C-Control Pro
Mega Series

Inhoudsopgave

Hoofdstuk

1 Belangrijke aanwijzingen

Dit hoofdstuk behandelt belangrijke informatie voor de garantie en support en gebruik van de C-control.Por hardware en software.

1.1 Introductie

De C-Control Pro systemen zijn gebaseerd op de Atmel Mega 32 resp. de Atmel Mega 128 RISC microcontroller. Deze microcontroller wordt in zeer vele apparaten in grote aantallen toegepast. Van de amusementselektronica, via huishoudmachines tot verschillende toepassingsmogelijkheden in de industrie. Daar neemt de controller belangrijke besturingsopgaven over. C-Control Pro biedt u deze uiterst moderne technologie om uw besturingsproblemen op te lossen. U kunt analoge meetwaarden en schakelposities registreren en afhankelijk van deze ingangscondities de desbetreffende schakelsignalen afgeven, bijv. voor relais of servomotoren. In combinatie met een zendergestuurde DCF77 –antenne kan C-Control Pro de atoom –exacte tijd ontvangen en precieze schakelklokfuncties overnemen. Verschillende hardware –interfaces en bussystemen maken het mogelijk C-Control Pro te koppelen met sensoren, actoren en andere besturingssystemen. Wij willen onze technologie ter beschikking stellen aan een brede toepassingskring. Wij weten vanuit onze werkzaamheden bij de C-Control serviceafdeling, dat ook klanten zonder enige ervaring met elektronica en elektrotechniek maar graag iets daarover willen leren, geïnteresseerd zijn in C-Control. Als u tot deze toepassingsgroep behoort, sta ons dan toe op deze plaats een tip te geven:

C-Control Pro is slechts in beperkte mate geschikt om in te stappen in de programmering van microcomputers en de elektronische schakeltechniek! Wij stellen als voorwaarde dat u minimaal beschikt over basiskennis betreffende een hogere programmeertaal, zoals bijv. BASIC, PASCAL, C, C++ of Java. Bovendien nemen wij aan, dat u vertrouwd bent met de bediening van een PC onder één van de Microsoft Windows besturingssystemen (98SE/NT/2000/ME/XP). U dient ook enige ervaring te hebben met het hanteren van soldeerbouten, multimeters en elektronische componenten. We hebben ons best gedaan alle beschrijvingen zo eenvoudig mogelijk te formulieren. Helaas kunnen wij in een gebruiksaanwijzing over het onderhavige thema niet steeds afzien van het gebruik van vaktermen en anglicismen. Sla indien nodig de desbetreffende vakliteratuur er op na.

1.2 Lezen van deze gebruiksaanwijzing

Lees deze gebruiksaanwijzing helemaal door voor u de C-Control Pro unit in gebruik neemt. Terwijl sommige hoofdstukken alleen van belang zijn voor het begrijpen van de diepere samenhang, bevatten andere hoofdstukken belangrijke informatie; als u dit niet in acht neemt, kan dat leiden tot foutief functioneren of tot beschadigingen.

➔ Hoofdstukken en alinea's die belangrijke thema's bevatten, worden gekenmerkt door het symbool ➔. Lees deze aanwijzingen bijzonder intensief door.

Lees voor de ingebruikneming de volledige gebruiksaanwijzing door, er staan belangrijke aanwijzingen in betreffende het correcte gebruik. Bij materiële schade of persoonlijk letsel die/dat veroorzaakt wordt door onvakkundig gebruik of het niet in acht nemen van deze gebruiksaanwijzing, vervalt het recht op garantie! Wij zijn niet aansprakelijk voor schades die daarvan het gevolg zijn!

1.3 Gebruik

De C-Control Pro unit bevat gevoelige componenten. Deze kunnen door elektrostatische ontladingen vernield worden! Let op de algemene regels voor het gebruik van elektronische componenten. Richt uw werkplek vakkundig in. Aard uw lichaam voor u begint met de werkzaamheden, bijv. door het aanraken van een geaard, geleidend voorwerp (bijv. een radiator). Vermijd het aanraken van de aansluitpins van de C-Control Pro unit.

1.4 Correcte toepassing

De C-Control Pro unit is een elektronische component in de zin van een geïntegreerd schakelcircuit. De C-Control Pro unit is bedoeld voor de programmeerbare aansturing van elektrische en elektronische apparaten. De opbouw en het gebruik van deze apparaten moet gebeuren conform de geldende Europese toelatingsrichtlijnen (CE).

De C-Control Pro unit mag niet in galvanische verbinding staan met spanningen hoger dan beveiligde laagspanning. De koppeling aan systemen met een hogere spanning mag uitsluitend plaatsvinden via componenten met VDE –toelating. Daarbij moeten de voorgeschreven lucht – en kruipafstand aangehouden worden en moeten er tevens voldoende maatregelen getroffen worden ter bescherming tegen het aanraken van gevaarlijke spanningen.

Op de printplaat van de C-Control Pro unit werken elektronische componenten met hoogfrequente kloksignalen en steile pulsflanken. Bij onvakkundig gebruik van de unit kan dit leiden tot het uitzenden van elektromagnetische stoorsignalen. Het gebruik van desbetreffende maatregelen (bijv. het gebruik van smoorspoelen, limietweerstanden, blokcondensatoren en afschermingen) valt onder de verantwoordelijkheid van de gebruiker.

De maximaal toegestane lengte van aangesloten kabels zonder extra maatregelen bedraagt 0,25 meter (uitgezonderd de seriële interface). Onder invloed van sterke elektromagnetische wisselvelden of stoorimpulsen kan de functie van de C-Control Pro unit beïnvloed worden. Eventueel is in dat geval een reset en het opnieuw starten van het systeem noodzakelijk.

Let bij het aansluiten van externe modules op de maximaal toelaatbare stroom – en spanningswaarden van de aparte pins. Het aanleggen van een verkeerd gepoolde of te hoge spanning of een belasting met een te hoge stroom kan leiden tot de onmiddellijke vernieling van de unit. Houd de C-Control Pro unit uit de buurt van spatwater en condenswater. Let op het toelaatbare temperatuurbereik in de technische specificaties in de bijlage.

1.5 Garantie en aansprakelijkheid

Conrad Electronic biedt voor de C-Control Pro unit een garantieperiode van 24 maanden gerekend vanaf de datum van aankoop. Binnen deze periode worden defecte units gratis omgeruild, als het defect aantoonbaar terug te voeren is op een productiefout of aan transportschade.

De software in het besturingssysteem van de microcontroller alsmede de PC –software op CD-ROM worden in de aanwezige vorm geleverd. Conrad Electronic geeft geen garantie dat de prestatiekenmerken van deze software voldoen aan individuele eisen en dat de software in elk geval werkt zonder onderbrekingen en fouten. Conrad Electronic is niet aansprakelijk voor schade die rechtstreeks door of ten gevolge van de toepassing van de C-Control Pro unit ontstaan. Het gebruik van de C-Control Pro unit in systemen, die direct of indirect bedoeld zijn voor medische, gezondheid – of leven beschermende doelen, is niet toegestaan.

Als de C-Control Pro unit inclusief software niet aan uw eisen voldoet, of u bent het niet eens met de garantie – en aansprakelijkheidsbepalingen, maak dan gebruik van onze 14-daagse geld-terug-garantie. Stuur ons dan de unit binnen deze termijn zonder sporen van gebruik, in de onbeschadigde originele verpakking en inclusief alle accessoires terug voor terugbetaling of verrekening van de waarde van dit artikel!

1.6 Service

Conrad Electronic stelt u een team van ervaren servicemedewerkers ter beschikking. Als u vragen heeft over de C-Control Pro unit, kunt u onze klantenservice bereiken per brief, fax of e-mail.

Per brief Conrad Electronic
Technische Anfrage
Klaus Conrad-Strasse 2
92530 Wernjberg-Köblitz
Bundesrepublik Deutschland

Faxnr. 0049-9604 / 408848
e-mail webmaster@c-control.de

Onze voorkeur gaat uit naar communicatie per e-mail. Als u een probleem heeft, geef ons dan indien mogelijk een schets van uw aansluitschakeling als bijgevoegd beeldbestand (in JPG-formaat) alsmede de tot het probleem gereduceerde programma– brontekst (maximaal 20 regels). Meer informatie en actuele software om te downloaden vindt u op de C-Control homepage op internet onder www.c-control.de.

1.7 Open Source

Bij het maken van C-Control Pro is ook Open Source software gebruikt:

ANTRL 2.73	http://www.antrl.org
Inno Setup 5.15	http://www.jrsoftware.org
GPP (Generic preprocessor)	http://www.nothingisreal.com/gpp

Volgens de bepalingen van de “LESSER GPL” (www.gnu.org/copyleft/lesser) wordt bij de installatie van de IDE ook de originele Source Code van de Generic preprocessor meegeleverd, alsmede de brontekst van de aangepaste versie, die bij de C-Control Pro gebruikt wordt. Beide bronteksten zijn te vinden in het “GNU” submenu in een ZIP archief.

1.8 Historie

Versie 1.50 d.d. 08.11.2005

Nieuwe features

- IDE ondersteuning voor Mega128
- Verbeterde cache algorithme bij toegang IDE op looptijddata in de debugger
- Nieuwe bibliotheekroutines voor Timer 3 (Mega128)
- Programma's gebruiken de uitgebreide (>64Kb) adresplaats (Mega128)
- Ondersteuning externe 64Kb SRAM
- Externe interrupts 3-7 worden ondersteund (Mega128)
- Routines voor 2de seriele interface (Mega128)
- Mathematische functies (Mega128)

- Weergave van de geheugengrootte bij start van de interpreter
- Interne RAM check voor de herkenning wanneer globale vaiabelen te groot voor hoofdgeheugen
- Interne RAM check voor de herkenning wanneer thread configuratie te groot voor hoofdgeheugen
- Looptijdcontrole of stacklimiets beschadigd worden
- Bronbestanden kunnen in de project hiërarchie naar boven en onder beweegt worden
- Waarschuwing bij toewijzing van te lange strings
- De compiler maakt naar wens een map-bestand die de grootte van alle programmavariabelen beschrijft
- Nieuw adresmodel voor globale variabelen (hetzelfde programma loopt op verschillende RAM-groottes)
- Interruptroutines voor seriële interface (max. 256 Byte ontvangstbuffer / 256 Byte zendbuffer
- Vaste bedrading IRQ routines om een periodemeting van kleine tijdspannes mogelijk te maken
- Recursies kunnen nu onbeperkt gebruikt worden
- Willekeurig grote arrays kunnen in de debugger weergegeven worden in een eigen venster
- Strings (character arrays) worden nu als tooltip in de debugger getoond
- SPI kan uitgeschakeld worden om de pins als I/O te gebruiken
- De seriele interface kan uitgechakeld worden om de pins als I/O te gebruiken
- De hex-waarde wordt nu extra als tooltip in de debugger getoond
- Nieuwe functie Thread_MemFree()
- Extra EEPROM routines voor woord- en floating point-toegang
- Tijdmeting met Timer_TickCount()
- #pragma commando's om fouten of waarschuwingen te maken
- voorgedefinieerd symbool in de preprocessor: __DATE__,__TIME__ __FILE__,__FUNCTIE__;__LINE__
- Versienummer in splashscreen
- Aanvulling documentatie
- Interactieve grafiek bij "Jumper Application Board" in het helpbestand
- Nieuwe demo-programma's
- Ctrl-F1 start contexthulp

Fout-correcties
- Een fout wordt geproduceerd wanneer er geen return-aanwijzing op het einde van een functie is
- Breakpoint markeringen worden niet meer gewist
- Limieten bij EEPROM-toegang nauwkeuriger gecontroleerd (interne overloop opgevangen)
- Enkele stap kan in de debugger niet meer te vroeg het volgende commando afzetten

Versie 1.39 d.d. 09.06.2005

Nieuwe features
- BASIC ondersteuning
- CompactC en BASIC kunnen in een project gemengd worden
- Aanvulling documentatie
- Lusoptimalisatie voor For – Next in BASIC
- Thread-info functie

- Nieuwe demo-programma's

Foutcorrecties
- Bij umlauten crasht de compiler niet meer
- Interne bytecode commando StoreRel32XT gecorrigeerd
- Offset in stringtabel verbeterd

Versie 1.28 d.d. 26.04.2005
- Initiaalversie

Hoofdstuk

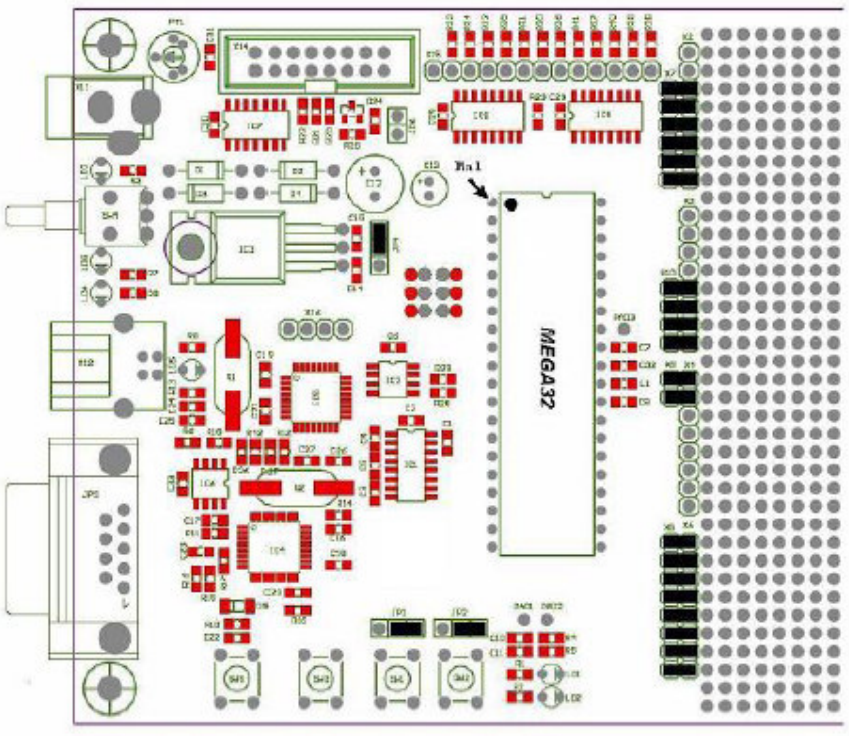
2 Installatie

In dit hoofdstuk wordt de installatie van de hard- en software beschreven.

