Bachelor Thesis

Ontwikkeling van een systeem voor opslag een doorgifte van energiemetingen

Een onderdeel van het EnerBox energiemonitoring systeem

Mohammad Hosseini
(1259431)
Folkert Roscam Abbing
(1192388)

Begeleider:
dr. Rudy Negenborn
Technisch Begeleider:
Leo van Velzen

Groep 5, EnerSense
13 juni2008

VOORWOORD

Deze thesis is geschreven in het kader van het bachelor afstudeerproject, behorende bij de studie elektrotechniek aan de Technische Universiteit Delft. Dit project werd vooraf gegaan door het vak *High Tech Startups*. In dit vak is in opdracht van *Delft Center for Systems and Control* onderzoek verricht naar mogelijkheden voor het monitoren van energieverbruik in huishoudens. Hieruit is de *EnerBox* naar voren gekomen als oplossing. Dit is een systeem bestaande uit meerdere onderdelen. De focus van deze thesis ligt op de ontwikkeling van de EnerStation, één van deze onderdelen. Dit betreft een deelsysteem dat communicatie tussen de overige onderdelen en buffering van meetgegevens bewerkstelligt.

Voorkennis met betrekking tot embedded systems is niet vereist, maar kan zeker bijdragen aan een goed begrip van de thesis. Lezers die geïnteresseerd zijn in de technische implementatie, alsook de alternatieve implementatiemogelijkheden die zijn overwogen, kunnen die vinden in hoofdstuk 4.

Voor de groepsbegeleiding van het project gaat dank uit naar dr. Rudy Negenborn en Leo van Velzen. Daarnaast zijn wij dank verschuldigd aan de algemene begeleiders van het project, dr. Koen Bertels en ir. drs. Liselotte van Dam-Schuringa. Voor het controleren van de thesis bedanken wij drs. Wim Blokzijl. We bedanken de overige groepsleden van EnerSense, Mohamad Alamili, Tarak Jabri, Bart Kersbergen en Wouter Mense voor de goede samenwerking.

INHOUDSOPGAVE

SAMENVATTING

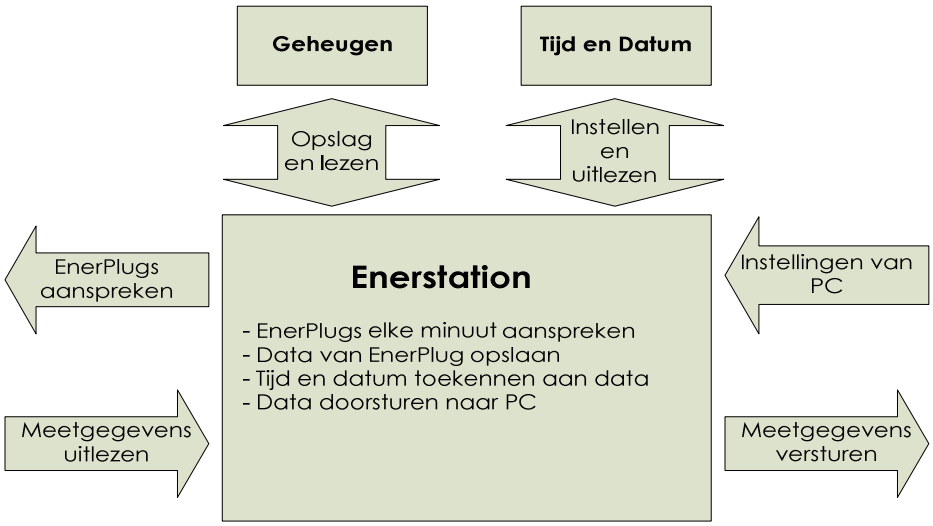
Deze thesis is geschreven in het kader van de ontwikkeling van de EnerBox. De EnerBox is een systeem dat energieverbruik in huishoudens kan monitoren en inzichtelijk kan maken voor de gebruiker. Het systeem bestaat uit drie onderdelen, te weten de EnerPlug, de EnerStation en EnerView. De EnerPlug is de sensor waarmee energieverbruik gemeten wordt. EnerView is software voor een PC, waarmee de gebruiker het gemeten energieverbruik kan analyseren.

Het hoofddoel van dit onderzoek is het ontwerpen van de EnerStation, het onderdeel dat de overige twee delen aan elkaar koppelt. De functies van de EnerStation zijn de volgende:

1. meetgegevens van de EnerPlugs verzamelen,
2. verzamelde meetgegevens bufferen,
3. een tijd- en datumreferentie bijhouden, die gekoppeld wordt aan de meetgegevens,
4. de gebufferde meetgegevens doorsturen naar EnerView.

Er is onderzocht hoe aan deze functionaliteiten voldaan kan worden. Uit dit onderzoek zijn een aantal implementaties naar voren gekomen, die vervolgens ook daadwerkelijk uitgevoerd zijn.

Als centrale deel van het systeem is gekozen voor een 8051 compatible microcontroller. Doordat dit onderdeel meerdere functionaliteiten bezit, konden beide communicatiefuncties (1 en 4) hier direct mee geïmplementeerd worden. Voor de communicatie met de EnerPlug is gekozen voor een zelf ontworpen dataprotocol, dat gebruik maakt van pulsbreedte-modulatie. De communicatie met de PC met daarop EnerView verloopt via een veelgebruikt protocol, RS-232[14]. In de microcontroller is hier specifieke functionaliteit voor ingebouwd. De buffer-functionaliteit (2) is ingevuld met een aparte geheugenmodule. Als tijdreferentie (3) is gekozen voor een externe zogenaamde *Real Time Clock*, een onderdeel dat speciaal ontworpen is om de tijdreferentie te bieden. Het geheel wordt door dezelfde microcontroller aan elkaar gekoppeld. Figuur 1 is een schets van de EnerStation, waarin de vier functies beschreven worden.



Figuur 1 - Blokschema van EnerStation

1 INLEIDING

Elektriciteitskosten stijgen razendsnel, van 2003 tot 2007 gemiddeld 6,9% per jaar[1]. Daarnaast wordt de schade aan het milieu door ons energieverbruik elke dag groter. Daarom is er in opdracht van *Delft Center for Systems and Control (TU Delft)* marktonderzoek uitgevoerd naar systemen voor energiebesparing. Hieruit bleek dat er veel vraag is naar een systeem dat energie kan besparen in huishoudens. Omdat de problemen gerelateerd aan energieverbruik zonder maatregelen groter zullen worden, is het belangrijk om naar mogelijke oplossingen te zoeken.

Naar aanleiding hiervan is ten doel gesteld een systeem te ontwerpen dat het elektriciteitsverbruik van huishoudens visualiseert en vervolgens de gebruiker adviseert. Er is vastgesteld wat de eisen van de opdrachtgever waren. Tevens is er onderzoek uitgevoerd naar de wensen van de eindgebruiker van het systeem. Hierna zijn randvoorwaarden, gebruikerseisen en technische eisen vastgesteld. Aan de hand daarvan is een globaal ontwerp gemaakt. Dit heeft tot een implementatie geleid. Ten slotte is geëvalueerd of de gekozen implementatie aan de vastgestelde eisen voldoet.

Het eerste wat besproken wordt is de probleemdefinitie in hoofdstuk 0. Hieruit volgt een programma van eisen. Tevens is in hoofdstuk 3 een bestaand apparaat met vergelijkbare functionaliteit onderzocht. Voor de implementatie van de onderdelen van de EnerStation zijn eerst diverse implementatiemogelijkheden afgewogen, waarna op basis van vooraf gestelde criteria een keuze is gemaakt. Deze keuze is vervolgens geïmplementeerd, zoals is beschreven in hoofdstuk 4. De resultaten worden beschreven in hoofdstuk 5, waarna in hoofdstuk 6 terug wordt geblikt op de keuzes die zijn gemaakt. De conclusies en aanbevelingen worden gegeven in hoofdstuk 7.

[1] Op basis van 3000KWh enkeltarief, bron: Centraal Bureau voor de Statistiek

2 PROBLEEMDEFINITIE

Het overkoepelende systeem, de EnerBox, is tot stand gekomen vanuit de wens energie te besparen in huishoudens. Hiervoor is het nodig om nauwkeurig te kunnen analyseren in welke mate de diverse apparaten in een huishouden bijdragen aan het energieverbruik. De EnerBox doet dit.

Binnen dit systeem draagt de EnerPlug zorg voor de meting van het energieverbruik. De EnerView software zorgt voor de presentatie en analyse van de verzamelde meetgegevens. De EnerStation koppelt deze systemen aan elkaar. Het programma van eisen voor dit gehele systeem is te vinden in Bijlage 1: Programma van eisen van de EnerBox. Hierbij dient vermeld te worden dat deze eisen gelden voor een hypothetisch eindproduct. In het prototype wordt slechts aan een gedeelte van deze eisen voldaan.

Voor de EnerStation gelden specifieke eisen, die niet voor de alle andere onderdelen van de EnerBox gelden. Deze eisen komen uit het algemene programma van eisen.

2.1 PROGRAMMA VAN EISEN

2.1.1 INSTALLATIE

1. Het systeem dient zonder gereedschap geïnstalleerd te kunnen worden.
2. Het systeem dient geen extra bekabeling te vereisen.

2.1.2 SYSTEEMVEREISTEN

Algemeen

3. Het systeem dient maximaal 250 te kunnen bevatten.

Opslag

4. Het opslagmedium van het systeem (uitgezonderd PC) dient zodanig gekozen te worden dat de gegevens zonder PC 1 jaar bewaard kunnen worden.
5. Het systeem moet KEMA gekeurd worden.
6. De verversingsfrequentie van de meetgegevens is 1 minuut.

3 VERWANT ONDERZOEK – VOLTCRAFT ENERGY CONTROL 3000

Om een betere indruk te krijgen van de implementatiemogelijkheden die er zijn voor de EnerBox, is een apparaat onderzocht dat wat opbouw betreft sterk lijkt op de EnerBox. Dit betreft de *Voltcraft Energy Control 3000*[1]. Van dit systeem zijn in het bijzonder de bediening en de implementatie bestudeerd. In dit hoofdstuk wordt vooral de werking van de *centrale* beschreven. Dit onderdeel is equivalent aan de EnerStation in het EnerBox systeem.

3.1 OPBOUW TOTALE SYSTEEM

Het systeem bestaat, net als de EnerBox, uit drie basiscomponenten:

- Sensor
- Centrale
- Pc software

De sensor meet enkele waarden (reëel vermogen en schijnvermogen) en verzendt deze naar de centrale. De sensor is geïmplementeerd in de vorm van een tussenstekker, die tussen een te meten apparaat en het stopcontact geplaatst wordt.

De software, genaamd *Energy Professional* (een Pc programma draaiend onder Microsoft Windows) kan de meetgegevens uit het centrale inlezen, om deze vervolgens inzichtelijk te maken met behulp van een aantal tabellen en grafieken. Gebruik van deze software is optioneel; het systeem werkt ook zonder deze software.

3.2 OPBOUW CENTRALE

3.2.1 UITERLIJKE KENMERKEN

De centrale is een apparaat met afmetingen van ruwweg 13x10x3cm (hoogte x breedte x diepte). De behuizing van het apparaat is van zilvergrijs plastic. Aan de voorkant van het apparaat zit een LCD display van ongeveer 6x5cm (breedte x hoogte), met daaronder drie knoppen, genaamd *SENSOR*, *ENERGY* en *POWER*. Aan de achterkant van het apparaat bevindt zich een mini-USB aansluiting en een klepje met daarachter ruimte voor drie AA-batterijen. Het apparaat kan opgehangen worden met een schroef, of het kan op een oppervlak gezet worden met een bijgeleverd voetje.

3.2.2 INWENDIGE KENMERKEN

Binnenin het apparaat bevindt zich een PCB (printed circuit board, printplaat), met daarop een aantal IC's (integrated circuits) en enkele discrete componenten. Daarnaast is er een autonome radio-ontvanger bevestigd aan het PCB. De belangrijkste componenten worden verderop beschreven en zijn de volgende:

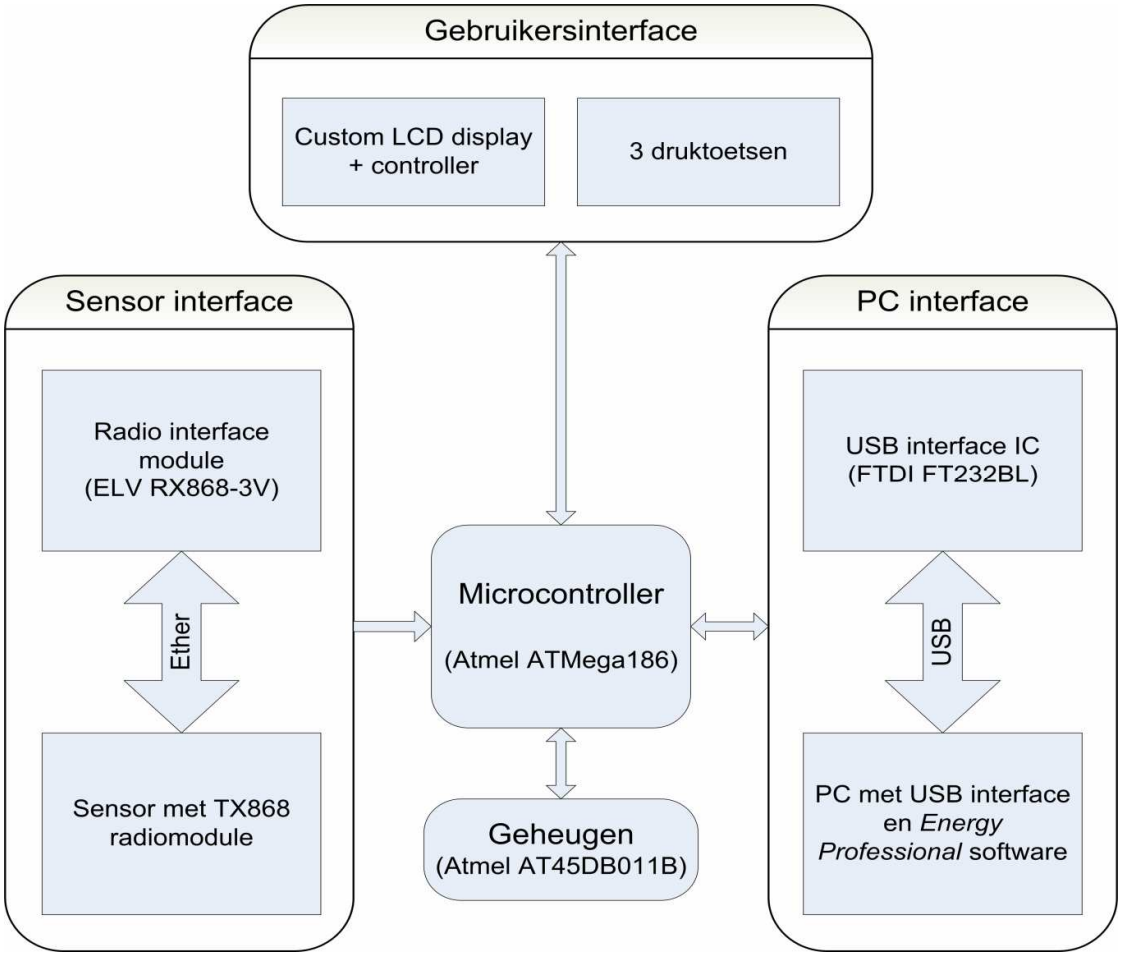
1. *FTDI FT232*BL. USB naar UART omzetter.
2. ST93C46. Geheugen van 1Kbit.
3. Atmel ATMega168. Microcontroller.
4. Atmel AT45DB011B. Geheugen van 1Mbit.
5. DC-DC converter, uitgangsspanning 3V.
6. LCD display driver IC zonder merk/typeaanduiding.
7. LCD display, specifiek ontworpen voor dit apparaat.

8. ELV RX868-3V. Radio-ontvanger module.

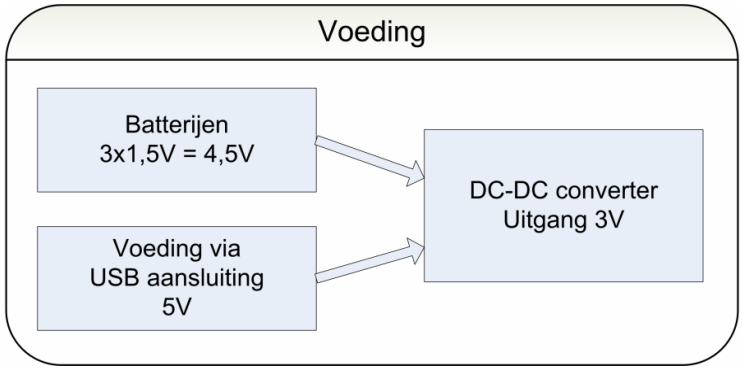
Op het PCB bevinden zich tevens een aantal kopereilandjes waar geen componenten op gesoldeerd zitten. Dit suggereert dat wellicht extra functionaliteit mogelijk is, maar is weggelaten.

3.2.3 INTERCONNECTIES TUSSEN ONDERDELEN

Wanneer gekeken wordt naar de functionaliteit en fysieke positie op de PCB van de afzonderlijke onderdelen, dan kan een beschrijving gegeven worden van de interconnecties binnen het apparaat, en daarmee de wijze van functioneren. Figuur 2 beschrijft de verbindingen tussen de onderdelen van het apparaat. Voor de duidelijkheid zijn verschillende groepen van onderdelen samen genomen. Figuur 3 beschrijft de werking van de voeding van het apparaat.



Figuur 2 - Schematische opbouw centrale



Figuur 3 - Schematische opbouw voeding centrale

3.2.4 FUNCTIONALITEIT

Het is niet mogelijk om de programmatuur die op de microcontroller draait uit te lezen. Daarom is de enige methode om hier iets over te zeggen te kijken naar de gebruikers in- en uitvoer van het apparaat. Hieruit kunnen de volgende functionaliteiten opgemaakt worden:

- Communicatie met sensoren.
- Opslag van meetgegevens.
- Berekeningen uitvoeren aan meetgegevens.
- Uitvoer naar gebruiker van diverse gegevens, per sensor.
- Communicatie met PC.

Deze functionaliteiten worden hierna verder uitgewerkt.

3.2.5 COMMUNICATIE MET SENSOREN

Het ontbreken van een radio-zendmodule in de centrale suggereert dat de informatiestroom slechts één kant op gaat: van de sensor naar de centrale. De communicatie met de sensoren is in verschillende niveaus op te delen.

