

Aansturen van een modulair free-form LED display

Tom Puttemans

Promotor: prof. dr. Jan Doutreloigne, prof. dr. Herbert De Smet
Begeleider: Pieter Bauwens

Afstudeerwerk ingediend tot het behalen van de academische graad van
Master in de ingenieurswetenschappen: elektrotechniek

Vakgroep Elektronica en informatiesystemen
Voorzitter: prof. dr. ir. Jan Van Campenhout
Faculteit Ingenieurswetenschappen
Academiejaar 2008-2009



Voorwoord

Allereerst wil ik verschillende mensen bedanken. Om te beginnen zijn dat natuurlijk de promotoren, Pieter Bauwens, Jan Doutreloigne en Herbert De Smet voor het mogelijk maken en begeleiden van dit eindwerk. Hun bijdrage heeft een uitstekende basis gelegd voor het behaalde resultaat. Verder bedank ik de mensen die hebben geholpen voor het beter leesbaar maken van de meeste teksten in dit eindwerk.

Toelating tot bruikleen

“De auteur geeft de toelating deze masterproef voor consultatie beschikbaar te stellen en delen van de masterproef te kopiëren voor persoonlijk gebruik. Elk ander gebruik valt onder de beperkingen van het auteursrecht, in het bijzonder met betrekking tot de verplichting de bron uitdrukkelijk te vermelden bij het aanhalen van resultaten uit deze masterproef.”

Driver for a free-form LED display

Tom Puttemans

Supervisor(s): Pieter Bauwens, Jan Doutreloigne, Herbert De Smet

Abstract—

A modular free-form LED display is a display that's divided into different parts (displaymodule with a LED display) [1]. This also means that each module acts as an individual display. With this technique to create a display, a lot of displayshapes can be formed, even not rectangular shapes. It's also possible to change the shape of the display during the use of it. Therefore there has to be a system that automatically detects the shape of the display and visualize it for the user. In that way the user can insert a picture that has to be shown on the display. This article provides the basics of such a system and some testresults.

Keywords— modular display, networkprotocol, autodetection, display-driver

I. INTRODUCTION

BY the great use of displays in a lot of applications (TV, car, GSM), there's a need of displays with different sizes and forms. But it has always a rectangular shape. Some applications needs a display in another shape than a rectangle. This can be accomplished by using basic displays and connect them in such a way that a specific shape is formed. It's also usefull that the shape of the display can change during the use of it, depending on the needs of the application.

In this abstract, we will first propose the structure of a system that makes it possible to detect the shape of a display and shows a picture inserted by the user. The task and operation mode of each component will be discussed. After positive simulations, the system is implemented on existing hardware modules. Some results and remarks are given in the conclusion.

II. THE SYSTEM

The entire system consists of three parts (see figure 1). First there's a chain of displaymodules. A displaymodule consists of harware to let the display work. Each module can be seen as a node in a network. Owing to this, a networkprotocol makes it possible that the different nodes can communicate with each other without causing problems. Each displaymodule has a display which can only work with a specific driver. This driver together with the networkprotocol is called the modulair display driver. The modular display driver is installed in a FPGA (Field Programmable Gate Array) on the displaymodule. Secondly there's the interface. The interface forms the connection between the network of displaymodules and the PC. The third part is the GUI (Graphical User Interface), which is installed on a PC. The GUI makes it possible for the user to insert data en parameter information like the dimensions of the display and the refreshrate. The data information inserted in the GUI has to be displayed on the entire display.

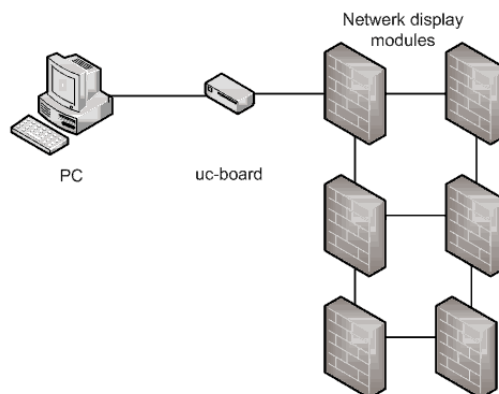


Fig. 1. Structure of the proposed system

A. The displaymodule

A displaymodule consists of a display that has to show the data. There's also an FPGA that execute the networkprotocol and the driver for the display. Those two forms the modulair display driver. For simplicity we call the modulair display driver in the future the driver. A displaymodule has also important supply hardware. This generates the correct voltages for the different integrated circuits on the displaymodule. There's also programming hardware for the FPGA.

The networkprotocol uses messages that the displaymodules send to each other. Each message starts with a startcondition followed by the command bits. The startcondition is a falling edge. After the startcondition follows three command bits. Those bits specify the type of the message. In figure 2 you can find a list of possible commands. The commands can be divided in three different categories. First we have the commands to get an identification (Adressasked and Adress_in). Secondly there are some commands to detect the presence of the displaymodules (Presencequestion, Networkconfirmation and Confirmation). The last categorie of commands are used to drive the display (Data and Param).

Commando	Binary value
Module_away	000
Param	001
Data	010
Adress_in	011
Adressasked	100
Presencequestion	101
Networkconfirmation	110
Confirmation	111

Fig. 2. Overview of the used commands

A part of the driver implemented on the FPGA consist of the networkprotocol. After the displaymodule is booted, a local Power-on-reset (POR) (see figure 3) is generated and the module wants an identification. The identification is an address that specifies the location in the network (row and column of a matrix). This identification is important for the system to know which displaymodule is present and makes it also possible to send specific data to a displaymodule. Figure 3 shows how displaymodules gets an address. After sending an address question message to adjacent modules, an address is received on data_input_0. This address is saved in a register and is the identification until the modules falls away. When another adjacent module asks an address the modules send an adres when it's got an own address. The numbers 0 to 4 mentioned in the simulation are the in- and outputs on the four edges of the displaymodule

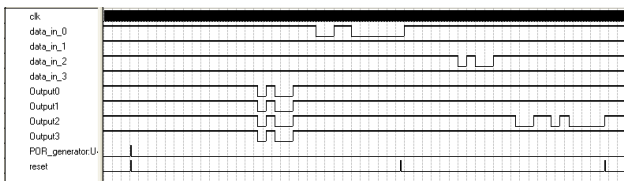


Fig. 3. The identification principle

The system wants to know which modules are present. This is accomplished by a periodical message sended by the interface. The interface sends a presence question into the network. When the displaymodule receives such a question, it starts sending periodically a networkconfirmation message. This networkconfirmation message comes on the output consistent with the input that received the presence question message. The networkconfirmation message follows a path in the network right to the interface. When the interface receives the networkconfirmation message, it seds a confirmation message back with the same address as the received networkconfirmation message. This confirmation message is ment for a specific module and signals that the displaymodule registered. It also stops the periodically sending of the network confirmation messages by the specific displaymodule. The interface seds also data and parmeter messages into the network.

When a adjacent displaymodule falls away, this is detected by the Module.away command. In fact it's not really a command but it just detects a falling edge due the voltage drop on the output pin of the vanished displaymodule.

B. The interface

The interface is the link between the network and the user (PC). The code of the interface is executed by a microcontroller. The testboard has a 8051- compatible microcontroller [2]. The interface communicates also with the GUI. This communication passes via a serial connection. The received data and parameter information is stored in the RAM-memory. Information about the presence of the displaymodules is sended to the GUI. The interface communicates with the network via an in- and ouput pin. It gives an address to the adjacent displaymodule after an incoming adres question message. This starts the whole identification principle. Periodically a presence question mes-

sage is send to the network. When a networkconfirmation message is received, the address is stored and the interface seds a regular confirmation message back.

C. The GUI

The GUI offers the possibility for the user to easily control the entire system. The user can enter some settings like the refreshrate and dimensions of a basic display. The shape of the display is also visualized on the computer screen. This makes it possible for the user to enter an image that has to appear on the display. To visualize the display, the GUI communicates with the interface via a serial interface. The settings (data and parameters) inserted by the user are send to the interface and the interface seds the information about the presence of diplaymodules to the GUI.

III. CONCLUSION

The simulation results shows the wanted behaviour of the displaymodules. It's also possible to simulate an entire network, so several networkshapes where simulated with good results. There are 6 address bits so 64 displaymodules can have an identification. The driver and interface code is implemented and tested on the existing hardware boards. Due to noise, incoming messages where detected on unconnected entries. This problem can easily be solved by placing extra pull-down resistors on the entries. Now the changing shape of a displaynetwork is automatically detected and visualized for the user. By the GUI, the user can draw figures that are shown on the display. It's not possible for the interface to detect when the displaymodule connected to the interface falls away. The use of another microcontroller or hardware can be a solution and a issue for further research.

REFERENCES

- [1] P.Bauwens J. Doutreloigne A.Monté *A Driver for Modular Passive-Matrix Displays*, Ghent Univ. ELINTEC/TFCG Microsystems
- [2] *Ultra-High-Speed Flash Microcontroller Users Guide*, Rev: 6 12/04, Maxim integrated products, 2004 USA/

Inhoudsopgave

Voorwoord	i
Toelating tot bruikleen	ii
Extended abstract	iii
Inhoudsopgave	v
Gebruikte afkortingen	vii
1 Inleiding	1
2 De modulaire display driver	5
2.1 Netwerk functionaliteit	7
2.1.1 De ingangscontrole	8
2.1.2 Commando onderzoek	10
2.1.3 Verkrijgen van een adres	11
2.1.4 Aanwezigheidsopvraging	13

<i>INHOUDSOPGAVE</i>	vi
2.1.5 Parameter en data berichten ontvangen	15
2.1.6 Buur weg detectie	16
2.1.7 Uitgangscontrole	16
2.2 Display aansturing	16
3 De interface	19
3.1 Het hoofdprogramma	22
3.2 Seriële poort interrupt service routine	26
3.3 Externe interrupt service routine	27
4 De grafische user-interface	30
5 Conclusie	34
bibliografie	37
Lijst van figuren	38
Lijst van tabellen	40

Gebruikte afkortingen

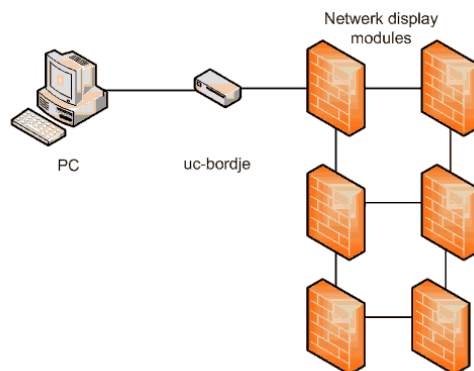
PC	Personal Computer
LED	Light Emitting Diode
FPGA	Field-programmable gate array
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
IDE	Integrated Development Environment
GUI	Grafische User Interface
USB	Universal Serial Bus
MSB	Most significant bit
LSB	Least Significant Bit
POR	Power On Reset
ISR	Interrupt Service Routine

Inleiding

Door het frequent gebruik van displays in elektronische apparaten, is het interessant om de vorm van een display te kunnen aanpassen aan zijn gebruik. De functionaliteit kan ook veranderen tijdens de werking zodat een andere displayvorm vereist is. Door het aaneenschakelen van verschillende basisdisplays wordt een groter display gevormd. Elk basisdisplay gedraagt zich als een apart display. De manier waarop de verschillende basisdisplays aaneengeschakeld worden bepaald de vorm van het totale display. Het wordt dus mogelijk een display te vormen voor toepassingen waarbij een standaard rechthoekig display niet geschikt is (free-form display).

Er bestaat reeds een systeem waarbij de gebruiker zelf de vorm van het totale display ingeeft in een grafische user-interface (GUI) die uitgevoerd wordt op een PC. De gebruiker geeft de dimensies (aantal rijen en kolommen van basisdisplays) in van het display. Dit heeft als beperking dat enkel rechthoekige displays gevormd kunnen worden. Ook is er geen zekerheid dat de basismodules effectief aanwezig zijn. De (rechthoekige) vorm van het display verschijnt in de grafische user-interface. Door pixels aan te klikken in de GUI wordt een beeld gevormd dat op het werkelijke display dient te verschijnen. Verder geeft de gebruiker ook nog parameters in die noodzakelijk zijn voor de goede werking van de basisdisplays. In het huidige systeem kunnen de displaymodules enkel bij de opstart een identificatie verkrijgen. Door de identificatie kunnen berichten die specifiek voor een display bedoeld zijn, verstuurd worden. Nadat een identificatie is verkregen, kunnen enkel berichten ontvangen worden op de ingang waarop de identificatie werd verkregen. Dit geeft problemen als er modules wegvallen.