2.1 Application Board

Belangrijke aanwijzing voor het in- en uitbouwen van een Mega-module
Voor de verbinding van de module en het Application Board moeten hoogwaardige insteekpinnen gebruikt worden, die een goed contact garanderen. De in- en uitbouw van een module mag uitsluitend met uitgeschakelde voedingsspanning (spanningsvrij) uitgevoerd worden, anders kunnen vernielingen op het Application board of de module ontstaan. Door het aantal contacten (40/64 pins) is er aardig wat kracht nodig om de module in- en uit te bouwen. Bij de inbouw moet u er op letten dat de module gelijkmatig, d.w.z. niet gekanteld, in de fitting gedrukt wordt. Leg het Application board hiertoe op een vlakke ondergrond. Monteer de module Mega32 met de juiste plaatsbepaling. Hiertoe de pin 1- markering in acht nemen. De opschriften van de module wijzen dan naar de bedieningselementen op het Application board.

Inbouwrichting module Mega32



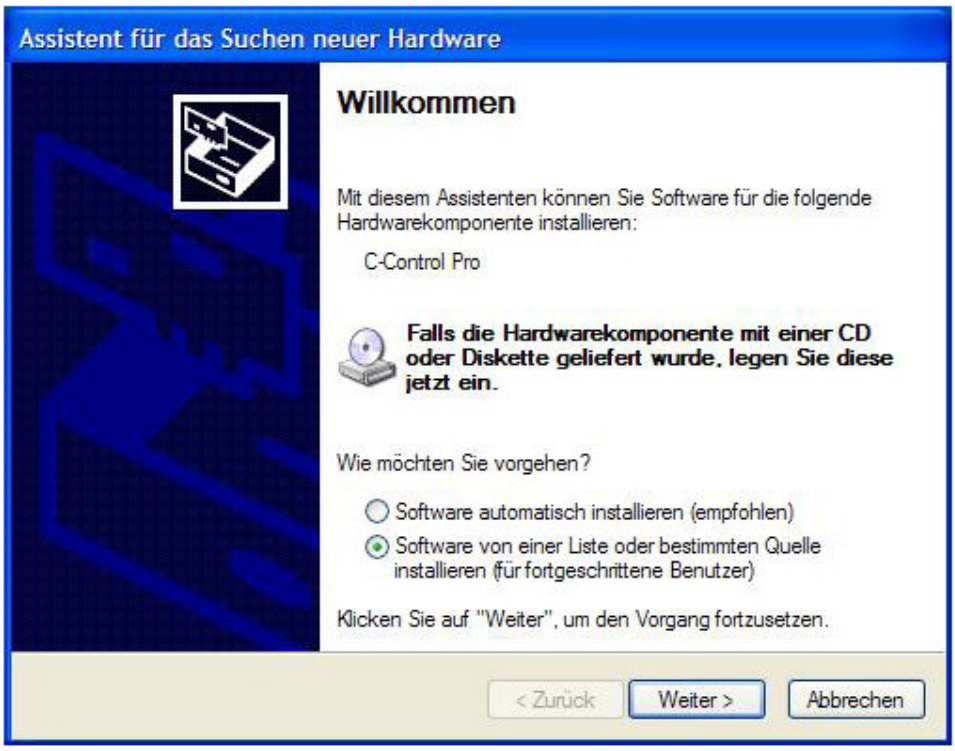
Bij de module Mega 128 zijn de insteekpinnen zo geplaatst dat er geen verkeerde inbouw mogelijk is. Bij de uitbouw wordt de module voorzichtig met een geschikt gereedschap uit de fitting getild. Om de aansluitingen niet om te buigen moet het omhoogheffen op verschillende plaatsen van de module gebeuren.

Installatie van de USB driver

Verbind het Application board met een netvoeding. U kunt hiervoor een standaard stekkernetvoeding met 9V/250mA gebruiken. U kunt de poling zelf uitkiezen, deze wordt door diodes steeds correct omgezet. Afhankelijk van extra schakelingen kan het later noodzakelijk zijn een netvoeding met een hogere capaciteit te gebruiken. Maak een verbinding tussen het Application board en uw PC met behulp van een USB kabel. Schakel het Application board in.

➔ Een Windows besturingssysteem ouder dan Win98 SE (*"Second Edition")* zal vermoedelijk geen betrouwbare USB verbinding mogelijk maken tussen PC en Application board. De USB drivers van Microsoft functioneren pas vanaf Win98 SE betrouwbaar met alle USB apparatuur. In een dergelijk geval kunnen we u alleen maar aanraden over te stappen naar een actueler besturingssysteem, of alleen de seriële verbinding naar het Application board te gebruiken.

Als het Application board voor de eerste keer aangesloten is, zal er nog geen driver voor de FTDI chip aanwezig zijn. Onder Windows XP wordt dan het volgende venster getoond:



U dient hier "Software uit een lijst of een bepaalde bron installeren" te kiezen en op "Volgende" te klikken.

Daarna dient u het pad naar de map van de driver aan te geven. Als u de software naar “C:\Programma’s” geïnstalleerd heeft, is het pad “C:\Programma’s\C-Control\FTDI USB Driver”.

Het bericht “C-Control Pro USB Device hat den Windows-Logo-Test nicht bestanden…” (“C-Control Pro USB Device heeft de Windows-Logo-Test niet doorstaan…”) is heel normaal. Het betekent **niet**, dat de driver bij de Windows-Logo-Test gefaald heeft, maar dat de driver niet deelgenomen heeft aan de (tamelijk dure) test in Redmond.

Op deze plek drukt u gewoon op “Installatie voortzetten”. Na een paar seconden moet de driver dan volledig geïnstalleerd zijn.

In de PC –software klikt u in het menu Opties op IDE en selecteert u het bereik “Interfaces”. Kies daar de communicatiepoort "USB0".

Seriële aansluiting
Vanwege de langzame overdrachtsnelheid van de seriële interface heeft een USB aansluiting de voorkeur. Als er echter vanwege de hardware geen USB interface beschikbaar is, kan de bootloader naar de seriële modus gebracht worden.

Hiertoe moet bij het inschakelen van het Application board de toets SW1 ingedrukt gehouden worden. Daarna is de seriële bootloader modus geactiveerd.

In de PC-software klikt u op het punt IDE in het menu opties en daar kiest u het bereik interfaces. Daar kiest u een communicatieport “COMx”, die bij de interface op de PC past, waarop het board aangesloten is.

2.2 Software

Als de meegeleverde CD in de computer gelegd wordt, dient de installer automatisch te starten, om de C-Control Pro software te installeren. Als dat niet gebeurt, bijv. omdat de “Autostart” –functie voor CD’s of DVD’s in Windows uitgeschakeld is, start dan de installer handmatig met “C-ControlSetup.exe” in het hoofdbestand van de CD-ROM.

➔ Voor de installatie van de software en de installatie van de USB –drivers moet u als "Administrator" aangemeld zijn. Bij het normale werken met C-Control Pro is dit niet nodig.

➔ Om de consistentie van de Demo-programma's te behouden, wordt bij een nieuwe installatie op een eerdere installatie de oude map van de demoprogramma's vervangen door een neiuwe. Om die reden raden wij u aan, uw eigen programma's buiten de map C-Control-Pro te bewaren.
Aan het begin van de installatie kiest u in welke taal de installatie uitgevoerd moet worden. Daarna kunt u uitzoeken, of C-Control Pro in de standaardmap geïnstalleerd moet worden, of dat u een eigen doelmap wilt aangeven. Aan het eind van de installatie wordt u nog gevraagd, of er iconen op uw desktop geïnstalleerd moeten worden.

Als de installatieprocedure afgesloten is, kunt u naar wens eerst het “ReadMe” –bestand (korte introductie) weergeven, of de C-Control Pro ontwikkelingsomgeving starten.

Hoofdstuk

3 Hardware

In dit hoofdstuk wordt de hardware beschreven die bij de C-Control Pro serie gebruikt wordt. Hier wordt de module beschreven van C-Control Pro Mega32 en C-Control Pro Mega128. Verdere paragrafen verklaren de opbouw en functie van het bijhorende Application board en de meegeleverde LCD module en het toetsenbord.

3.1 Firmware

Het besturingssysteem van de C-Control Pro bestaat uit de volgende componenten:
- *Bootloader*
- *Interpreter*

Bootloader
De bootloader staat altijd tot uw beschikking. Deze zorgt voor de USB of seriële communicatie met de IDE. Via regel–commando's kunnen de interpreter en het toepassingsprogramma van de PC naar de Atmel Risc chip overgebracht worden. Als een programma gecompileerd wordt en overgebracht wordt naar de mega chip, dan wordt tegelijkertijd ook de actuele interpreter mee overgebracht.

➔ Als er in plaats van de USB interface een seriële verbinding van de IDE naar de C-Control Pro module opgebouwd moet worden, dan dient u bij het inschakelen van de module de toets SW1 (Poort **M32**:D.2 resp. **M128**:E.4 op low) ingedrukt te houden. In deze modus wordt elke communicatie via de seriële interface geleid. Dit is praktisch, als de module al in de hardware applicatie is ingebouwd, en het Application board daarom niet ter beschikking staat. De seriële communicatie is echter aanzienlijk langzamer dan een USB verbinding. In de seriële modus worden de pins voor USB niet gebruikt en staan de gebruiker voor andere doeleinden ter beschikking.

➔ Omdat de SW1 bij het starten van de module de seriele bootloader inleidt, mag op de Port **M32**:D.2 resp. **M128**:E.4 bij het inschakelen van de applicatie geen signaal aanwezig zijn. Deze poorten kunnen namelijk ook als uitgangen gebruikt worden.

SPI uitschakeling (alleen Mega128)

Een signaal op de SPI interface, bij het inschakelen van de module, kan de USB communicatie activeren. Om dit te voorkomen kan men PortG.4 (LED 2) bij het inschakelen op low zetten en daardoor wordt de SPI interface niet geconfigureerd. De SPI interface kan ook later door de interpreter handmatig met SPI Disable() uitgeschakeld worden.

Interpreter
De interpreter bestaat uit meerdere componenten:
- Bytecode interpreter
- Multithreading ondersteuning
- Interrupt -verwerking
- Toepassingsfuncties
- RAM en EEPROM interface

In de hoofdzaak werkt de interpreter de bytecode af, die door de compiler gegenereerd is. Verder zijn de meeste bibliotheekfuncties in de interpreter geïntegreerd, opdat het bytecode –programma bijv. toegang kan krijgen tot hardware-poorten. De RAM en EEPROM interface wordt gebruikt door de debugger in de IDE, om toegang te krijgen tot variabelen, als de debugger gestopt is bij een breakpoint.

Autostart

Als er geen USB interface is aangesloten, en u heeft bij het inschakelen niet op SW1 gedrukt om in de seriële bootloadermodus te komen, dan wordt de bytecode (voor zover aanwezig) in de interpreter gestart. Dat wil zeggen, als de module in een hardware applicatie ingebouwd wordt, dan is het aanleggen van de voedingsspanning voldoende om het toepassingsprogramma automatisch te starten.

3.2 Mega32

Mega32 overzicht

De microcontroller ATmega32 komt uit de AVR –familie van ATMEL. Dit betreft een low-power microcontroller met Advanced RISC Architecture. Hier volgt een korte samenstelling van de hardware resources:

- **131 Powerful instructions – Most Single-clock Cycle Execution**
- **32 x 8 General purpose Working Registers**
- **Up to 16 MIPS Throughput at 16 MHz**

- **Nonvolatile Program and Data Memories**
  32K Bytes of In-System Self-Programmable Flash
  Endurance: 10,000 Write/Erase Cycles
  In-System Programming by On-chip Boot Program

- **1024 Bytes EEPROM**
- **2K Byte Internal SRAM**

- **Peripheral Features:**
  Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  One 16-bit Timer/Counter with Separate Prescaler, Compare mode and Capture Mode
  Four PWM Channels
  8-channel, 10-bit ADC
  8 Single-ended Channels
  2 Differential Channels with Programmable Gain at 1x, 10x or 200x
  Byte-oriented Two-wire Serial Interface (I2C)
  Programmable Serial USART
  On-chip Analog Comparator
  External and Internal Interrupt Sources
  32 Programmable I\O Lines

- **40-pin DIP**
- **Operating Voltages 4.4 – 5.5 V**

3.2.1 Module

Modulegeheugen
In de C-Control Pro module zijn 32kB FLASH, 2kB EEPROM en 2kB SRAM geïntegreerd. Op het Application board bevindt zich een extra EEPROM met een geheugen van 8kB. Dit EEPROM kan aangesproken worden via een 12C interface.

Aanwijzing: U vindt gedetailleerde informatie in de PDF-bestanden van de IC –fabrikanten op de C-Control Pro software CD.

ADC Referentiespanning –opwekking

De microcontroller beschikt over een analoog –digitaal –omvormer met een resolutie van 10Bit. Dit betekent dat gemeten spanningen als gehele getallen van 0 tot 1023 weergegeven kunnen worden. De referentiespanning voor de ondergrens is het GND- niveau, dus 0V. De referentiespanning voor de bovengrens kan door de gebruiker gekozen worden:

* 5V voedingsspanning (VCC)
* interne referentiespanning van 2,56V
* externe referentiespanning bijv. 4,096V gegenereerd door referentiespannings –IC

Als x een digitale meetwaarde is, wordt de desbetreffende spanningswaarde als volgt berekend:

u = x * referentiespanning / 1024

Genereren van klokfrequentie

Het genereren van de klokfrequentie gebeurt door een 14,7456MHz kwartsoscillator. Alle tijdprocedures van de controller zijn van deze klok-frequentie afgeleid.

Reset

Een reset zorgt voor het terugkeren van het microcontroller –systeem naar een gedefinieerde begintoestand. De C-Control Pro module kent in principe twee reset –bronnen:

- Power-On -Reset: wordt automatisch uitgevoerd na het inschakelen van de voedingsspanning
- Hardware -Reset : wordt uitgevoerd als de RESET (pin 9) van de module op “low” getrokken en weer losgelaten wordt, bijv. door het kort indrukken van de aangesloten reset –toets RESET1 (SW3).

Door een “Brown-Out-Detection” wordt voorkomen dat de controller bij het te laag worden van de voedingsspanning in een ongedefinieerde toestand terecht kan komen.

Digitale poorten (PortA, PortB, PortC, PortD)

De C-Control Pro module beschikt over vier digitale poorten met elk 8 pinnen. Op de digitale poorten kunnen bijv. toetsen met pull -up weerstanden, digitale –IC's, opto–koppelingen of driverschakelingen voor relais aangesloten worden. De poorten kunnen apart, d.w.z. pin- of bytewijze aangesproken worden. Elke pin kan of uitgang of ingang zijn.