Op het fysieke niveau vindt communicatie plaats door middel van radiogolven. Analyse van de radiozender- en ontvanger modules (die overigens ook los verkocht worden aan particulieren, prijs rond €13, -) wijst uit dat de specificaties van het radiokanaal onder andere de volgende zijn[2]:

- Draaggolf: 868,35Mhz
- Modulatie: Amplitude modulatie
- Maximale frequentie data: 2Khz
- Uitgang ontvanger is digitaal
- Bereik: tot 100m (line-of-sight, d.w.z. zonder obstakels)

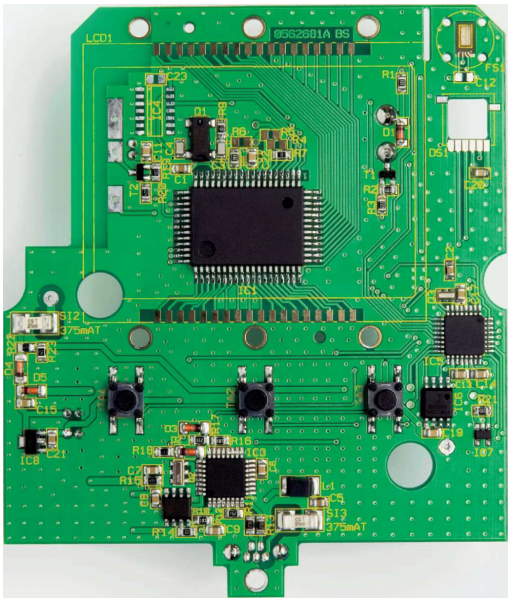
Wanneer met een oscilloscoop aan de uitgang van de ontvangermodule gemeten wordt, wordt inderdaad een digitaal signaal gemeten. Omdat de wijze van coderen onbekend is, kan over deze data echter weinig gezegd worden.

Op softwareniveau wordt de digitale data door de microcontroller in de centrale gedecodeerd. Dit levert onder andere informatie op met betrekking tot welke sensor de data zendt, en daarnaast natuurlijk de meetgegevens. De centrale kan de sensoren van elkaar onderscheiden doordat ze een vooraf door de gebruiker in te stellen adres hebben.

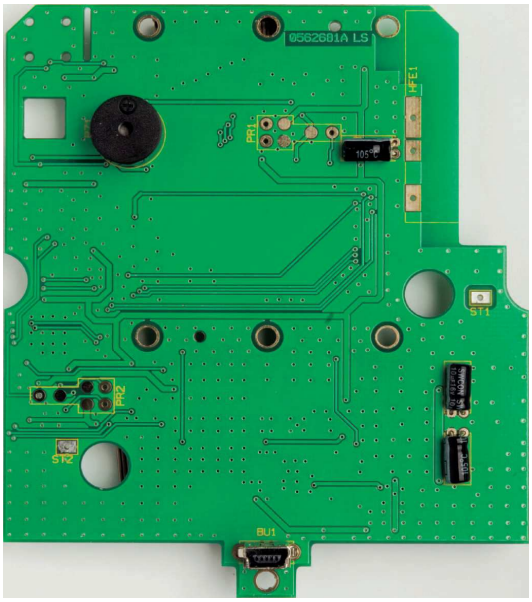
Sensoren lijken niet hetzelfde adres te kunnen/mogen hebben. Nadat de sensoren op verschillende adressen zijn ingesteld, doet het centrale er na inschakelen 12 minuten over om te synchroniseren. Daarna kunnen er meetgegevens ontvangen worden.

3.2.6 BESCHRIJVING VAN DE COMPONENTEN

In deze paragraaf staan de belangrijkste componenten van de centrale beschreven. Figuur 4 en Figuur 5 zijn afbeeldingen van de twee zijden van de PCB in het apparaat. Hierop zijn een aantal componenten zichtbaar.



Figuur 4 - Printplaat voorkant



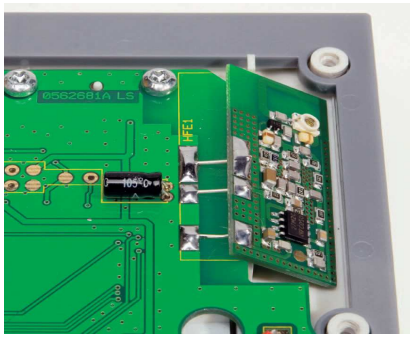
Figuur 5 - Printplaat achterkant

FTDI FT232BL - USB interface IC

De FT232BL[17] wordt gebruikt voor de USB interface. De FT232BL heeft geen processor, noch een geïntegreerde USB stack nodig om te functioneren. In een bestaand design kan het RS232 line driver IC vervangen worden door de FT232BL om te dienen als een USB slave interface. Alle verwerking van USB signalen wordt verzorgd door de FT232BL. De externe componenten betreffen een EEPROM, een kristal en enkele weerstanden en condensators.

ELV RX868-3V - Radio interface module

De radio interface module RX868-3V[2] is een kant en klare ontvangermodule, die communiceert met TX868 verzendmodule van de sensor. In Figuur 6 is het kleine PCB met daarop de radio-interface te zien.



Figuur 6 - Radio interface module

ATMEL0622 45DB011- Seriëel flashgeheugen

De AT45DB011[3] is een flash geheugen, voorzien van een seriële interface. De capaciteit is 1 Mbit. Het geheugen bevat daarnaast een SRAM data buffer van 264 bytes.

ATmega168 - Microcontroller

De ATmega168[4] is een 8-bit microcontroller van fabrikant Atmel. De ATmega168 haalt een troughput van 1 MIPS per MHz. Enkele karakteristieke eigenschappen zijn 16 Kbyte in-system Programmeerbaar flashgeheugen, 512 byte EEPROM, 1Kbyte SRAM, 23 I/O pinnen, 32 registers, drie flexibele timer/tellers met vergelijking modes, interne en externe interrupts, een seriële programeerbare USART, een byte georiënteerde 2-draad seriële interface, een SPI seriële poort, een 6-kanaal 10-bit ADC en een programmeerbare watchdog timer met interne oscillator.

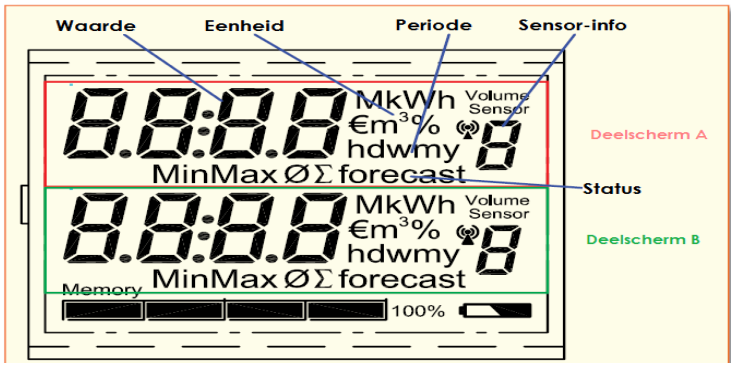
ST93C46 – Seriëel EEPROM geheugen

ST93C46[5] is een 1K bit EEPROM met een SPI interface.

LCD display

LCD display heeft afmetingen 40,5mm x 56mm. Zoals is te zien is in Figuur 7, bestaat de display uit twee deelschermen. Elke van de deelschermen kan verbruiksgegevens van een sensor laten zien.

Het verbruik kan in periodes van uur/dag/week/maand/jaar basis worden getoond. De verbruikskosten van de sensoren kunnen in euro's in uur/dag/week/maand/jaar basis getoond worden. Op het onderste deel van het scherm wordt het verbruik van het geheugen en de batterij aangegeven.



Figuur 7 - LDC display in Volcraft Energy Control 3000

4 MOGELIJKE ONTWERPEN EN AFWEGINGEN

In dit hoofdstuk wordt voor een aantal onderdelen van het systeem gekeken welke verschillende mogelijke implementaties er zijn. De mogelijkheden worden afgewogen, waarna aan de hand van diverse criteria een keuze wordt gemaakt. Vervolgens wordt de toepassing van de gekozen implementatie beschreven.

4.1 GEHEUGEN

Deze paragraaf beschrijft het geheugen dat wordt gebruikt om data afkomstig van de EnerPlug op te slaan. Volgens het programma van eisen dienen de meetgegevens van maximaal 250 EnerPlugs gedurende 1 jaar te worden bijgehouden zonder gebruik van een PC. Veder staat in het Programma van eisen beschreven, dat energieverbruik gemeten door elke EnerPlug met tijdsintervallen van 1 minuut wordt opgeslagen.

Aan de metingen die gedaan worden door de EnerPlugs moeten tijd en datum worden toegekend. Dat kan op twee manieren: door deEnerPlugs of door deEnerStation. Een uitgebreidere analyse hierover vindt u in de paragraaf *Tijd en Datum*. Daar is er gekozen om tijd en datum door de EnerStation bij te laten houden, omdat op deze manier minder data getransporteerd hoeft te worden van EnerPlugs naar EnerStation.

Verder moet er gekozen worden wat wordt gemeten door EnerPlugs. Deze keuze is ook van groot belang voor de EnerStation, omdat dataopslag en -transport ook hier een grote rol spelen. Voor een snel systeem moet de hoeveelheid datatransport zo laag mogelijk blijven. De hoeveelheid gegevens die opgeslagen dient te worden moet zo klein mogelijk zijn, om de benodigde hoeveelheid geheugen, en daarmee de kosten, te beperken.

4.1.1 GEGEVENS VAN ENERPLUG

De EnerPlug meet een aantal gegevens. Daarnaast heeft de EnerPlug de mogelijkheid om met die gegevens berekeningen uit te voeren. Er blijven vervolgens een aantal mogelijkheden over ten aanzien van de gegevens die worden verzonden naar de EnerStation:

1. stroom door elk aangesloten apparaat, de spanning over het apparaat en de fasehoek van de stroom,
2. vermogen,
3. energieverbruik in een tijdsinterval (de afgelopen minuut).

Bij de eerste optie moet er relatief veel data tussen EnerPlugs en EnerStation worden uitgewisseld. Dit betreft 2 bytes voor stroom, 1 byte voor spanning, 1 byte voor fasehoek en 2 bytes voor het adres van elke EnerPlug, totaal 6 bytes.

De tweede en de derde optie beperken zich tot 2 bytes voor vermogen/energieverbruik en 2 bytes voor het adres, totaal 4 bytes.

Gelet op het aantal te verzenden bytes genieten opties twee en drie dus de voorkeur. Omdat voor de doelgroep van de EnerBox, huishoudens, in principe alleen het energieverbruik van belang is, wordt gekozen voor optie drie.

4.1.2 BEOORDELINGSCRITERIA

In dit hoofdstuk worden eisen voor het geheugen nader beschreven. De criteria waaraan de afweging gemaakt wordt, staan in volgorde van prioriteit beschreven.

Elektrische interface

Het geheugen moet aan te sturen zijn door een microcontroller, in het geval van het prototype een 8051 compatible microcontroller.

Non-volatiliteit

Het systeem dient zonder PC 1 jaar lang te functioneren. Daarnaast mag het systeem zonder netspanning of batterijen de opgeslagen gegevens niet kwijt raken.

Deze voorwaarden beperken de keuzes voor het geheugen tot non-volatiel geheugen. Dit houdt in dat het geheugen gegevens kan bewaren wanneer het niet verbonden is aan de voeding.

Opslagcapaciteit

Het belangrijkste aspect is de capaciteit van het geheugen. Volgens het programma van eisen moet het systeem gebruikt kunnen worden met maximaal 250 EnerPlugs, en zoals is vermeld moet systeem gedurende 1 jaar al deze metingen kunnen opslaan. In Tabel 1 staat een overzicht van de opslag data.

Soort data	*Opslagfrequentie*	*Aantal bytes*	*Geheugen per jaar per EnerPlug*	*Totaal per jaar*
Energieverbruik per meting	1 x per minuut	2	1,05 MB	263 MB
Tijd en datum	Elke reeks metingen	4	210 KB	52,6 MB
Serienummer van EnerPlugs	Elke reeks metingen	2	105 KB	26,3 MB
Totaal				**341,9 MB**

Tabel 1 - Benodigde hoeveelheid geheugen voor opslag meetgegevens

Het is nodig om een referentie van tijd en datum aan de meetgegevens te koppelen. In Tabel 1 is te zien dat voor elke tijdsaanduiding 4 bytes aan data nodig is. Om hiermee rekening te kunnen houden moet een schatting worden gemaakt van hoe vaak een apparaat gemiddeld aan en uitgeschakeld wordt. Als aangenomen wordt dat in gemiddeld maximaal in 10 minuten een apparaat aan en uit geschakeld zou worden dan kosten de tijdreferenties per jaar 4 byte x 6 x 24 uur x 365 dagen = 210240 bytes aan opslagruimte. Het totaal aan benodigde opslag geheugen komt daarmee op 341,9 MB.

Kostprijs

De kosten van het geheugen zijn een belangrijk aspect, omdat in latere massaproductie de kosten zo laag mogelijk moeten blijven. Daarom is het bij het kiezen van opslaggeheugen noodzakelijk om te zoeken naar een zo goedkoop mogelijk alternatief.

Herschrijfbaarheid

Het is van belang om rekening te houden met het feit dat het geheugen heel vaak beschreven wordt. Dit wordt veroorzaak doordat er elke minuut meetgegevens opgeslagen moeten worden.

Hierdoor is noodzakelijk dat het aantal toegestane *read/write cycles* van het opslag geheugen zo hoog mogelijk is, zodat het geheugen lang genoeg meegaat.

Energieverbruik

Omdat EnerSense een energie-bezuinigingssysteem is, moet het apparaat zelf zo zuinig mogelijk zijn. De voorkeur gaat dus naar een zuinig type geheugen.

4.1.3 MOGELIJKE TYPEN GEHEUGEN

Uit de criteria blijkt dat er veel beperkingen zitten aan de geheugenkeuze. De belangrijkste zijn de non-volatiliteit en voldoende capaciteit. De mogelijkheden die in aanmerking komen zijn:

- EEPROM geheugen
- Compact flash
- xD-Picture Card
- SD-Card (Secure Digital)

EEPROM

EEPROM (Electrically Erasable Programmable Read-Only Memory) is het meest flexibele non-volatiele geheugen, omdat het per byte beschreven kan worden. Er zijn talloze soorten op de markt, waarbij met name de interface, het lage aantal pinnen en de kosten-effectiviteit[6] verschillen.

Specificaties:

- Capaciteiten liggen tussen: 1kbit en 1Mbit.
- 2 industriestandaarden voor seriële interfaces (I2C, SPI/Microwire).
- Verschillende voltages (5.5V, 2.5V, 1.8V).
- Kleinste mogelijk aantal pinnen is 8-pin IC.
- Schrijf capaciteit 1 miljoen *cycles* en data bewaartijd van 40 jaar.

Compact flash

Compact Flash[7], afgekort CF, is een non-volatiele geheugenkaart, geschikt voor gegevensopslag door middel van flashgeheugen. De opgeslagen data blijft oneindig lang bestaan. Deze zogenoemde CF-kaartjes worden vooral gebruikt in Pc's, digitale camera's en PDA's.

Specificaties:

- Capaciteiten liggen tussen: 8MB en 32GB.
- Kan tot 100 jaar gebruikt worden zonder verlies van data.
- Vermogensdissipatie is 5% van het verbruik door kleine diskdrives.
- Twee mogelijke voltages: 3.3V en 5.0V.
- 512MB kaart kost €15.70 .
- Afmetingen zijn 36mm bij 3.3mm, gewicht is 14 gram.
- Standaard 50 pinnen aansluiting.

xD-Picture Card

De xD-Picture Card[8] is een non-volatiele geheugenkaart die opgebouwd is met flashgeheugen. De afkorting *xD* staat voor *extreme digital*. xD geheugenkaarten worden gebruikt in digitale camera's van *Olympus*, Kodak en *Fujifilm*.

Specificaties:

- Capaciteiten liggen tussen: 16MB en 2GB.
- Afmetingen zijn: 20mm x 25mm x 1.78mm (lxbxh) .
- Gewicht is 2,8 gram.
- 512MB Kingston kaart kost €9.50.
- Heeft 18-pinnen en werkt op een voltage van 3.3V.
- Schrijf capaciteit is 10,000 *read/write cycles* en data bewaartijd van 10 jaar[9].

SD-Card

Een Secure Digital[10] card is een geheugenkaart, opgebouwd met flashgeheugen. SD is specifiek ontworpen voor beter beveiliging, hoge capaciteit en prestaties.

Specificaties:

- Capaciteiten liggen tussen: 128 MB en 4 GB, maar de hoog capacitive versie(SDHC) kan een capaciteit van 32 GB bereiken.
- Afmetingen zijn: 32 mm x 24 mm x 2.1 mm.
- DATA transport snelheid tussen 10-20 MByte/sec.
- Schrijf capaciteit is 100000 *read/write cycles* en data bwaartijd is10 jaar.
- Kingston 512 MB SD-Card kost €5, -.
- 9-pinnen interface (Clock, Command, 4xData en 3xPower lijnen).

4.1.4 AFWEGINGEN EN KEUZE

De genoemde alternatieven worden op grond van vermeldde specificaties afgewogen. De criteria worden op volgorde van prioriteit getoetst. Omdat genoemde alternatieven allemaal behoren tot non-volatiel opslag geheugen, valt deze voorwaarde buiten beschouwing.

Als gekeken wordt naar de capaciteit van de genoemde geheugenkaarten, dan blijkt dat EEPROM geheugen afvalt. De overige typen geheugen beschikken over voldoende capaciteit.

xD-Picture card en SD zijn het beste wanneer gekeken wordt naar afmetingen en read/write cycles. Wanneer de kosten worden vergeleken, dan is SD het meest kosteneffectief, met daaropvolgend xD en ten slotte CF.

De elektrische interface van de SD-Card is relatief eenvoudig in vergelijking met Memory Stick en CF. Er zijn vier aansluitingen nodig voor de communicatie. Desondanks kunnen snelheden in megabits per seconden gehaald worden. Dit is een groot voordeel ten opzichte van de andere geheugenstandaarden, aangezien de complexiteit van de geheugeninterface een belangrijke afwegingsfactor vormt.

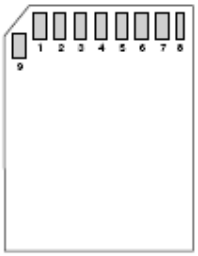
Voor het prototype is er vanwege de ruime capaciteit en lage kosten gekozen voor een SD-Card. SD-Card is tevens klein genoeg om in een schakeling op de PCB te monteren en daarmee op een microcontroller aan te sluiten.

Ook de herschrijfbaarheid van dit non-volatiele geheugen is groot genoeg om enkele jaren te kunnen functioneren.

4.2 DE WERKING VAN DE SD-CARD

4.2.1 INTERFACE

Als eerste stap is onderzocht hoe de SD-Card werkt. Figuur 8 is een schets van de fysieke vorm en aansluitingen van een SD-Card. In Tabel 2 staan de aansluitingen van de SD-Card (in SPI mode) beschreven.



Figuur 8 - Uiterlijk SD-Card

Pin #	Name	Type[1]	SPI Description
1	CS	I	Chip Select (Active low)
2	DataIn	I	Host to Card Commands and Data
3	VSS1	S	Supply Voltage Ground
4	VDD	S	Supply Voltage
5	CLK	I	Clock
6	VSS2	S	Supply Voltage Ground
7	DataOut	O	Card to Host Data and Status
8	RSV(2)	I	Reserved
9	RSV(2)	I	Reserved

NOTES: 1) S=power supply; I=input; O=output.
2) The 'RSV' pins are floating inputs. It is the responsibility of the host designer to connect external pullup resistors to those lines. Otherwise non-expected high current consumption may occur due to the floating inputs.

Tabel 2 - Aansluitingen SD-Card (SPI mode) [11]

De SD-Card ondersteunt SPI (Serial Peripheral interface) mode. Dit is een secundair communicatieprotocol van de SD-Card, en beschikt over een subset van het complete SD-Card protocol. Deze mode is ontworpen voor communicatie over een SPI verbinding, iets wat bij veel microcontrollers aanwezig is. De SPI standaard zelf beschrijft alleen de fysische link, niet het data transfer protocol.