De communicatie tussen de GUI en de basisdisplays gebeurt via berichten. Eerst wordt via een seriële link de nodige gegevens (data of parameters) vanuit de GUI verzonden naar een interface. Deze interface zet de gegevens om naar berichten die op hun beurt naar de verschillende basisdisplays verstuurd worden, ook al zijn deze niet aanwezig. De configuratie is weergegeven in figuur 1.1. Het geheel van een basisdisplay en extra hardware, wordt een displaymodule genoemd.



Figuur 1.1: Voorbeeld netwerk

Door elke aanwezige displaymodule als een node in een netwerk te beschouwen, kan een netwerkprotocol opgesteld worden. Via dit netwerkprotocol worden extra functionaliteiten toegevoegd die de beperkingen van het bestaande systeem wegwerken. Het bestaande systeem vormt de basis voor het netwerkprotocol en de aansturing van de displays. Het wordt uitgebreid met een systeem dat ervoor zorgt dat de displaymodules op ieder moment een identificatie verkrijgen. Door de identificatie wordt de displaymodule in het netwerk opgenomen en is deze bereikbaar om specifieke berichten naartoe te zenden. De identificatie wordt verder ook gebruikt om de aanwezigheid van een displaymodule te kunnen bepalen zodat de vorm van het display automatisch wordt gedetecteerd. Op deze manier wordt iedere vormverandering van het display gevisualiseerd voor de gebruiker. Iedere ingang kan berichten ontvangen en verwerken. Bij het verdwijnen van een naburige module worden nog steeds berichten ontvangen en verwerkt via een andere ingang.

Via verschillende types van berichten kunnen de displaymodules met elkaar communiceren. Het systeem wordt controleerbaar gehouden doordat de displaymodules niet periodiek hun aanwezigheid dienen te bevestigen, maar er wordt periodiek aan het netwerk gevraagd welke modules aanwezig zijn. De displaymodules reageren op deze aanvraag door hun aanwezigheid te bevestigen.

Om een werkend geheel te bekomen moet eerst een netwerkprotocol uitgewerkt worden. Als hulp bij de uitwerking van het protocol werd gebruik gemaakt van Omnet++ software. Deze software is ontwikkeld voor de simulatie van communicatienetwerken en hardware-architecturen. Het gedrag van de nodes wordt geprogrammeerd in C++. De uitwisseling van de berichten tussen de verschillende nodes wordt gevisualiseerd zodat fouten op eenvoudige wijze gedetecteerd worden.

Het gedrag van de nodes, de netwerkfunctionaliteit en de aansturing van het display, beschreven in C++, moet hardwarematig worden geïmplementeerd. Hier is het voordeel dat de netwerkfunctionaliteit en aansturing parallel kunnen uitgevoerd worden. De implementatie gebeurt in een FPGA via VHDL code. Het geheel van de code wordt in het verder verloop de driver genoemd. Een FPGA is een geïntegreerde schakeling met programmeerbare logische componenten. Hierin worden logische functies zoals AND en OR functies geprogrammeerd. Het geheel van deze functies vormt een bepaald gedrag. De VHDL programmeertaal wordt gebruikt om programmeerbare logica in FPGA's te beschrijven en te modelleren. Als IDE-software wordt Active-HDL gebruikt. Active-HDL is een tool om digitale schakelingen te simuleren en te testen. Nadat de VHDL-code geschreven is, dient de code geanalyseerd en gesynthetiseerd te worden. Deze voert een synthese uit die de logische functies minimaliseert en voert ook een technologie mapping uit, zodat het design wordt afgebeeld op de logische elementen van de FPGA. Nadat dit gebeurd is, moet het geheel terug gesimuleerd worden om te onderzoeken of het design na de mapping nog steeds goed functioneert. De analyse en synthese wordt uitgevoerd met de Quartus II software.

Het gedrag van de interface wordt geregeld via een microcontroller. De microcontroller voert de instructies seriëel uit. Er dient een seriële connectie opgezet te worden voor de communicatie met een PC. Verder is er een in- en uitgang nodig voor de communicatie met het netwerk. De C-programmeertaal beschrijft het gedrag. De ontwikkeling van software gebeurt via de Keil software. Deze bevat een C51 compiler die de C-code omzet naar code die uitvoerbaar is door een microcontroller met een 8051 core. De gebruikte software biedt ook goede debug mogelijkheden zoals het monitoren van de poorten via een Logic Analyzer en het bekijken van de inhoud van bepaalde registers.

Het systeem wordt controleerbaar gemaakt voor de gebruiker via een grafische user interface (GUI). Deze wordt uitgevoerd op de PC. Hier wordt de vorm van de displays op grafische wijze voorgesteld zodat de gebruiker bepaalde pixels kan aanklikken om een bepaalde figuur op het display af te beelden. Deze data wordt vervolgens naar de betreffende basisdisplays verstuurd zodat de ingegeven vorm op de displays verschijnt. In de GUI worden ook

bepaalde parameters ingesteld die specifiek zijn voor de gebruikte display modules. Deze parameters zijn bijvoorbeeld de dimensies van een basisdisplay en de refreshrate. De Visual C++ ontwikkelomgeving laat toe om op een eenvoudige manier een grafische user interface te maken. De layout van de GUI kan grafisch opgebouwd worden en de functies bij de verschillende componenten worden dan via de C++ programmeertaal ingegeven.

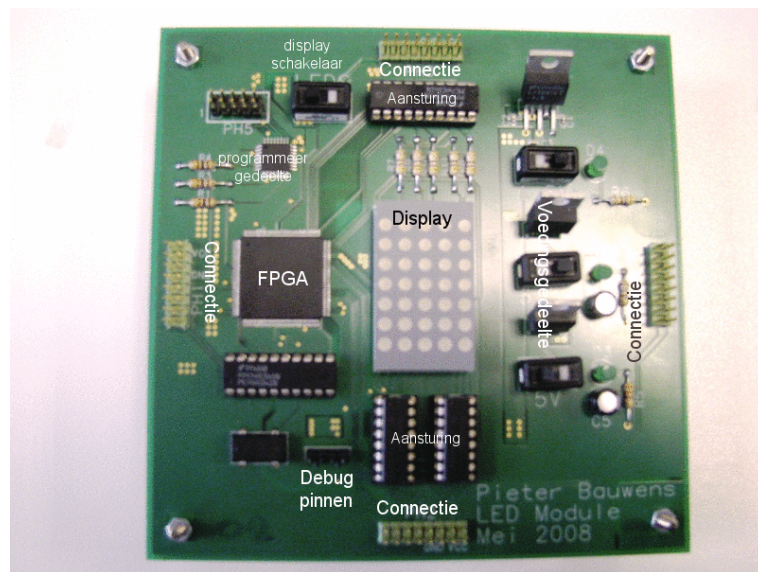
Nadat de verschillende delen van het systeem gemaakt en gesimuleerd zijn, wordt alles getest op de bestaande hardwarebordjes. Na de assemblage van de testbordjes wordt alles geprogrammeerd. De microcontroller bevat een bootloader waardoor deze eenvoudig via de seriële interface kan geprogrammeerd worden. De driver wordt in een configuratie chip geprogrammeerd. Telkens de displaymodule opstart, zorgt de configuratiechip (EPC2) dat de driver naar de FPGA wordt overgezet. Dit is nodig omdat de gebruikte FPGA een SRAM-gebaseerde chip is en de configuratie wegvalt als de spanning wegvalt. De EPC2 wordt geprogrammeerd via de ByteBlaster II die gebruik maakt van de parallelle poort van de PC.

De modulaire display driver

Een displaymodule bevat een display waarop de data afgebeeld zal worden. Vervolgens is er het FPGA gedeelte waarin het netwerkprotocol en de aansturing van het display (de modulaire display driver of kortweg driver) in opgeslaan wordt. Er is ook een belangrijk voedingsgedeelte die de spanningen voor de verschillende componenten genereert. Ook is hardware aanwezig voor de programmatie van de FPGA. Een voorbeeld van een displaymodule is weergegeven in figuur 2.1. Het display is een 7x5 LED matrix. Er worden drie verschillende spanningen (5V, 3.3V, 1.5V) gegenereerd en er is een Cyclone-FPGA van Altera aanwezig.

Het netwerkprotocol maakt gebruik van berichten die de modules naar elkaar versturen, elk bericht begint met een startconditie gevolgd door een bepaald commando die de aard van het bericht bepaald. De mogelijke commando's zijn weergegeven in tabel 2.1. De commando's worden in drie verschillende categoriën opgedeeld, eerst zijn er commando's die voor de adressering zorgen. Deze zorgen ervoor dat iedere module een geldig adres krijgt zodat er adres specifieke berichten naartoe kunnen verstuurd worden (Adres_in en Adresvraag). Er zijn de commando's die nodig zijn voor de aansturing van het display, deze verzorgen het ontvangen van data en parameterinformatie (Data en Param). De derde categorie zijn de commando's om de aanwezigheid van modules te detecteren (Aanwezigheidsvraag, Netwerkbevestiging en Bevestiging).

De displaymodules bevatten een 20 MHz klok, dit is de frequentie alle simulaties gebeuren tijdens de ontwikkeling. De bits van een commando duren echter 1 μs (1MHz) hiervoor wordt een kloksignaal (bitklok) afgeleid van het ingangskloksignaal. In de opstartfase wordt

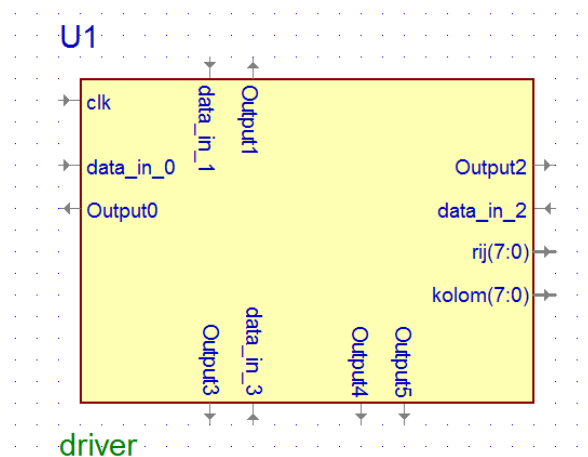


Figuur 2.1: Voorbeeld display module

na $3 \mu s$ een locale Power On Reset (POR) gegenereerd waardoor alle registers juist worden ingesteld en de uitgangssignalen hoog komen te staan. Het hoog komen van de uitgangen wijst erop dat de module actief is. Het begin van de berichten wordt gedetecteerd aan de hand van een dalende flank. Het laag komen van een uitgang wijst op de startconditie van een bericht dat verzonden wordt. In figuur 2.2 is een schema gegeven van de driver. Hierop zijn de in- en uitgangen van de FPGA weergegeven. In- en uitgang 0 tot 4 zijn verbonden met in- en uitgangen van het bordje. Rij [0-7] en Kolom [0-7] zorgen voor de aansturing van het display. Uitgang 4 en 5 zijn verbonden met pinnen op het bord. Deze pinnen zijn extra pinnen voor te debuggen en hebben verder geen functie. In de simulaties worden in- en uitgangen 0 tot 4 getoond. De ingang voor het kloksignaal is ook telkens weergegeven.