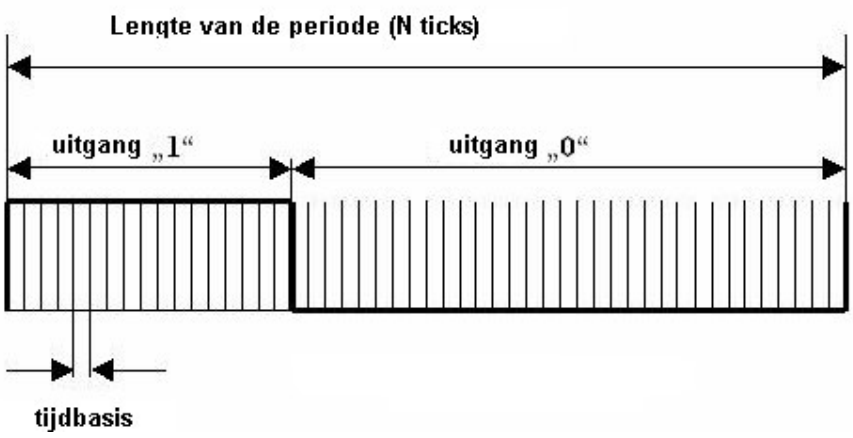
➔ Schakel nooit direct twee poorten samen, die gelijktijdig als uitgang moeten werken!

Digitale ingangpinnen zijn hoogohmig of met een interne pull-up weerstand geschakeld en zetten een aanwesend spanningssignaal om in een logische waarde. Voorwaarde daarvoor is, dat het spanningsignaal zich binnen het voor TTL – resp. CMOS –IC's gedefinieerde bereiken voor low - of high- niveau bevindt. In de verdere bewerking in het programma worden de logische waarden van aparte ingangspoorten als 0 (“low”) of –1 (“high”) weergegeven. Pinnen nemen waarden van 0 of 1 aan, bytepoorten 0 tot 255. Uitgangspoorten kunnen via een interne driverschakeling digitale spanningssignalen afgeven. Aangesloten schakelingen kunnen een bepaalde stroom uit de poorten trekken (bij high niveau) resp. deze poorten er mee voeden (bij low niveau).

➔ Let op de maximaal toelaatbare laststroom voor een afzonderlijke poort en voor alle poorten bij elkaar. Een overschrijding van de maximale waarden kan leiden tot vernieling van de C-Control Pro module. Na de reset is in eerste instantie elke digitale poort als ingangspoort geconfigureerd. Via bepaalde commando’s kan de datarichting omgeschakeld worden.

➔ Het is belangrijk om vóór de programmering de pintoewijzing van M32 en M128 te bestuderen, aangezien belangrijke functies van de programma- ontwikkeling (bijv. de USB –interface van het Application board) op bepaalde poorten liggen. Als deze poorten omgeprogrammeerd worden of als de bijbehorende jumpers op het Application board niet meer gezet zijn, kan het gebeuren dat de ontwikkelingsomgeving geen programma’s meer kan overbrengen naar de C-Control Pro. Ook in- en uitgangen van de timer, A/D omvormer, I2C en de seriële interface zijn met bepaalde poortpinnen verbonden.

PLM -ports
Er zijn twee timers beschikbaar voor de PLM, *Timer_0* met 8 bit en *Timer_1* met 16 bit. Deze kunnen gebruikt worden voor de D/A –omvorming, voor het aansturen van servomotoren in de modelbouw of voor het afgeven van audio -frequenties. Een pulslengte –gemoduleerd signaal heeft een periode van N zogenaamde “Ticks”. De duur van een tick is de tijdbasis. Als u de uitvoerwaarde van een PLM –poort op X zet, dan houdt deze gedurende X ticks van een periode high niveau en valt voor de rest van de periode op low. Voor de programmering van de PLM –kanalen zie Timer.



De PLM –kanalen voor *Timer_0* en *Timer_1* hebben een onafhankelijke tijdbasis en periodelengte. In toepassingen voor pulsbreedte –gemoduleerde digitaal- analoog -omvorming worden tijdbasis en periodelengte eenmalig ingesteld en daarna wordt alleen de uitvoerwaarde veranderd. De PLM –poorten zijn vanwege hun elektrische eigenschappen digitale poorten. Let op de technische randvoorwaarden voor digitale poorten (max. stroom).

Technische specificaties module

Aanwijzing: u vindt gedetailleerde informatie in de PDF –bestanden van de IC –fabrikanten op de C-Control Pro software CD.

Alle spanningen hebben betrekking op gelijkspanning (DC).

Omgevingscondities	
Bereik van de toelaatbare omgevingstemperatuur	0 °C … 70 °C
Bereik van de toelaatbare relatieve luchtvochtigheid van de omgeving	20% …60%
Voedingsspanning	
Bereik van de toelaatbare voedingsspanning	4.5V … 5,5V
Stroomverbruik van de module zonder externe lasten	ca. 20mA
Puls	
Pulsfrequentie (kwarts –oscillator)	14,7456MHz
Mechanische deel	
Buitenafmetingen zonder pinnen ca.	53 mm x 21 mm x 8 mm
Gewicht	ca. 90g
Pinraster	2,54mm
Aantal pins (2 rijen)	40
Afstand van de rijen	15,24mm
Poorten	
Max. toelaatbare stroom uit digitale poorten	± 20mA
Toelaatbare totaal van de stromen op digitale poorten	200mA
Toelaatbare ingangsspanning op poortpins (digitaal en A/D)	-0,5V … 5,5V
Interne pull –up weerstanden (uitschakelbaar)	20 – 50 kOhm

3.2.2 Application Board

USB

Het Application board beschikt over een USB interface voor het laden en debuggen van het programma. Door de hoge datasnelheid van deze interface zijn de dataoverdracht –tijden aanzienlijk korter vergeleken met de seriële interface. De communicatie vindt plaats via een USB –controller van FDTI en een AVR Mega8 controller. De mega8 heeft een eigen reset –toets (SW5). Tijdens het USB gebruik wordt de status van de interface getoond via twee

lichtdiodes (LD4 rood, LD5 groen). Als alleen de groene LED oplicht, dan is de USB interface klaar voor gebruik. Als er een dataoverdracht plaatsvindt, branden beide LEDs. Dit geldt ook voor de debug –modus. Het knipperen van de rode LED geeft een foutconditie aan. Voor de USB –communicatie wordt de SPI –interface van de mega32 gebruikt (PortB.4 t/m PortB.7, PortA.6, PortA.7) en deze moeten via de desbetreffende jumpers verbonden zijn.

Aanwijzing: U vindt meer gedetailleerde informatie over de Mega8 in de PDF –bestanden van de IC –fabrikanten op de C-Control Pro software CD.

Aan - /Uitschakelaar
De schakelaar SW4 bevindt zich aan de voorkant van het Application board en is bedoeld voor het in -/uitschakelen van de spanningvoorziening.

Lichtdiodes
Er zijn 5 lichtdiodes beschikbaar. LD3 (groen) bevindt zich aan de voorkant onder de DC –aansluiting en brandt, als er voedingsspanning aanwezig is. LD4 en LD5 geven de status van de USB interface aan (zie "USB" hiervoor). De groene lichtdiodes LD 1 en LD2 bevinden zich naast de vier toetsen en staan de gebruiker vrij ter beschikking. Ze zijn via een voorweerstand aan VCC gelegd. Via een jumper kan LD1 aangesloten worden op PortD.6 en LD2 op PortD.7. De lichtdiodes branden als de desbetreffende poortpin low (GND) is.

Toetsen
Er zijn vier toetsen gepland. Met SW3 (RESET1) wordt bij de Mega32 een reset geactiveerd en met SW5 (RESET2) een reset voor de mega8. Over de toetsen SW1 en SW2 kan de gebruiker beschikken. SW1 kan via een jumper met PortD.2 verbonden worden en op dezelfde manier SW2 met PortD.3. Er bestaat de mogelijkheid SW1/2 of tegen GND of tegen VCC te schakelen. Deze keuzemogelijkheid wordt vastgelegd met JP1 resp. JP2. Om bij een open schakelaar een gedefinieerd niveau op de ingangspoort te krijgen, moet de desbetreffende pull –up ingeschakeld zijn (zie hoofdstuk Digitale poorten).

➔ Als u op SW1 drukt bij het inschakelen van het board, wordt de seriële bootloader –modus geactiveerd.

LCD
Er kan een LCD –module met het Application board verbonden worden. Deze laat 2 rijen met elk 8 tekens zien. Ook anders georganiseerde displays kunnen in principe via deze interface gebruikt worden. Elk teken bestaat uit een monochrome matrix van 5x7 punten. Een knipperende cursor onder één van de tekens kan de actuele uitvoerpositie aanduiden. Het besturingssysteem biedt een eenvoudige software –interface voor uitvoer naar het display. Het display wordt aangesloten op de stekker X14 (16-polig, twee rijen). Door een mechanische bescherming tegen verkeerde poling is het verkeerd insteken niet mogelijk.

De gebruikte LCD module is van het type Hantronix HDM08216L-3. Verdere informatie vindt u op de Hantronix website http://www.hantronix.com en onder datasheets op de CD-ROM.

LCD –contrast (LCD-ADJ)
U heeft de beste zichtbaarheid van de tekens als u er frontaal naar kijkt. Eventueel moet u het contrast een beetje bijregelen. Het contrast kan ingesteld worden via de draaiweerstand PT1.

Toetsenbord
Voor het invoeren van data heeft de gebruiker een 12-delig toetsenbord (0 … 9, *, #) ter beschikking (X15: 13-polige stekker). Het toetenbord is 1 uit 12 georganiseerd, d.w.z. aan iedere toets is een leiding toegewezen. De toetsinformatie wordt serieel via een schuifregister ingelezen. Als er geen toetsenbord gebruikt wordt, kunnen de 12 ingangen als extra digitale ingangen gebruikt worden. Het toetsenbord beschikt over een 13-polige aansluiting (één rij) en wordt met X15 zo verbonden dat het toetsenveld naar het Application board wijst.

12C-interface

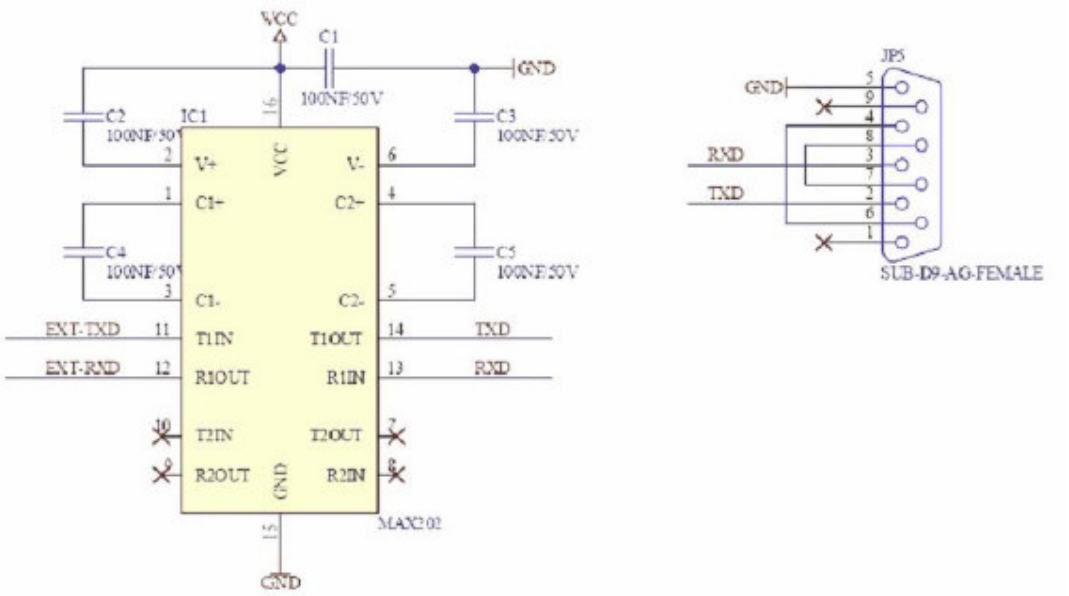
Via deze interface kunnen seriële data met hoge snelheid verzonden worden. Er zijn daarvoor slechts twee signaalleidingen nodig. De overdracht gebeurt via het 12C –protocol. Voor het effectieve gebruik van deze interface worden speciale functies ter beschikking gesteld (zie de softwarebeschrijving 12C).

12C SCL	12C-bus pulsleiding	PortC.0
12C SDA	12C-bus dataleiding	PortC.1

Voedingsspanning (POWER, 5 Volt, GND)
Het Application board wordt via een stekkernetvoeding (9V/250mA) van spanning voorzien. Afhankelijk van de extra schakelingen van het Application board kan het later nodig zijn een netvoeding met een hogere capaciteit te gebruiken. Een vaste spanningsregelaar produceert de interne gestabiliseerde voedingsspanning van 5V. Alle delen van de schakeling op het Application board worden met deze spanning gevoed. Door de capaciteitsreserve van de netvoeding staan deze 5V ook ter beschikking als voeding voor externe IC's.

➔ Let op de maximaal afneembare stroom. Een overschrijding kan leiden tot vernieling! Vanwege het relatief hoge stroomverbruik van het Application board in het bereik van 125mA is deze niet aan te bevelen voor toepassing in permanent op batterijen werkende apparaten. Let op de aanwijzing betreffende het kort uitvallen van de voedingsspanning (“zie Reset - gedrag”).

Seriële interface
De microcontroller Atmega32 bezit voor wat betreft de hardware over een asynchrone seriële interface volgens RS232 standaard. Het formaat kan vastgelegd worden bij de initialisering van de interface (databits, pariteitbit, stopbit). Op het Application board bevindt zich een hoogwaardige niveau- omvormer- IC voor het omzetten van de digitale bitstromen in Non-Return-Zero-signalen volgens de RS232 standaard (positieve spanning voor lowbits, negatieve spanning voor highbits). De niveau- omvormer- IC beschikt over een verhoogde bescherming tegen spanningspieken. Spanningspieken kunnen in een elektromagnetische omgeving, bijv. in industriële toepassingen, in de interfacekabel geïnduceerd worden en aangesloten schakelcircuits vernielen. D.m.v. jumpers kunnen de datakabels RxD en TxD met de controller PortD.0 en PortD.1 verbonden worden. In rusttoestand (geen actieve dataoverdracht) kunt u op pin TxD een negatieve spanning van een paar volt tegen GND meten. RxD is hoogohmig. Op de 9-polige SUB-D bus van het Application board ligt RxD aan pin 3 en TxD aan pin 2. De GND –aansluiting ligt op pin 5. Er worden voor de seriële dataoverdracht geen handshake- signalen gebruikt.



Een kabelverbinding met aansluiting aan de NRZ –pinnen TxD, RxD, RTS mag maximaal 10 meter lang zijn. U dient waar mogelijk afgeschermde normkabels te gebruiken. Bij langere kabels of onafgeschermde kabels kunnen storende invloeden de dataoverdracht beïnvloeden. Sluit alleen verbindingskabels aan waarvan de aansluitbezetting bekend is.