Het SD-Card protocol is een eenvoudig commando-response protocol. Alle commando's worden gegeven door de master, waarna de SD-Card een antwoord geeft op de commando's. Afhankelijk van het commando volgt een data token, die het begin van een gegevensoverdracht of een fout aangeeft. Tijdens elk soort transactie met de SD-Card (commando, response en token), dient de CS pin laag gehouden te worden. Het SD-Card protocol heeft een set commando's, bestaande uit in totaal 60 commando's[9].

In deze context worden SD commando's aangeduid als "CMDXX" (algemene commando's) of "ACMDXX" (applicatie commando's), waarbij "XX" voor het commandonummer staat. In Tabel 3 zijn een aantal relevante commando's en beschrijvingen daarvan opgenomen.

Command	Argument	Type	Description
CMD0	None	R1	Tell the card to reset and enter its idle state.
CMD16	32-bit Block Length	R1	Select the block length.
CMD17	32-bit Block Address	R1	Read a single block.
CMD24	32-bit Block Address	R1	Write a single block.
CMD55	None	R1	Next command will be application-specific (ACMDXX).
CMD58	None	R3	Read OCR (Operating Conditions Register).
ACMD41	None	R1	Initialize the card.

Tabel 3 - Belangrijke SD-Card commando's [11]

Zoals in Tabel 4 staat, worden SD commando's gestuurd in pakketten van 6 bytes. Het commando begint altijd met "01", gevolgd door het 6-bit commando nummer. Daarna wordt een argument van 4 bytes verstuurd. Als laatste wordt een 7-bit CRC (Cyclic Redundancy Check) met een stop bit aan het eind verstuurd.

First Byte			Bytes 2-5	Last Byte	
0	1	Command	Argument (MSB First)	CRC	1

Tabel 4 - Indeling SD-Card commando

Voor de SPI mode is alleen in het begin een CRC byte belangrijk. Na dat de SD-Card in SPI mode is gezet, wordt de CRC check automatisch uitgeschakeld. Bij initialisatie wordt CMD0 gestuurd met nullen als argument. De bijbehorende CRC is 0x95, dus het complete CMD0 commando wordt 40 00 00 00 00 95 (hexadecimaal).

Als reactie op elk commando stuurt de SD-Card een response terug. Elk commando heeft een voorspelbaar antwoordtype. Dit type is alleen afhankelijk van het commando nummer, niet van de inhoud van het argument. De drie andwoordtypes zijn gedefinieerd voor de SPI mode als R1, R2 en R3. Omdat men voor eenvoudige communicatie alleen te maken krijgt met antwoord R1, wordt alleen dit responsetype beschreven in Tabel 5.

Byte	Bit	Meaning
1	7	Start Bit, Always 0
	6	Parameter Error
	5	Address Error
	4	Erase Sequence Error
	3	CRC Error
	2	Illegal Command
	1	Erase Reset
	0	In Idle State

Tabel 5 - Antwoord type R1

Antwoord token

Elke keer dat er geschreven wordt naar de SD-Card, wordt er een 1-byte token terug gestuurd dat de status van de handeling aan geeft. Deze token is "XXX0AAA1", waar XXX niet uitmaken, en AAA de status van de operatie geeft. "010" betekent bijvoorbeeld dat de data is geaccepteerd, "101" betekent dat de data niet is geaccepteerd vanwege CRC error en "110" betekent dat data niet is geaccepteerd vanwege een schrijffout[11].

4.2.2 AANSTURING

Initialisatie

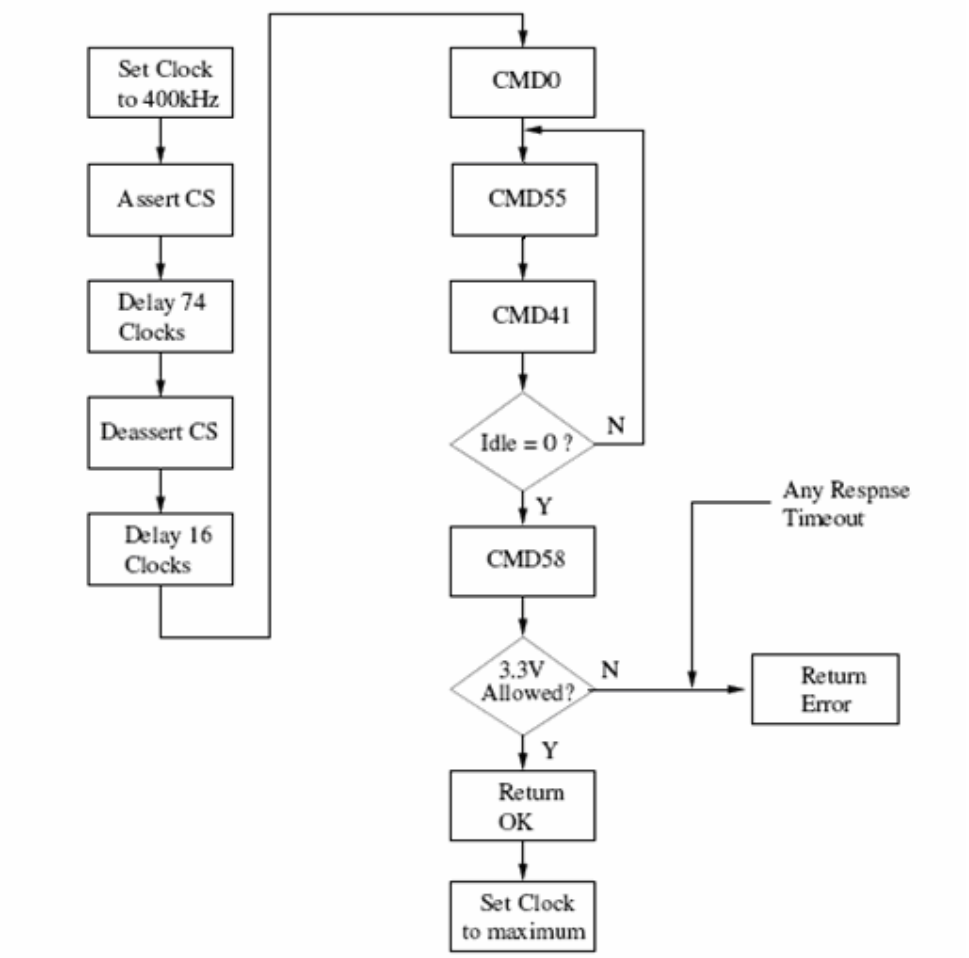
Nadat SD-Card een voedingsspanning krijgt, gaat deze direct in zijn SD mode. De SD-Card gaat in SPI mode gaan als de CS pin laag gehouden wordt en het reset commando CMD0 naar de DI pin wordt gestuurd.

De SD-Card initialisatie begint met het instellen van SPI clock op 400kHz. Vervolgens moet er minstens 74 klokpulsen worden gewacht, voor dat de master (de microcontroller) communiceert met de SD-Card. Dit zorgt ervoor dat de SD-Card alle interne state registers initialiseert voordat het initialisatieproces doorgaat.

Daarna wordt de SD-Card gereset door het sturen van commando CMD0 en tegelijkertijd laag houden van de CS pin. Deze beide handelingen resetten de SD-Card en stellen deze in op SPI mode.

Voor elk applicatiecommando moet eerst een commando CMD55 worden gegeven aan de SD-Card. Daarna wordt applicatiecommando ACMD41, dat aangeeft dat de SD Card zich moet initialiserejn, gestuurd naar de card. Deze twee stappen worden herhaald totdat de idle bit van het antwoordregister nul is. Dit betekent de SD-Card is geinitialiseerd en klaar is om andere commando's te ontvangen.

Hierop volgt commando CMD58. Dit commando test of de SD-Card de uitgangsspanning van de microcontroller wel aan kan. Het antwoord op CMD58 is een bitveld met het toegestane spanningsbereik. Voor de SD-Card is dat tussen 2,7V en 3,6V. Ten slotte wordt de SPI klokfrequentie op de maximaal toelaatbare waarde gezet. Het totale initialisatie proces is uitgewerkt in de vorm van een state diagram in Figuur 9.



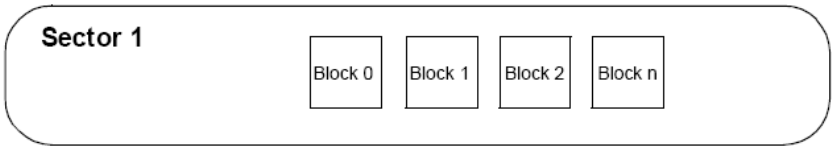
Figuur 9 - SD-Card initialisatieproces

Gegevensoverdracht

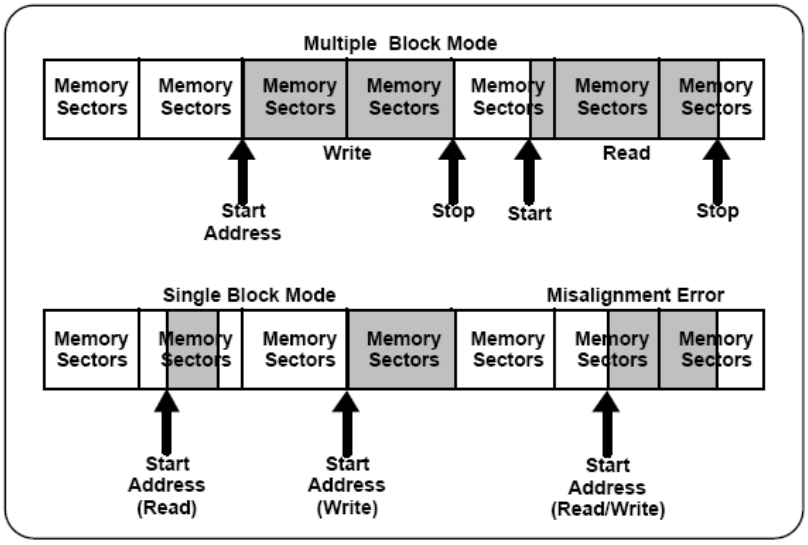
Voor elke schrijf- of leesoperatie moet de SC-Card in transfer mode worden gezet met het transfer commando (CMD7). De SPI mode ondersteunt gegevensoverdracht met één en met meerdere datablokken.

De basiseenheid voor gegevensoverdracht van en naar de SD-Card is één byte. Alle gegevensoverdracht gebeurt in een veelvoud van bytes.

Behalve het datablok is nog de zogenaamde sector, die uit meerdere datablokken bestaat. Dit wordt weergegeven in Figuur 10. Sectoren zijn gerelateerd aan wisfunctie van de SD-Card. De grootte van een sector staat vast voor elke SD-Card.



Figuur 10 - Sector op een SD-Card



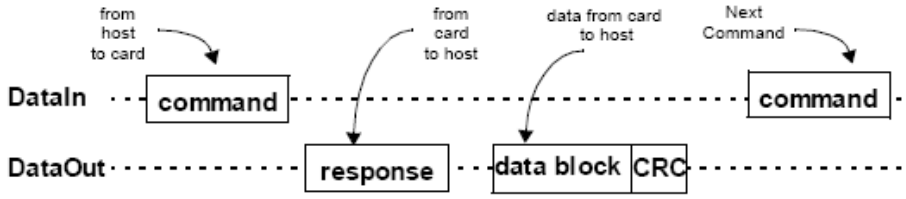
Figuur 11 - Data transfer formaten [12]

In Figuur 11 staat een diagram dat de lees- en schrijfmodes van de SD-Card beschrijft. Gegevensoverdracht met één datablok heeft een van tevoren gedefinieerde grootte. De blokgrootte van de lees operatie kan variëren tussen 1 en 512 bytes. Het is echter niet toegestaan de bovengrens van een blok te overschrijden (misalaignment, zie Figuur 11). De blokgrootte van een schrijfoperatie moet identiek zijn aan de sector grootte, en het startadres moet precies op de sectorgrens zitten.

Gegevensoverdracht met meerdere blokken is verglijkbaar met gegevensoverdracht met één blok, met het verschil dat de master meerdere datablokken kan lezen en schrijven. Deze worden gelezen of geschreven op het start adres dat in het argument van het commando is meegestuurd. Deze gegevensoverdracht stopt met een stop transmissie commando. De master dient de geschreven en gelezen adressen bij te houden.

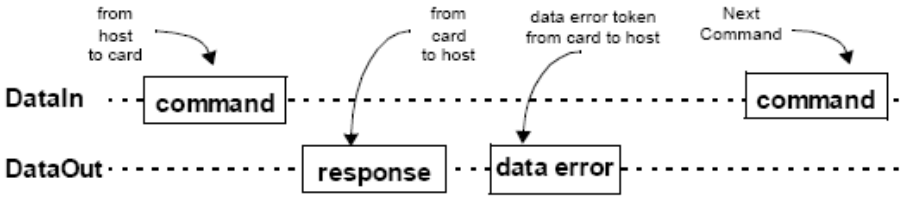
Lees datablok

In Figuur 12 staat een leesoperatie weergegeven. Nadat een geldig leescommando (CMD17) wordt ontvangen door de SD-Card, zal deze een antwoordbyte terugsturen, gevolgd door een datablok. De lengte van het blok moet eerder gedefinieerd zijn met het SET_BLOCKLEN (CMD16) commando.



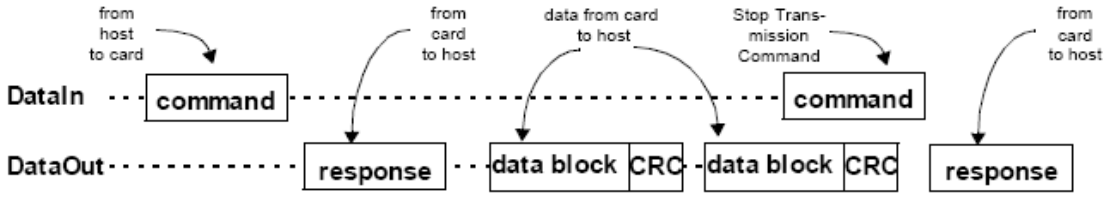
Figuur 12 – Leesoperatie met enkel datablok

In het geval van een fout in de leesoperatie, zal de SD-Card geen data terug sturen. Zoals in Figuur 13 te zien is, stuurt deze in plaats daarvan een datafout token. De microcontroller moet het leescommando dan opnieuw sturen.



Figuur 13 - Leesoperatie met datafout

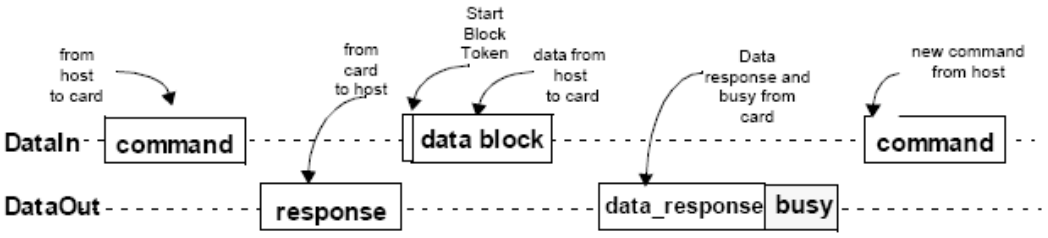
In het geval van een leesoperatie met meerdere datablokken, stuurt de SD-Card een continue reeks datablokken, nadat de microcontroller een lees commando heeft gestuurd. De data reeks stopt na een stop transmissie commando (CMD12). Figuur 14 is een diagram van deze leesoperatie.



Figuur 14 - Leesoperatie met meerdere datablokken

Schrijf datablok

Net als bij een leesoperatie, moet de SD-Card bij een schrijfoperatie eerst in transfer mode gezet worden met CMD7. De bloklengte moet worden ingesteld met CMD16. Figuur 15 geeft deze operatie weer. De operatie start met een schrijfcommando (CMD24), met het startadres in het argument. Nadat antwoord van de SD-Card ontvangen is, wordt het datablok verstuurd, met aan het begin het startblok token (1 byte). De enige geldige blok lengte is 512 bytes.



Figuur 15 - Schrijfoperatie met enkel datablok

Zodra het datablok ontvangen is door de SD-Card, en er geen foutrespons wordt terug gestuurd, begint de SD-Card de data in het geheugen cellen programmeren. Gedurende de tijd dat SD-Card bezig is met programmeren wordt een continue reeks van "bezig" datatokens gestuurd. Dit zorgt ervoor dat DataOut laag blijft. Nadat de programmeeroperatie voltooid is, moet de microcontroller het resultaat checken met een zend status commando (CMD13).

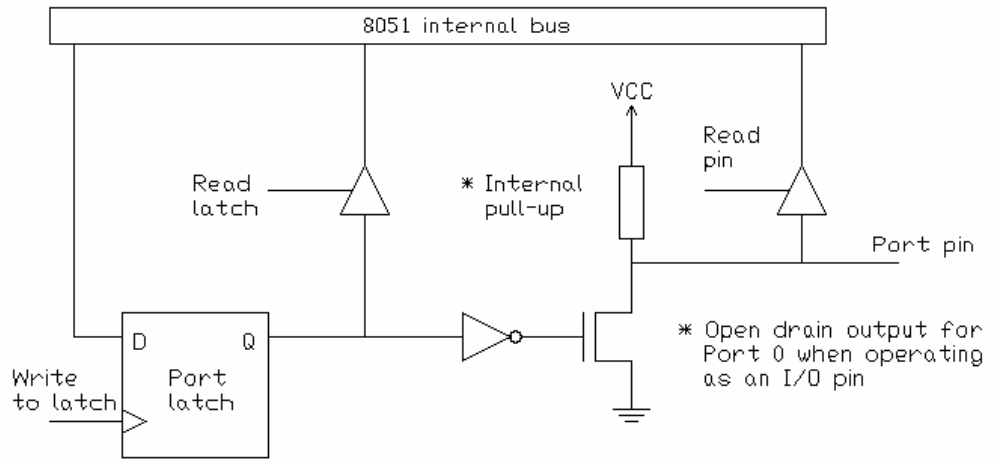
Bij schrijfoperatie met meerdere datablokken, beginnend met commando CMD25, moet de microcontroller wachten op een response. Daarna kunnen de datablokken (met startblok

token aan het begin van elk blok) worden verstuurd naar SD-Card. De operatie wordt gestopt met een "stop transmissie" token.

4.3 IMPLEMENTATIE AANSTURING SD-CARD

4.3.1 POORTEN VAN DE MICROCONTROLLER

Figuur 16 geeft een beschrijving van de pinnen van 8051 poorten P1, P2 en P3. Wanneer de poortregisters worden beschreven, wordt er een bit geladen in de bijbehorende port latch. Deze latch stuurt via een inverter een transistor aan. Wanneer de transistor in geleiding is, is de pin verbonden met de aarde en staat er een logische nul op de pin. Wanneer de transistor niet in geleiding is, is de pin met een pull-up weerstand verbonden met Vcc en staat er een logische één op de poort.