Commando	Binaire waarde
Module_weg	000
Param	001
Data	010
Adres_in	011
Adresvraag	100
Aanwezigheidsvraag	101
Netwerkbevestiging	110
Bevestiging	111

Tabel 2.1: Gebruikte commando's



Figuur 2.2: Layout driver

2.1 Netwerk functionaliteit

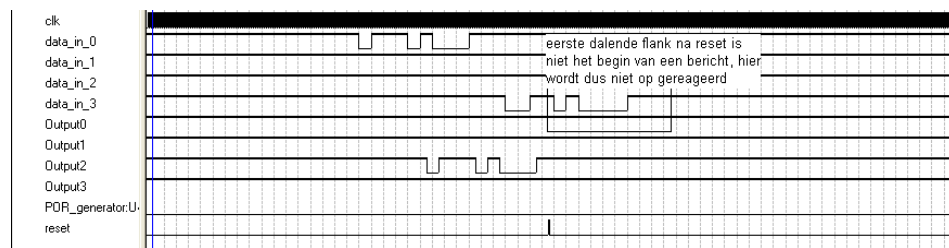
Een deel van de driver bestaat uit het netwerkprotocol. Na de opstart (POR) zorgt het protocol via een adresvraag dat de module een adres verkrijgt. De gebruiker wil weten welke modules aanwezig zijn, hiervoor vraagt de interface periodiek aan het netwerk welke modules aanwezig zijn. Dit gebeurt via een aanwezigheidsvraag. Als een displaymodule een aanwezigheidsvraag ontvangt, start de displaymodule met het periodiek uitzenden van een netwerkbevestiging. Deze netwerkbevestiging beweegt doorheen het netwerk tot ze

tenslotte bij de interface terechtkomt. De interface stuurt telkens een bevestiging terug naar de module zodat de module de periodieke uitzending van de netwerkbevestiging kan stoppen. Gedurende een bepaalde tijd verwacht de interface netwerkbevestigingen. Per aanwezigheidsvraag houdt de interface bij van welke modules een netwerkbevestiging werd ontvangen. Verder zal de interface data en parameterinformatie naar de displaymodules versturen.

2.1.1 De ingangscntrole

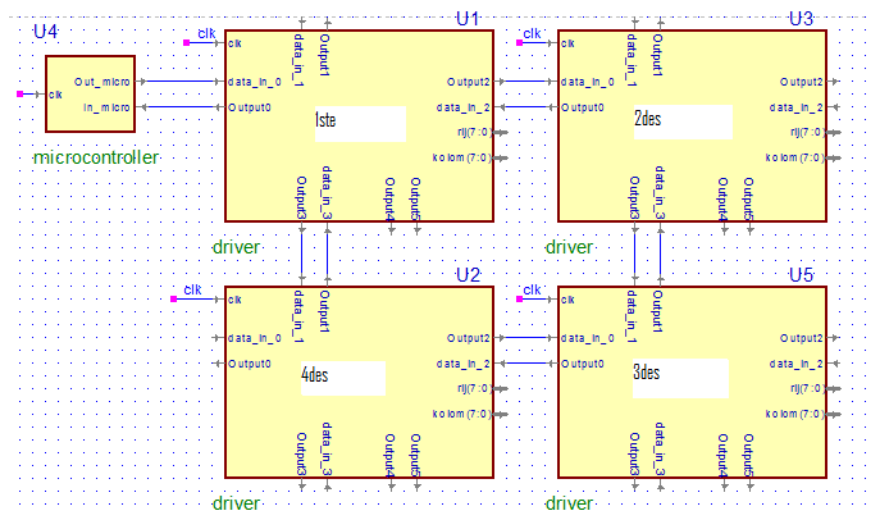
De ingangscntrole zorgt voor een correcte detectie van de start van een bericht. Elke module heeft vier in- en uitgangen (zie figuur 2.2). De start van een bericht op een bepaalde ingang wordt gedetecteerd aan de hand van de overgang van een hoog naar laag niveau op de ingang. De daarop volgende bits vormen het commando en afhankelijk van het bericht ook gegevens. Het is dus belangrijk dat gedurende een bericht dezelfde ingang bekeken wordt en er niet gewisseld wordt bij een dalende flank op een andere ingang. Er wordt niet overgegaan op een andere ingang tot er een reset is opgetreden. Een reset treedt op nadat een bericht is afgehandeld en de module klaar is voor een nieuw bericht. De reset zet alle instellingen terug zoals de begintoestand, dit zorgt ervoor dat er opnieuw een bericht kan ontvangen worden op elke ingang. Het is nodig van telkens opnieuw alle ingangen te bekijken want er kan een module zijn weggevallen zodat de berichten via een andere weg door het netwerk moeten komen. Met de huidige configuratie is het niet mogelijk om de verschillende ingangen parallel te observeren en te verwerken omdat de er dan problemen optreden bij gebruik van de verschillende sources. De verschillende ingangen hebben de dezelfde uitgangen nodig voor het doorzenden van de berichten en deze kunnen slechts serieel berichten doorzenden.

Bij de ontvangst van een bericht worden de andere ingangen wel geobeserveerd. Er wordt bijgehouden welke ingangen bezig zijn met het ontvangen van berichten en welke niet. Zo wordt vermeden dat een dalende flank midden in een bericht op een bepaalde ingang wordt aanzien als de start van een bericht indien er juist voor de dalende flank een reset is opgetreden na het einde van een bericht op een andere ingang. Deze situatie is weergegeven in figuur 2.3.



Figuur 2.3: Door bij te houden welke ingangen bezig zijn met het ontvangen van berichten, kan de start van en bericht eenduidig bepaald worden

De ingangscntrole houdt ook bij welke naburige modules aanwezig zijn. Dit gebeurt op basis van bepaalde commando's, nl. het adresvraag en aanwezigheidsvraag commando. De detectie gebeurt bij observatie dus ook als het binnenkomende bericht niet verwerkt wordt. Als de module wegvalt wordt dit ook bijgehouden. Dit controlesysteem is nodig want de modules kunnen op willekeurig tijdstip opstarten. Indien ze opstarten midden een bericht en er treedt een dalende flank op in het bericht, zal dit aanzien worden als het begin van een bericht wat fouten met zich meebrengt. Nadat een aanwezigheidsvraag of adresvraag wordt ontvangen van een naburige module zal het volgende uitgezonden bericht ook verzonden worden naar de nieuwe naburige module. Hierdoor zal de nieuwe naburige module enkel volledige berichten ontvangen. Er wordt zowel naar adresvraag als aanwezigheidsvraag gekeken, omdat afhankelijk van de opstartsequentie het niet voldoende is om enkel te kijken naar het adresvraag commando. Indien de opstartsequentie volgens figuur 2.4 verloopt, zal de module 10 (rij 1 kolom 0) niet meer weten dat module 11 aanwezig is als module 01 wegvalt. Hierdoor zal module 11 geen berichten meer ontvangen als er enkel rekening wordt gehouden met het adresvraag commando bij de burendetectie.



Figuur 2.4: Opstartsequentie waarbij het adresvraag commando niet voldoende is als busrendetectie

Tenslotte zorgt de ingangscontrole er ook voor dat gedurende de periodieke uitzendingen (adresvraag en netwerkbevestiging), afkomstig van de module zelf, geen berichten kunnen ontvangen worden. Dit is nodig omdat een ontvangen bericht de uitgangen nodig heeft en deze niet beschikbaar zijn. Indien de module bezig is met een bericht te verwerken, en de timer van de periodiek uitzending loopt af, zal er gewacht worden met de periodieke uitzending tot het bericht verwerkt is en de uitgangen beschikbaar zijn.

2.1.2 Commando onderzoek

Nadat een dalende flank is gedetecteerd, worden de volgende drie bits onderzocht om te bepalen welk commando er ontvangen is. Nadat dit bepaald is, wordt het commando check signaal geactiveerd. Dit signaal zorgt ervoor dat het systeem weet wanneer het commando geldig is. Bij het commando onderzoek wordt er ook voor gezorgd dat herhalingen van berichten tegengehouden worden. Deze berichten zouden in het netwerk blijven hangen en voor overbelasting zorgen. Er is ingesteld dat een bepaald type commando slechts om de $207 \mu s$ opnieuw kan verzonden worden. De reden voor deze tijd wordt later duidelijk. De tegengehouden commando's zijn de bevestiging en parameters berichten. Data mag niet tegengehouden worden. Bij meerdere modules zal de interface de databerichten voor de verschillende modules kort na elkaar verzenden. Indien databerichten ook gedurende een bepaalde tijd worden tegengehouden zal enkel de eerste module data ontvangen maar nooit

doorgeven omdat de tijd nog niet verstreken is. Hierdoor zullen de andere displaymodules nooit hun data ontvangen.

2.1.3 Verkrijgen van een adres

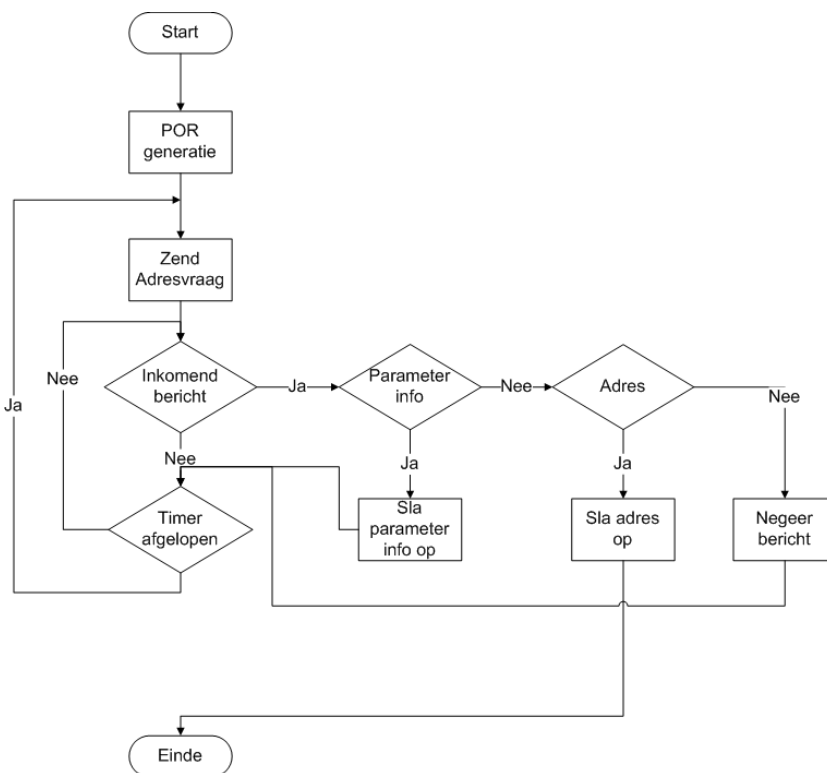
In figuur 2.7 wordt schematisch weergegeven hoe de displaymodule een adres verkrijgt. Na de POR wordt het Adresvraag commando (structuur zie figuur 2.5) uitgestuurd op de uitgangen, dit duurt tot de module een bepaald adres verkregen heeft. De tijd tussen de Adresvraag commando's wordt ingesteld via een timer (zie figuur 2.8). De aanvraag wordt op de vier uitgangen geplaatst omdat bij opstart nog niet geweten is welke naburige modules aanwezig zijn. Het adres is voor een module zeer belangrijk als identificatie in het netwerk, het wordt gebruikt als er data moet verstuurd worden en als identificatie als de aanwezigheid van de module wordt opgevraagd. Indien de module geen adres heeft worden andere commando's behalve de parameter informatie genegeerd, dit omdat de andere commando's adres specifiek zijn. Het verkregen adres is een binair getal van 6 bits waarin de rij en kolom verwerkt zit. De structuur is weergegeven in figuur 2.6. Na de startbit en commandobits ontvangen te hebben, wordt de LSB van de rij doorgestuurd, vervolgens de LSB van de kolom en dit tot het gehele adres is doorgestuurd. Er worden voor de rij en kolom 3 bits voorzien, waardoor er het netwerk maximaal 64 (8 rijen en 8 kolommen) modules kan bevatten, Een uitbreiding is wel mogelijk door meer adresbits te gebruiken. In figuur 2.9 wordt weergegeven hoe de display module een adres binnenkrijgt van een naburige module, het adres is 000000. Verder is ook weergegeven hoe de module een adresvraag behandelt. Dit kan enkel als de module zelf een adres heeft verkregen.