➔ Verbind nooit de seriële zenduitgangen van twee apparaten met elkaar! U herkent de zenduitgangen meestal aan de negatieve uitgangsspanning in rusttoestand.

Testinterfaces

De 4-polige stiftstrip X16 wordt alleen voor interne testdoeleinden gebruikt en zal ook niet op alle application boards gemonteerd worden. Voor de gebruiker is deze stiftstrip zonder betekenis.

Een andere testinterface is de 6-polige stiftstrip (twee rijen met elk 3 pinnen) bij JP4. Ook deze stiftstrip is alleen voor intern gebruik en wordt op latere boardseries waarschijnlijk niet meer gemonteerd.

Technische specificaties application board

Aanwijzing: gedetailleerde informatie vindt u in de PDF- bestanden van de IC –fabrikanten op de C-Control Pro software CD.

Alle spanningsaanduidingen hebben betrekking op gelijkspanning (DC).

Mechaniek	
Buitenafmetingen ca.	160 mm x 100 mm
Pinraster bedradingsveld	2,54 mm
Omgevingscondities	
Bereik van de toelaatbare omgevingstemperatuur	0°C … 70° C
Bereik van de toelaatbare omgevingsluchtvochtigheid	20% … 60%

Voedingsspanning	
Bereik van de toelaatbare voedingsspanning	8V … 24V
Stroomverbruik zonder externe lasten	ca. 125mA
Max. toelaatbare permanente stroom uit gestabiliseerde 5V-spanning	200mA

3.2.3 Pintoewijzing

PortA t/m PortD worden voor directe pin-functies (bijv. Port WriteBit) van 0 tot 31 geteld, zie “Poortbit”.

Pinbezetting voor application board Mega32

PIN	Poort	Poort	Poortbit	Naam	Schakelschema	Opmeringen
1	PB0	PortB.0	8	T0		Ingang timer/counter0
2	PB1	PortB.1	9	T1		Ingang timer/counter1
3	PB2	PortB.2	10	INT2/AIN0		(+) analoge comparator, externe interrupt2
4	PB3	PortB.3	11	OTO/AIN1		(-)analoge comparator, uitgang timer0
5	PB4	PortB.4	12		SS	USB -communicatie
6	PB5	PortB.5	13		MOSI	USB –communicatie
7	PB6	PortB.6	14		MISO	USB –communicatie
8	PB7	PortB.7	15		SCK	USB –communicatie
9				RESET		
10				VCC		
11				GND		
12				XTAL2		Oscillator: 14,7456MHz
13				XTAL1		Oscillator: 14,7456MHz
14	PD0	PortD.0	24	RXD	EXT-RXD	RS232, seriële interface
15	PD1	PortD.1	25	TXD	EXT-TXD	RS232, seriële interface
16	PD2	PortD.2	26	INT0	EXT-T1	SW1 (toets 1); externe interrupt0
17	PD3	PortD.3	27	INT1	EXT-T2	SW2 (toets 2): externe interrupt1
18	PD4	PortD.4	28	OT1B	EXT-A1	Uitgang B timer 1
19	PD5	PortD.5	29	OT1A	EXT-A2	Uitgang A timer 1
20	PD6	PortD.6	30	ICP	LED1	Lichtdiode; input capture pin; puls -/periodemeting
21	PD7	PortD.7	31		LED2	Lichtdiode
22	PC0	PortC.0	16	SCL	EXT-SCL	12C.interface
23	PC1	PortC.1	17	SDA	EXT-SDA	12C-interface
24	PC2	PortC.2	18			
25	PC3	PortC.3	19			
26	PC4	PortC.4	20			
27	PC5	PortC.5	21			
28	PC6	PortC.6	22			
29	PC7	PortC.7	23			
30				AVCC		
31				GND		
32				AREF		
33	PA7	PortA.7	7	ADC7	RX_BUSY	ADC7 ingang:

						USB-communicatie
34	PA6	PortA.6	5	ADC6	TX_REQ	ADC6 ingang: USB-communicatie
35	PA5	PortA.5	5	ADC5	KEY_EN	ADC5 ingang; LCD/toetsenbord interface
36	PA4	PortA.4	4	ADC4	LCD_EN	ADC4 ingang; LCD/toetsenbord interface
37	PA3	PortA.3	3	ADC3	EXT_SCK	ADC3 ingang; LCD/toetsenbord interface
38	PA2	PortA.2	5	ADC2	EXT_DATA	ADC2 ingang; LCD/toetsenbord interface
39	PA1	PortA.1	1	ADC1		ADC1 ingang
40	PA0	PortA.0	0	ADC0		ADC0 ingang

3.2.4 Jumper Application board

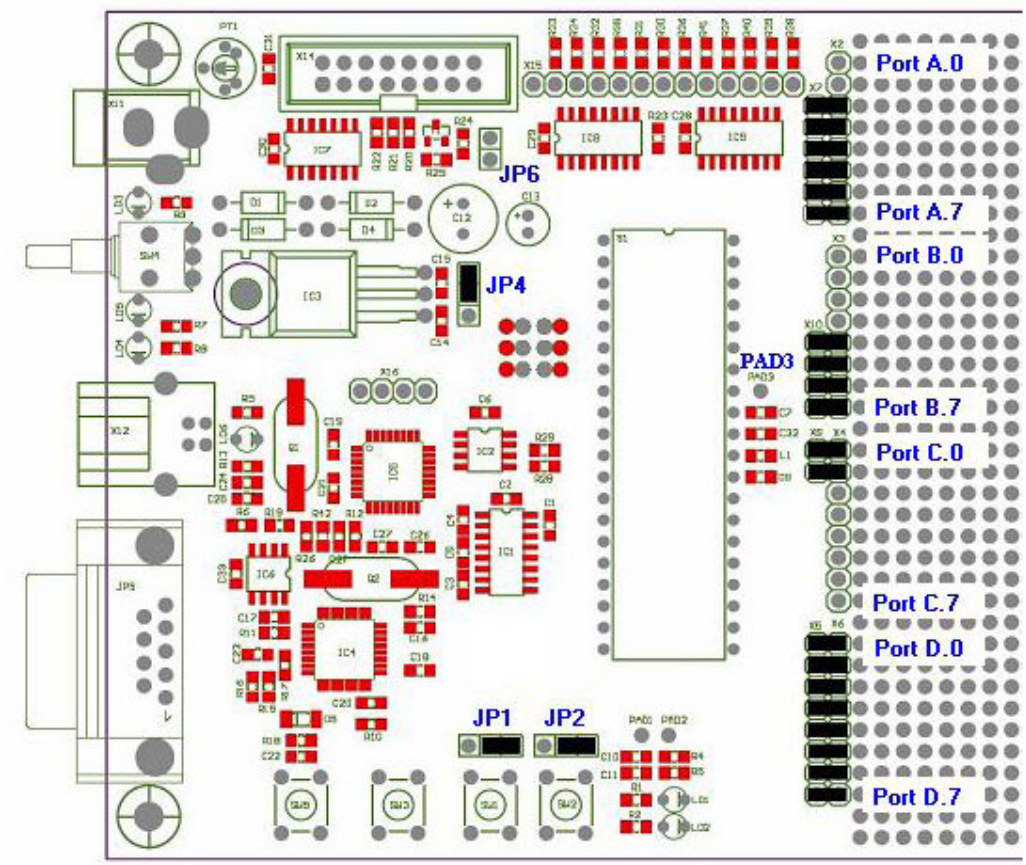
Jumper

Met behulp van jumpers kunnen bepaalde opties geselecteerd worden. Dit geldt voor bepaalde poorten welke speciale functies bezitten (zie tabel pintoewijzing M32). Bijvoorbeeld is de seriële interface via de pinnen PortD.0 en PortD.1 gerealiseerd. Als de seriële interface niet gebruikt wordt, kunnen de desbetreffende jumpers verwijdert worden en deze pinnen zijn dan voor andere functies beschikbaar. Naast de jumpers voor deze poorten zijn er nog extra jumpers, deze worden hierna beschreven.

Poorten A t/m D

De bij de Mega32- module beschikbare poorten zijn in deze grafiek ingetekend. Daarbij is de rechter kant met de module verbonden en de linker kant verbindt naar componenten van het Application board. Als een jumper getrokken wordt, onderbreekt dit de verbinding naar het Application board

JP1 en JP2
Deze jumpers zijn toegewezen aan de toetsen SW1 en SW2. Er bestaat de mogelijkheid de toetsen te laten werken tegen GND of VCC. In de standaardinstelling schakelen de toetsen tegen GND.



Jumperposities bij uitlevering

JP4

JP4 is bedoeld voor het omschakelen van de voedingsspanning (netvoeding of USB). Het Application board moet gevoed worden via netvoeding en spanningsregelaar (toestand bij uitlevering). De maximale stroomlevering van het USB interface is kleiner dan door een netzvoeding. Een overschrijding kan leiden tot beschadiging van de USB interface van de computer.

JP6

Bij gebruik van het display kan met JP6 de verlichting (back light) uitgeschakeld worden.

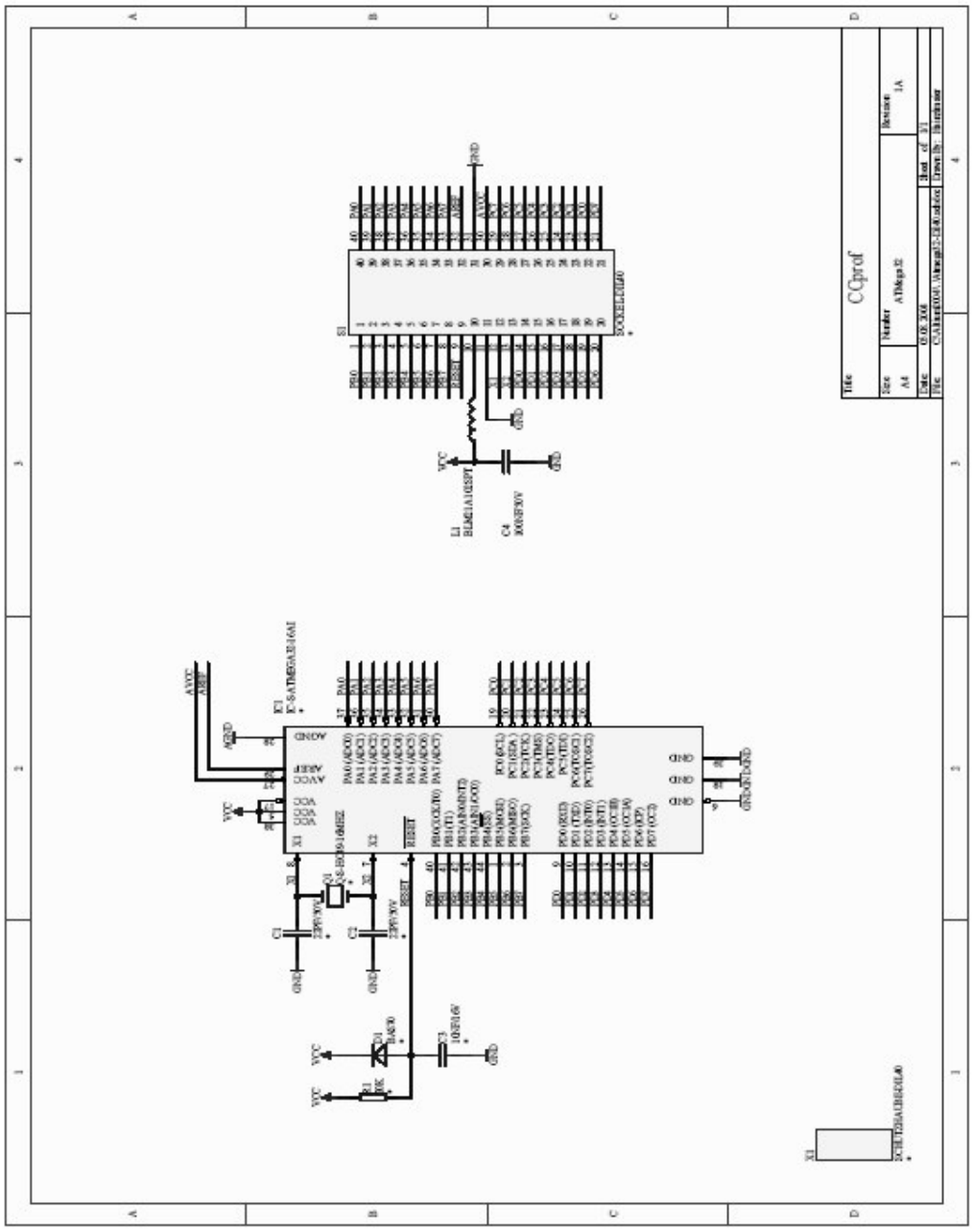
PAD3

PAD3 (rechts naast de module, onder de blauwe opschrift) wordt als ADC_VREF_EXT voor de functies ADC Set en ADC SetInt benodigd.