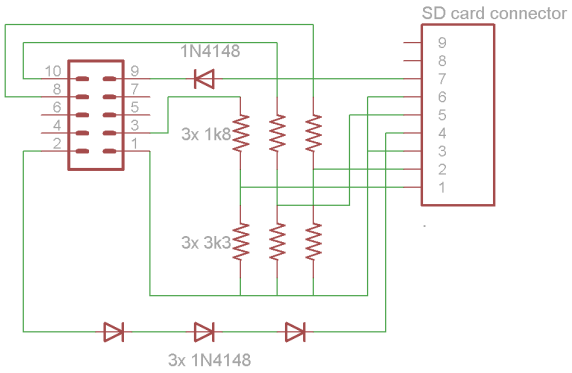
Figuur 16 - Interne schakeling poortpinnen 8051

Op elk moment is het mogelijk om een poort te gebruiken als output. Om de pin te gebruiken als input, moet de transistor uit zijn, omdat de pin anders permanent met de aarde verbonden zou zijn.

De SPI-module van de 8051 zit verbonden aan pinnen 4-7 van poort 1. Indien de microcontroller in SPI-mode wordt ingesteld, kan de microcontroller als slave functioneren, met behulp van slave-select pin [P1.4]. Deze pin kan dus niet gebruikt worden om een extern apparaat slave te maken.

4.3.2 HARDWARE-INTERFACE

Aan de hand van bovengenoemde informatie is een schakeling ontworpen om een SD-Card te kunnen aansturen met behulp van een 8051 microcontroller. Deze schakeling is te zien in Figuur 17.

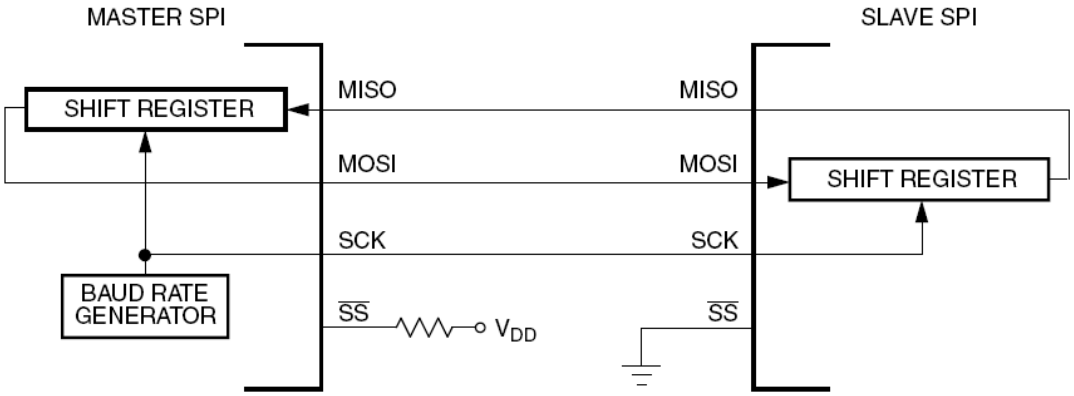


Figuur 17 - Schakeling voor aansturen SD-Card

De schakeling zorgt er middels spanningsdelers voor dat uitgangsspanning van microcontroller van 5V omgezet wordt naar de ingang spanning voor SD-Card van ongeveer 3,5V. De voedingsspanning wordt verlaagd door de drie diodes in serie. Hierover onstaat een spanningsval van rond de 2v, zodat de voedingsspanning van de SD-Card ongeveer 3V bedraagt. De enige verbinding die zonder spanningsconversie is aangesloten, is Data Out. Dit is mogelijk doordat de SD-Card intern een aansturing heeft met een laagohmig pad naar Vcc. De SD-Card forceert de pin daardoor naar een veilig spanningsniveau.

4.3.3 SPI COMMUNICATIE

Om een beter begrip te krijgen van SPI communicatie is eerst onderzocht hoe deze functie werkt bij de 8051 microcontrollers. SPI staat voor Serial Peripheral Interface. In SPI mode communiceren de betrokken apparaten in master/slave mode. Hierbij draagt de master zorg voor de aansturing. Het is mogelijk om meerdere individuele slaves met individuele slave select (SS) aansluitingen in een databus te hebben.



Figuur 18 - SPI interface diagram

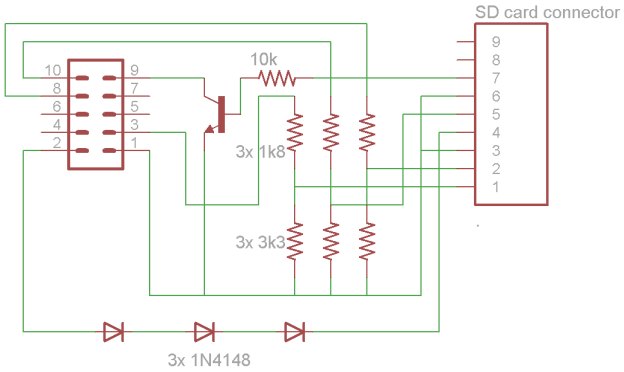
De SD-Card SPI interface is geschikt voor alle SPI controllers. Zoals in Figuur 18 is weergegeven, bestaat SPI bus uit vier signalen. De aansluitingen zijn als volgt:

- [P1.0] – SS (chip select, actief laag).
- [P1.7] – CLK (clock).
- [P1.5] – MOSI – "Master Out Slave In"(data van microcontroller naar SD-Card).
- [P1.6] – MISO – "Master In Slave Out" (data van SD-Card naar microcontroller).

Zoals te zien is aan de beschrijving is er voor communicatie met de SD-Card gekozen voor poort P1 van de P89V51RD2. Reden hiervoor is dat de SPI module van de P89V51RD2 met deze poort verbonden is. De 5V I/O pinnen P1.0, P1.5 en P1.7 van de microcontroller worden elk met een aparte spanningdeler van 1k8 en 3k3 omgezet naar de 3,3V uitgangsspanning die geschikt is voor SD-Card.

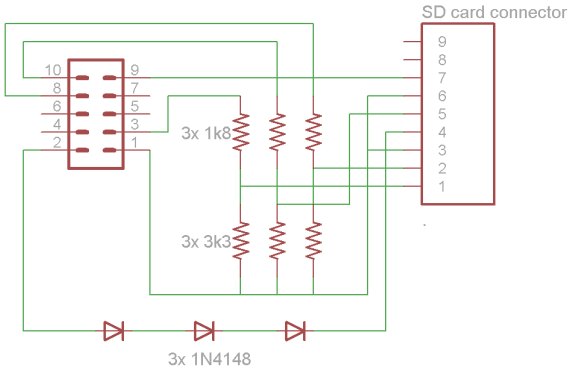
Nadat deze schakeling is vervaardigd, is er een poging gedaan om een reset commando via SPI mode naar de SD-Card te sturen. Uiteindelijk is erin geslaagd de SD-Card een antwoord te laten geven, maar het is niet gelukt dit antwoord in de microcontroller te ontvangen.

Toen bleek dat door de diode tussen de MISO verbinding, het signaal bij de uitgang van de SD-Card niet overeen kwam met het signaal bij pin P1.6 van de microcontroller. Hierna is onderzocht hoe de poorten van de P89V51RD2 in elkaar zitten. Dit is te vinden in paragraaf 4.3.1. Vervolgend is de schakeling aangepast naar Figuur 19, om het DO signaal van de SD-Card te kunnen ontvangen bij pin P1.6 pin van microcontroller.

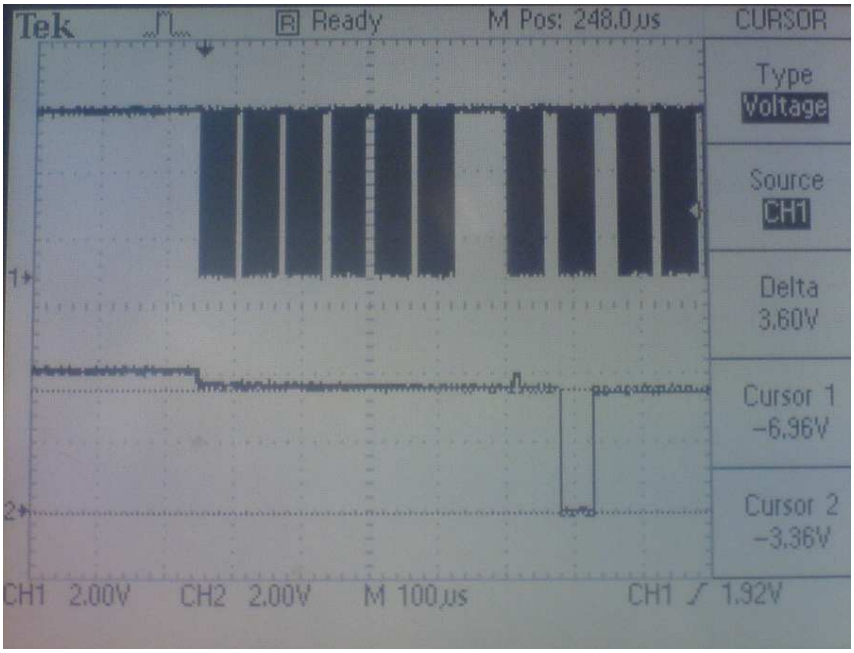


Figuur 19 - Schakeling voor aansturen SD-Card met transistor

Na het testen van deze schakeling is gebleken dat het signaal wel met geschikte spanning van 3,5 V op de DO pin van de SD-Card kwam te staan. Het probleem dat volgde was dat het DataOut signaal bij de bij pin P1.6 van de microcontroller geïnverteerd was door de toegevoegde transistor en weerstand. Na veel zoeken op internet naar de SD interfaces gemaakt voor andere microcontrollers die op 5 V werken, is gebleken dat het verbinding tussen DO van SD-Card en pin P1.6 van microcontroller ook rechtstreeks kan. Deze uiteindelijke schakeling is afgebeeld in Figuur 20.

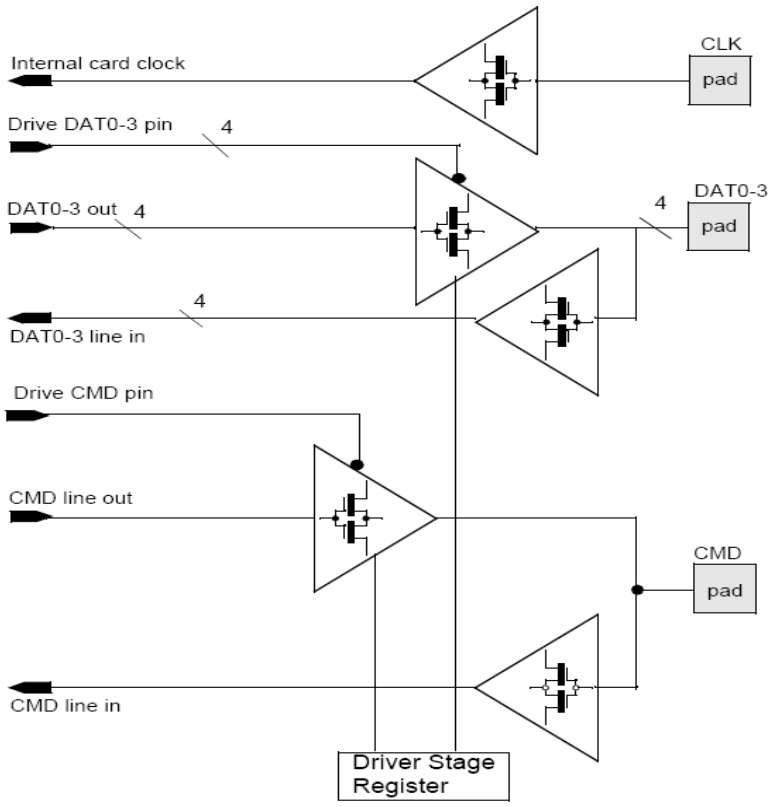


Figuur 20 - Schakeling voor aansturen SD-Card met directe verbinding



Figuur 21 - Spanningsval op DO aansluiting

Van deze schakeling is onderzocht hoe het DO signaal zich gedraagt tijdens gegevensoverdracht. Figuur 21 is een afbeelding van het oscilloscoop scherm. De bovenste zwarte lijn is het kloksignaal (CLK), de onderste lijn geeft de DO aansluiting weer. Op de afbeelding is zichtbaar dat het DO signaal in eerste instantie (links) 4,1V is. Wanneer de transmissie begint (te herkennen aan de zwarte blokken van het kloksignaal), wordt de spanning 3,6V. Een betere oplossing zou het gebruik van een voltage level translator zijn, een specifiek onderdeel voor de omzetting van 5V naar 3,3V en omgekeerd.



Figuur 22 - SD-Card bus driver

Figuur 22 is een diagram van de aansturing van de pinnen binnen de SD-Card. Te zien is dat de DO lijn (de CMD pad) aangestuurd wordt door een actieve driver, bestaande uit FET's. Dit veroorzaakt een zeer laag-ohmige uitgang. De combinatie met de 90 kΩ pull-up weerstand in de microcontroller, verklaart het feit dat de SD-Card de spanning op 3,6V kan instellen, terwijl de pin van de microcontroller hoog staat ingesteld.

4.4 MICROCONTROLLER

De microcontroller is meestal het belangrijkste onderdeel in een embedded systeem, aangezien dit onderdeel alle regelfuncties vervult. Daarom is het bij de keuze van een microcontroller belangrijk dat er goede selectiecriteria gesteld worden ten aanzien van de benodigde functionaliteit.

In dit hoofdstuk worden de benodigde functionaliteiten van de microcontroller op een rijtje gezet en worden er afwegingen gemaakt.

Naast de genoemde criteria wordt tevens gekeken naar de kosten van de microcontroller.

4.4.1 GEHEUGEN

Ten aanzien van het geheugen zijn er twee eisen. Ten eerste moet het mogelijk zijn om een aantal instelbare parameters te bewaren, ook als het appraat uit staat. Hiervoor is een kleine hoeveelheid EEPROM geheugen geschikt. Ten tweede moeten meetgegevens gebufferd kunnen worden, voordat deze in het permanente geheugen worden opgeslagen. Samen met de overige variabelen, is 1 kbyte voldoende.

4.4.2 POWER MANAGEMENT EN LAAG VOLTAGE

Voor een energiezuinig systeem dient de gekozen microcontroller softwarematig te kunnen worden ingesteld in een zogenaamde idle modus. In deze modus blijven registers en uitgangen gelijk, en worden alle activiteiten gestaakt, behalve de volgende:

- toegevoegde on-board oscillator schakeling,
- watchdog logic (watchdog timer),
- clock monitor,
- idle timer (een onafhankelijke timer).

Het energieverbruik van de microcontroller bedraagt in deze modus ongeveer 30% van het normale energieverbruik.

4.4.3 I/O

Omdat er meerdere externe devices worden aangesloten op de microcontroller in de EnerStation, dient de microcontroller te beschikken over verschillende I/O mogelijkheden. De volgende I/O mogelijkheden zijn van belang voor EnerStation:

- UART (Universal Asynchronous Receiver Transmitter), een seriële port adapter voor asynchrone seriële communicatie, bijvoorbeeld RS-232.,
- SPI (Serial Comunications Interface), een synchrone seriële poort.

4.4.4 INTERRUPTS

Een interrupt dient ervoor een programma te onderbreken, wanneer een programma voor een systeemonderdeel uitgevoerd moet worden. Bijvoorbeeld wanneer er gegevens ontvangen zijn via de UART of wanneer een timer afloopt. Omdat er meerdere devices zullen aangesloten worden op de microcontroller en er verschillende interupts zullen worden gebruikt, is het van belang dat er prioriteiten kunnen worden toegekend aan de interrupts.

4.4.5 KEUZE MICROCONTROLLER

In eerste instantie is gekozen voor de 8051 microcontroller. Deze microcontroller bezit standaard de genoemde functionaliteiten. Daarnaast is de instructieset van de 8051 eenvoudig. Dit heeft als voordeel dat programmeren eenvoudig is, en dat geschreven code eenvoudig omgezet kan worden naar de andere 8051 modellen. Veel van de instructies adresseren direct de I/O pinnen, waardoor een snelle manipulatie (bit-banging) van externe devices mogelijk wordt. Er zijn veel verschillende 8051 compatible microcontrollers, waardoor beschikbaarheid goed is en de kosten laag zijn.

De 8051 microcontroller heeft de volgende kenmerken:

- Beschikt over meerdere functies zoals: CPU, RAM, ROM , I/O, interrupt logic, timer, etc. in een enkel pakket.
- 8-bit databus, hij kan 8 bits van de data in één operatie bewerken.
- 16-bit adres bus, met ondersteuning voor 64 kB van zowel RAM en ROM.
- vier bidirectionele input/output (I/O) poorten
- UART (seriële poort)
- twee 16-bit timers
- twee niveaus van interrupt prioriteiten
- IDLE mode

De 8051 microcontroller is een 8-bit microcontroller. Deze is generatie microcontrollers zijn zo populair dat er wel meer dan twintig verschillende types op de markt zijn die allemaal opgebouwd zijn rond dezelfde architectuur. Deze controllers worden aangeboden door ondermeer de fabrikanten Intel, NXP Semiconductors, Infineon, Texas Instruments en Atmel.

Voor het implementeren van EnerStation is er gekozen voor P89V51RD2 van NXP Semiconductor. De belangrijkste reden om deze microcontroller te gebruiken voor het prototype is dat de hardware reeds beschikbaar is. Daarnaast beschikt deze microcontroller over de benodigde functionaliteiten voor de EnerStation. De onderwerpers hebben tevens ervaring met deze microcontroller. Voor een volledige beschrijving van hardware schakeling waar in P89V51RD2 is geïntegreerd wordt gerefereerd naar Computer Architectuur en Organisatie (CAO) practicumhandleiding[13].

4.4.6 NXP P89V51RD2 MICROCONTROLLER

De P89V51RD2 is een 8051 microcontroller met 64 kB Flashgeheugen en 1024 bytes data RAM. Een interessante eigenschap van deze microcontroller is de zogenaamde X2-modus. Deze functionaliteit stelt de microcontroller instaat om de effectieve snelheid van de 8051 (12 klokpulsen per instructiecyclus) te verdubbelen (6 klokpulsen per instructiecyclus). Het is mogelijk om de microcontroller te programmeren met In-System Programming (ISP). Dit houdt in dat de microcontroller geprogrammeerd kan worden via de seriële poort.

Daarnaast is de P89V51RD2 ook In-Application Programmable (IAP), waardoor het mogelijk is om Flash programma geheugen vanuit de applicatie te programmeren[24].

De belangrijkste kenmerken voor de EnerStation van de P89V51RD2 zijn:

- 5V voedingsspanning bij een klokfrequentie tot 40 Mhz,
- SPI (Serial Peripheral Interface) en UART,
- vier 8-bit I/O poorten,
- drie 16-bit timers/counters,
- programmeerbare Watchdog timer (WDT),
- acht interruptbronnen met vier prioriteitsniveaus,
- TTL- en CMOS-compatible logische niveaus,
- low power modi,
- power down modus with external interrupt wake-up,
- Idle modus,
- PDIP40, PLCC44 and TQFP44 behuizingen.

4.5 COMMUNICATIE MET ENERVIEW

De opzet van het totale systeem vereist dat er gegevens kunnen worden verzonden van de EnerStation naar de PC, met daarop EnerView en vice versa. Dit hoofdstuk beschrijft de verschillende mogelijkheden die er voor communicatie zijn, weegt deze af en beschrijft vervolgens in meer detail de gekozen implementatie.

4.5.1 MOGELIJKHEDEN VOOR COMMUNICATIE

Om een beeld te krijgen van de verschillende manieren die bestaan om apparatuur met een PC te laten communiceren, wordt geanalyseerd over welke communicatiekanalen een moderne PC beschikt. Uit het programma van eisen volgt dat de PC niet fysiek aangepast mag worden, en dat realtime weergeven van de meetgegevens mogelijk moet zijn.