Startbit 0	Cmd bit 1	Cmd bit 0	Cmd bit 0
---------------	--------------	--------------	--------------

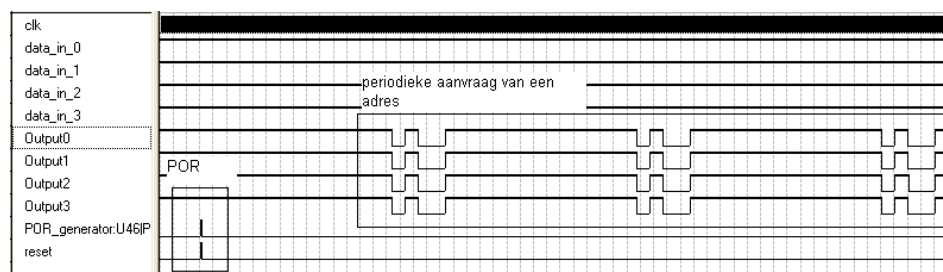
Figuur 2.5: Structuur van het Adresvraag bericht

Startbit 0	Cmd bit 0	Cmd bit 1	Cmd bit 1	Adres bit LSB Rij	Adres bit LSB Kolom	Adres bit	Adres bit	Adres bit MSB Rij	Adres bit MSB Kolom
---------------	--------------	--------------	--------------	----------------------	------------------------	-----------	-----------	----------------------	------------------------

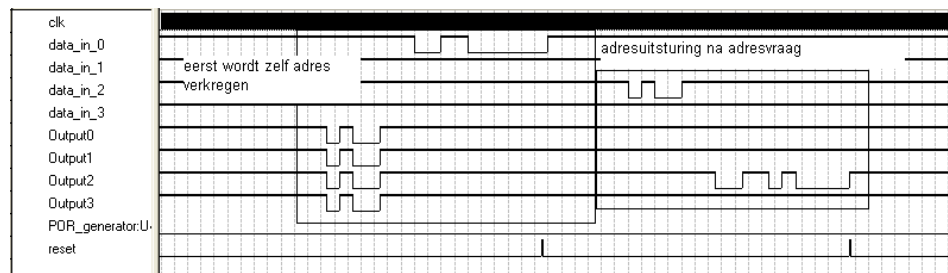
Figuur 2.6: Structuur van het Adres_in bericht



Figuur 2.7: Flowchart voor het bekomen van een adres



Figuur 2.8: Periodieke aanvraag van een adres na de POR (klok is 20MHz en grid is 1 μs)

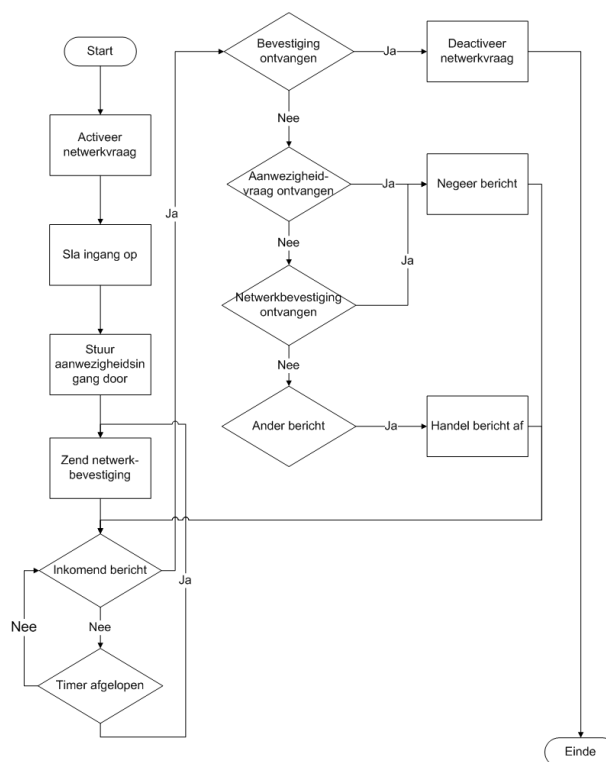


Figuur 2.9: Inkomend adres op data_ingang_0 na een adresvraag, er wordt ook een adresvraag afgehandeld van data_ingang_2 (klok is 20MHz en grid is 1 μ s)

2.1.4 Aanwezigheidsopvraging

In figuur 2.10 wordt schematisch weergegeven hoe de displaymodule zijn aanwezigheid in het netwerk meedeelt aan de interface. Bij ontvangst van een aanwezigheidsvraag (structuur zie figuur 2.11) uitgestuurd door de interface, wordt de Netwerkvraag conditie geactiveerd. De Netwerkvraag conditie duidt erop dat het netwerk vraagt naar de aanwezigheid en zorgt voor een periodieke uitzending van het Netwerkbevestigingscommando (structuur zie figuur 2.12). Deze periodieke uitzending gebeurt op de uitgang overeenkomstig met de ingang waar het Aanwezigheidsvraag commando werd ontvangen (zie figuur 2.14). Door de Netwerkbevestiging slechts op een uitgang te versturen wordt een specifiek pad gevormd van een displaymodule naar de interface doorheen het netwerk. De Netwerkbevestigingen uitgezonden door de modules het dichtst bij de interface zullen het eerst toekomen bij de interface. Na ontvangst door de interface zal deze een Bevestiging (structuur zie figuur 2.13) terugsturen met overeenkomstig adres. De bevestiging is noodzakelijk omdat op deze manier de displaymodule weet dat de aanwezigheid opgenomen is en hierdoor kan de periodieke uitzending van de Netwerkbevestiging stoppen. Inkomende Netwerkbevestigingen worden nu ook doorgegeven zodat deze ook via het specifieke pad bij de interface terechtkomen.

De ingestelde tijd om bepaalde berichten tegen te houden bij het commando onderzoek is ingesteld op 207 μ s. De netwerkbevestigingen worden periodiek uitgestuurd met kleinste periode 216 μ s. De bevestigingen worden dan ook ongeveer met deze frequentie door de interface verstuurd en deze moeten doorgelaten worden.



Figuur 2.10: Flowchart voor de afhandeling van het aanwezigheidsvraagscommando

Startbit	Cmd bit	Cmd bit	Cmd bit
0	1	0	1

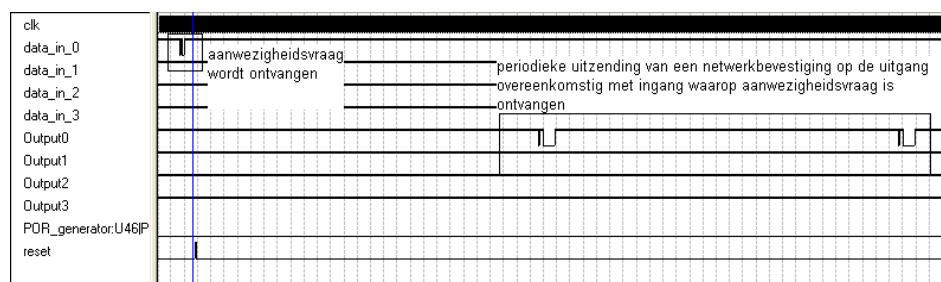
Figuur 2.11: Structuur van het aanwezigheidsvraagscommando

Startbit	Cmd bit	Cmd bit	Cmd bit	Adres bit	Adres bit	Adres bit	Adres bit	Adres bit	Adres bit
0	1	1	0	LSB Rij	LSB Kolom			MSB Rij	MSB Kolom

Figuur 2.12: Structuur van het netwerkbevestigingscommando

Startbit	Cmd bit	Cmd bit	Cmd bit	Adres bit	Adres bit	Adres bit	Adres bit	Adres bit	Adres bit
0	1	1	1	LSB Rij	LSB Kolom			MSB Rij	MSB Kolom

Figuur 2.13: Structuur van het bevestigingscommando

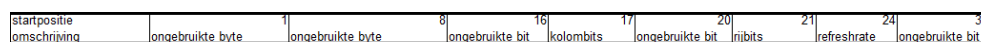


Figuur 2.14: Periodieke uitsturing van een netwerkbevestiging na de ontvangst van een aanwezigheidsvraag (klok is 20MHz en grid is $1 \mu s$)

2.1.5 Parameter en data berichten ontvangen

Aangezien er verschillende displays mogelijk zijn, dienen bepaalde parameters ingesteld te worden. Deze parameters zijn de refreshrate en de dimensie van het display. De ontvangen parameters worden na ontvangst opgeslaan in een buffer. De structuur van het registerbuffer is weergegeven in figuur 2.15. De eerste 7 bits stellen de refreshrate voor. De twee laatste bytes worden niet gebruikt en zijn beschikbaar voor later gebruik. Tijdens de ontvangst wordt het bericht doorgestuurd naar de naburige modules behalve naar de buur waar het bericht van ontvangen werd.

Anders dan de parameterinformatie is de datainformatie specifiek voor een bepaalde module. Bij ontvangst van een data bericht wordt eerst het adres gecontroleerd. Indien het adres overeenkomt met het eigen adres, start de opslag van de inkomende data. Deze data wordt opgeslaan in een RAM geheugen. De lengte van het bericht is afhankelijk van het aantal rijen van de module Er is 1 byte per rij voorzien. Bij de ontvangst moet dan ook rekening gehouden met de parameterbits die het aantal rijen bijhoudt. Indien het adres niet overeen komt met het eigen adres, wordt na adrescontrole het bericht doorgezonden naar de naburige modules, maar niet naar de module waar het bericht van ontvangen werd.



Figuur 2.15: Structuur van de Registerbuffer

2.1.6 Buur weg detectie

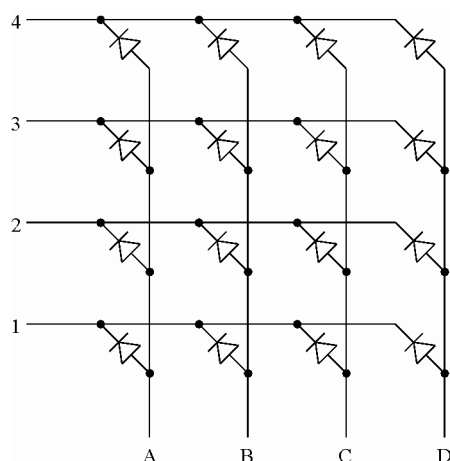
Indien een module wegvalt zal ook de spanning op de uitgangen van die module wegvallen. Het wegvallen van de spanning zorgt voor een dalende flank bij de naburige modules wat gezien wordt als het begin van een commando. De opeenvolgende bits zijn laag wat tot de detectie leidt dat een module is weggevallen. Hierdoor zal de `is_Buur` conditie laag gedeactiveerd worden. Indien op een later tijdstip terug een module zou opstarten wordt dit opnieuw gedetecteerd en wordt er opnieuw voor gezorgd dat laatst bijgekomen module terug start met de ontvangst van een volledig bericht.

2.1.7 Uitgangscontrole

De uitgangscontrole controleert welke berichten naar buiten gebracht worden en op welke uitgangen. Het uitgestuurde adres komt enkel op de uitgang overeenkomstig met ingang waarop de adresvraag is gekomen. De aanwezigheidsvraag, bevestigingen, parameters en data worden enkel naar naburige modules doorgestuurd, dat om problemen te voorkomen die reeds aangehaald zijn bij de ingangscontrole. De netwerkbevestigingen worden enkel op de uitgangen doorgestuurd overeenkomstig met de ingang waarop de aanwezigheidsvraag ontvangen. Bij de netwerkbevestigingen wordt geen rekening gehouden met de naburige burens. Er wordt verwacht dat indien er een aanwezigheidsvraag ontvangen is, de buur wel aanwezig is.

2.2 Display aansturing

Parallel met het ontvangen en zenden van berichten gebeurt de aansturing van het display. Vanaf het moment dat de displaymodule een adres verkrijgt, is de displaymodule geïntegreerd in het netwerk en kan data afgebeeld worden. Het aanwezige display is een 30mm 5 X 7 passieve dot matrix (Blok-schema zie figuur 2.16). Om een pixel (BV rij 3 kolom C) te activeren dient rij 3 met de ground verbonden te worden. De spanning op C bepaald of de LED oplicht of niet. Indien er een hoog niveau op C aangelegd wordt, licht de LED op indien laag zal de LED gedoofd blijven.