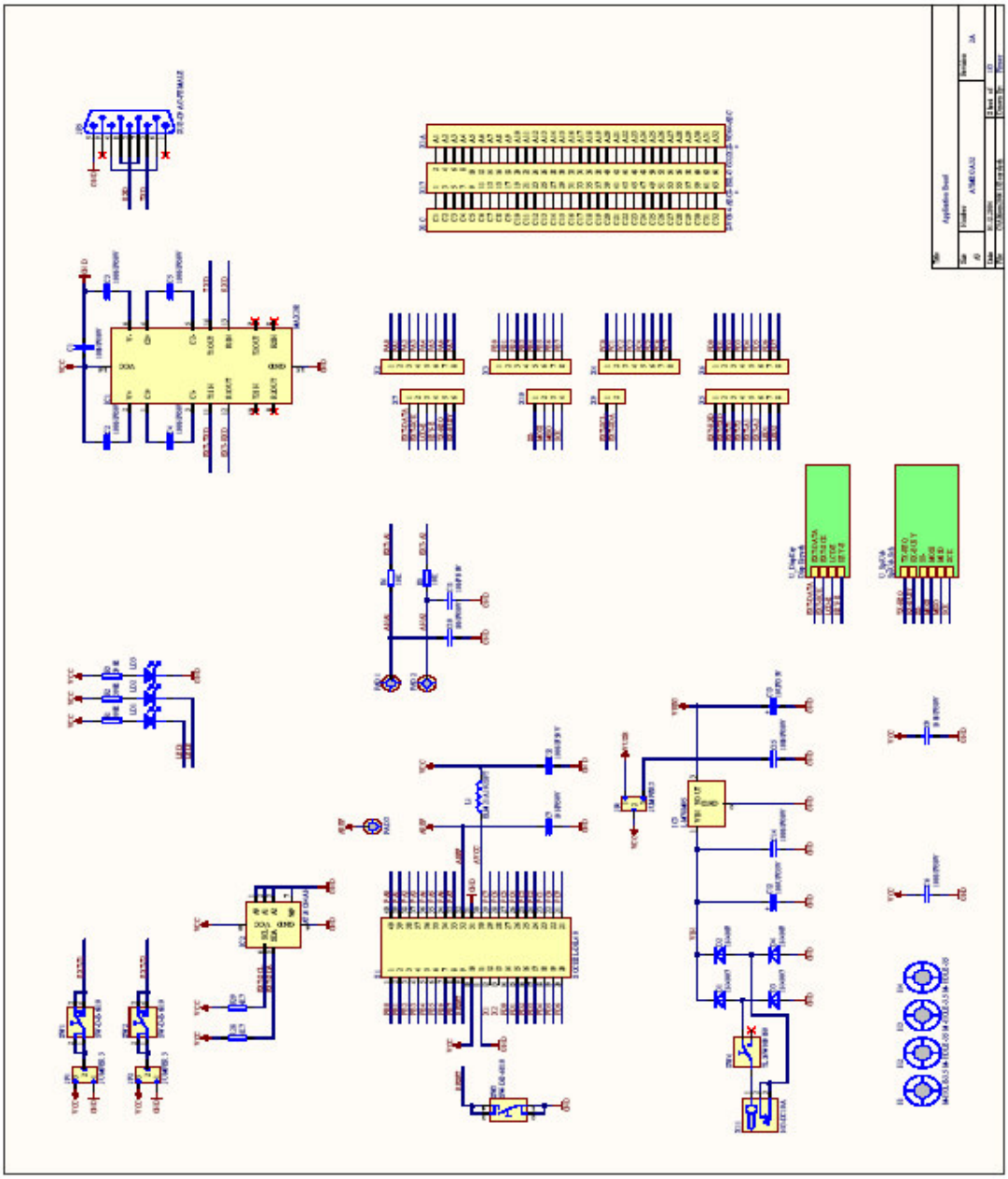
3.2.5 Schakelschema’s

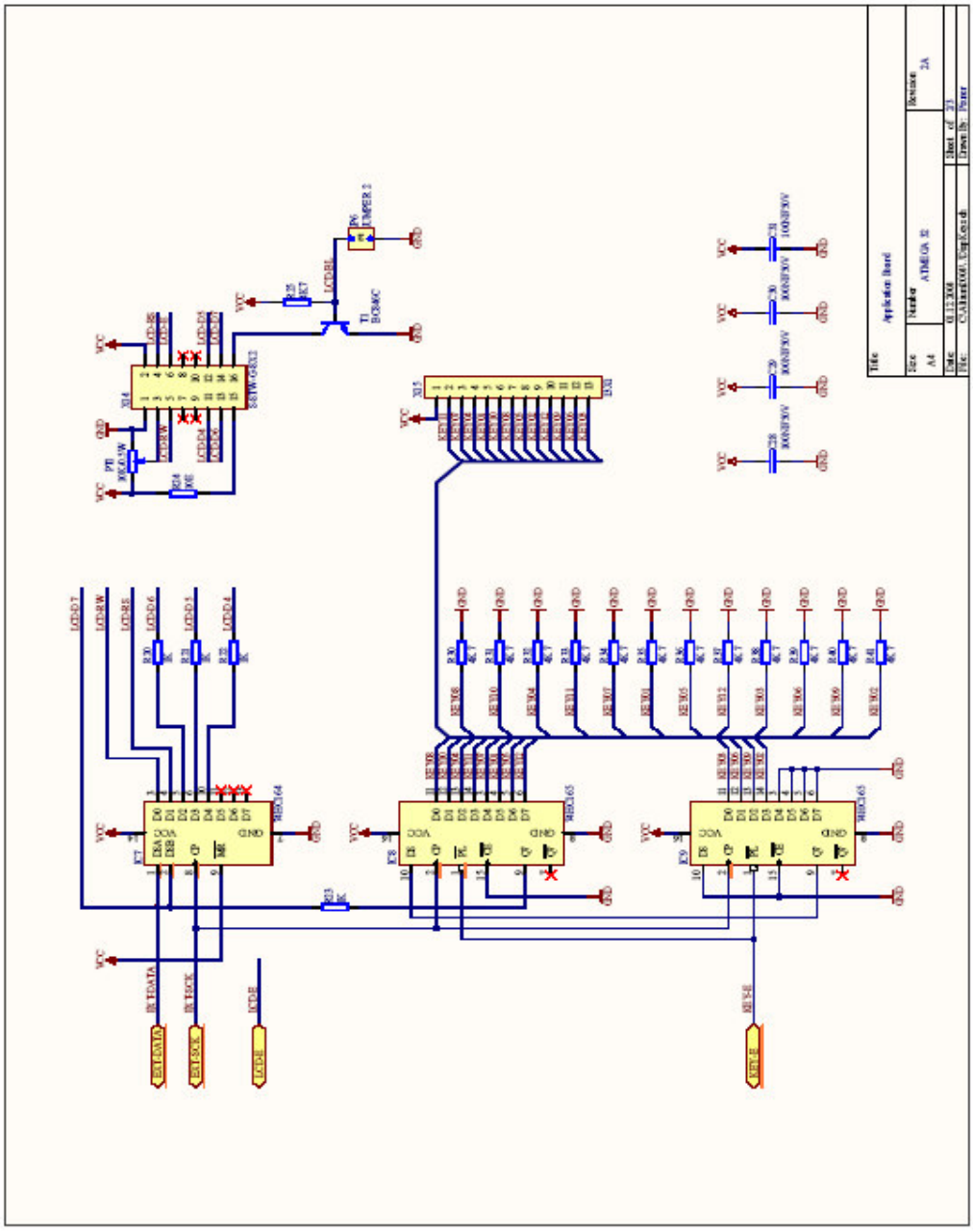
De schakelschema's zijn ook als pdf-bestand op de installatie-CD.

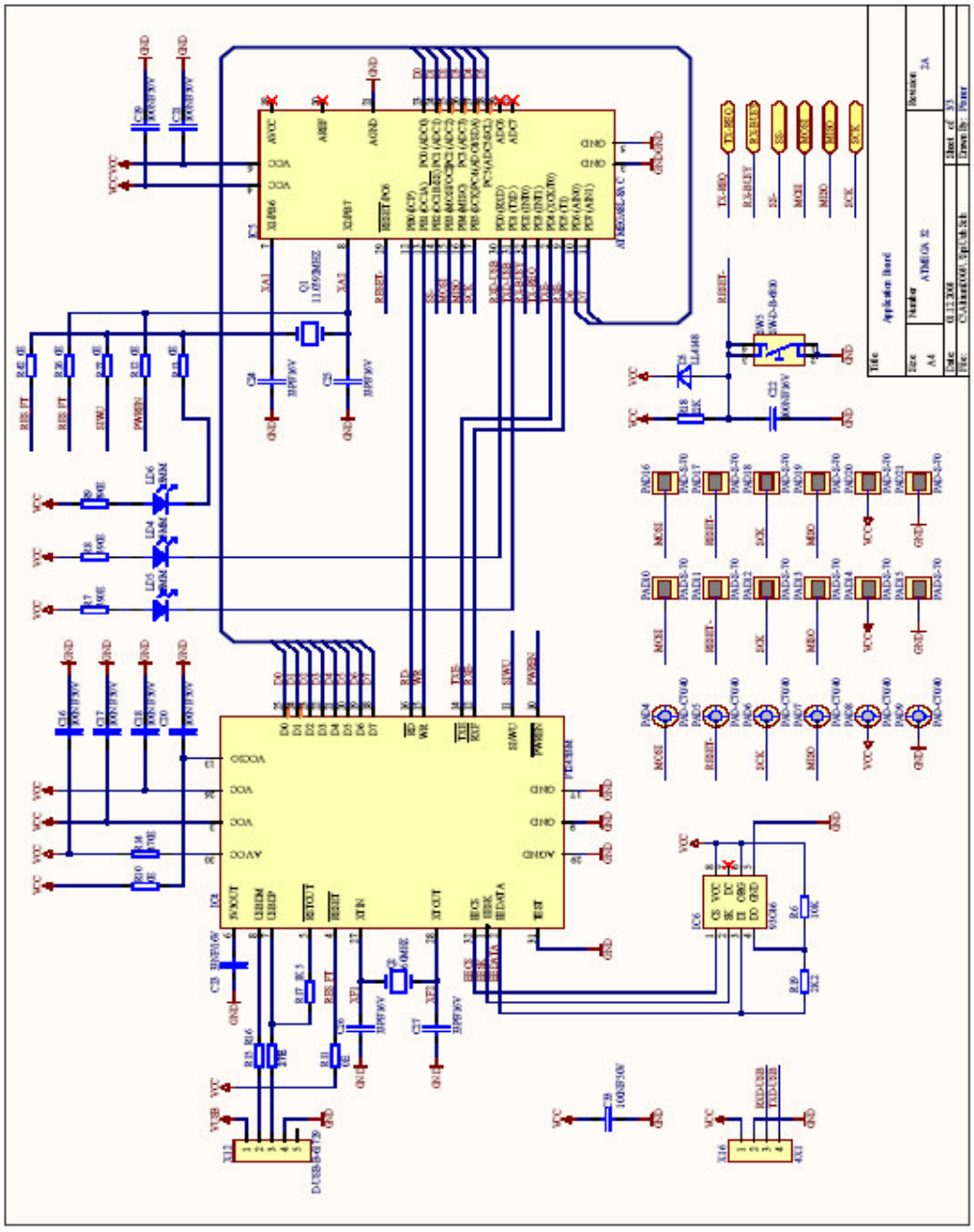
3.2.5.1 Mega 32 module



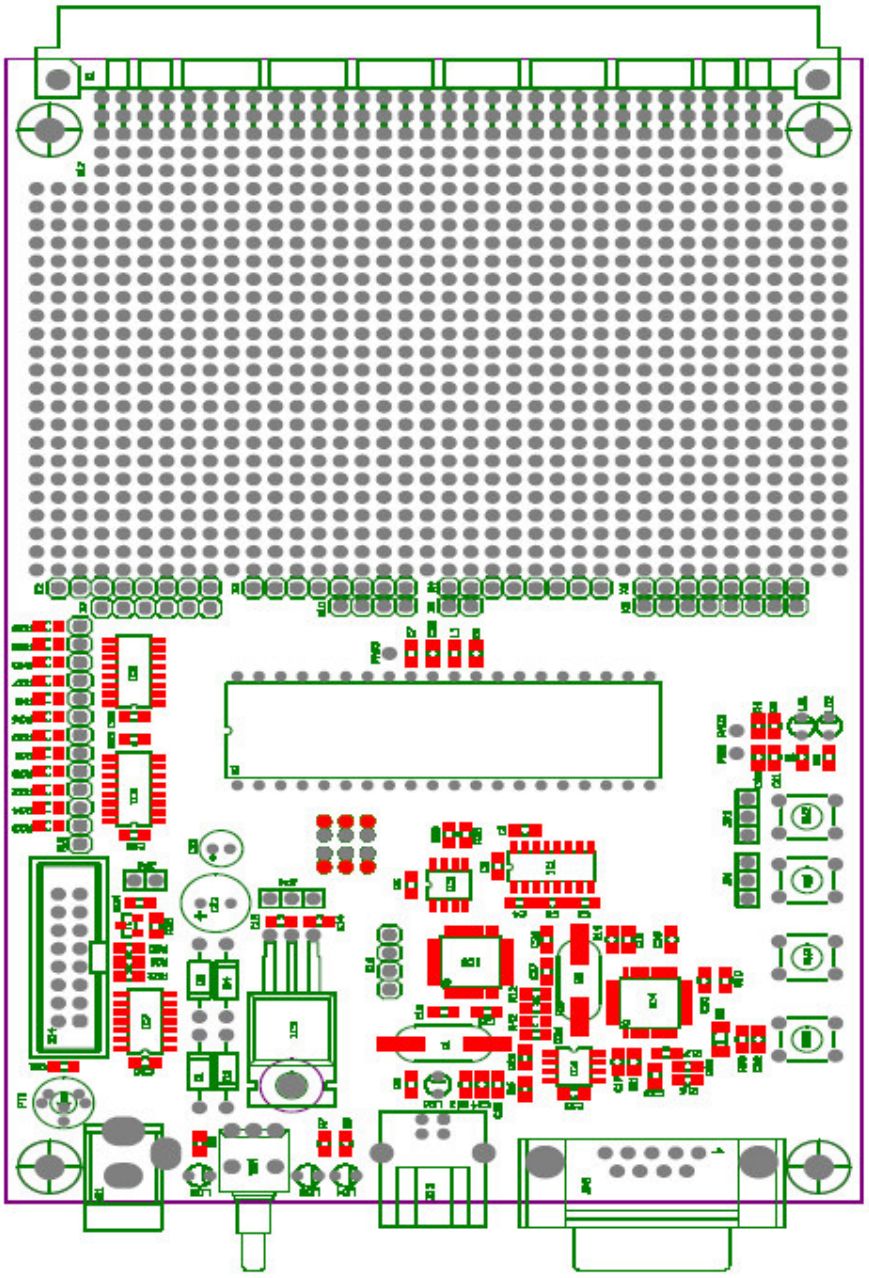
3.2.5.2 Application board







3.2.5.3 Onderdelenschema



3.3 Mega 128

Mega128 overzicht

De microcontroller Atmega 128 komt uit de AVR –familie van ATMEL. Hij is een low-power microcontroller met Advanced RISC Architecture. Hier volgt een korte samenstelling van de hardware resources:

- **133 Powerful Instructions – Most Single Clock Cycle Execution**
- **32 x 8 General Purpose Working Registers + Peripheral Control Registers**
- **Fully Static Operation**
- **Up to 16 MIPS Throughput at 16 MHz**
- **On-chip 2-cycle Multiplier**

- **Nonvolatile Program and Data Memories**
  128K Bytes of In-System Reprogrammable Flash
  Endurance: 10,000 Write/Erase Cycles
  Optional Boot Code Section with Independent Lock Bits
  In-System Programming by On-chip Boot Program

- **True Read-While-Write Operation**
  4K Bytes EEPROM
  Endurance: 100,000 Write/Erase Cycles
  4K Bytes Internal SRAM
  Up to 64K Bytes Optional External Memory Space
  Programming Lock for Software Security
  SPI Interface for In-System Programming

- **JTAG (IEEE std. 1149.1 Compliant) Interface**
  Boundary-scan Capabilities According to the JTAG Standard
  Extensive On-chip Debug Support
  Programming of Flash, EEPROM, Fuses and Lock Bits through the JTAG Interface