De mogelijkheden die dan overblijven zijn verbindingen, in de vorm van een kabel of een draadloze verbinding. Uit een kort onderzoek naar de PC's en laptops die in gebruik zijn op faculteit EWI van de TU Delft blijkt dat deze over de volgende relevante verbindingsmogelijkheden beschikken:

- Seriële poort (RS-232[14])
- Parallelle poort (IEEE 1284[15])
- USB
- Netwerkaansluiting (Ethernet)
- Bluetooth (draadloze radioverbinding)

Deze verbindingen worden vergeleken op basis van snelheid, kosten, implementatiegemak, gebruiksgemak en beschikbaarheid. De kosten zijn een ruwe schatting aan de hand van prijzen van diverse elektronicaleveranciers, waaronder Farnell.

Seriële poort

De maximale snelheid van de seriële poort is 115,2 Kbit/s[2]. Hardware om een seriële verbinding te implementeren kost ongeveer 3 euro, een gedeelte van de benodigde hardware zit in de meeste microcontrollers geïntegreerd. Het implementeren van deze verbinding is eenvoudig, daar er veel voorbeelden en documentatie voorhanden is[3]. Het gebruiken van deze verbinding is eenvoudig, op een PC vereist een seriële verbinding geen extra stuurprogramma's en volstaat het om een kabel aan te sluiten. Een nadeel van de seriële poort is dat deze steeds minder gebruikt wordt, en daarom niet vanzelfsprekend op elke standaard PC aanwezig is.

Parallelle poort

De maximale snelheid van de parallelle poort ligt rond de 2 Mbit/s[16]. Hardware om een parallelle verbinding te implementeren kost ongeveer 3 euro. Implementatie van een parallelle verbinding is hardwarematig gezien eenvoudig omdat een apparaat (bijvoorbeeld een microcontroller) in principe direct aangesloten kan worden op een PC. Voor de gebruiker is een parallelle verbinding eenvoudig tot stand te brengen, het aansluiten van een kabel volstaat. Ook de parallelle poort wordt steeds minder gebruikt.

USB

De maximale snelheid van een USB verbinding is 12 Mbit/s[17]. Wanneer gebruikt wordt gemaakt van een USB naar RS-232 converter[4] daalt die snelheid tot maximaal 1 Mbit/s. Hardware om een USB verbinding te implementeren kost rond de 8 euro. Implementatie van een USB verbinding is lastiger dan bij een seriële verbinding. Dit komt doordat er meer en gecompliceerdere componenten gebruikt worden. Softwarematig vereist een USB verbinding aanzienlijk meer werk, aangezien het verzenden van data significant meer instellingen en acties vereist dan bij bijvoorbeeld een seriële verbinding. Het gebruikersgemak van een USB verbinding is meestal vergelijkbaar met dat van een seriële verbinding, maar uitzonderingen doen zich voor wanneer de gebruiker niet alle beheerdersrechten op een PC

[2] Dit is de maximale snelheid die ingesteld kan worden op een PC.

[3] Het kostte in dit project slechts enkele uren om het 8051 bordje te laten communiceren met een PC.

[4] Op basis van de implementatie in de Conrad Energycontrol 3000[1].

heeft. In dit geval kan een verbinding niet tot stand gebracht worden. De beschikbaarheid van USB is goed, vrijwel elke moderne PC heeft een aantal USB aansluitingen.

Netwerkaansluiting

De maximale snelheid van een netwerkaansluiting is, afhankelijk van de gekozen standaard, 10, 100 of 1000 Mbit/s. Dit is het theoretisch maximaal haalbare, voor een 100 Mbit/s ethernet verbinding is de maximale snelheid in de praktijk minder dan 11 Mbyte/s[18]. Hardware om een verbinding via een netwerkaansluiting te implementeren kost rond de 8 euro[5]. Implementatie is ingewikkeld, omdat er op hogere niveaus gecommuniceerd wordt dan bij bijvoorbeeld USB, het simpelweg verzenden van bytes volstaat niet. Voor een gebruiker kost het tot stand brengen van een netwerkverbinding meer moeite, daar er doorgaans specifieke dingen ingesteld moeten worden en er soms extra apparatuur benodigd is. De beschikbaarheid varieert. Hoewel de meeste Pc's over een netwerkverbinding beschikken, is deze meestal in gebruik. Extra netwerkaansluitingen kunnen wel gecreëerd worden, maar dit vereist dan extra apparatuur.

Bluetooth

Momenteel is Bluetooth[19] de enige veelgebruikte draadloze verbinding voor randapparatuur van Pc's. Met behulp van het *Cable Replacement Protocol* (RFCOMM), kan een RS-232 verbinding tot stand gebracht worden. De maximale snelheid is dan 1 Mbit/s[20]. Hardware om een Bluetooth verbinding te implementeren kost, in het geval van een kant en klare module rond de 30 euro. Implementatie van een Bluetooth verbinding met behulp van een kant en klare module is eenvoudig, daar het in principe vergelijkbaar is met implementatie van een seriële verbinding. Gebruik van deze verbinding is iets omslachtiger dan een seriële verbinding omdat er op de PC een aantal zaken ingesteld moeten worden. De beschikbaarheid van Bluetooth varieert. Vooralsnog heeft een groot deel van de Pc's nog geen Bluetooth.

4.5.2 AFWEGING

De gevonden mogelijkheden kunnen op basis van de genoemde criteria beoordeeld en afgewogen worden. Omdat voor het prototype andere eisen gelden dan voor een hypothetisch op de markt te brengen eindproduct, zullen twee verschillende keuzes gemaakt worden.

Snelheid

Om tussen de verschillende mogelijkheden te kunnen kiezen, wordt eerst gekeken naar de benodigde verbindingssnelheid. Wanneer er vanuit wordt gegaan dat er maximaal 250MB aan data verzonden moet worden naar de PC, dan kan uitgerekend worden hoeveel tijd dit ongeveer per verbinding kost:

1. Netwerk – 3 minuten
2. Parallelle poort – 15 minuten
3. USB en Bluetooth als RS-232 converter – 30 minuten
4. Seriële poort – 4 uur 13 minuten

Wat direct opvalt, is dat de seriële verbinding er in negatieve zin uit springt. De overige opties zijn allemaal acceptabel, wanneer er rekening mee gehouden wordt dat de gegeven hoeveelheid data slechts zelden zal voorkomen.

[5] Op basis van implementatie mbv Microchip ENC28J60 ethernet controller

Kosten

Wanneer gekeken wordt naar de kosten van de verschillende mogelijkheden, ontstaat de volgende indeling:

1. Seriële en parallelle poort – 3 euro
2. USB en Netwerkaansluiting – 8 euro
3. Bluetooth – 30 euro

Hieruit blijkt dat de kosten van een seriële of een parallelle verbinding het laagst zijn. Bluetooth is een dure uitschieter, maar hier moet bij vermeld worden dat deze kosten zijn berekend op basis van een kant en klare oplossing. Deze kosten kunnen in massaproductie aanzienlijk lager uitvallen.

Implementatiegemak

Het gemak van implementatie hangt onder meer af van ervaring met een bepaalde technologie, die natuurlijk kan verschillen per ontwerper. Omdat hier geen algemene uitspraken over te doen zijn, zal er vanuit gegaan worden van een situatie waarbij de ontwerper geen ervaring heeft met de betreffende technologie. Daarnaast kan onderscheid gemaakt worden tussen de hardware- en de softwarekant (microcontroller) van de implementatie.

De complexiteit van de benodigde hardware is (van laag naar hoog):

1. Parallelle poort – Directe verbinding tussen microcontroller en PC.
2. Seriële poort – Conversie van signaalniveau nodig tussen microcontroller en PC, bestaande uit IC en enkele discrete componenten.
3. Bluetooth – Bij onderzochte configuratie is een converter nodig tussen RS-232 en Bluetooth, bestaande uit geïntegreerde module. Er is geen kabel nodig.
4. USB – Bij onderzochte configuratie is een converter nodig tussen RS-232 en USB, bestaande uit IC, kristal en enkele discrete componenten.
5. Netwerkaansluiting – Bij onderzochte configuratie is een netwerkinterface nodig, bestaande uit een IC, 15 discrete componenten een ethernet transformator.

Op software niveau is de complexiteit vergelijkbaar (van laag naar hoog):

1. Parallelle poort – Bytes kunnen simpel gelezen en geschreven worden door de i/o pinnen direct aan te sturen.
2. Seriële poort, Bluetooth, USB – De UART dient ingesteld te worden, waarna bytes eenvoudig gelezen en geschreven kunnen worden.
3. Netwerkaansluiting – Het gebruik van een netwerkaansluiting vereist dat er op hogere niveaus gecommuniceerd wordt. Hoe dit in zijn werk gaat valt buiten deze thesis, zie voor meer informatie [21].

Op zowel hardware- als softwareniveau is de parallelle poort het makkelijkst te implementeren, gevolgd door de seriële poort. Overige implementaties zijn lastiger, met op enige afstand van de rest de netwerkaansluiting.

Gebruiksgemak

Om het gebruiksgemak te vergelijken wordt gekeken wat een gebruiker zou moeten doen om een bepaalde verbinding tot stand te brengen. In volgorde is dit:

1. Parallelle en seriële poort – Kabel aansluiten.
2. USB – Kabel aansluiten en eventueel stuurprogramma installeren.

3. Bluetooth – PC instellen.
4. Netwerkaansluiting – Kabel aansluiten en PC instellen.

Hieruit blijkt dat wat gebruiksgemak betreft de parallelle en seriële kabel het winnen. Het installeren van een stuurprogramma voor USB telt niet zwaar, aangezien de gebruiker toch al een programma moet installeren.

Beschikbaarheid

Als wordt gekeken naar beschikbaarheid, kan de volgende indeling gemaakt worden, met de hoogste beschikbaarheid bovenaan:

1. USB – Vrijwel alle moderne Pc's.
2. Seriële en parallelle poort – Een redelijk groot deel van de Pc's, maar dit wordt wel steeds minder.
3. Netwerkaansluiting – Vrijwel alle moderne Pc's beschikken erover, maar deze is niet altijd beschikbaar.
4. Bluetooth – Vrij weinig Pc's, maar dit worden er wel steeds meer.

Uit deze gegevens blijkt dat USB duidelijk de beste beschikbaarheid heeft.

4.5.3 KEUZE THEORETISCH EINDPRODUCT

Op basis van de genoemde afwegingen kan een keuze gemaakt worden voor een hypothetisch eindproduct. Hierbij zijn vooral gebruiksgemak, beschikbaarheid en snelheid belangrijk in de afweging.

Zowel de seriële als de parallelle verbindingen zullen steeds minder beschikbaar zijn. Daarom zijn dit beiden geen duurzame keuzes. USB is wat beschikbaarheid betreft een goede keuze. Het gebruiksgemak van USB is goed en van de overgebleven opties is het één van de goedkoopste. Het alternatief, een netwerkverbinding, biedt als enige voordeel een hogere snelheid, dit weegt op tegen minder gebuiks- en implementatiegemak. Daarom valt voor het eindproduct de keuze op USB.

4.5.4 KEUZE PROTOTYPE

Voor het prototype zijn kosten en gebruiksgemak nauwelijks van belang. De snelheid is eveneens niet relevant aangezien er met veel minder meetgegevens gewerkt wordt. Het belangrijkste criterium is implementatiegemak, de keuze voor beschikbaarheid beperkt zich tot de beschikbare ontwikkelmiddelen.

Gelet op implementatiegemak hebben de seriële en de parallelle verbinding de voorkeur. Deze verbindingsmogelijkheden zijn beide beschikbaar in de ontwikkelomgeving. Aangezien het gebruikte microcontroller-bordje reeds beschikt over een seriële verbinding, wordt gekozen voor een seriële verbinding.

4.6 IMPLEMENTATIE COMMUNICATIE MET ENERVIEW

De hardware die nodig is voor het implementeren van een seriële verbinding bestaat uit twee delen. Hierbij wordt het gedeelte dat in de Pc zit buiten beschouwing gehouden. Het eerste deel is het digitale deel, de UART. Het tweede deel is het gedeelte dat voor de spanningconversie van het signaal zorgt.

4.6.1 DE UART

UART staat voor Universal Asynchronous Receiver/Transmitter. Dit is een generiek onderdeel, dat onder andere met het RS-232 protocol kan communiceren. De UART draagt zorg voor de

conversie van parallelle commuinicatie naar seriële communicatie en terug. De seriële communicatie wordt volledig door de UART geregeld, en hoeft slechts eenmalig ingesteld te worden.

De geïntegreerde UART in de microcontroller is ingesteld als 8-bit UART (er worden 8 bits per keer verstuurd), waarbij één van de ingebouwde timers als klok voor de UART wordt ingesteld. Deze timer heeft een instelbare overloopfrequentie. Op deze overloopfrequentie (de *baudrate*) worden door de UART bits verstuurd. Vooraf dient de timer gekalibreerd te worden. Er is gekozen om deze kalibratie uit te voeren na elke reset van de EnerStation. Hiertoe dient vanuit de PC een karakter ('a') gestuurd te worden. Tijdens het versturen van dit karakter wordt gemeten hoe lang het duurt om één bit te verzenden en met deze informatie wordt de timer gekalibreerd. Hiermee is het mogelijk om op elke snelheid te communiceren.

Vanwege het discrete karakter van de timer, kan de ingestelde baudrate afwijkingen vertonen ten opzichte van de gewenste baudrate. De keuze van een specifieke klokfrequentie voor de microcontroller (in dit geval 18,432 Mhz) zorgt ervoor dat alle standaard baudrates zonder afwijking in te stellen zijn.

4.6.2 SPANNINGSCONVERSIE

De in- en uitgang van de UART werken met spanningsniveaus van 0V en 5V[24] voor respectievelijk een nul en een één. De RS-232 standaard gebruikt daarentegen spanningsniveaus van -3V tot -15V en 3V tot 15V voor respectievelijk een nul en een één. Hierdoor is het noodzakelijk de signalen te converteren. Een veelgebruikt IC dat deze conversie doet is de ST232[22] van ST Microelectronics. Dit IC converteert de logische signalen tussen de twee spanningsniveaus.

4.6.3 SOFTWARE

De UART kan wanneer deze is ingesteld in de software eenvoudig aangesproken worden met de functies die bij de software-ontwikkelomgeving meegeleverd worden. Karakters en tekstregels kunnen met eenvoudige functies verzonden en ontvangen worden. De implementatie van deze functies valt buiten het bereik van deze tekst.

Op de Pc is de seriële poort te benaderen met een speciaal programma. Een voorbeeld onder Windows is Hyperterminal. In dit programma zijn karakters over de seriële poort te sturen en worden alle karakters die naar de Pc gestuurd zijn zichtbaar. In EnerView is een specifieke implementatie toegepast om de seriële poort aan te sturen. De details hiervan vallen echter buiten het bereik van deze tekst.

4.7 COMMUNICATIE MET ENERPLUGS

Het EnerBox systeem bevat onder meer een groot aantal , EnerPlugs. Elke EnerPlug genereert elke minuut enkele bytes aan meetgegevens. De EnerStation moet ervoor zorgen dat deze meetgegevens verzameld worden. Daarom is het noodzakelijk dat er een communicatiekanaal bestaat tussen de EnerPlugs en de EnerStation.

De communicatie met de EnerPlugs kan worden verdeeld in meerdere lagen van functionaliteit:

1. fysieke communicatie,
2. bitniveau,
3. byteniveau,
4. communicatie op netwerkniveau (het 'netwerk' is de verzameling van EnerPlugs en de EnerStation).

Deze vier niveaus zullen in deze paragraaf besproken worden.

4.7.1 FYSIEK COMMUNICATIEKANAAL

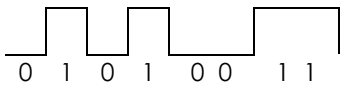
De keuze voor het communicatiekanaal is gedaan door de ontwerpgroep die de EnerPlug heeft ontworpen. Deze groep heeft ervoor gekozen om de communicatie met de EnerPlugs via het lichtnet te laten verlopen. De enige beperking die deze keuze oplegt, is de maximale snelheid waarmee een digitaal signaal verzonden kan worden. Het communicatiekanaal kan als een zogenaamde *black box* gezien worden. Daarop zijn de EnerStation en alle EnerPlugs aangesloten. Mits binnen de gestelde limieten gebleven wordt, kunnen hier logische nullen en enen doorheen gestuurd worden. Deze thesis beperkt zich tot de keuze van het protocol om de gegevens door deze bidirectionele, seriële verbinding te verzenden.

4.7.2 COMMUNICATIE OP BITNIVEAU

Het beschikbare communicatiekanaal biedt ondersteuning voor digitale communicatie, waarbij er twee mogelijke states zijn waarin het communicatiekanaal zich kan bevinden: *hoog* en *laag*. Er zijn een aantal mogelijkheden om met deze mogelijke states enen en nullen te representeren. De eenvoudigste protocollen worden beschreven, waarna op basis van de verschillen een keuze wordt gemaakt.

Unipolaire codering

In het geval van unipolaire codering, wordt een nul gerepresenteerd door een laag spanningsniveau (bijvoorbeeld 0V), en een één door een hoog spanningsniveau (5V). Op vaste tijdstippen wordt gekeken wat het niveau van het signaal is.

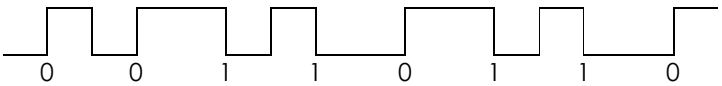


Figuur 23 - Unipolaire codering

Een voordeel van deze wijze van coderen is een eenvoudig te herkennen representatie. Een nadeel is het feit dat de codering geen synchronisatie met de klok geeft. Het samplen dient op het juiste moment te gebeuren en een kleine afwijking in klokfrequentie tussen zender en ontvanger kan er al voor zorgen dat er bitfouten optreden, doordat te vroeg of te laat gesampled wordt.

Manchester codering

Bij manchester codering wordt een nul gerepresenteerd door een overgang van laag naar hoog. Een één is een overgang van hoog naar laag.

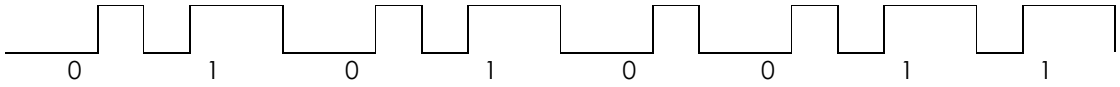


Figuur 24 - Manchester codering

In tegenstelling tot bij unipolaire codering, geeft deze codering bij elke bit een verandering in niveau. Hiermee is de klok te synchroniseren en is de kans op bitfouten als gevolg van een verkeerde kloksynchronisatie klein. Een nadeel is dat dit protocol voor elke verzonden bit twee verschillende states gebruikt. Dit halveert de effectieve bandbreedte van het communicatiekanaal.

Pulsbreedte codering

In het geval van pulsbreedte codering, wordt het onderscheid tussen enen en nullen gemaakt door de pulsbreedte te variëren. Een nul is een korte puls (hoge periode), een één is een langere puls. De duur van de lage periode kan onbepaald zijn. Optioneel kan de breedte van de lage periode worden gedefinieerd, om ervoor te zorgen dat alle bits evenveel tijd innemen. Dit maakt het protocol beter voorspelbaar, omdat elke bit even lang duurt.