Figuur 2.16: Blokschema van een passieve LED matrix

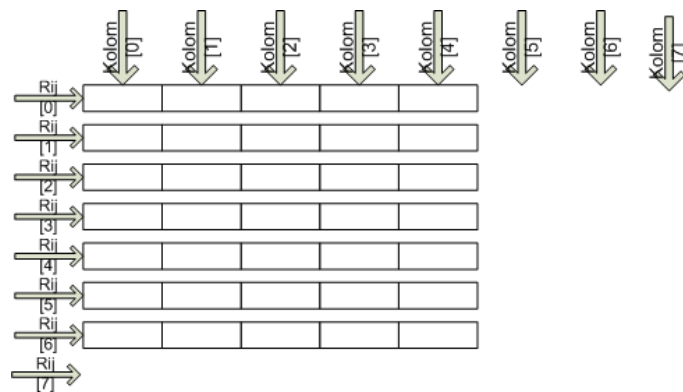
Via de parameter en datagegevens wordt de aansturing van het display geregeld. Aan de hand van de ingestelde refreshrate, wordt een read klokfrequentie afgeleid. Voor de correcte waarde van de refreshrate te bepalen wordt gebruik gemaakt van een look-up table waarbij de ingestelde parameter waarde wordt omgezet naar een bepaalde klokfrequentie. De eigenlijke refreshrate is afhankelijk van het aantal ingestelde rijen, aangezien de refreshrate slechts bepaald hoelang een bepaalde rij actief is. Hoe minder rijen er zijn, hoe vlugger het display gerefreshed zal zijn. Voor een display met 8 rijen is een refreshrate mogelijk van $\pm 0,6$ tot ± 76 Hz (komt overeen met resp. 2097152 tot 16384 kloktikken, klok = 20 MHz). De omzetting van kloktikken naar een bepaalde frequentie gebeurt volgens volgende formule:

$$freq = \frac{2097152}{128 - X} \quad (2.1)$$

Hierbij is X de decimale waarde van de ontvangen refreshrate. Indien de read klokfrequentie te laag is ingesteld, zal de afbeeldcyclus traag optreden waardoor het display begint te flinkeren.

De afbeeldcyclus (zie ook schema 2.17) zal elke aanwezige rij (Rij[0]-Rij[6]), om de beurt met de ground verbinden via analoge schakelaars. De andere rijen worden open gelaten door aan de schakelaar een logische 0 aan leggen, deze toestand zorgt ervoor dat de schakelaar open blijft. Zo wordt rij per rij geactiveerd. De data uit het RAM-geheugen wordt op het juist moment opgehaald en op de kolomuitgangen Kolom[0]-Kolom[7]) gezet (van de driver zie figuur 2.2. Via een buffer komen deze waarden op de kolomingangen van het display.

Dit bepaald of een bepaald pixel van een rij oplicht of niet. Op de niet-aanwezige pixelrijen (Rij[7]) en kolommen (Kolom[5]-Kolom[7]) wordt steeds een logische 0 aangelegd. Aan de hand van de parameterdata, wordt niet doorgeteld tot elke rij geselecteerd is, maar worden enkel de rijen afgelopen die aanwezig zijn. Dit is nodig omdat er maar evenveel bytes in het RAM geheugen zitten als er rijen aanwezig zijn. Na een read reset wordt een teller op nul gezet zodat terug de data van het eerste adres (adres 0) van het geheugen wordt uitgelezen. Aangezien de data 8 bits zijn, en de rij en kolom uitgangen ook acht bits zijn, kan er maar een display aangestuurd worden van 8x8 pixels.

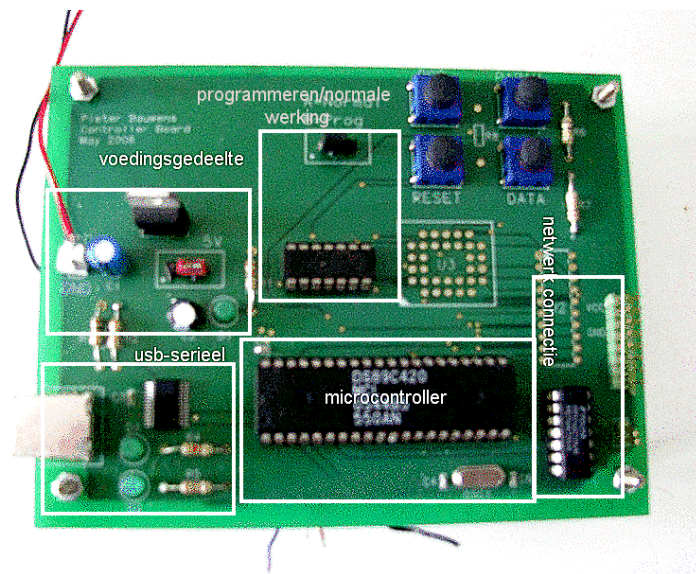


Figuur 2.17: Display verbindingen

De interface

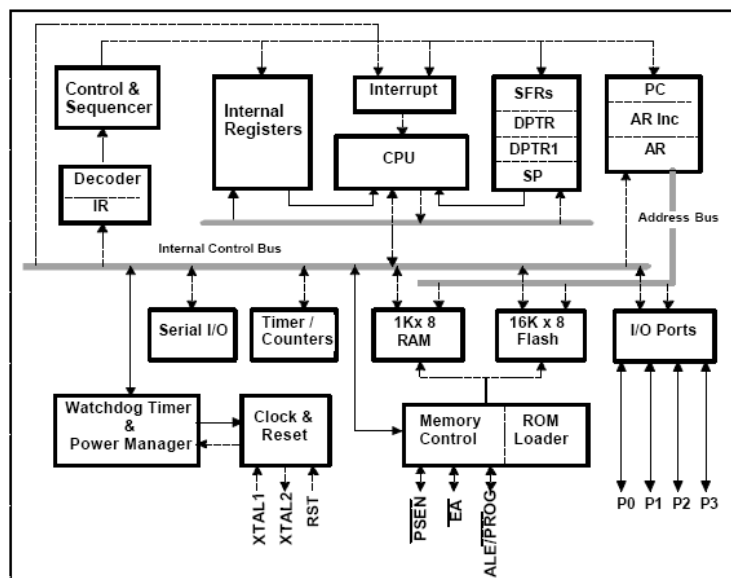
De verbinding tussen het displaynetwerk en de PC wordt voorzien via een interfacebord (zie figuur 3.1). Dit hardwarebord bevat als belangrijkste componenten een microcontroller, voedingsgedeelte, programmeerhardware en USB naar serieel omzetter. De gebruikte microcontroller is de ds89c420 van Dallas (blokdigram zie figuur 3.2). Deze is 8051-compatibel en wordt gekenmerkt door de snelle uitvoering van de instructies. De maximale klokfrequentie is 33Mhz, er zijn 3 timers, 2 seriële poorten en 1kb RAM geheugen aanwezig. Het indirecte RAM geheugen wordt gebruikt om de parameters en de data in op te slaan. De controller heeft een inwendige bootloader die seriële programmatie mogelijk maakt. Via automatische detectie van de baudrate wordt er automatisch een connectie opgezet zonder de seriële poort te configureren. De autodetectie is wel beperkt en afhankelijk van de klokfrequentie. Voor de klok van 32Mhz is een baudrate van 14400 het meest geschikt. De programmatie gebeurt in C via de Keil C51 compiler. Het Keil software pakket ondersteunt de ds89C420 en biedt veel debug mogelijkheden. Op het interfacebord zijn er voorzieningen voor extra flash geheugen maar deze worden niet gebruikt. De serieel naar USB omzetting gebeurt aan de hand van een FTDI chip waardoor er geen seriële poort op de PC aanwezig moet zijn, toch dient opgemerkt te worden dat de verbinding serieel blijft.

De communicatie tussen het interfacebord en het netwerk gebeurt via een 8-bit bidirectionele poort, die de externe interrupts bevat. Wegens het bestaande hardware-ontwerp wordt voor de ontvangst van berichten een externe interrupt INT2 (zie figuur 3.3) gebruikt die reageert op een stijgende flank. Een stijgende flank op de ontvangst pin zorgt voor de

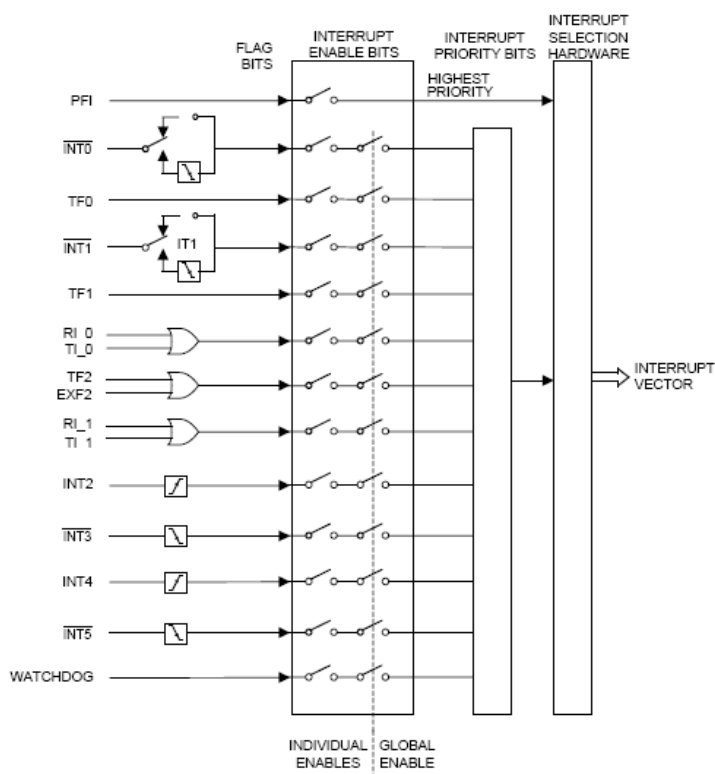


Figuur 3.1: Interface

onderbreking van het hoofdprogramma en na de afhandeling van de ontvangst wordt het hoofdprogramma verder uitgevoerd. Het zenden van berichten naar het netwerk gebeurt in de hoofdloop. De connectie van het interface met de PC gebeurt via de seriële poort. Bij de ontvangst van gegevens (data of parameters) zal via een interrupt RI_0 (zie figuur 3.3 en figuur 3.5) de gegevens in het geheugen worden opgeslaan, nadien wordt de hoofdloop verder uitgevoerd. Na het opvragen van de displaymodules worden de bytes met informatie naar de GUI gestuurd.



