals eerder vermeld bestaat een frame uit 11 bits en zullen die ook verzonden moeten worden. Aangezien dit geen standaard is zullen de bits per 16 doorgestuurd moeten worden als data type uint16. Door het combineren van frames zijn de 2 bytes gevuld. Bijvoorbeeld bij de volgende frames 0000000011, 01101101011 en 00101101111. Om met deze frames uint16 te vullen, worden deze achter elkaar gezet dus zal het er als volgt uit komen te zien 0000000001101101 en 01100101101111xx.

Nadat alle data juist was verzonden, werd er een bij de led par geconstateerd dat de doorgestuurde data ook overeenkwam met de kleur die waar te nemen was op de lamp. Echter viel ook op dat er andere kleuren doorheen knipperde. Dit is helaas tot op heden niet opgelost.

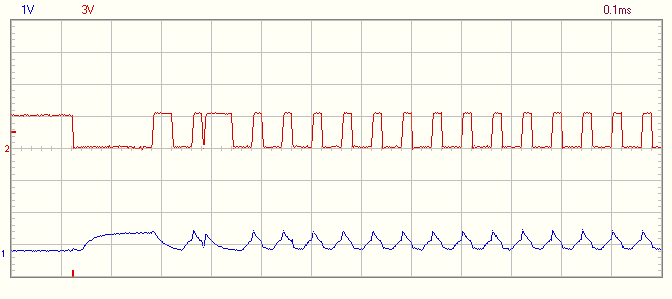
# Resultaten

Als resultaat bleek dat het generen van een start sequentie nogal moeilijk ging met het UART-protocol zonder de bestaande bibliotheken aan te moeten passen, wat een grote hoeveelheid extra werk zou opleveren dus is er met inspraak van de professor gekozen voor het gebruik van het SPI-protocol in de plaats van het UART-protocol. Dit gaf een aantal nieuwe uitdagingen gezien er bij het SPI-protocol meer zelf gegenereerd moest worden in vergelijking met het vorige protocol. Uiteindelijk werd er een redelijk correct signaal verkregen dat voldeed aan de procedures van een DMX-signaal, vervolgens is de volgende stap ondernomen om aan de hand van een DMX-controller en een oscilloscoop het signaal dat gegenereerd werd door de Psoc en het signaal dat gegenereerd werk door de DMX-controller met elkaar te vergelijken. Er werden vervolgens een aantal timings-problemen vastgesteld deze zijn tot zover mogelijk opgelost.

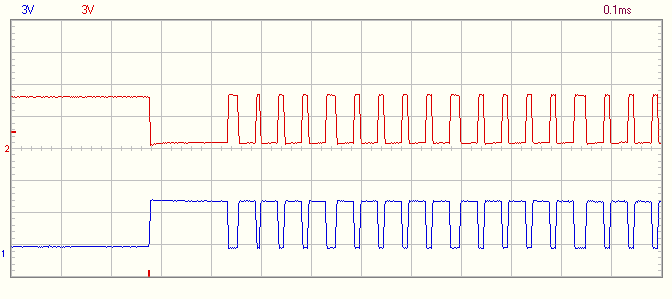
# Discussie

Omdat het Project uiteindelijk niet 100% werkend is door een gebrek aan tijd wordt er vastgesteld dat dit geen voorbeeld zou moeten zijn voor opvolgende projecten, losstaand van de verworven informatie deze is namelijk wel functioneel om een inzicht te krijgen in de werking van het protocol en de aanpak methode om het werkend te krijgen aan de hand van de CYPRESS PSOC 4 kit.

Het resultaat kunnen we best bespreken aan de hand van de volgende foto’s (Figuur 3 en Figuur 4)



Figuur Output SN75176 (met kanaal 1 geactiveerd op 255)



Figuur Output DMX controller (met geen kanalen geactiveerd)

Het grootste probleem dat er zichtbaar is dat de niet geïnverteerde uitgang van de SN75176 chip. In figuur 2 ziet men hoe een ideale uitgang er uit ziet en dus kunnen we zien dat de niet-geïnverteerde uitgang van de chip een probleem is. Dit kon helaas niet worden oplossen door een gebrek aan tijd en materiaal. Nu is er een vermoeden dat dit niet het enigste probleem is. Maar dit is voorlopig het eerste probleem dat we kunnen vaststellen aan de hand van data. Tegenover het vermoeden dat er nog een fout zit rond de timing van het signaal maar is tot op heden nog niet vastgelegd op beeld. En blijft dus een vermoeden.

In de toekomst wordt er getracht dit probleem nog te kunnen oplossen door eerst de chip te vervangen en vervolgens verder te debuggen om een perfect werkend signaal te krijgen.

# Conclusie

Zoals voorheen gezegd is, is het advies om de chip een te vervangen of ten miste een stress-test er op uit te voeren en te controleren dat dit niet een eenmalig probleem is veroorzaakt door de huidige connecties. Ook moet, nadat dit gebeurd is de timing van het signaal gecontroleerd worden door de twee signalen boven elkaar te zetten met een logic analyzer.

Door de problemen die er bij dit project zijn opgedoken was er helaas geen tijd meer om meer functionaliteit aan het project te geven. Er wordt getracht in de toekomst dit project af te kunnen ronden op de manier zoals oorspronkelijk bedoeld.

# Referentielijst

Cypress Community. (2010, Januari 13). *UART to DMX Bridge - AN49187*. Opgehaald van Cypress: https://www.cypress.com/documentation/reference-designs/uart-dmx-bridge

Hareendran, T. (/, / /). *An Introduction to the Digital Multiplex Protocol*. Opgehaald van Electro Schematics: https://www.electroschematics.com/an-introduction-to-the-digital-multiplex-protocol/

shabaz. (2017, Augustus 24). *DMX Explained, DMX512 and RS-485 Protocol Detail for Lighting Applications*. Retrieved from element 14: https://www.element14.com/community/groups/open-source-hardware/blog/2017/08/24/dmx-explained-dmx512-and-rs-485-protocol-detail-for-lighting-applications

tdvdesign. (2014, oktober 14). *Functional DMX Transmitter and Receiver Software*. Opgehaald van Cypress: https://www.cypress.com/comment/220886

Wikipedia Community. (2019, oktober 7). *DMX512*. Opgehaald van Wikipedia: https://en.wikipedia.org/wiki/DMX512

# Bijlagens