Figuur 25 - Pulsbreedte codering

Nog makkelijker dan bij manchester codering, kan de klok gesynchroniseerd worden aan het signaal: elke neergaande flank geeft het begin van een nieuwe bit aan. De effectieve bandbreedte van het communicatiekanaal is drie keer zo klein, omdat per bit drie transities van elkaar onderscheiden moeten kunnen worden.

Overige protocollen

Naast de genoemde protocollen, zijn er een aantal andere manieren om bits te coderen. Een groot deel hiervan, de zogenaamde *non-return-to-zero* protocollen, vereisen een derde state voor het datakanaal en zijn daarom geen geschikte opties. De protocollen waarvoor dit niet geldt, zijn combinaties of afgeleiden van de genoemde protocollen en zijn over het algemeen lastiger te implementeren. Daarom zijn deze protocollen niet meegenomen in de afweging.

Keuze protocol

Om de bedrijfszekerheid van de communicatie met de sensor te kunnen garanderen, is het noodzakelijk een protocol te kiezen dat voor synchronisatie met de klok zorgt. Unipolaire codering valt dan ook af.

Bij de keuze tussen manchester of pulsbreedte codering, wordt de voorkeur gegeven van implementatiegemak boven bandbreedte. Dit aangezien het prototype slechts een beperkt aantal hoeft te ondersteunen. Het feit dat pulsbreedte codering eenvoudig te synchroniseren is doet de keuze op dit protocol vallen.

4.7.3 COMMUNICATIE OP BYTENIVEAU

Om op een efficiënte manier informatie te kunnen versturen, is het noodzakelijk om meerdere bits te combineren. De kleinste hoeveelheid bits die verzonden moeten kunnen worden tussen EnerStation en EnerPlug is 8 bits. Daarom is de byte een logische eenheid om bij de communicatie vanuit te gaan.

Om bytes te kunnen onderscheiden van elkaar, is het noodzakelijk een scheiding aan te brengen, die aangeeft wanneer een byte begint. Wanneer er geen gegevens verzonden worden is het communicatiekanaal hoog, dit komt doordat alle aangesloten 8051 microcontrollers de communicatiepin voor ontvangst hoog hebben ingesteld, zie ook hoofdstuk 4.3.1. Een mogelijkheid om het begin van een byte aan te geven, is het versturen van een nul. Hierdoor wordt de lijn even laag, dan weer hoog. De klok kan gesynchroniseerd worden met de periode dat de lijn hoog is, er is immers bekend wat er is verstuurd.

Een beperking hiervan is dat er geen onderscheid gemaakt kan worden tussen de starbit en de databits. Door de hoge periode van de startbit langer te maken dan bij een één, kan de starbit onderscheiden worden van de databits.

4.7.4 COMMUNICATIE OP NETWERKNIVEAU

Wanneer er meerdere apparaten van hetzelfde communicatiekanaal gebruik maken, dient voorkomen te worden dat meerdere apparaten tegelijkertijd proberen te zenden (dit veroorzaakt zogenaamde *collissions*)[23]. Een master-slave systeem zorgt hiervoor. Dit houdt in dat één apparaat (de EnerStation) master is en dat de overige apparaten (de EnerPlugs) slave zijn. Zij sturen slechts data wanneer de master daarom vraagt. Een voordeel van dit systeem is dat collissions niet voor kunnen komen. Een algemeen nadeel is het systeem niet werkt, wanneer de master niet werkt. Aangezien dit altijd geldt wanneer de EnerStation niet werkt, doet dit niet ter zake.

4.8 IMPLEMENTATIE COMMUNICATIE MET ENERPLUGS

Aangezien er in de gebruikte microcontroller geen specifieke hardware aanwezig is om de gekozen manier van communicatie te implementeren, wordt alles in software geïmplementeerd. De functionaliteit die geïmplementeerd dient te worden betreft het kunnen opvragen van meetgegevens van een aangesloten sensor met een specifiek adres. Daarbij wordt gebruik gemaakt van de eerder bepaalde methoden.

4.8.1 VERZENDEN EN ONTVANGEN VAN EEN BYTE

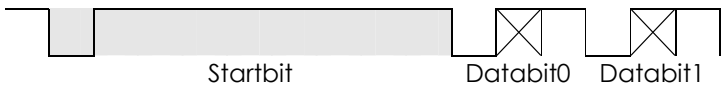
Een byte bestaat uit een startbit, gevolgd door acht databits. Zowel het ontvangen als het verzenden van een byte is een basale functionaliteit voor zowel EnerStation als EnerPlug. In deze paragraaf wordt beschreven hoe deze functies zijn geïmplementeerd.

Startbit

Een starbit begint met periode t laag, gevolgd door een periode $a * t$ hoog, waarbij:

1. $t \geq f_{max}^{-1}$, waarbij f_{max} is de maximale datasnelheid,
2. $t = f^{-1}$, waarbij $f \leq f_{max} / 0{,}4a$ is de gebruikte datasnelheid.

Restrictie (1) geldt omdat de lage periode anders niet gedetecteerd zou worden. Restrictie (2) dient om de startbit anders te laten zijn dan een één, met een hoge periode van maximaal $0{,}8a * f_{max}^{-1}$. Voor a wordt 8 gekozen. Dit geeft een factor >6 tussen minimale en maximale datarate. Een startbit ziet er dan uit als in Figuur 26.



Figuur 26 - Startbit en twee databits

Het verzenden van een startbit gebeurt als volgt:

1. Maak datapin laag.
2. Wacht t.
3. Maak datapin hoog.
4. Wacht a * t.

Bij het verzenden van een databit, wat direct dient te gebeuren, zal de datapin weer laag worden. Een startbit wordt als volgt gedetecteerd:

1. Wacht tot datapin laag is.
2. Wacht tot datapin hoog is.
3. Reset en start ingebouwde timer.
4. Wacht tot datapin laag is.
5. Stop timer.
6. Vergelijk timer met vooraf ingesteld waarde om te checken of de periode lang genoeg was (restrictie 3). Zo niet, ga terug naar stap 1.
7. Bereken referentietijd t_{ref} = timer * 1,5 / 8 (nodig voor ontvangen databits).

Databits

Wanneer een startbit verstuurd is, volgen direct acht databits. Het verzenden van deze databits gebeurt als volgt:

1. Maak datapin laag.
2. Wacht t.
3. Maak datapin hoog.
4. Als databit = 1, wacht t, als databit = 0, wacht 2*t
5. Maak datapin laag.
6. Als databit = 0, wacht t, als databit = 1, wacht 2*t

Nadat de startbit ontvangen is, is een tijdreferentie bepaald. De hoge periode wordt vergeleken met deze referentie om te bepalen of een nul of een één is verstuurd:

1. Wacht tot datapin laag is.
2. Wacht tot datapin hoog is.
3. Reset en start ingebouwde timer
4. Wacht tot datapin laag is.
5. Stop timer.
6. Vergelijk timer met t_{ref}. Als timer < t_{ref} dan databit = 0, als timer ≥ t_{ref} dan databit = 1.

Bitfouten

Wanneer bitfouten optreden, kan het gebeuren dat er op data gewacht wordt, die niet meer komt. Daarom kan bij het wachten totdat een datapin hoog of laag wordt op een time-out gecheckt worden. Hiertoe wordt voordat er gewacht wordt, een ingebouwde timer gereset en gestart. Er wordt vervolgens ook gecheckt op de bijbehorende *timer flag* bit. Dit is een bit die hoog wordt wanneer de timer overloopt[24]. Wanneer dit gebeurt wordt het ontvangen gestaakt.

4.8.2 IMPLEMENTATIE IN ENERPLUG

Wat de communicatie betreft heeft de EnerPlug een eenvoudige functionaliteit. Elke EnerPlug heeft een uniek adres. Wanneer een byte ontvangen wordt, wordt deze vergeleken met het eigen adres. Komt de ontvangen byte overeen met het adres, dan worden twee bytes aan meetgegevens teruggestuurd. Deze worden gebufferd om verder verwerkt te worden.

4.8.3 IMPLEMENTATIE IN ENERSTATION

De EnerStation verzamelt elke minuut meetgegevens van de EnerPlugs. Dit gebeurt in een iteratief proces. Naar het eerste adres wordt een adresbyte gestuurd, waarna gewacht wordt op meetgegevens en een timer gestart wordt. Wanneer er meetgegevens ontvangen worden of een time-out optreedt, wordt het volgende adres verstuurd. Dit proces wordt 250 maal herhaald.

4.9 TIJD- EN DATUMREFERENTIE

Voor het bijhouden van metingen is het nodig om elke minuut een tijdreferentie te hebben. De eenvoudigste oplossing is een externe Real Time Clock. Er is een keuze gemaakt op basis van prijs en beschikbaarheid. Het goedkoopste type uit het assortiment van elektronicaleverancier Farnell betreft de DS1602 van fabrikant Maxim.

4.9.1 BESCHRIJVING VAN DE DS1602

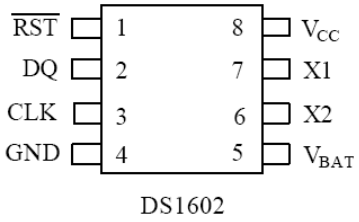
De DS1602 is een real-time clock/ teller, geschikt om secondes tellen.

De real-time clock telt continu secondes en maakt daarbij gebruik van een externe batterij waneer er geen voedingsspanning aanwezig is. Met behulp van software algoritmen kunnen de getelde secondes omgerekend worden naar tijd en datum.

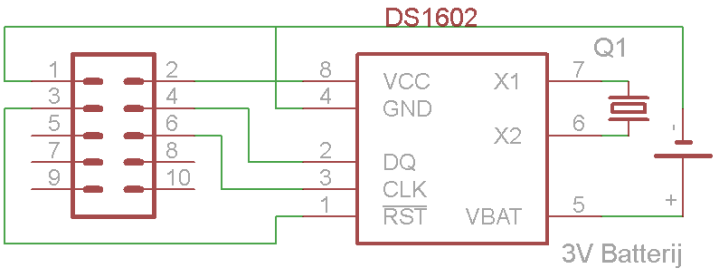
De teller telt secondes wanneer er een voedingsspanning beschikbaar is. Deze functie is bruikbaar voor het meten van de tijd dat een apparaat aan staat. Ook kan deze teller gebruikt worden om softwarematig real-time gebeurtenissen van de processen bij te houden. De 32-bit tellers in DS1602 kunnen secondes bij houden voor de maximale duur van 125 jaar [1].

4.9.2 INTERFACE

Figuur 27 geeft een schets van de aansluitingen van de DS1602[25]. Dit zijn naast de voedingsaansluitingen (V_{CC}, GND) de data-aansluitingen (/RST, DQ en CLK), de aansluitingen voor een kristal (X1, X2) en een aansluiting voor een externe batterij (V_{BAT}).



Figuur 27 - Pinnen van de DS1602



Figuur 28 – DS1602 Interface voor microcontroller

Om de DS1602 aan te sluiten op de microcontroller, is een schakeling ontworpen (Figuur 28). Hierin is een kristal Q1 te zien, van 32,768KHz. De batterij die nodig is voor DS1602 moet een spanning leveren tussen 2,5V en 3,5V. In de schakeling wordt een 3V lithium batterij gebruikt. Volgens de specificaties van DS1602 kan een lithium batterij van 35 mAh of groter tot 10 jaar lang meegaan.

Communicatie van en naar de DS1602 vindt plaats door een seriële interface met drie verbidingen. Voor de communicatie is gekozen voor P0 port van de microcontroller. De volgende pinnen worden gebruikt:

- P0.0 = /RST
- P0.1 = DQ
- P0.3 = CLK

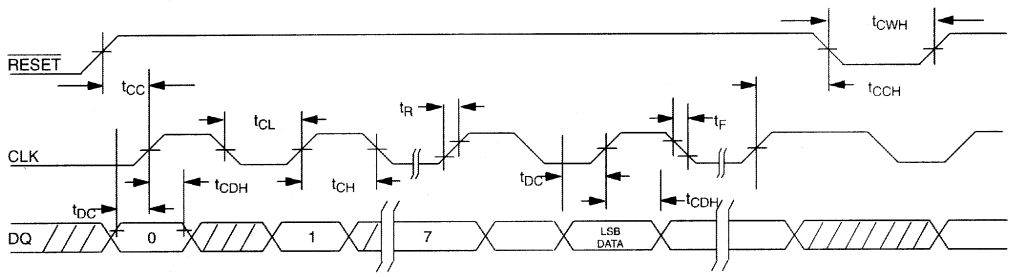
De seriële interface wordt geactiveerd waneer /RST hoog wordt. Vervolgens worden er 8 bits geladen in het protocol-schuifregister. Via dit register kan er een opdracht gegeven worden om een timer in te stellen, uit te lezen of te resetten. Ook kan hiermee de oscillator getrimd worden om kleine afwijkingen te corrigeren. Elke bit komt serieel binnen op een opgaande flank van de CLK pin. Wanneer er een geldige opdracht in het protocol-register geladen wordt, worden op de volgende 32 klokcycly bits ingelezen of uitgevoerd.

Communicatie vereist dat er een voedingsspanning aanwezig is. Wanneer de voedingsspanning lager wordt dan de batterijspanning, gaat de DS1602 in een zuinige modus en wordt de seriële interface uitgeschakeld. De 32-bit continue teller blijft tellen zolang er een geldige spanning (batterij of VCC) aanwezig is en er een oscillator verbonden is aan DS1602.

4.9.3 SOFTWARE

Tijd instellen

De functie DS1602_settime(void) stelt de begintijd van real-time klok in. Eerst wordt /RST hoog gemaakt. Vervolgens wordt het protocol commando 0x01 gestuurd naar DS1602 om aan te geven dat er een schrijfinstructie volgt. Daarna wordt de ingestelde tijdwaarde verstuurd, waarna /RST weer laag wordt.

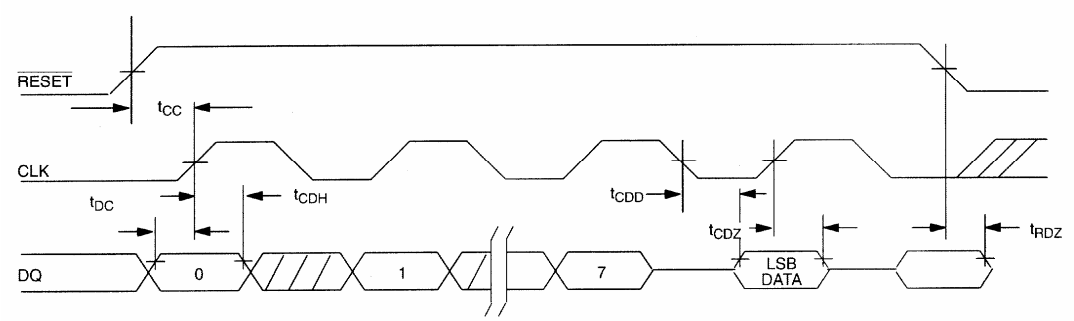


Figuur 29 - Schrijfinstructie DS1602

Figuur 29 is een tijddiagram van schrijf procedure. Zodra het 8-bit protocol commando de DS1602 in de schrijfmode zet, worden 32 bits aan data naar de geselecteerde teller, op elke opgaande klokflank van de input CLK pin. Deze functie hoeft maar één keer gebruikt te worden. Zolang de batterij aanwezig is in de schakeling, hoeft deze tijd niet opnieuw ingesteld worden.

Tijd uitlezen

De functie DS1602_readtime(void) leest de tijd uit de DS1602. Eerst wordt /RST hoog gemaakt. Vervolgens wordt het protocol commando 0x81gestuurd naar DS1602. Dit commando geeft aan dat de waarde van teller op de output (DQ) moet worden gezet. De eerste data bit die verstuurd wordt van de geselecteerde 32-bit counter gebeurt op de neergaande klokflank (t_{CDD}) na dat de laatste bit van het protocol is geschreven in DS1602. Figuur 30 is een tijddiagram van een leesinstructie.



Figuur 30 - Leesinstructie DS1602

5 RESULTATEN

De verschillende deelsystemen zijn, na het kiezen van een mogelijke implementatie, eerst los van elkaar toegepast. Drie van deze systemen zijn vervolgens met gezamelijk in een microcontroller geïntegreerd, om een werkend geheel te krijgen. In dit hoofdstuk worden eerst de afzonderlijke resultaten beschreven, waarna het resultaat van het gehele systeem wordt beschreven.

De broncode van de beschreven software is terug te vinden in bijlage 2.

5.1 COMMUNICATIE MET ENERVIEW

De eerste versie van het programma om te communiceren met EnerView had als doen het kunnen verzenden van een karakter van de microcontroller naar de Pc. Dit is met succes geïmplementeerd, waarna een tweede versie is gemaakt. Deze versie voegde automatische baudrate detectie toe, zoals beschreven in paragraaf 4.6.2. Hiermee was het mogelijk om vanuit een Pc de baudrate te laten instellen, waarna met eenvoudige functies bytes (en daarmee tekst) naar een Pc gestuurd kan worden.

De software is op zodanige wijze geïmplementeerd, dat deze eenvoudig gebruikt kan worden in andere programma's. Dit programma is tevens goed van pas gekomen om andere software te debuggen, omdat op eenvoudige wijze informatie uitgevoerd kan worden.

De hardware voor de communicatie was reeds geïntegreerd in het gebruikte microccontroller-bord.

5.2 COMMUNICATIE MET ENERPLUG

Voor de communicatie met de EnerPlug is uitgegaan van een directe elektrische verbinding tussen twee microcontrollers, in de vorm van een kabel.

In eerste instantie zijn functies voor het verzenden van een bit geïmplementeerd, vervolgens voor een byte, waarbij een nul als startbit is gebruikt. Dit leverde geen noemenswaardige problemen op. Vervolgens zijn functies voor het ontvangen van bytes geïmplementeerd. Deze bleek te werken, mits er geen bitfouten optraden in de verbinding. In dit geval werd niet langer de startbit, maar een willekeurige databit geïnterpreteerd als startbit. Ook bleken bytes in omgekeerde volgorde ontvangen te worden.

Aanpassing van zender en ontvanger voor het gebruik van een andere startbit met een vooraf bepaalde minimumlengte, zoals beschreven in paragraaf 4.7.3, bleek deze problemen op te lossen.

De resulterende programmatuur is eveneens zodanig geïmplementeerd dat deze gebruikt kan worden in andere programma's.

5.2.1 ENERPLUG SIMULATOR

Met de beschikbare communicatiefucties is een programma geschreven, dat een EnerPlug simuleert. Dit programma voert de volgende taken uit:

1. Wacht tot byte b ontvangen wordt.
2. Als b niet gelijk is aan eigen adres, ga naar 1.
3. Verstuur 2 bytes aan nep-meetgegevens.
4. Verhoog bytes met nep-meetgegevens.
5. Ga naar 1.

Door verschillende microcontrollers met een ander ingebouwd adres te programmeren, was het mogelijk om meerdere sensoren te simuleren. Deze sensoren zijn allen aangesloten op dezelfde elektrische verbinding.

5.3 TIJD- EN DATUMREFERENTIE

Om de DS1602 real-time clock te kunnen gebruiken in combinatie met een microcontroller, is de schakeling uit paragraaf 4.9.2 opgebouwd, in eerste instantie met V_{BAT} verbonden aan V_{CC}. Hiervoor is gebruik gemaakt van een universele gaatjesprintplaat, zoals in figuur XXX te zien is.