Figuur 3.2: Blokdiagram van de microcontroller



Figuur 3.3: Interruptschema van de microcontroller

3.1 Het hoofdprogramma

Bij het opstarten van de microcontroller dienen enkele instellingen te gebeuren. De timers worden ingesteld op 8-bit autoreload. Hierdoor zal na de overflow de teller beginnen te tellen van ingestelde waarde in het MSB register van de timer. De waarde van het LSB register dient op dezelfde waarde ingesteld te worden om de eerste maal ook van de autoreload waarde te beginnen tellen, anders wordt er van 0 geteld. Het ingangskloksignaal voor de tellers wordt ingesteld op de klokfrequentie/4, dit is 8 MHz. Timer1 wordt gebruikt als klok voor de seriële interface. De startwaarde wordt ingesteld op 0xCC. Volgens formule 3.4 wordt een baudrate van 9600 verkregen. Er moet rekening worden gehouden in de formule (figuur 3.4) dat een frequentieverdubbeling is ingesteld. Timer0 wordt gebruikt voor de generatie van uitgangssignalen naar het netwerk. De signalen hebben een frequentie van 1 MHz. De ingangsklok is 8 Mhz waardoor er na 8 tellen 1 μs verstreken is. De startwaarde (autoreload waarde) is 248 (0xF8) zodat het 8 tellen duurt tot een overflow optreedt bij een counterwaarde van 255 (0xFF). Na het instellen van de timers, wordt de seriële poort geïnitieerd en worden de interrupts geactiveerd. De default parameters worden ook ingesteld tijdens de initialisatie, er wordt verondersteld dat het default display een dimensie heeft van 7 x 5 pixels en een refreshrate van +/-60 Hz geen flikker veroorzaakt. De eerste waarde die de dimensies bepaalt, wordt ingesteld op 0x31, de waarde wordt als volgt bepaald:

$$[8\text{-most significant nibble}][8\text{-least significant nibble}] = [\text{aantal kolommen}][\text{aantal rijen}]$$

Voor de refreshrate wordt een waarde van 0x51 ingesteld. De refreshwaarde van 50 (=0101000) geeft, bij een 7 rij matrix, een refreshrate van +/-60Hz. De eerste bit is deze van het commando en wordt niet gebruikt voor de refreshfrequentie te bepalen. De omzetting van de waarde naar een frequentie is terug te vinden in vorig hoofdstuk (vergelijking 2.1).

$$\text{Modes 1, 3 baud rate} = \underbrace{\frac{2^{\text{SMOD}_x}}{32}}_{\text{Number of serial bits / Number of timer 1 rollovers}} \times \underbrace{\frac{\text{Timer 1 input clock frequency}}{(256 - \text{TH1})}}_{\text{Timer 1 rollover frequency}}$$

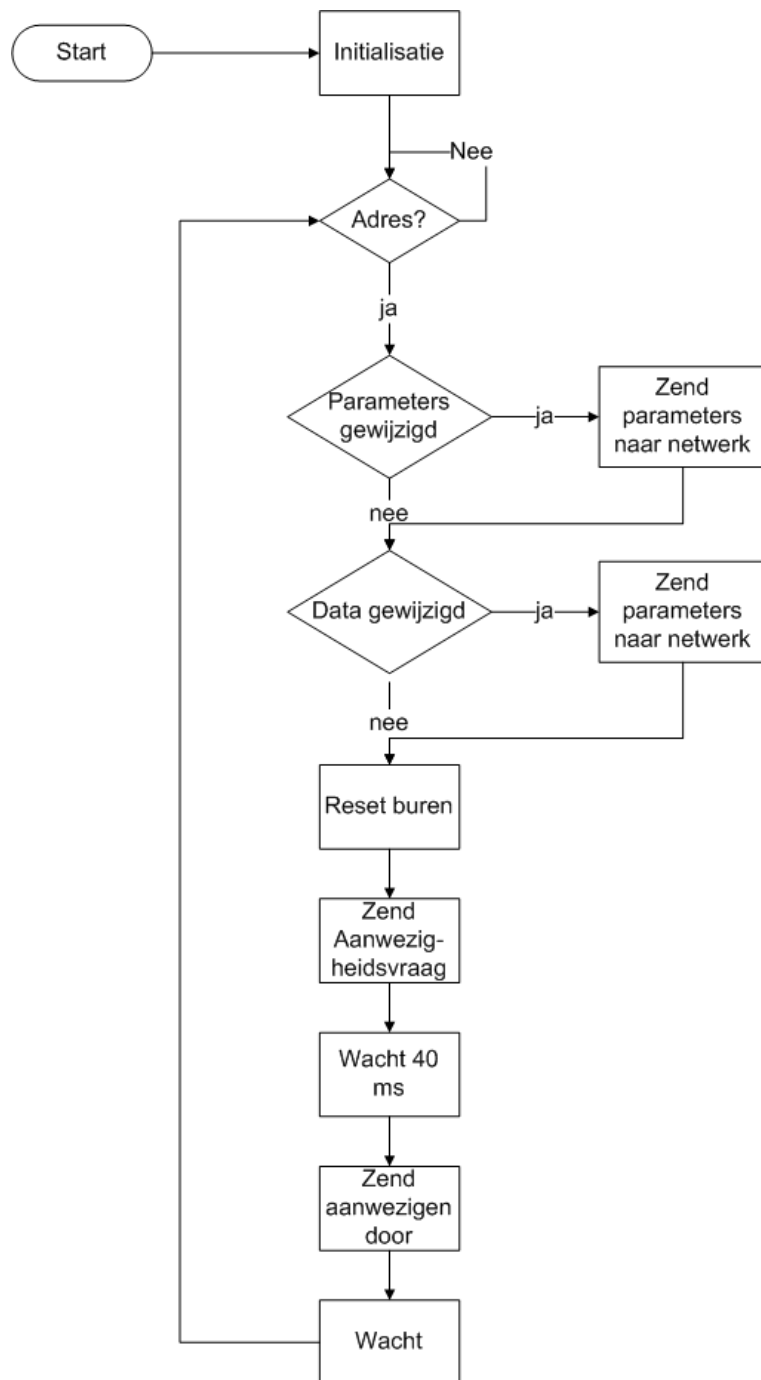
Figuur 3.4: Formule voor het bepalen van de baudrate

Na de initialisatie wordt het hoofdprogramma gestart, dit is schematisch weergegeven in figuur 3.6. Voor de start van de hoofdlus wordt telkens gekeken of er een adres is opgevraagd. Dit is nodig omdat een adressaanvraag willekeurig gebeurt waardoor de hoofd- lus onderbroken wordt en er fouten kunnen optreden. Verder is het zinloos om berichten te verzenden als er toch geen displaymodule aanwezig is. Nadat een adres is opgevraagd, zal er geen adresvraag meer optreden totdat de module uitgeschakeld is. Het wegvallen wordt ook gedetecteerd en dit zal de adres conditie deactiveren.

Indien er data of parameter gegevens gewijzigd zijn, worden deze eerst naar de displaymod- ules gezonden. Voor het verzenden van data wordt een teller gestart die elk adres afloopt en onderzoekt aan de hand van de aanwezigheidsrij of de module met dat bepaald adres aanwezig is. Vervolgens worden het aantal databytes klaargezet in een buffer. Dit is nodig omdat de databytes uit het indirecte RAM geheugen worden gehaald en dit kost tijd. Om een mooi signaal te verkrijgen met een bittijd van $1 \mu s$, ook tussen de overgang tussen de bytes, worden de bytes dus eerst in een buffer in het directe RAM geheugen opgeslaan. Dit geheugen heeft een lagere toegangstijd. Het aantal bytes wordt bepaald aan de hand van het aantal rijen terug te vinden in de parameter informatie.

Voor het sturen van een aanwezigheidsvraag om de aanwezige modules op te vragen, wordt de rij die de aanwezigen bijhoudt gereset. Na het zenden van een aanwezigheidsvraag, verwacht de microcontroller inkomende netwerkbevestigingen, waarna er een bevestiging teruggestuurd wordt. Na onderzoek werd bekomen dat de worst case tijd rond de 40 ms bedraagt. Deze tijd is bepaald voor het maximaal aantal displaymodules bij een klokfre- quentie van 20 MHz. Gedurende 40 ms kunnen netwerkbevestigingen toekomen. Tijdens deze tijd wacht de microcontroller zodat de ontvangst van netwerkbevestigingen geen problemen veroorzaakt. Na 40 ms zijn alle netwerkbevestigingen ontvangen en opgeslaan in de aanwezigheidsrij. Elk adres wordt gezien als een getal. Dit getal is de bit posi- tie in de aanwezigheidsrij. Na ontvangst van een netwerkbevestiging wordt de bitpositie die overeenkomt met het adres hoog gezet. Zo is de aanwezigheidsrij 64 bits lang of 8 bytes.

Nadat de aanwezigen zijn opgevraagd wordt de aanwezigheidsrij doorgestuurd via de seriële interface (blokschema zie figuur 3.5). De 8 bytes van de aanwezigheidsrij worden één voor één naar het SBUF register geschreven. De bits worden dan aan het tempo van de ingestelde baudrate naar buiten geschoven. De seriële interface meldt via een conditie (TI vlag) aan de processor (CPU) dat de byte verzonden is, er kan dan een nieuwe byte verzonden worden. Na ontvangst zal de GUI de vorm van het netwerk zal visualiseren.



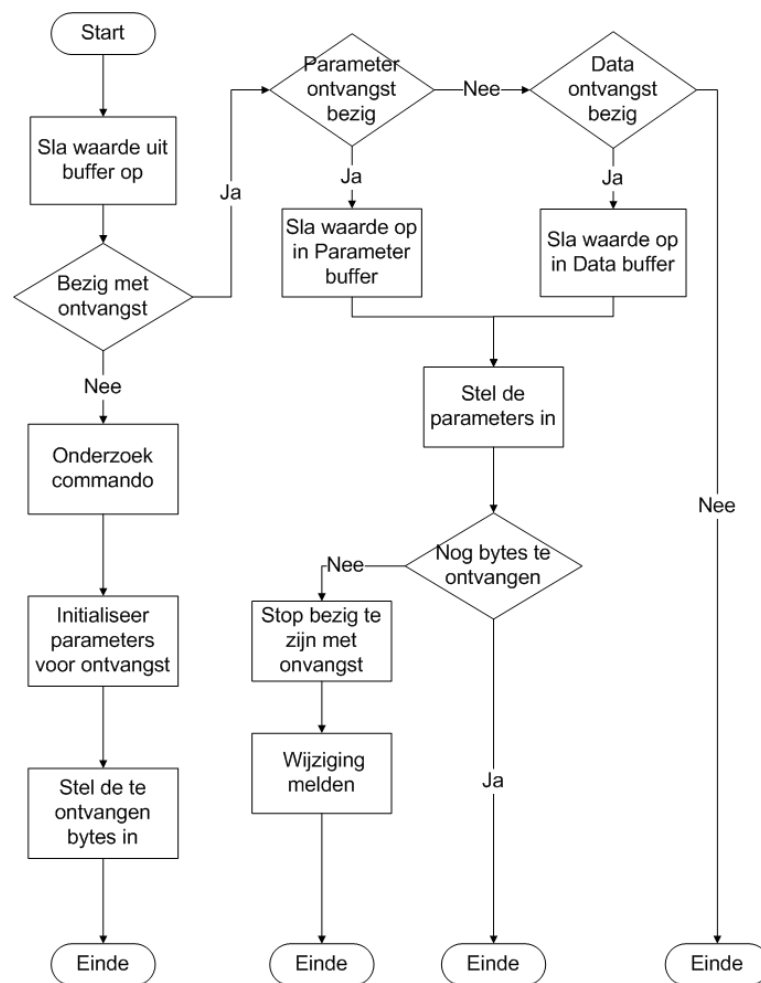
Figuur 3.6: Flowchart Hoofdprogramma

3.2 Seriële poort interrupt service routine

Om de data en parameters die ingegeven worden door de gebruiker naar de interface te sturen, wordt de seriële interface gebruikt. Het doorsturen van de gegevens gebeurt als de gebruiker daartoe opdracht geeft en is dus niet periodiek. Deze taak maakt gebruik van interrupts. In figuur 3.7 wordt het verloop weergegeven van het programma dat doorlopen wordt indien de interrupt (RI_0 zie figuur 3.3 en figuur 3.5) optreedt.

Nadat de waarde uit de seriële buffer (SBUF register) is opgeslaan in het RAM geheugen, wordt onderzocht of de microcontroller reeds bezig is met ontvangst van gegevens. Indien dit niet het geval is, wordt aan de hand van het ontvangen karakter, onderzocht wat de komende karakters zijn. Dit kunnen zowel data als parameters gegevens zijn. Na onderzoek worden instellingen zoals de juist booleaanse waarden en het aantal te ontvangen bytes ingesteld. Voor parameters zullen er telkens twee bytes verwacht worden. De eerste byte omvat informatie over de dimensies van een display, de tweede byte bevat de refreshrate waarvan het display gebruik maakt. De LSB van de tweede bit wordt gebruikt als eerste commandobit bij het doorsturen van de parameters naar de display modules zelf. Het aantal te ontvangen databits is afhankelijk van het display. Voor elke rij van het display wordt een byte voorzien waar de bitpositie binnen die byte de waarde van een kolom voorstelt. Indien er slechts vijf kolommen aanwezig zijn, worden de eerste vijf bits gebruikt de rest wordt genegeerd. Het te ontvangen databytes is dus afhankelijk van het aantal aanwezige modules en het aantal rijen de displaymodule heeft.

Na de ontvangst en opslag van de bytes, wordt ingesteld dat de gegevens gewijzigd zijn. Hierdoor zal in het hoofdprogramma de data of parameters naar de displaymodules verzonden worden.