- **Peripheral Features**
  Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  Two Expanded 16-bit Timer/Counters with Separate Prescaler, Compare Mode and Capture Mode
  Real Time Counter with Separate Oscillator
  Two 8-bit PWM Channels
  6 PWM Channels with Programmable Resolution from 2 to 16 Bits
  Output Compare Modulator
  8-channel, 10-bit ADC
  8 Single-ended Channels
  7 Differential Channels
  2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  Byte-oriented Two-wire Serial Interface
  Dual Programmable Serial USARTs
  Master/Slave SPI Serial Interface
  Programmable Watchdog Timer with On-chip Oscillator
  On-chip Analog Comparator

- **Special Microcontroller Features**
  Power-on Reset and Programmable Brown-out Detection
  Internal Calibrated RC Oscillator
  External and Internal Interrupt Sources
  Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby

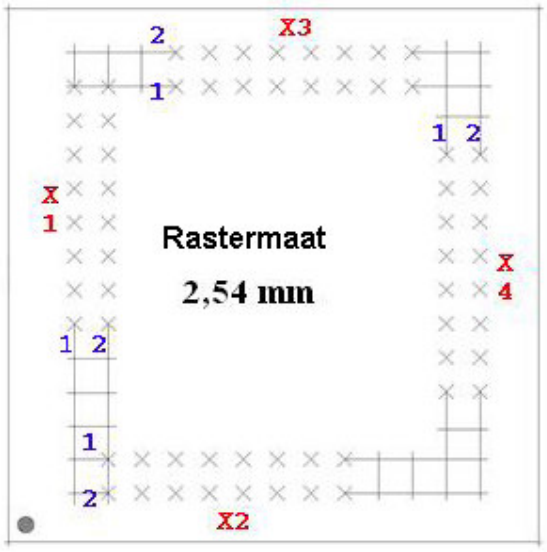
Software Selectable Clock Frequency
ATmega103 Compatibility Mode Selected by a Fuse
Global Pull-up Disable

- **I/O and Packages**
  53 Programmable I/O Lines
  64-lead TQFP and 64-pad MLF

- **Operating Voltages**
  2.7 - 5.5V for ATmega128L
  4.5 - 5.5V for ATmega128

3.3.1 Module

Pinlayout van de module
De Mega128 module wordt geleverd op 4 dubbele rijen (2x8) vierkante pinnen. Voor een hardware applicatie moeten overeenkomstige busstrippen in het onderstaande rasterformaat ingericht worden:



In de afbeelding worden de busstrippen X1-X4 aangegeven en de eerste twee pinnen van de busstrippen. Pin 1 van strip X1 komt overeen met de aansluiting X1_1 (zie Mega128 Pintoewijzing).

Modulegeheugen
In de C-Control Pro 128 module zijn 128kB FLASH, 4kB EEPROM en 4kB SRAM geïntegreerd. Op het Application board bevindt zich een extra EEPROM met een geheugen van 8kB en een SRAM met 64kB geheugen. Het EEPROM kan aangesproken worden via een 12C interface.

Aanwijzing: U vindt gedetailleerde informatie in de PDF-bestanden van de IC –fabrikanten op de C-Control Pro software CD.

ADC Referentiespanning –opwekking
De microcontroller beschikt over een analoog –digitaal –omvormer met een resolutie van 10Bit. Dit betekent dat gemeten spanningen als gehele getallen van 0 tot 1023 weergegeven kunnen worden. De referentiespanning voor de ondergrens is het GND- niveau, dus 0V. De referentiespanning voor de bovengrens kan door de gebruiker gekozen worden:

* 5V voedingsspanning (VCC)
* interne referentiespanning van 2,56V
* externe referentiespanning bijv. 4,096V gegenereerd door referentiespannings –IC

Als x een digitale meetwaarde is, wordt de desbetreffende spanningswaarde als volgt berekend:

u = x * referentiespanning / 1024

Genereren van klokfrequentie
Het genereren van de klokfrequentie gebeurt door een 14,7456MHz kwartsoscillator. Alle tijdprocedures van de controller zijn van deze klokfrequentie afgeleid.

Reset
Een reset zorgt voor het terugkeren van het microcontroller –systeem naar een gedefinieerde begintoestand. De C-Control Pro module kent in principe twee reset –bronnen:

- Power-On -Reset: wordt automatisch uitgevoerd na het inschakelen van de voedingsspanning
- Hardware -Reset : wordt uitgevoerd als de RESET (X2_3) van de module op “low” getrokken en weer losgelaten wordt, bijv. door het kort indrukken van de aangesloten reset –toets RESET1 (SW3).

Door een “Brown-Out-Detection” wordt voorkomen dat de controller bij het te laag worden van de voedingsspanning in een ongedefinieerde toestand terecht kan komen.

Digitale poorten (PortA, PortB, PortC, PortD, PortE, PortF, PortG)

De C-Control Pro module beschikt over zes digitale poorten met elk 8 pinnen en een digitale poort met 5 pinnen. Op de digitale poorten kunnen bijv. toetsen met pull -up weerstanden, digitale IC's, opto-koppelingen of driverschakelingen voor relais aangesloten worden. De poorten kunnen apart, d.w.z. pinwijze of bytewijze aangesproken worden. Elke pin kan of uitgang of ingang zijn.

➔ Schakel nooit direct twee poorten samen, die gelijktijdig als uitgang moeten werken!

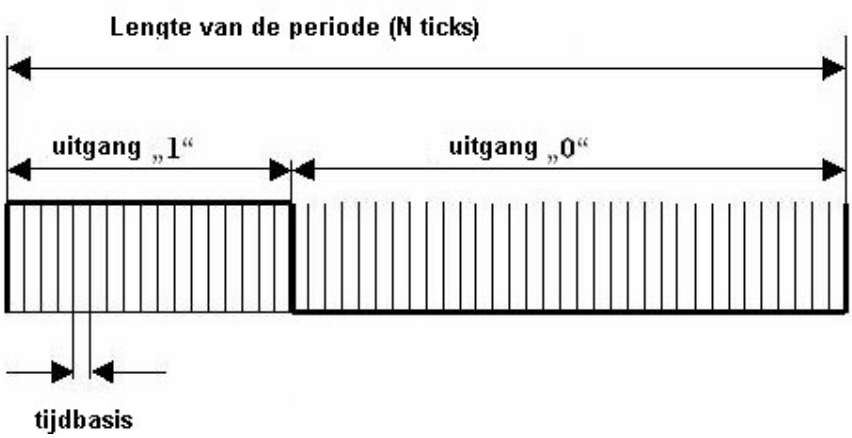
Digitale ingangspinnen zijn hoogohmig of met een interne pull –up weerstand geschakeld en zetten een aanliggend spanningssignaal om in een logische waarde. Voorwaarde daarvoor is, dat het spanningsignaal zich binnen de voor TTL – resp. CMOS –IC's gedefinieerde bereiken voor Low - of High niveau bevindt. In de verdere bewerking in het programma worden de logische waarden van aparte ingangspoorten als 0 (“low”) of –1 (“high”) weergegeven. Pinnen nemen waarden van 0 of 1 aan, bytepoorten 0 tot 255. Uitgangspoorten kunnen via een interne driverschakeling digitale spanningssignalen afgeven. Aangesloten schakelingen kunnen een bepaalde stroom uit de poorten trekken (bij High niveau) resp. deze poorten er mee voeden (bij Low niveau).

➔ Let op de maximaal toelaatbare laststroom voor een aparte poort en voor alle poorten bij elkaar. Een overschrijding van de maximale waarden kan leiden tot vernieling van de C-

Control Pro module. Na de reset is in eerste instantie elke digitale poort als ingangspoort geconfigureerd. Via bepaalde commando's kan de datarichting omgeschakeld worden.

➔ Het is belangrijk om vóór de programmering de pintoewijzing M32 en M128 te bestude-ren, aangezien belangrijke functies van de programma –ontwikkeling (bijv. de USB –interface van het Application board) op bepaalde poorten liggen. Als deze poorten omgeprogram-meerd worden of als de bijbehorende jumpers op het Application board niet meer gezet zijn, kan het gebeuren dat de ontwikkelingsomgeving geen programma's meer kan overbrengen naar de C-Control Pro. Ook in- en uitgangen van de timer, A/D omvormer, I2C en de seriële interface zijn met bepaalde pinnen verbonden.

PLM -poorten



Er staan twee timers ter beschikking voor de PLM, *Timer_0* met 8 bit en *Timer_1* en *Timer_3* met elk 16 bit. Deze kunnen gebruikt worden voor de D/A –omvorming, voor het aansturen van servomotoren in de modelbouw of voor het afgeven van geluidsfrequenties. Een pulslengte –gemoduleerd signaal heeft een periode van N zogenaamde "Ticks". De duur van een tick is de tijdbasis. Als men de uitvoerwaarde van een PLM –poort op X stelt, dan houdt deze geduren-de X ticks van een periode high niveau en valt voor de rest van de periode op low. Voor de programmering van de PLM –kanalen zie Timer.

De PLM –kanalen voor *Timer_0* en *Timer_1* en *Timer_3* hebben een onafhankelijke tijdbasis en periodelengte. In toepassingen voor pulsbreedte –gemoduleerde digitaal – analoog - omvorming worden tijdbasis en periodelengte eenmalig ingesteld en daarna wordt alleen de afgiftewaarde veranderd. De PLM –poorten zijn vanwege hun elektrische eigenschappen digitale poorten. Let op de technische randvoorwaarden voor digitale poorten (max. stroom).

Technische specificaties module

Aanwijzing: u vindt gedetailleerde informatie in de PDF –bestanden van de IC –fabrikanten op de C-Control Pro software CD.

Alle spanningen hebben betrekking op gelijkspanning (DC).

Omgevingscondities	
Bereik van de toelaatbare omgevingstemperatuur	0 °C … 70 °C
Bereik van de toelaatbare relatieve luchtvochtigheid van de omgeving	20% …60%

Voedingsspanning	
Bereik van de toelaatbare voedingsspanning	4.5V … 5,5V
Stroomverbruik van de module zonder externe lasten	ca. 20mA
Puls	
Pulsfrequentie (kwarts –oscillator)	14,7456MHz
Mechanische deel	
Buitenafmetingen zonder pinnen ca.	40 mm x 40 mm x 8 mm
Gewicht	ca. 90g
Pinraster	2,54mm
Aantal pinnen (2 rijen)	64
Poorten	
Max. toelaatbare stroom uit digitale poorten	± 20mA
Toelaatbare totaal- stroom aan digitale poorten	200mA
Toelaatbare ingangsspanning op poortpinnen (digitaal en A/D)	-0,5V … 5,5V
Interne pull –up weerstanden (uitschakelbaar)	20 – 50 kOhm

3.3.2 Application Board

USB
Het Application board beschikt over een USB interface voor het laden en debuggen van het programma. Door de hoge datasnelheid van deze interface zijn de dataoverdrachttijden aanzienlijk korter vergeleken met de seriële interface. De communicatie vindt plaats via een USB –controller van FDTI en een AVR Mega8 controller. De Mega8 heeft een eigen reset –toets (SW5). Tijdens het USB gebruik wordt de status van de interface getoond via twee lichtdiodes (LD4 rood, LD5 groen). Als alleen de groene LED oplicht, dan is de USB interface bedrijfsklaar. Als er een dataoverdracht plaatsvindt, branden beide LED's. Dit geldt ook voor de debug –modus.Het knipperen van de rode LED geeft een foutconditie aan. Voor de USB-communicatie wordt de SPI –interface van de Mega128 gebruikt (PortB. 0 t/m PortB.4, PortE.5) en deze moeten via de desbetreffende jumpers verbonden zijn.

Aanwijzing: U vindt meer gedetailleerde informatie over de Mega8 in de PDF –bestanden van de IC –fabrikant op de C-Control pro software CD.

Aan - /Uitschakelaar
De schakelaar SW4 bevindt zich aan de voorkant van het Application board en dient voor het in -/uitschakelen van de spanningvoorziening.

Lichtdiodes

Er zijn 5 lichtdiodes beschikbaar. LD3 (groen) bevindt zich aan de voorkant onder de DC –aansluiting en brandt, als er voedingsspanning aanwezig is. LD4 en LD5 geven de status van de USB interface aan (zie par. USB). De groene lichtdiodes LD1 en LD2 bevinden zich naast de vier toetsen en staan de gebruiker vrij ter beschikking. Ze zijn via een voorweerstand aan VCC gelegd. Via jumper kan LD1 aangesloten worden op PortG.3 en LD2 op PortG.4. De lichtdiodes branden als de desbetreffende poort pin low (GND) is.

Toetsen

Er zijn vier toetsen gepland. Met SW3 (RESET1) wordt bij de Mega128 een reset geactiveerd en met SW5 (RESET2) een reset voor de Mega8. De toetsen SW1 en SW2 staan ter beschikking van de gebruiker. SW1 kan via een jumper met PortE.4 verbonden worden en op dezelfde manier SW2 met PortE.6. Er bestaat de mogelijkheid SWS1/2 of tegen GND of tegen VCC te schakelen. Deze keuzemogelijkheid wordt vastgelegd meet JP1 resp. met JP2. Om bij een open schakelaar een gedefinieerd niveau op de ingangspoort te hebben, moet de desbetreffende pull –up ingeschakeld zijn (zie par. Digitale poorten).

➔ Als u op SW1 drukt bij het inschakelen van het board, wordt de seriële bootloader –modus geactiveerd.

LCD

Er kan een LCD –module met het Application board verbonden worden. Deze geeft 2 regels met elk 8 tekens weer. Ook anders georganiseerde displays kunnen in principe via deze interface gebruikt worden. Elk teken bestaat uit een monochrome matrix van 5x7 punten. Een knipperende cursor onder één van de tekens kan de actuele uitvoerpositie aanduiden. Het besturingssysteem biedt een eenvoudige software –interface voor uitvoer naar het display. Het display wordt aangesloten op de stekker X14 (16-polig, twee rijen). Door een mechanische bescherming tegen verkeerde polariteit is het verkeerd insteken niet mogelijk.

De gebruikte LCD module is van het type Hantronix HDM08216L-3. Verdere informatie vindt u op de Hantronix website http://www.hantronix.com en in het datasheets register op de CD-ROM.

LCD –contrast (LCD-ADJ)

De beste zichtbaarheid van de LCD- tekens wordt verkregen door frontaal naar te kijken. Eventueel moet het contrast iets bijgeregeld worden. Het contrast kan ingesteld worden via de draaiweerstand PT1.