Hier plaatje van DS1602 bordje

Vervolgens is de software, beschreven in paragraaf 4.9.3, geschreven. De snelheid waarmee een pin van een microcontroller hoog gemaakt kan worden (de rise-time) bleek een beperkende factor te zijn voor de communicatiesnelheid. Aanpassing van de software loste deze problemen op.

Het gebruik van een batterij leek noodzakelijk te zijn, deze is daarom toegevoegd aan de schakeling. Timingproblemen in de software bleken uiteindelijk nog voor enkele problemen te zorgen. Deze zijn succesvol opgelost, waarna het programma is gestructureerd voor gebruik in andere software.

5.4 ENERSTATION

Met de geïmplementeerde functionaliteiten was het mogelijk om een primitief programma voor de EnerStation te schrijven. Dit programma heeft geen opslagfunctionaliteit, maar stuurt de ontvangen gegevens en de huidige tijd direct naar de Pc. Het programma werkt als volgt:

1. Initialiseer UART.
2. Stel tijd in.
3. Wacht 1 seconde.
4. Stuur adresbyte.
5. Als er antwoord komt:

   ontvang meetgegevens,

   lees tijd uit,

   stuur adres, tijd en meetgegevens naar Pc.
6. Verhoog adres (totdat alle adressen zijn geweest, anders resetten).
7. Ga naar 3.

Met dit programma was het mogelijk om meerdere EnerPlug simulators aan te spreken, waarna de ontvangen data eenmaal per seconde werd verstuurd naar de Pc.

5.5 SD-CARD

Om een beter begrip te krijgen van het SPI protocol, is een programma gemaakt, dat twee microcontrollers laat communiceren via SPI. Dit bleek wegens zeer beperkte informatie in de datasheet van de gebruikte microcontroller lastig te zijn. Dit heeft eraan bijgedragen dat dit vrij veel tijd heeft gekost.

Toen eenmaal duidelijk was hoe het SPI protocol werkt, is getracht een SD-Card aan te sturen. Hiervoor werd de interface uit 4.3.2 opgebouwd. Nadat de interface schakeling meerdere

was malen aangepast, werd uiteindelijk duidelijk hoe deze schakeling opgebouwd had moeten zijn, met een *voltage level translator* om de spanningen te regelen.

Met de laatste versie van de schakeling is het uiteindelijk gelukt om een R1 response van de SD Card te ontvangen. Door eerdere tegenslagen is dit het enige resultaat met betrekking tot het aansturen van de SD-Card.

6 DISCUSSIE

In paragraaf 4.1.2 is een berekening gemaakt voor de totale benodigde hoeveelheid geheugen. Hierbij is uitgegaan van een geheugenstrategie, waarbij elke meting wordt opgeslagen. In de praktijk zal dit veel verspilling van geheugenruimte opleveren. Men kan er immers vanuit gaan dat, wanneer bjivoorbeeld een lamp één uur aan staat, er 60 metingen zullen worden opgeslagen die nagenoeg gelijk zijn. In een toekomstig systeem verdient het dan ook de aanbeveling om hier een zekere vorm van compressie op toe te passen.

Zoals te lezen is in paragraaf 4.4.5, is gekozen voor de NXP P89V51RD2FA microcontroller. De ervaringen met deze microcontroller zijn helaas niet uitsluitend positief. Een belangrijk nadeel van deze microcontroller is de gebrekkige documentatie. Dit kwam met name naar voren bij het tot stand brengen van SPI communicatie met de SD-Card. Nergens in de datasheet staat duidelijk vermeld hoe gegevens ontvangen kunnen worden. Ook voor het instellen van de UART is de informatie summier. Fabrikanten als Microchip[6][26] en Atmel[7][27] leveren betere documentatie bij hun microcontrollers, daar gaat in dit opzicht dan ook de voorkeur naar uit.

De communicatie met SD Card is zeker mogelijk te voldoen nu het sturen van een commando en het ontvangen van een response goed gaat. Helaas is veel tijd verloren gegaan voordat de interface schakeling is aangepast. Gebruik van een specifiek interface IC had veel tijd kunnen besparen.

[6] De datasheet[26] van de vergelijkbare 16f87x microcontroller serie levert bijvoorbeeld tabellen voor het instellen van de baudrate van de UART.

[7] Het hoofdstuk over de SPI module in de datasheet[27] van de vergelijkbare AT89C51RD2 microcontroller beslaat maar liefst 9 pagina's, tegenover 2,5 voor de datasheet van de P89V51RD2

7 CONCLUSIE

In deze thesis is onderzocht hoe een systeem ontwikkeld kan worden voor het verzamelen, opslaan, en doorsturen van meetgegevens. Het ontwerpproces is in vier stappen verlopen: probleemdefinitie, verwant onderzoek, onderzoek naar mogelijke oplossingen en de daadwerkelijke implementatie.

In de probleemdefinitie is er volgens de wensen van de opdrachtgever en de eindgebruiker een programma van eisen opgesteld. Hiermee zijn richtlijnen gegeven aan het opstellen van een concreter beeld voor de inrichting van het systeem. Omdat deze richtlijnen gericht zijn op te ontwerpen eindproduct, is het niet realistisch gebleken om aan alle eisen te voldoen. Voor een demonstratie van het werkingsprincipe is dit echter ook niet nodig.

Vervolgens is er verwant onderzoek verricht naar een bestaand product, om een betere indruk krijgen van implementatie mogelijkheden. De opzet van het onderzochte systeem blijkt in grote lijnen overeen te komen, hoewel er toch fundamentele andere keuzes zijn gemaakt, bijvoorbeeld voor wat betreft de communicatie.

Bij het onderzoek naar de mogelijke oplossingen voor delen van de EnerStation, bleek soms dat er veel keuzes zijn. In het geval van de communicatie met de EnerPlugs is een goede keuze gemaakt, aangezien dit een werkend onderdeel heeft opgeleverd. Ook de keuze voor de communicatie met de Pc heeft goed uitgepakt, en heeft een werkend onderdeel opgeleverd. De keuze van een kant- en klare real time clock bleek eveneens een goede geweest te zijn. Van het geheugen kan helaas niet hetzelfde gezegd worden. Dit bleek een dermate gecompliceerde implementatie te vereisen, dat het resultaat niet zoals gewenst is. Hoewel het in principe mogelijk lijkt te zijn om een SD-Card aan te sturen, bleek dat voor een spoedig verloop ervaring een nuttige toevoeging had kunnen vormen.

Het apart oplossen van de deelproblemen heeft over het algemeen goed gewerkt. Hierna is dan ook in korte tijd een werkende EnerStation geïmplementeerd, waarmee duidelijk is dat het concept daadwerkelijk te implementeren is in de vorm van een primitief prototype.

LITERATUUR

1 ELV Elektronik AG, *Funk-Energiemonitor EM 1010 PC/EM 1010*, september 2006

2 ELV Elektronik AG, *868-MHz-Empfangsmodul RX868-3V*, http://www.elv.de/868-MHz-Empfangsmodul-RX868-3V/x.aspx/cid_74/detail_10/detail2_12822

3 Atmel, *1-megabit 2.7-volt Only Serial DataFlash® AT45DB011*, 2001

4 Atmel, *Datasheet: 8-bit Microcontroller with 8K Bytes In-System Programmable Flash*, 2007

5 ST Electronics, 1K (64 x 16 or 128 x 8) SERIAL MICROWIRE EEPROM, juni 1997

6 Holtek Semiconductor inc., *New 2-wire Serial Interface EEPROM - HT2201*, November 2004, http://www.holtek.com.tw/english/news/products/111504.htm

7 CompactFlash Association, *CF and CompactFlash® Frequently Asked Questions*, http://www.compactflash.org/faqs/faq.htm

8 Fuji Foto Film, *Exhibition of Newly Developed xD-Picture Card, September 2002*, http://home.fujifilm.com/photokina2002/data/pr_pdf/di_3.pdf

9 Legend Memory, *xD Picture Card High Performance Media Cards for todays portable devices*, 2005, http://www.legend.com.au/files/article_pdfs/416.pdf

10 SD Group, *SD Specifications Part 1Physical Layer, Simplified Specification Version 2.00* september 2006

11 SanDisk Secure Digital Card , Product Manual Version 1.9, Document No. 80-13-00169

12 Freescale Semiconductor, Serial Peripheral Interface (SPIV3) Block Description, Block Guide, V3.06

13 Velzen, J.L.J.M van, B.H.H Juurlink, S. Prins, J.C Verwer, *Computer Architectuur en Organisatie (CAO) practicumhandleiding*, februari 2007

14 Electronics Industries Association, *EIA Standard RS-232-C Interface Between Data Terminal Equipment and Data Communication Equipment Employing Serial Data Interchange*, August 1969, reprinted in Telebyte Technology *Data Communication Library*, Greenlawn NY, 1985

15 IEEE Std 1284-1994, *IEEE Standard Signaling Method for a Bidirectional Parallel Peripheral Interface for Personal Computers*, 1994

16 IEEE Std 1284-1994, *IEEE Standard Signaling Method for a Bidirectional Parallel Peripheral Interface for Personal Computers*, 1994

17 Future Technology Devices International Limited, *FT232R USB UART IC Data Sheet*, United Kingdom, januari 2006

18 Marc E. Fiuczynski, Vincent K. Lam, Brian N. Bershad, *The Design and Implementation of an IPv6/IPv4 Network Address and Protocol Translator*, Proceedings of the 1998 USENIX Conference, New Orleans, LA., juni 1998

19 Bluetooth SIG inc., *Specification of the Bluetooth system*, version 2.1, 2007

20 Sparkfun Electronics, Boulder, CO (USA), *Bluetooth SMD Module WRL-08474 specifications*, http://www.sparkfun.com

21 Microchip Technology inc., *Application note AN1120 "Ethernet Theory of Operation"*, 2008

22 ST Microelectronics, *ST232 5V POWERED MULTI-CHANNEL RS-232 DRIVERS AND RECEIVERS*, 2001

23 Mieghem, Piet van, *Data Communications Networking* , Techne Press, Amsterdam, 2006, ISBN: 90-8594-008-7

24 NXP B.V., *P89V51RB2/RC2/RD2 Product data sheet*, rev. 04, 2007

25 Maxim, *DS1602 Elapsed Time Counter*, 2002

26 Microchip, *PIC16F87x datasheet*, 2001

27 Atmel, *8-bit Flash microcontroller AT89C51RD2, AT89C51ED2*, 2007

N.B. Alle URL's zijn in juni 2008 geverifieerd.

BIJLAGE 1: PROGRAMMA VAN EISEN VAN DE ENERBOX

De EnerBox is een systeem dat het energieverbruik van verschillende apparaten in een huishouden kan meten. De verzamelde informatie wordt vervolgens op verschillende manieren gepresenteerd aan de gebruiker.

Het systeem is in eerste plaats bedoeld voor consumenten, maar kan daarnaast gebruikt worden voor bedrijfsomgevingen, zoals kantoren.

Hoewel er een aantal concurrerende apparaten bestaan die gedeeltelijk dezelfde functionaliteit bezitten, is dit een innovatief en nieuw product.

INSTALLATIE

1. Het systeem dient zonder gereedschap geïnstalleerd te kunnen worden.
2. Het systeem dient geen extra bekabeling te vereisen.
3. Installatie en updaten van software dient zo eenvoudig te geschieden, dat iedere eindgebruiker dit zelf kan doen.

GEBRUIK

In- en uitvoer van gegevens

4. Het moet voor de gebruiker mogelijk zijn om de energiekosten van verschillende termijnen te zien.
5. Het moet voor de gebruiker mogelijk zijn om de energiekosten van een specifieke selectie van apparaten te zien.
6. Het moet voor de gebruiker mogelijk zijn om het momentane energieverbruik te zien.
7. Het moet voor de gebruiker mogelijk zijn om de energiekosten van verschillende termijnen (bijv. dag, week, jaar) met elkaar te vergelijken.
8. De verzamelde informatie moet te bekijken zijn door middel van een PC.
9. Het moet mogelijk zijn om zowel handmatig als automatisch de variabele systeemparameters in te voeren. De variabele systeemparameters zijn:
    a. Energietarief op verschillende momenten,
    b. Huidige tijd en datum,
    c. Standaard netspanning.

Stand-alone gebruik

Zie ook de eisen onder *Opslag*.

10. Zonder PC het momentane energieverbruik te zien zijn.
11. Zonder PC moet het mogelijk zijn actuatoren te bedienen.

SYSTEEMVEREISTEN

Algemeen

12. Het systeem dient maximaal 250 te kunnen bevatten.

Meting

13. Het bereik van de vermogensmeting dient van 0,1 – 4000W te zijn.
14. De nauwkeurigheid van de vermogensmeting dient 1% te zijn over een bereik van 2-4000W.
15. De volgende grootheden dienen gemeten te worden:
    d. Stroom (absoluut),
    e. Spanning (afwijking van standaard netspanning),
    f. Fasehoek.
16. De verversingssnelheid/meetperiode van de metingen is 1 minuut.
17. Een meting dient het gemiddelde te zijn van de grootheid tijdens de meetperiode.

Opslag

18. Het opslagmedium van het systeem (uitgezonderd PC) dient zodanig gekozen te worden dat de gegevens zonder PC 1 jaar bewaard kunnen worden.
19. Het systeem mag op een PC niet meer dan 1GB aan opslagruimte innemen.

Actuatorfunctie

20. Het systeem moet de mogelijkheid hebben om een apparaat automatisch uit te schakelen, door de elektriciteit van dit apparaat uit te schakelen.
21. Het systeem moet weten of een bepaald apparaat automatisch uitgeschakeld kan worden.
22. In het geval van apparaten die soms automatisch uitgeschakeld kunnen worden, moet het systeem weten onder welke omstandigheden dit geldt.
23. In het geval van apparaten die niet automatisch uitgeschakeld kunnen worden, moet het systeem aan de gebruiker een melding geven wanneer dit zou kunnen.

Veiligheid en omgeving

24. Het systeem moet KEMA gekeurd worden.
25. Het systeem moet voldoen aan de Europese EMC-richtlijn voor elektromagnetische compatibiliteit.
26. Het systeem mag de ingangsstroom, spanning en fasehoek niet meer dan 1% doen afwijken.
27. Het moet mogelijk zijn om meerdere EnerBox systemen op hetzelfde elektriciteitsnet te gebruiken, zonder dat deze elkaar kunnen beïnvloeden.
28. Het systeem, exclusief PC, mag maximaal gemiddeld 10W gebruiken, wanneer er 1 EnerStation en 10 EnerPlugs gebruikt worden.

BIJLAGE 2: PROGRAMMACODE

COMMUNICATIE MET ENERVIEW

```
/* Auto Baud functie
/* Werkt alleen wanneer er na de stopbit een 1 wordt gestuurd
/* Bijvoorbeeld een 'a'  */

#include <REG51F.H> /* 80c51 standaard definities */

void AutoBaud (void) {

   T2CON ^= 0x30;       //Initialiseer Timer2 als Baud rate generator
   TH2 = 0;          //Zet timer in 16-bit counter mode
   TL2 = 0;
   TCON = 0;

   while(RXD == 1){;}      //Wacht op start bit (falling edge)
   TR2 = 1; //Start timer

   while(RXD == 0) {;}     //Wacht op volgende edge (rising edge)
   TR2 = 0; //Stop timer

   /* Preload waarde voor Timer2 wordt door 16 gedeeld */
   TL2 >>= 4;
   TL2 ^= (TH2 << 4);
   TH2 >>= 4;

   /* Preload Timer2 */
   RCAP2L = 0xFF - TL2;
   RCAP2H = 0xFF - TH2;

   /*Stel UART in */
   SCON  = 0x50;   // SCON: mode 1, 8-bit UART, enable rcvr
   TR2   = 1;      // TR2:  timer 2 run
    TI    = 1;     // TI:   set TI to send first char of UART
}
```

COMMUNICATIE MET ENERPLUGS

```
/* PLC.c
*  Functies voor pulslengtecodering
*
*  De codering is als volgt:
*           __          ____
*  ----____|  |---__|    |---
*
*         0          1
*
*  Een 0 is 2xT laag, 1xT hoog
*  Een 1 is 1xT laag, 2xT hoog
*  T wordt aan het begin van een byte bepaald en blijft die byte gelijk.
*
*  Een byte bestaat uit een startbit en 8 databits.
*  Door de lengte van de hoge periode van de startbit te meten, kan de
*  grens voor toekomstige bits bepaald worden.
*
*  Een startbit is:
*  -Minimaal 2ms (3072 klokcycly @ 18.432Mhz)
*  -Exact 8 keer zo lang als de hoge periode van een '0' (klok
*  wordt zo bepaald)
*/

#include <REG51f.H>
#include "hulpfuncties.h"
#include "PLC.h"
#include <STDIO.H>

/* Definieer interface */
sbit DATA = P2^0;

/* Variabelen */
unsigned int timer_compare;

void PLC_sendbit(bit b){
   DATA = 0;
   if(b)
      wait1_3ms();
   else{
      wait1_3ms();
      wait1_3ms();
   }
   DATA = 1;

   if(b){
      wait1_3ms();
      wait1_3ms();
   }
   else
      wait1_3ms();
   DATA = 0;
}

void PLC_sendchar(char c){
   /* Startbit */
   DATA = 0;
   wait1_3ms();
   DATA = 1;
   wait1_3ms();
   wait1_3ms();
   wait1_3ms();
   wait1_3ms();
   wait1_3ms();
```

```
   wait1_3ms();
   wait1_3ms();
   wait1_3ms();
   /* Databits */
   PLC_sendbit(c & 0x80);
   PLC_sendbit(c & 0x40);
   PLC_sendbit(c & 0x20);
   PLC_sendbit(c & 0x10);
   PLC_sendbit(c & 0x08);
   PLC_sendbit(c & 0x04);
   PLC_sendbit(c & 0x02);
   PLC_sendbit(c & 0x01);
}

unsigned char PLC_readchar(void){
   unsigned char c = 0;

   DATA = 1;

   /* Timer0 en Timer1 als 16-bit timer */
   TMOD |= 0x01;
   TF0 = 0;
   TR0 = 0;
   TR1 = 1;

   do{
      /* Reset Timer1 */
      TF1 = 0;
      TL1 = 0;
      TH1 = 0;
      /* Wacht op begin van starbit */
      while(!DATA && !TF1);

      /* Startbit begonnen, start timer */
      TL0 = 0;
      TH0 = 0;
      TR0 = 1;

      /* Reset Timer1 */
      TL1 = 0;
      TH1 = 0;
      /* Wacht op eind van startbit */
      while(DATA && !TF1);

      /* Starbit klaar, sla timer op */
      TR0 = 0;
      timer_compare = ((int)TH0 << 8) + TL0;

   /* Check of het inderdaad een startbit was */
   } while(timer_compare <= 3072);

   /* Bij Timer1 overflow timeout, return 0 */
   if(TF1) return 0;

   /* tijdreferentie / 8, * 3/2 om tijdreferentie te krijgen */
   timer_compare *= 3;
   timer_compare >>=4;

   /* Startbit is ontvangen, nu databits ontvangen */
   c |= PLC_readbit();
   if(TF1) return 0;
   c <<= 1;
   c |= PLC_readbit();
   if(TF1) return 0;
   c <<= 1;
   c |= PLC_readbit();
   if(TF1) return 0;
   c <<= 1;
   c |= PLC_readbit();
   if(TF1) return 0;
   c <<= 1;
   c |= PLC_readbit();
   if(TF1) return 0;
   c <<= 1;
   c |= PLC_readbit();
   if(TF1) return 0;
   c <<= 1;
   c |= PLC_readbit();
   if(TF1) return 0;
   c <<= 1;
   c |= PLC_readbit();
   if(TF1) return 0;

   DATA = 0;

   return c;

   /* todo: Bij timeout (timer overflow) exit routine */
}
bit PLC_readbit(void){
   /* Reset Timer1 */
   TF1 = 0;
   TL1 = 0;
   TH1 = 0;
   /* Wacht op opgaande flank */
   while(!DATA && !TF1);

   /* Start timer0 */
   TL0 = 0;
   TH0 = 0;
   TR0 = 1;

   /* Reset Timer1 */
   TL1 = 0;
   TH1 = 0;
   /* Wacht op neergaande flank */
   while(DATA && !TF1);

   /* Stop timer0 */
   TR0 = 0;
```

```
   /* Vergelijk met timer_compare */
   if((((int)TH0 << 8) + TL0) <= timer_compare)
      return 0;
   else
      return 1;
}
```