Figuur 3.7: Flowchart Seriële poort ISR

3.3 Externe interrupt service routine

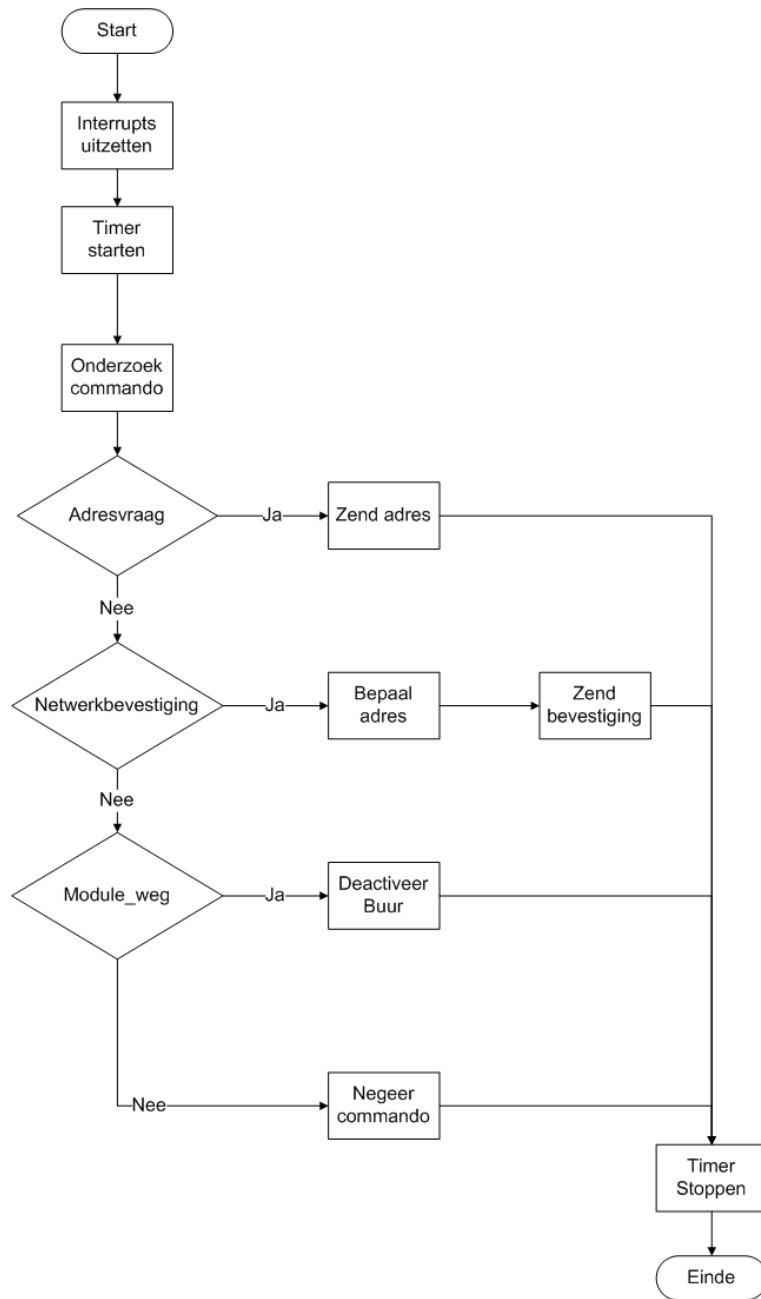
Na het resetten van de microcontroller wordt de ontvangstpin hoog gehouden via een zwakke pullup. Deze staat wordt gedomineerd door externe componenten. Als de software de waarde van de poort opvraagt, wordt de staat van de poort gelezen. Als het externe circuit een logische 1 aanlegt, zal een 1 gelezen worden. Indien het externe circuit een laag niveau aanlegt zal de zwakke pullup overheerst worden, waardoor een laag niveau wordt ingelezen. Indien er geeningangssignaal aanwezig is, zal de zwakke pullup de pin hoog houden via de interne geheugenschakeling. Hierdoor wordt een hoog niveau ingelezen als de staat wordt opgevraagd.

Het signaal afkomstig van de displaymodule wordt geïnverteerd naar de ingang van de microcontroller gebracht. De microcontroller genereert een interrupt op een stijgende flank. Na het optreden van de interrupt worden de interrupts afgezet omdat de ISR niet mag onderbroken worden. Vervolgens dient de timer ingesteld te worden om op de juiste tijdstippen te samplen. Na drie maal samplen wordt het binnenkomende commando bepaald. Indien het commando een adresvraag is, wordt een adres teruggestuurd naar de displaymodule. Het adres staat ingesteld op 000000. Hierdoor stellen we eenduidig vast dat de displaymodule die verbonden is met de microcontroller op rij 0 en kolom 0 staat.

Indien er een netwerkbevestiging ontvangen wordt, zal eerst het adres bepaald worden via de sampling van de 6 adresbits (zie figuur 2.12). Vervolgens wordt een bevestiging uitgestuurd die het ontvangen adres bevat (zie figuur 2.13). Bij de aanwezigheid zal een bit hoog gezet worden op de positie die overeenkomt met het ontvangen adres, zo wordt bijgehouden welke displaymodules aanwezig zijn.

Bij het wegvallen van de displaymodule zal er een stijgende flank optreden gevolgd door een hoog niveau (wegens inversie en pull-up). De microcontroller detecteert dat de module is weggefallen en zal dit doorsturen naar de GUI. Zolang er geen adres wordt aangevraagd (zie figuur 3.6) zal de microcontroller standby blijven en het hoofdprogramma niet uitvoeren. Dit is een beveiliging omdat een adresvraag willekeurig kan optreden en dus het hoofdprogramma kan onderbreken, waardoor er mogelijks fouten optreden.

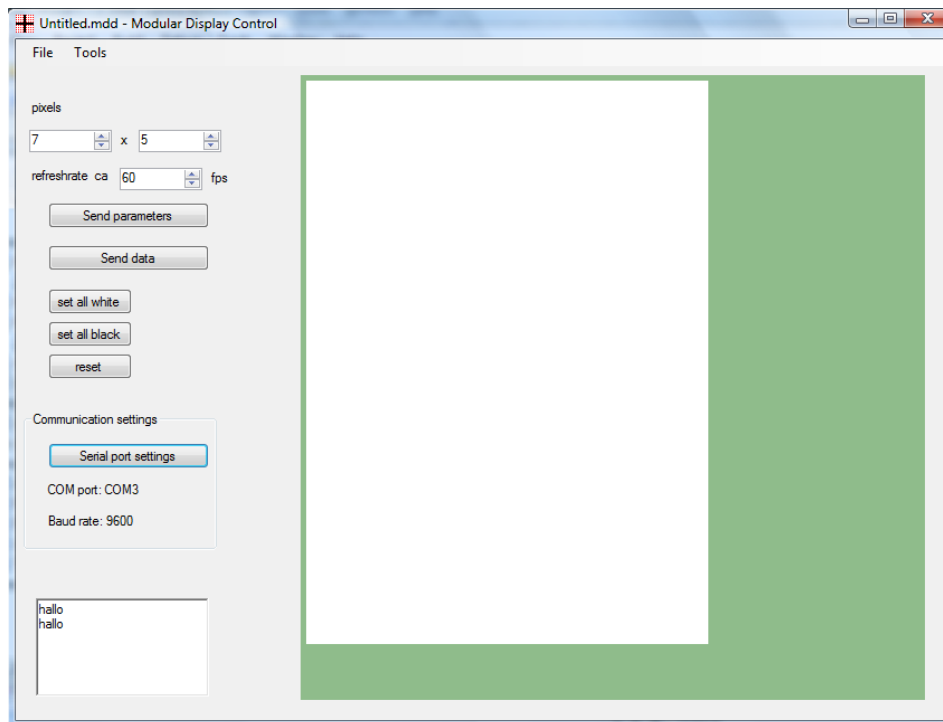
Na de afhandeling van de interrupt wordt de timer gestopt en worden de interrupts opnieuw geactiveerd zodat er weer berichten kunnen ontvangen worden.



Figuur 3.8: Flowchart Externe ISR

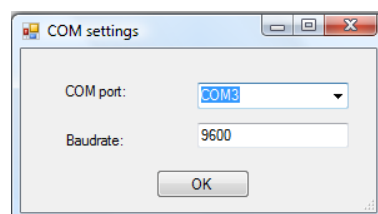
De grafische user-interface

Om het geheel op eenvoudige manier controleerbaar te maken voor de gebruiker, is een GUI voorzien. De GUI biedt een manier om de instellingen van de displaymodules in te geven. De vorm van het display wordt ook getoond op het computerscherm. Hierbij worden de aaneengekoppelde basisdisplays als een geheel beschouwd. Dit laat toe om op een eenvoudige manier in te geven wat op het display moet verschijnen. De user-interface is volledig gemaakt in C++. Microsoft Visual studio maakt het mogelijk om de layout op een grafische manier in te geven. De specifieke functies van de panels en knoppen wordt vervolgens door de programmeur ingegeven. In figuur 4.1 is het opstartscherm van de GUI weergegeven. Links in het opstartscherm is er de mogelijkheid bepaalde parameters in te stellen zoals de dimensies van het display en de refreshrate. Deze parameters worden standaard ingesteld op de waarden die ook standaard zijn voor de interface. Rechts is een wit vlak weergegeven waar de vorm van het display verschijnt en wordt upgedate na de ontvangst van de gegevens over de aanwezige displaymodules. Links onderaan is nog een tekstvak weergegeven dat wordt gebruikt voor debugging.



Figuur 4.1: Opstartscherm van de grafische user interface

De dataoverdracht tussen de PC en interface gebeurt serieel. Via de ontwikkelomgeving wordt een seriële interface toegevoegd aan het programma die de communicatie zal verzorgen. Deze wordt opgestart als het programma wordt opgestart, hierbij worden de standaardinstellingen gebruikt. Een extra mogelijkheid is dat de gebruiker de poort zelf dient op te starten via een extra drukknop. Het configureren van de seriële interface gebeurt via een pop-up window (zie figuur 4.2), hier kan de gewenste baudrate en COM-poort ingegeven worden. Voor het gebruikersgemak te verhogen, hoeft de gebruiker enkel te kiezen uit een lijst met aanwezige COM-poorten, er moet wel gekozen worden voor de COM-poort die met de FTDI-chip geconnecteerd is. Indien er fouten optreden betreffende de seriële connectie wordt dit via een pop-up window gemeld aan de gebruiker.



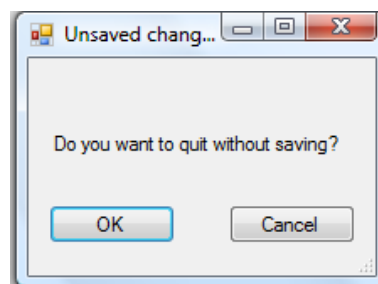
Figuur 4.2: Pop-up venster om de seriële poort te configureren

Via de “*send parameter*” knop (zie figuur 4.1) worden de ingestelde parameters naar het interface bordje gestuurd. Eerst worden de dimensie-instellingen verzonden en daarna de refreshrate. Er dienen telkens twee bytes verstuurd te worden aangezien de interface beide parameters verwacht. De refreshrate wordt voor het verzenden eerst omgezet naar een waarde die de driver aankan. Als er op de “*send data*” knop wordt gedrukt, wordt eerst berekend hoeveel bytes er moeten verzonden worden. Hiervoor wordt bijgehouden hoeveel modules aanwezig zijn. Dit aantal wordt vermenigvuldigd met het aantal pixelrijen. Het resultaat is het aantal bytes dat de interface verwacht. De aanwezige adressen worden dan rij per rij afgelopen en de waarde van de bytes wordt bepaald door de kolomdata. Er wordt per rij een byte gestuurd, als er geen acht kolompixels aanwezig zijn, worden de resterende bits op nul gehouden. De “*set all black*” en “*set all white*” knoppen zetten alle pixels respectievelijk op zwart en wit.