Toetsenbord

Voor de invoeren is een 12-delig toetsenbord (0 … 9, *, #) beschikbaar (X15: 13-polige stekker). Het toetsenbord is 1 uit 12 georganiseerd, d.w.z. aan iedere toets is een leiding toegewezen. De toetsinformatie wordt serieel via een schuifregister ingelezen. Als er geen toetsenbord gebruikt wordt, kunnen de 12 ingangen als extra digitale ingangen gebruikt worden. Het toetsenbord beschikt over een 13-polige aansluiting (één rij) en wordt zo in de X15 gestoken, dat het toetsenveld naar het Application board wijst.

SRAM

Op het Application board bevindt zich een SRAM-chip (K6X1008C2D) van Samsung. Hierdoor wordt het beschikbare SRAM-geheugen uitgebreid op 64KB. De SRAM gebruikt voor de aansturing de poorten A, C en deels poort G. Als het SRAM niet gebruikt wordt kan deze met JP7 deactiveert worden en de poorten zijn weer beschikbaar.

➔ Oftewel de gebruikte RAM chip een capaciteit van 128 KB heeft, kan opgrond van het geheugenmodel slechts 64kb daarvan benut worden.

12C-interface

Via deze interface kunnen seriële data met hoge snelheid verzonden worden. Er zijn daarvoor slechts twee signaalleidingen nodig. De overdracht gebeurt via het I2C –protocol. Voor het effectieve gebruik van deze interface worden speciale functies ter beschikking gesteld (zie de softwarebeschrijving I2C).

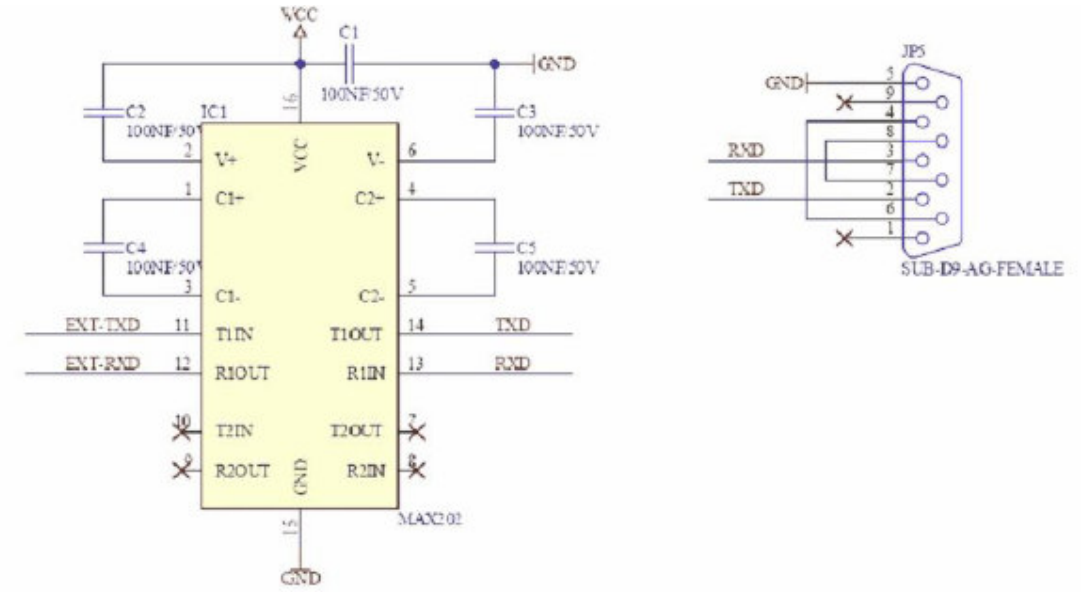
I2C SCL	12C-bus pulsleiding	PortD.0
I2C SDA	12C-bus dataleiding	PortD.1

Voedingsspanning (POWER, 5 Volt, GND)

Het Application board wordt via een stekkernetvoeding (9V/250mA) van spanning voorzien. Afhankelijk van de verdere schakelingen van het Application board kan het later nodig zijn een netvoeding met een hogere capaciteit te gebruiken. Een vaste spanningsregelaar produceert de interne gestabiliseerde voedingsspanning van 5V. Alle delen van de schakeling op het Application board worden met deze spanning gevoed. Door de capaciteitsreserve van de netvoeding staat deze 5V ook ter beschikking als voeding van externe ICs.

➔ Let op de maximaal afneembare stroom. Een overschrijding kan leiden tot vernieling! Vanwege het relatief hoge stroomverbruik van het Application board in het bereik van 125 mA is deze niet aan te bevelen voor toepassing in permanent op batterijen werkende apparaten. Let op de aanwijzing betreffende het kort uitvallen van de voedingsspanning (“zie Reset gedrag”).

Seriële interface



De microcontroller Atmega128 bezit voor wat betreft de hardware twee asynchrone seriële interfaces volgens RS232 standaard. Het formaat kan vastgelegd worden bij de initialisering van de interface (databits, pariteitbit, stopbit). Op het application board bevindt zich een hoogwaardige niveau-omvormer-IC voor het omzetten van de digitale bitstromen in Non-Return-Zero-signalen volgens de RS232 standaard (positieve spanning voor lowbits, negatieve spanning voor highbits). De niveau-omvormer-IC beschikt over een verhoogde bescherming tegen spanningspieken. Spanningspieken kunnen in een elektromagnetische omgeving, bijv. in industriële toepassingen, in de interfacekabel geïnduceerd worden en aangesloten schakelcircuits vernielen. D.m.v. jumpers kunnen de datakabels RxD0 (PortE.0), TxD (PortE.1) en RxD1 (PortD.2), TxD (PortD.3) van de controller met de niveau-omvormer verbonden worden. In rusttoestand (geen actieve dataoverdracht) kunt u op pin TxD een negatieve spanning van een paar volt tegen GND meten. RxD is hoogohmig. Op de 9-polige SUB-D bus van het application board ligt RxD0 aan pin 3 en TxD0 aan pin 2. De GND –aansluiting ligt op pin 5. Er worden voor de seriële dataoverdracht geen handshake –signalen gebruikt. De tweede seriële interface is op een 3-polige stiftstrip geleid. RxD1 ligt aan pin 2 en TxD1 aan pin 1, pin3=GND.

Een kabelverbinding met aansluiting aan de NRZ –pinnen TxD, RxD, RTS mag maximaal 10 meter lang zijn. U dient waar mogelijk afgeschermde normkabels te gebruiken. Bij langere kabels of onafgeschermde kabels kunnen storende invloeden de dataoverdracht beïnvloeden. Verbind alleen verbindingskabels waarvan de pinbezetting bekend ist.

➔ Verbind nooit de seriële zenduitgangen van twee apparaten met elkaar! U herkent de zenduitgangen meestal aan de negatieve uitgangsspanning in rusttoestand.

Testinterfaces

De 4-polige stiftstrip X16 wordt alleen voor interne testdoeleinden gebruikt en zal ook niet op alle application boards gemonteerd worden. Voor de gebruiker is deze stiftstrip zonder betekenis.

Een andere testinterface is de 6-polige stiftstrip (twee rijen met elk 3 pinnen) rechts onder bij JP4. Ook deze stiftstrip is alleen voor intern gebruik en wordt op latere boardseries waarschijnlijk niet meer gemonteerd.

Technische specificaties application board

Aanwijzing: U vindt gedetailleerde informatie in de PDF –bestanden van de IC –fabrikant op de C-Control Pro software CD.
Alle spanningsaanduidingen hebben betrekking op gelijkspanning (DC).

Mechaniek	
Buitenafmetingen ca.	160 mm x 100 mm
Pinraster bedradingveld	2,54 mm
Omgevingscondities	
Bereik van de toelaatbare omgevingstemperatuur	0℃ … 70° C
Bereik van de toelaatbare omgevingsluchtvochtigheid	20% … 60%
Voedingsspanning	
Bereik van de toelaatbare voedingsspanning	8V … 24V
Stroomverbruik zonder externe lasten	ca. 125mA
Max. toelaatbare permanente stroom uit gestabiliseerde 5V-spanning	200mA

3.3.3 Pintoewijzing

PortA tot PortD worden voor directe pin-functies (b.v. Port WriteBit) van 0 tot 52 geteld, zie “Portbit”.

Pinbezetting voor application board Mega128

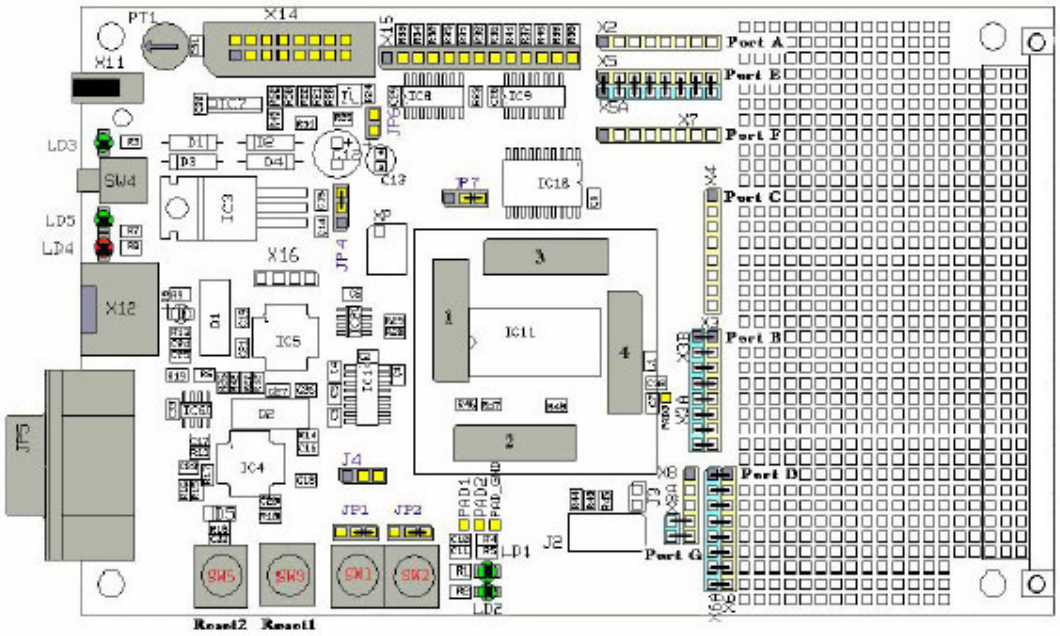
Modul	M128	Port	Port #	PortBit	Name1	Name2	Intern	Bemerkungen
	1				PEN			prog. Enable
X1_16	2	PE0	4	32	RXD0	PDI	EXT-RXD0	RS232
X1_15	3	PE1	4	33	TXD0	PDO	EXT-TXD0	RS232
X1_14	4	PE2	4	34	AIN0	XCK0		Analog Comparator
X1_13	5	PE3	4	35	AIN1	OC3A		Analog Comparator
X1_12	6	PE4	4	36	INT4	OC3B	EXT-T1	Taste1 (SW1)
X1_11	7	PE5	4	37	INT5	OC3C	TX-REQ	SPI_TX_REQ
X1_10	8	PE6	4	38	INT6	T3	EXT-T2	Taste2 (SW2)
X1_9	9	PE7	4	39	INT7	IC3	EXT-DATA	LCD_Interface
X1_8	10	PB0	1	8	SS			SPI
X1_7	11	PB1	1	9	SCK			SPI
X1_6	12	PB2	1	10	MOSI			SPI
X1_5	13	PB3	1	11	MISO			SPI

X1_4	14	PB4	1	12	OC0		RX-BUSY	SPI_RX_BUSY
X1_3	15	PB5	1	13	OC1A		EXT-A1	DAC1
X1_2	16	PB6	1	14	OC1B		EXT-A2	DAC2
X1_1	17	PB7	1	15	OC1C	OC2	EXT-SCK	LCD_Interface
X2_5	18	PG3	6	51	TOSC2		LED1	Leuchtdiode
X2_6	19	PG4	6	52	TOSC1		LED2	Leuchtdiode
X2_3	20				RESET			
X4_10	21				VCC			
X4_12	22				GND			
	23				XTAL2			Oscillator
	24				XTAL1			Oscillator
X2_9	25	PD0	3	24	INT0	SCL	EXT-SCL	I2C
X2_10	26	PD1	3	25	INT1	SDA	EXT-SDA	I2C
X2_11	27	PD2	3	26	INT2	RXD1	EXT-RXD1	RS232
X2_12	28	PD3	3	27	INT3	TXD1	EXT-TXD1	RS232
X2_13	29	PD4	3	28	IC1	A16		IC Timer 1, SRAM bank select
X2_14	30	PD5	3	29	XCK1		LCD-E	LCD_Interface
X2_15	31	PD6	3	30	T1			
X2_16	32	PD7	3	31	T2		KEY-E	LCD_Interface
X2_7	33	PG0	6	48	WR			WR SRAM
X2_8	34	PG1	6	49	RD			RD SRAM
X4_8	35	PC0	2	16	A8			ADR SRAM
X4_7	36	PC1	2	17	A9			ADR SRAM
X4_6	37	PC2	2	18	A10			ADR SRAM
X4_5	38	PC3	2	19	A11			ADR SRAM
X4_4	39	PC4	2	20	A12			ADR SRAM
X4_3	40	PC5	2	21	A13			ADR SRAM
X4_2	41	PC6	2	22	A14			ADR SRAM
X4_1	42	PC7	2	23	A15			ADR SRAM
X2_4	43	PG2	6	50	ALE			Latch
X3_16	44	PA7	0	7	AD7			A/D SRAM
X3_15	45	PA6	0	6	AD6			A/D SRAM
X3_14	46	PA5	0	5	AD5			A/D SRAM
X3_13	47	PA4	0	4	AD4			A/D SRAM
X3_12	48	PA3	0	3	AD3			A/D SRAM
X3_11	49	PA2	0	2	AD2			A/D SRAM
X3_10	50	PA1	0	1	AD1			A/D SRAM
X3_9	51	PA0	0	0	AD0			A/D SRAM
X4_10	52				VCC			
X4_12	53				GND			
X3_8	54	PF7	5	47	ADC7	TDI-JTAG		
X3_7	55	PF6	5	46	ADC6	TDO-JTAG		
X3_6	56	PF5	5	45	ADC5	TMS-JTAG		
X3_5	57	PF4	5	44	ADC4	TCK-JTAG		
X3_4	58	PF3	5	43	ADC3			

X3_3	59	PF2	5	42	ADC2			
X3_2	60	PF1	5	41	ADC1			
X3_1	61	PF0	5	40	ADC0			
X4_11	62				AREF			
X4_12	63				GND			
X4_9	64				AVCC			

3.3.4 Jumper Applicatie board

Jumper
D.m.v. de jumpers kunnen bepaalde opties gekozen worden. Dit zijn bepaalde poorten welke speciale functies bezitten (zie tabel pintoewijzing van M128). Bijvoorbeeld is de seriële interface via de pinnen PortE.0 en PortE.1 gerealiseerd. Als de seriële interface niet gebruikt wordt, kunnen de desbetreffende jumpers verwijdert worden en deze pinnen zijn dan voor andere functies beschikbaar. Naast de jumpers voor deze poorten zijn er nog extra jumpers, deze worden hierna beschreven.