TIJD- EN DATUMREFERENTIE

```
/* ds1602.c
 Functies om DS1602 RTC ic aan te sturen
 */

#include <REG51F.H>        /* 80c51 standaard definities */
#include "ds1602.h"

/* Interface definitie */
sbit RST = P0^0;
sbit DQ = P0^1;
sbit CLK = P0^3;

/* Deze functie stelt de tijd in.
*  Stel eerst t0...t3 in, deze waardes worden naar de DS1602 gestuurd
*/
void DS1602_settime(unsigned long tijd){
   /* Begin transfer, RST hoog */
   DS1602_rst(1);

   /* Stuur '1000 0000' */
   DS1602_sendchar(0x01);

   DS1602_sendlong(tijd);

   /* Eind transfer, RST laag */
   DS1602_rst(0);
}

unsigned long DS1602_readtime(void){
   unsigned long tijd;
   /* Begin transfer, RST hoog */
   DS1602_rst(1);
   /* Stuur '1000 0001' */
   DS1602_sendchar(0x81);
   DQ = 1;  //pin in input mode
   tijd = DS1602_readlong();
   DS1602_rst(0);
   CLK = 0;
   DQ = 0;
   return tijd;
}

/* Deze functie stuurt een char naar de DS1602 */
void DS1602_sendchar(char c){
   DS1602_sendbit(c&0x80);
   DS1602_sendbit(c&0x40);
   DS1602_sendbit(c&0x20);
   DS1602_sendbit(c&0x10);
   DS1602_sendbit(c&0x08);
   DS1602_sendbit(c&0x04);
   DS1602_sendbit(c&0x02);
   DS1602_sendbit(c&0x01);
}
/* Deze functie stuurt een long naar de DS1602 */
void DS1602_sendlong(long l){

   DS1602_sendbit(l&0x00000001);
   DS1602_sendbit(l&0x00000002);
   DS1602_sendbit(l&0x00000004);
   DS1602_sendbit(l&0x00000008);
   DS1602_sendbit(l&0x00000010);
   DS1602_sendbit(l&0x00000020);
   DS1602_sendbit(l&0x00000040);
   DS1602_sendbit(l&0x00000080);

   DS1602_sendbit(l&0x00000100);
   DS1602_sendbit(l&0x00000200);
   DS1602_sendbit(l&0x00000400);
   DS1602_sendbit(l&0x00000800);
   DS1602_sendbit(l&0x00001000);
   DS1602_sendbit(l&0x00002000);
   DS1602_sendbit(l&0x00004000);
   DS1602_sendbit(l&0x00008000);

   DS1602_sendbit(l&0x00010000);
   DS1602_sendbit(l&0x00020000);
   DS1602_sendbit(l&0x00040000);
   DS1602_sendbit(l&0x00080000);
   DS1602_sendbit(l&0x00100000);
   DS1602_sendbit(l&0x00200000);
   DS1602_sendbit(l&0x00400000);
   DS1602_sendbit(l&0x00800000);

   DS1602_sendbit(l&0x01000000);
   DS1602_sendbit(l&0x02000000);
   DS1602_sendbit(l&0x04000000);
   DS1602_sendbit(l&0x08000000);
   DS1602_sendbit(l&0x10000000);
   DS1602_sendbit(l&0x20000000);
   DS1602_sendbit(l&0x40000000);
   DS1602_sendbit(l&0x80000000);
}

/* Deze functie leest een long van de DS1602
*  Stuur eerst een "lees" commando, anders slaat het resultaat nergens
op.
*/
unsigned long DS1602_readlong(void){
   unsigned long l = 0;

   l |= ((long)DS1602_readbit() <<31);
```

```
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;

l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;

l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;

l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;
l |= ((long)DS1602_readbit() <<31);
l >>= 1;
```

```
   l |= ((long)DS1602_readbit() <<31);
   l >>= 1;
   l |= ((long)DS1602_readbit() <<31);
   l >>= 1;
   l |= ((long)DS1602_readbit() <<31);
   l >>= 1;
   l |= ((long)DS1602_readbit() <<31);
   l >>= 1;
   l |= ((long)DS1602_readbit() <<31);

   return l;
}


/* Deze functie stuurt een bit naar de DS1602 */
void DS1602_sendbit(bit b){
   CLK = 0;
   /* Herhaal vanwege hoge risetime */
   DQ = b;
   DQ = b;
   DQ = b;
   DQ = b;
   DQ = b;
   DQ = b;
   DQ = b;
   DQ = b;
   DQ = b;
   DQ = b;
   DQ = b;
   CLK = 1;
   CLK = 1;
   CLK = 1;
   CLK = 1;
   CLK = 1;
   CLK = 1;
   CLK = 1;
   CLK = 1;
   CLK = 1;
   CLK = 1;
   CLK = 1;
   CLK = 1;
}


/* Deze functie leest een bit van de DS1602 */
bit DS1602_readbit(void){
   bit b;

   /* Herhaal vanwege hoge risetime */
   CLK = 1;
   CLK = 1;
   CLK = 1;
   CLK = 1;
   CLK = 1;
   CLK = 1;
   CLK = 1;
```

```
   CLK = 1;
   CLK = 1;
   CLK = 1;
   CLK = 1;
   CLK = 1;
   CLK = 0;
   CLK = 0;
   CLK = 0;
   CLK = 0;
   CLK = 0;
   CLK = 0;
   b = DQ;
   return b;
}

/* Deze functie maakt de reset lijn hoog of laag */
void DS1602_rst(bit b){

   /* Herhaal vanwege hoge risetime */
   RST = b;
   RST = b;
   RST = b;
   RST = b;
   RST = b;
   RST = b;
   RST = b;
   RST = b;
   RST = b;
   RST = b;
   RST = b;
   RST = b;
}
```

DIVERSE HULPFUNCTIES

```
/* hulpfuncties.c
*  -Een ongedefinieerd instelbaar delay
*  -Een delay van 1/3 ms
*  -Een functie die wacht tot de testknop wordt ingedrukt
*/

#include <REG51F.H>        /* 80c51 standaard definities */

/* Wacht t cycles */
void wait(unsigned int t){
   unsigned int i;
   for(i=0;i<t;i++){
      ;
   }
}

/* Wacht 1/3 ms, dit komt overeen met 512 klokcycles */
void wait1_3ms(void){
   unsigned char c = 252;
   /* Wacht tot c == 0 */
   while(c){
      c--;
   }
}

/* Wacht tot testbutton wordt ingedrukt en weer losgelaten */
void TestButton(void){
   while(INT0 == 1){}
   wait(1000);
   while(INT0 == 0){}
   wait(1000);
}
```

ENERSTATION

```
/* EnerStation
*  Dit programma communiceert met alle sensoren in het netwerk.
*
*  Eerst wordt een adres gestuurd. De sensor met het juiste adres
*  reageert daarop door twee bytes meetgegevens terug te sturen. Deze
*  gegevens  worden opgeslagen, samen met adres en tijdreferentie.
*
*  Het geheugen kan via de seriële poort uitgelezen worden door een PC.
*/

#include <REG51F.H>
#include <STDIO.H>
#include "autobaud.h"
#include "DS1602.h"
#include "PLC.h"
#include "hulpfuncties.h"

sbit DATA = P2^0;

void main(void){
   unsigned char i;
   /* opslagruimte voor meetgegevens */
   xdata unsigned char adres;
   xdata unsigned long tijd;
   xdata unsigned int meetdata = 100;

   /* Initialiseer PLC link */
   DATA = 0;

   /* Initialiseer seriële poort, zet 'i' op display */
   P1 = 0xAF;
   AutoBaud();
   /* Zet "." op display */
   P1 = 0x7F;
   printf("Initialisatie voltooid.\n");

   /* Stel tijd in */
   tijd = 10000;
   DS1602_settime(10000);
```

```
   while(1){
      /* Wacht tot 1 seconde verstreken is */
      while(tijd == DS1602_readtime());

      for(i=1; i<=5; i++){
         /* Stuur adres */
         adres = i;
         PLC_sendchar(adres);

         /* Ontvang meetdata */
         meetdata = PLC_readchar();
         meetdata <<= 8;
         meetdata |= PLC_readchar();
         meetdata++;


         /* Sla tijd op */
         tijd = DS1602_readtime();

         /* Als meetdata leeg is was er geen sensor, niet printen */
         if(meetdata)
            printf("Adres: %u, Tijd: %lu, Meetdata: %u\n",(int)adres,
               tijd, meetdata);
      }
   }
}
```

SD-CARD

```
// deze code is bedoeld om SPI comunicatie met SD testen stuurt alleen
comand0
//als er op test knoopje wordt gedrukt en zet wordt initialisatie
commando's
// gestuurd naar de SD en de Respons ontvangen en in P2 gezet

// 80c51 standaard definities
#include <REG51F.H>

// Special Function Registers
sfr SPCR = 0xD5;        //SPI Control Register (Reset value 00000000B).
   #define SPIE 0x80    //If both SPIE and ES are set to one, SPI
interrupts are enabled.
   #define SPEN 0x40    //SPI enable bit.  When set enables SPI.
   #define DORD 0x20    //Data trans. order. 0=MSB first; 1=LSB first.
   #define MSTR 0x10    //1=master mode.  0=slave mode.
   #define CPOL 0x08    //1=SCK is high when idle (active low), 0=SCK is
low when idle (active high).
   #define CPHA 0x04    //1=shift triggered on the trailing edge of SCK.
0=shift trig. on leading edge.
   #define SPR1 0x02    //SPI Clork Rate select bit 1.
   #define SPR0 0x01    //SPI Clork Rate select bit 0.
                     //00 = Fosc/4
                     //01 = Fosc/16
                     //10 = Fosc/64
                     //11 = Fosc/128
sfr SPDAT = 0x86;       //SPI data register
sfr SPSR = 0xAA;        //SPI Configuration Register (Reset value
00000000B).
   #define SPIF 0x80    //SPI interrupt flag.
   #define SPWCOL 0x40     //Write collision Flag.
sbit CS = 0x90;            //Chip select

// Function prototypes
void spi_init(void);
void spi_send_byte(unsigned char);
unsigned char spi_rcv_byte(void);
void spi_send_byte(unsigned char input);
void spi_isr(void);
bit command(unsigned char befF, unsigned char AdrH1, unsigned char
AdrH2, unsigned char AdrL1, unsigned char AdrL2);
bit sd_init(void);
void wait(unsigned int t);
void wait1(unsigned int t);

// Global Variables
char serial_data;
bit transmit_completed;
unsigned char t, g;

void main(void)
{
   unsigned char i;
   // SPI initialiseren
   spi_init();

   // SD kaart initialiseren
   sd_init();
   P2=0;
   while(1)
   {
         while(INT0 == 1)
      {
       wait(1000000);
      }

      while(INT0 == 0)
      {
       wait(1000000);
      }

/*    hier wordt twee keer reset commando CMD0 gestuurd
   naar de SD Card, SD Card stuurt vervolgens 1 byte
   terug, deze byte wordt met wait1() functie op gevangen
   en op de P2 gezet zodat je kan zien wat de response in
   houdt en tijdens testen was deze response gelijk aan
   00000001 dat betekent dat SD is in SPI mode, en mag de
   initialisatie proces beginnen */

    command(0, 0, 0, 0, 0);
```

```
/* ook tijdens response ontvangen dient de chip worden geselecteerd */
     CS=0;
     wait1(2);

    command(0, 0, 0, 0, 0);
     CS=0;
     wait1(2);

   for (i=0 ; i<=1 ; i++)
   {
/*  hier wordt de CMD55 gestuurd naar de SD Card
   opvolgend met  ACMD41 die zorgen dat de SD Card wordt
   geinitialisserd, de P2 blijkt uit eindelijk 00000000
   aan tegeven en dat is teken dat SD is ge initialiseerd
   de idle bit is 0
*/
    command(55, 0, 0, 0, 0);
     CS=0;
     wait1(2);

    command(41, 0, 0, 0, 0);
     CS=0;
     wait1(3);
    }
      while(1);
    }
}

void spi_init(void)
{
   /* Stel SPI in */
   SPCR = SPEN               // Enable SPI
       + MSTR                // SPI = Master
       + CPOL                // CLK is high when idle
       + CPHA                // shift triggered on trailing edge of CLK
       + SPR1                // Clock setting 00 = fosc/4; 01 = fosc
/16; 10 = fosc/64; 11 = fosc/128
       + SPR0;
         P1=0;             // maak de P1 eerst 0
}

// Send a byte to SPI
void spi_send_byte(unsigned char d)
{
   SPSR &= ~SPIF;            // Clear interrupt flag
   SPDAT = d;                // Load data register to start transfer
   while((SPSR & SPIF) == 0);   // Wait for the transfer to complete
}

// Receive a byte from SPI
unsigned char spi_rcv_byte(void)
{  T2=0;
   SPSR &= ~SPIF;            // Clear interrupt flag
   SPDAT = 0xff;             // Send a dummy byte to generate clock
pulses
   while ((SPSR & SPIF) == 0);     // Wait for the transfer to complete
   return SPDAT;             // Return the received byte
   T2=1;
}
// Send command   to SD Card
bit command(unsigned char cmd,
          unsigned char AdrH1,
          unsigned char AdrH2,
          unsigned char AdrL1,
          unsigned char AdrL2)
{
   CS = 0;
   spi_send_byte((cmd & 0x3f) | 0x40);
   spi_send_byte(AdrH1);
   spi_send_byte(AdrH2);
   spi_send_byte(AdrL1);
   spi_send_byte(AdrL2);
   spi_send_byte(0x95);
   CS=1;
}

//* SD Initialisatie functie */
bit sd_init()
{
   unsigned char i,s, ans = 1;
   CS = 0;

   // Delay 74 clocks
   for(i = 0; i < 10; i++)
   {
      spi_send_byte(0xFF);    // send 10 * 8 = 80 clock pulses
   }

   CS = 1;

   // Delay 16 clocks
   for(i = 0; i < 2; i++)
   {
      spi_send_byte(0xFF);    // send 2 * 8 = 16 clock pulses
   }
   return 1;
}

void wait(unsigned int t)
{
   unsigned int i;
   for(i = 0; i < t; i++);
}

void wait1(unsigned int t)
{
   unsigned int i;
   for(i = 0; i < t; i++)
   {
     spi_send_byte(0xFF);
     g =SPDAT;
     P2= g;
```

```
   }
}
```

SPI

```
// deze code is bedoeld om SPI comunicatie te testen tussen twee
// microcontrollers om, elke keer ene microcontroller slave maken
// en andere master, en in ene geval een byte sturen en in andere
// geval byte ontvangen en dat op de P2 zichbaar make (op ledkastje)
// 80c51 standaard definities
#include <REG51F.H>
#include "testSPI.H"

void main(void)
{
   unsigned char i,s;
   int j,d;

   spi_initialize();
   P2=0;
   while(1)
   {
      while(INT0 == 1)      // na reset begint de
                           //microcontroller ontvangen   (slave mode)
      {
         spi_slave();
          d = spi_rcv_byte();
         P2 = d ;
      }

      while(INT0 == 0)     // na eerste test druk gaat hij wachten
      {
         P2 = 1;
         wait(80000);
          wait(80000);
         wait(80000);
      }

      while(INT0 == 1)              // na tweede keer test druk wordt
hij master en gaat verzenden
      {

         spi_master();

         CS=0;
         wait(1);

         s = 0xAA ;
         spi_send_byte(s);          // bericht is 0xAA

         wait(1);
         CS=1;

         P2= 2;                     // flag : deze microcontroller is
aan het verzenden

      }

      while(INT0 == 0)           // na eerste test druk
                                 //gaat hij wachten
      {
         P2 = 3;
         wait(80000);
          wait(80000);
         wait(80000);
      }


      while(INT0 == 0)
      {                          // na derde test druk gaat hij wachten
         P2 = 4;
         wait(500000);
      }
   }
}

/* Initialize and enable the SPI module */
void spi_initialize(void)
{
   /* Stel SPI in */
   SPCR = ~SPIE               // Disable SPI Interrupt
       + MSTR                 // SPI = Master
       + CPHA;             //

   spi_enable();
}

void spi_enable(void)
{
   /* Enable the SPI module */
   SPCR |= SPEN;
}

/* Receive a byte. Output an 0xFF (the bus idles high) to receive the
byte */
unsigned char spi_rcv_byte(void)
{
   unsigned char tmp;
   SPSR &= ~SPIF;       // clear interrupt flag

   /* Send the byte */
   SPDAT = 0xFF;

   /* Wait for the byte to be received */
   while (SPIF == 0) { }
   tmp = SPDAT;
   return (tmp);

}

/* Send a single byte over the SPI port */
```

```
void spi_send_byte(unsigned char input)
{
   SPSR &= ~SPIF;             // clear interrupt flag
   SPDAT = input;             // load data register
   while(!SPIF);              // wait until byte sent
}

void wait(int t)
{
   int i;
   for(i = 0; i < t; i++);
}

void spi_master(void)
{
   SPCR |= MSTR;                 // SPI = Master

   EA = 1;                    // Enable interrupts
   ES = 1;                    // Enable SPI Interrupt

   /* Stel SPI in */
   SPCR = SPIE                // Enable SPI Interrupt
       + SPEN                 // Enable SPI
                           // DORD=0  hier door wordt MSB als eerst
verzonden
       + MSTR                 // SPI = Master
       + CPHA                 //
       + SPR1                 // Clock setting 00 = fosc/4; 01 = fosc
/16; 10 = fosc/64; 11 = fosc/128
       + SPR0;

}

void spi_slave(void)
{
   SPCR &= ~MSTR;                // SPI = Slave

   EA = 1;                    // Enable interrupts
   ES = 1;                    // Enable SPI Interrupt

   /* Stel SPI in */
   SPCR = SPIE                // Enable SPI Interrupt
       + SPEN                 // Enable SPI
       + DORD                 // voor het ontvangen is het nodig om LSB
eerst ontvangen
                           // omdat bij master MSB eerst verzonden wordt

                           // SPI = Slave
       + CPHA                 //
       + SPR1                 // Clock setting 00 = fosc/4; 01 = fosc
/16; 10 = fosc/64; 11 = fosc/128
       + SPR0;

}
```