Bij de ontvangst van de bytes wordt telkens gewacht tot acht bytes ontvangen zijn. Nadat de ontvangst van acht bytes, worden deze opgeslaan in een aanwezigheidsmatrix. Deze matrix is dan een kopie van de aanwezigenrij zoals beschreven bij de interface. De analyse is nodig zodat het scherm kan geupdate worden volgens de nieuwe data. Elk adres (6 bits) wordt afgelopen en er wordt gekeken naar de overeenkomstige bits uit de ontvangen bytes. Indien de bit hoog staat, is de module overeenkomstig adres aanwezig. De positie van het basisdisplay wordt via een conversie van het adres naar een bepaalde rij en kolom bepaald. Op deze locatie wordt dan een raster afgebeeld dat een basisdisplay voorstelt. Het raster wordt opgebouwd aan de hand van de ingestelde displaydimensies. Het groene vlak zal steeds dezelfde grootte behouden door de pixelgrootte aan te passen na verandering van de dimensies.

De gebruiker kan data ingeven via een linkermuisklik om het pixel zwart te maken en rechtermuisknop om het wit te maken. Na ingave van de data wordt eerst bepaald aan de hand van de positie in welk display er geklikt is. Enkel als het display aanwezig is, zal de specifieke locatie binnen het display bepaald worden. In een overeenkomstige 3-dimensionale matrix (datamatrix) zal een bool gezet worden. De eerste dimensie van de matrix is het adres. Vervolgens heeft elk adres twee dimensies, een rij en een kolomdimensie. Nadat de pixel is ingegeven wordt het display ge-update. Dit houdt in dat de drie dimensionale matrix wordt afgelopen en op de locaties waar een 1 staan wordt een zwart vierkantje getekend, op de plaatsen met een 0 blijft het pixel wit. Bij een wijziging van de dimensies, zal ook de datamatrix wijzigen. Er wordt een nieuwe matrix aangemaakt waarbij de rij- en kolomdimansie overeenkomen met de nieuwe instellingen. De data uit de oude matrix wordt per adres (per moduledisplay) gekopieerd naar de nieuwe matrix.

Er is een mogelijkheid voor de gebruiker om alle gegevens op te slaan zodat de gebruiker op een later tijdstip kan verder werken. Als het programma afgesloten wordt en de gegevens zijn niet opgeslaan, wordt de gebruiker verwittigd via een pop-up window. Dit pop-up window is weergegeven in figuur 4.3. Bij iedere wijziging in het programma wordt een bool gezet zodat het systeem weet dat laatste wijzigingen nog niet opgeslaan werden. Na opslag wordt deze bool laag. Voor de opslag wordt eerst een numerieke code geschreven naar een bestand, deze code dient om ervoor te zorgen dat geen andere file kan gelezen worden en er zo fouten optreden. Dan worden de instellingen zoals de COM-poort, baudrate, pixels en refreshrate als een string naar het bestand geschreven. Vervolgens worden de aanwezige modules opgeslaan en ook de drie dimensionale matrix die de data bevat. Als de gebruiker het bestand opent, worden alle instellingen teruggezet. De strings worden dan omgezet naar integer waarden. Indien er fouten optreden met de COM-poort bij het heropstarten wordt de gebruiker hiervan op de hoogte gebracht.



Figuur 4.3: Pop-up venster indien de gegevens niet opgeslaan zijn

Bij het afsluiten van het programma wordt ervoor gezorgd dat de seriële interface wordt afgesloten. Indien dit niet zou gebeuren zouden er problemen optreden aangezien de computer niet weet dat de poort niet meer in gebruik is. Om het systeem robuuster te maken werd er een systeem ingebouwd dat enkel bytes kon verzenden indien de interface beschikbaar was. Indien de gebruiker op een zend knop drukte werd er gekeken of de interface klaar was voor data ontvangst. Indien dit niet het geval was, werd er gewacht tot de acht bytes met aanwezige modules werden ontvangen, hierna gaat de interface immers in een wachttoestand waarbij ondertussen data kan ontvangen worden. Via een stopbyte kon de interface de GUI beletten om gegevens te verzenden, dit tot de GUI opnieuw de acht aanwezigheid bytes ontving. Na de praktische realistische bleek dit extra controlemechanisme overbodig. De reden hiervoor is dat de verwerkingstijd van de interface zeer kort is in vergelijking met zijn wachttijd, waardoor er nauwelijks problemen optreden.

Conclusie

Na grondig onderzoek van het bestaande systeem, werd al vlug duidelijk dat er een goede basis aanwezig was voor de driver. De displaymodules konden reeds data en parameters ontvangen. Aan de hand van de data en de parameters werd de vereiste figuur afgebeeld op het display. De interface verzorgde de communicatie tussen de PC en de aaneengeschakelde displaymodules. Via een seriële connectie kon de PC data en de parameters doorsturen naar de interface. Via berichten kwam de data en parameters bij de displaymodules terecht.

Door de invoering van een grotere variatie van berichten wordt het mogelijk dat de displaymodules op ieder moment een identificatie verkrijgen. Indien een displaymodule wegvalt, wordt dit ook gedetecteerd bij de naburige displaymodules. Andere types berichten zorgen voor het opvragen van de aanwezigheid. Na een aanwezigheidsvraag gaat elke module zich melden bij de interface. Dit maakt het mogelijk dat de vorm van het display automatisch wordt gedetecteerd. De simulaties in Active-HDL en Quartus vertonen het gewenste gedrag van een displaymodule. In Active-HDL is het mogelijk een geheel netwerk te testen. Verschillende vormen zijn gesimuleerd en leveren goede simulatieresultaten op. Bij de praktische implementatie traden timingsproblemen op (vertraging van het kloksignaal) die opgelost werden. Bij het testen van de driver op de bestaande displaymodules vormde de ruisgevoeligheid van de ingangen een probleem. De stoorsignalen zorgden voor de detectie van berichten op ingangen die niet verbonden waren. Via extra pull-down weerstanden toe te voegen, is dit probleem opgelost. De ingangen detecteren nu een laag niveau indien ze open gelaten worden. Voor het testen wordt het opstarten en wegvallen van een module gesimuleerd door de voedingsspanning respectievelijk aan of uit te schake-

len via een schakelaar. Dit veroorzaakt jitter op de uitgangen en kan storend werken voor naburige modules. Dit probleem kan opgelost worden door het toevoegen van condensatoren. Een andere oplossing is om enkel de FPGA's aan en uit te schakelen en niet het gehele voedingscircuit.

De bestaande interface was volledig geschreven in assembler. Keil bevat een C-compiler zodat geopteerd is om de interface in C te schrijven. Bestaande algoritmes vormden de basis voor het zenden van berichten. De interface moet in het nieuwe systeem ook berichten ontvangen. Via een externe interrupt wordt dit gerealiseerd. Het gedrag van de interface is via simulatie gecontroleerd. Na de praktische implementatie traden timingsproblemen op (samplen van een binnenkomende bericht van het netwerk). Deze problemen zijn opgelost. De gebruikte microcontroller (ds89c420) zorgt via een pull-up dat de ingelezen staat hoog is indien er geen extern circuit aanwezig is. Als de module die verbonden is met de interface, wegvalt, wordt dit niet altijd gedeteceerd. Dit vormt geen probleem want als de module wegvalt, is er niet langer sprake van een netwerk. Het gebruik van een andere microcontroller of extra hardware kan een oplossing bieden en een mogelijkheid die verder onderzocht kan worden.

De GUI maakt het mogelijk op een eenvoudige manier data en parameters in te geven. De layout is gebaseerd op de bestaande layout. De code is wel volledig aangepast aan het nieuwe systeem. De werking van de GUI is getest door een seriële connectie met een andere PC op te zetten. De karakters die de GUI verzond, kunnen op eenvoudige manier gecontroleerd worden via de Realterm terminal. Via Realterm worden ook karakters naar de GUI gestuurd. De implementatie van de GUI in het systeem gaf geen problemen.

De frequentie waarmee de aanwezigheid wordt opgevraagd, kan aangepast worden naargelang de toepassing. Een stijging van de aanwezigheidsvraag zorgt voor een extra belasting van netwerk en een hoger vermogensverbruik. Bij een kloksignaal van 20MHz duurt het 40ms opdat alle modules (64) zich gemeld hebben bij de interface. Dit is dan ook de maximale frequentie voor het uitsturen van een aanwezigheidsvraag.

De grootte van het netwerk is nu vastgelegd op een 8x8 matrix. Dit kan uitgebreid worden door meer adresbits te gebruiken of door in clusters te werken. Verschillende blokken van 8x8 displays worden dan aaneengeschakeld. Voor deze toepassing moet het systeem aangepast en uitgebreid worden. De invloed van een groter netwerk op het netwerkprotocol kan op die manier verder onderzocht worden.

Het controlesysteem dat berichten tegenhoudt zodat deze niet blijven hangen in het netwerk, voert een controle uit per type bericht. Door ook het adres te controleren kunnen gelijkaardige berichten met een verschillend adres onderscheiden worden. Gelijkaardige berichten, inclusief databerichten met een verschillend adres worden een langere tijd tegengehouden. De kans dat een bericht blijft hangen in het netwerk verkleint hierdoor aanzienlijk. Uit de simulaties blijkt het huidige systeem toch goed te functioneren. Tijdens de aanwezigheidsvraag blijven geen berichten in het netwerk hangen.

bibliografie

- A Driver for Modular Passive-Matrix Displays, P.Bauwens J. Doutreloigne A.Monté, Ghent Univ. ELINTEC/TFCG Microsystems
- David Buchla and Douglas Joksch, Experiments in digital fundamentals with VHDL, PEARSON Prentice Hall, 2003
- Chapter 5: Aldec Active-HDL Support Perform Simulation Using the Active-HDL Software (GUI Mode), Quartus II Handbook Version 9.0 Volume 3, Altera Corporation, 2009
- ir. L. Ammeraal, de programmeertaal C
- Dallas Semiconductor, Ultra-High-Speed Flash Microcontroller Users Guide Rev: 6 12/04, Maxim integrated products, 2004 USA
- Dallas Semiconductor, DS89C420 Ultra-High-Speed Microcontroller, Maxim integrated products, USA

Lijst van figuren

1.1	Voorbeeld netwerk	2
2.1	Voorbeeld display module	6
2.2	Layout driver	7
2.3	Door bij te houden welke ingangen bezig zijn met het ontvangen van berichten, kan de start van en bericht eenduidig bepaald worden	9
2.4	Opstartsequentie waarbij het adresvraag commando niet voldoende is als burendetectie	10
2.5	Structuur van het Adresvraag bericht	11
2.6	Strucruur van het Adres_in bericht	11
2.7	Flowchart voor het bekomen van een adres	12
2.8	Periodieke aanvraag van een adres na de POR (klok is 20MHz en grid is 1 μs)	12
2.9	Inkomend adres op data_ingang_0 na een adresvraag, er wordt ook een adresvraag afgehandeld van data_ingang_2 (klok is 20MHz en grid is 1 μs)	13
2.10	Flowchart voor de afhandeling van het aanwezigheidsvraagscommando	14
2.11	Structuur van het aanwezigheidsvraagscommando	14
2.12	Strucruur van het netwerkbevestigingscommando	14

2.13	Structuur van het bevestigingscommando	14
2.14	Periodieke uitsturing van een netwerkbevestiging na de ontvangst van een aanwezigheidsvraag (klok is 20MHz en grid is 1 μs	15
2.15	Structuur van de Registerbuffer	15
2.16	Blokschema van een passieve LED matrix	17
2.17	Display verbindingen	18
3.1	Interface	20
3.2	Blokdiagram van de microcontroller	21
3.3	Interruptschema van de microcontroller	21
3.4	Formule voor het bepalen van de baudrate	22
3.5	Blokschema van de seriële interrupt mode 1	24
3.6	Flowchart Hoofdprogramma	25
3.7	Flowchart Seriële poort ISR	27
3.8	Flowchart Externe ISR	29
4.1	Opstartscherm van de grafische user interface	31
4.2	Pop-up venster om de seriële poort te configureren	31
4.3	Pop-up venster indien de gegevens niet opgeslaan zijn	33

Lijst van tabellen

2.1	Gebruikte commando's	7
-----	--------------------------------	---