Jumperposities bij uitlevering

Poorten A tot G

De beschikbare poorten bij de Mega 128 module zijn in de grafiek ingetekend. Daarbij is de gele kant met de module verbonden, de lichtblauwe kant is verbonden met de componenten van het application board. Wanneer een jumper getrokken wordt dan is de verbinding naar het application board verbroken. Dit kan een storing aan de USB, RS232 etc. veroorzaken. De grijze markering toont de eerste pin (pin 0) van de poort.

JP1 en JP2

De jumpers zijn toegewezen aan de toetsen SW1 en SW2. Er bestaat de mogelijkheid de toetsen te laten werken tegen GND of VCC. In de basisinstelling schakelen de toetsen tegen GND.

JP4

JP4 is bedoeld voor het omschakelen van de voedingsspanning (netvoeding of USB). Het application board moet gevoed worden via netvoeding en spanningsregelaar (toestand bij uitlevering). De maximale uithaalbare stroom van de USB interface is lager dan van de netvoeding. Het overschrijden kan de USB interface van de computer beschadigen.

JP6

Bij gebruik van het display kan met JP6 de verlichting (back light) uitgeschakeld worden.

JP7

Als de SRAM op het application board niet gebruikt wordt dan kan es mit JP7 deactiveerd worden en deze poorten staan dan de gebruiker ter beschikking.

JP4

Aan de jumper J4 is de 2^{e} seriële interface van de Mega 128 aangesloten via een niveau-omvormer.

Pin 1 (links, grijs)	TxD
Pin 2 (midden)	RxD
Pin 3 (rechts)	GND

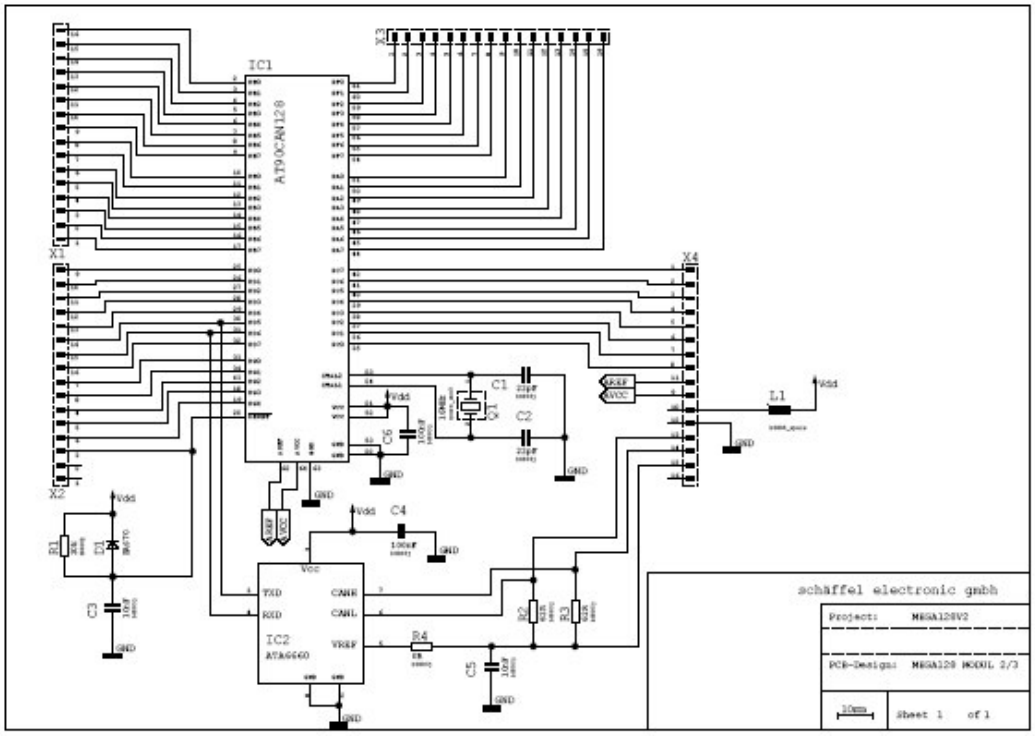
PAD3

PAD3 (rechts naast de module) wordt als ADC_VREF_EXT voor de functies ADC Set en ADC Setint benodigd.

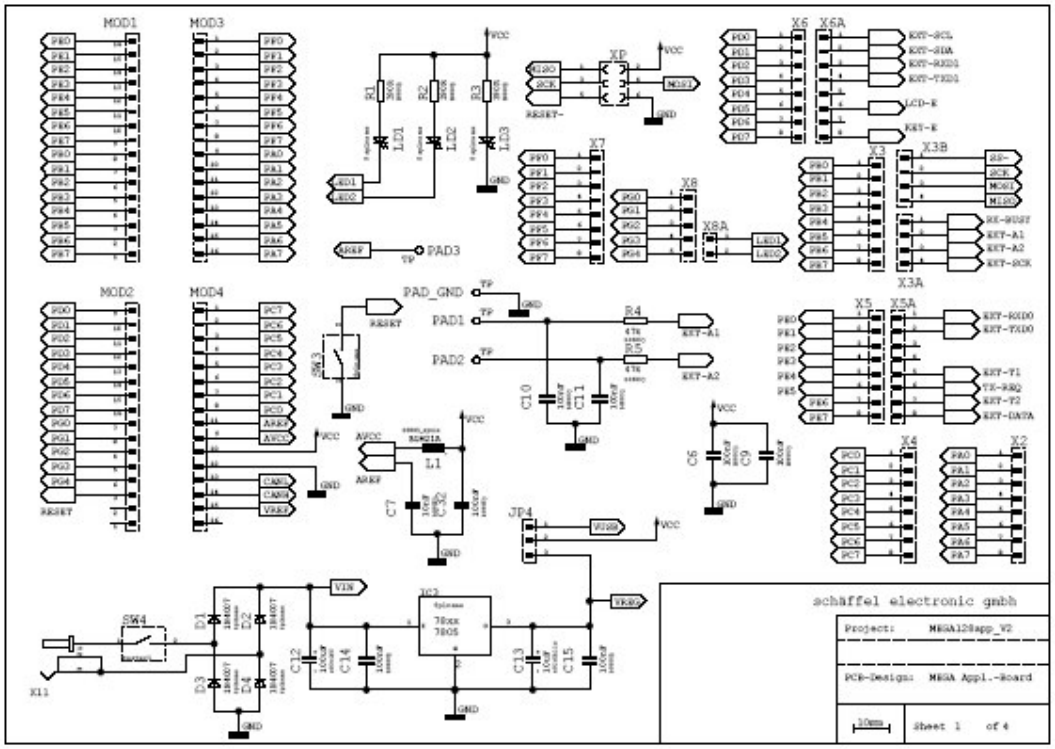
3.3.5 Schakelschema's

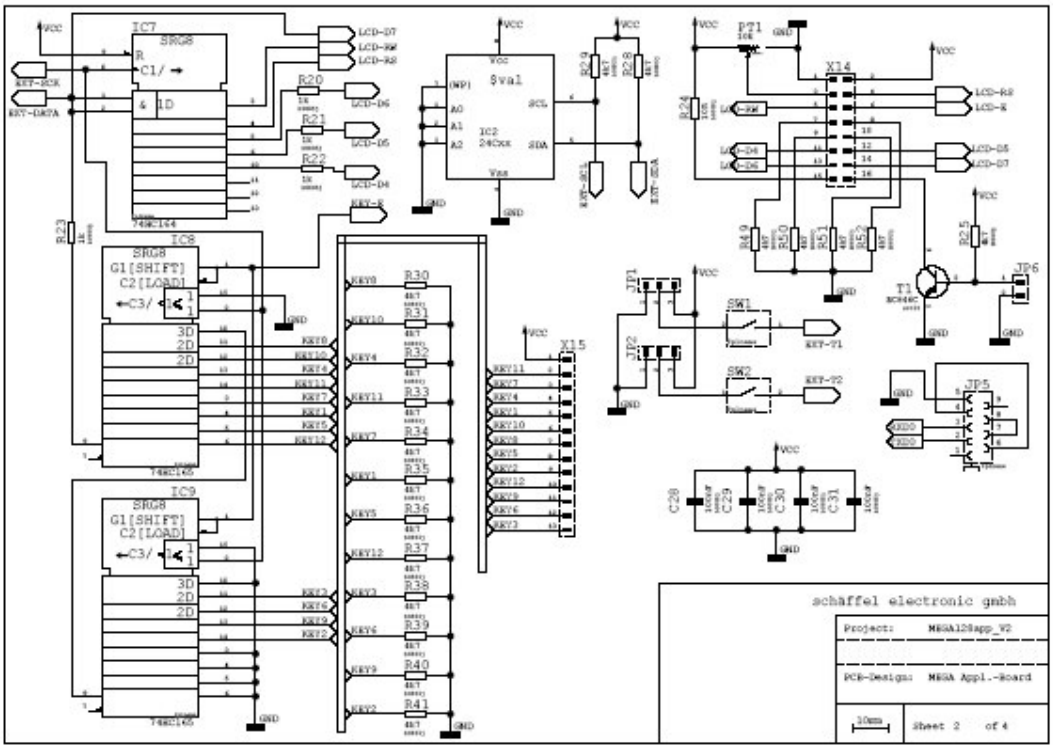
De schakelschema's bevinden zich eveneens op de installatie- CD als pdf-bestand.

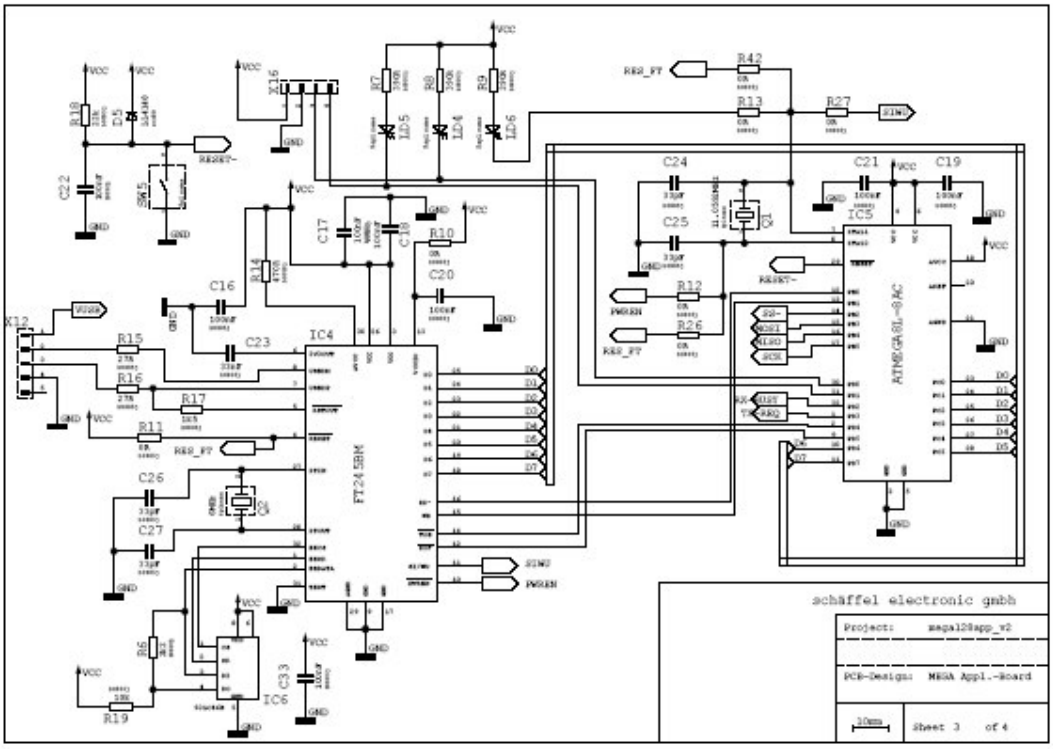
3.3.5.1 Mega 128 module

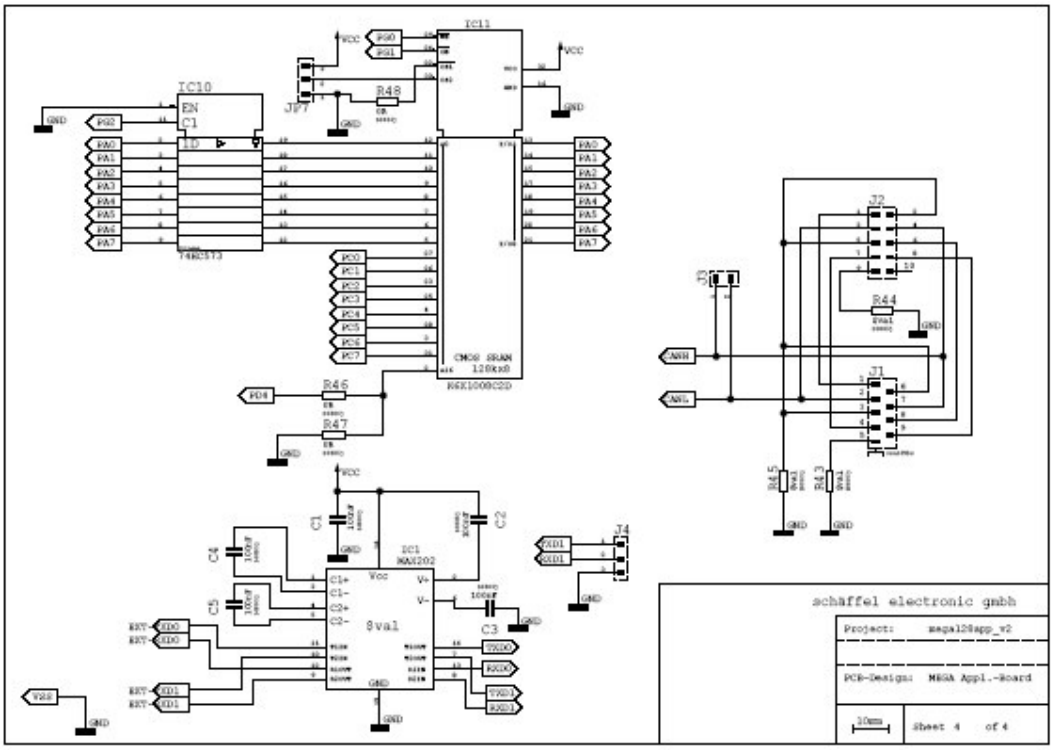


3.3.5.2 Application board

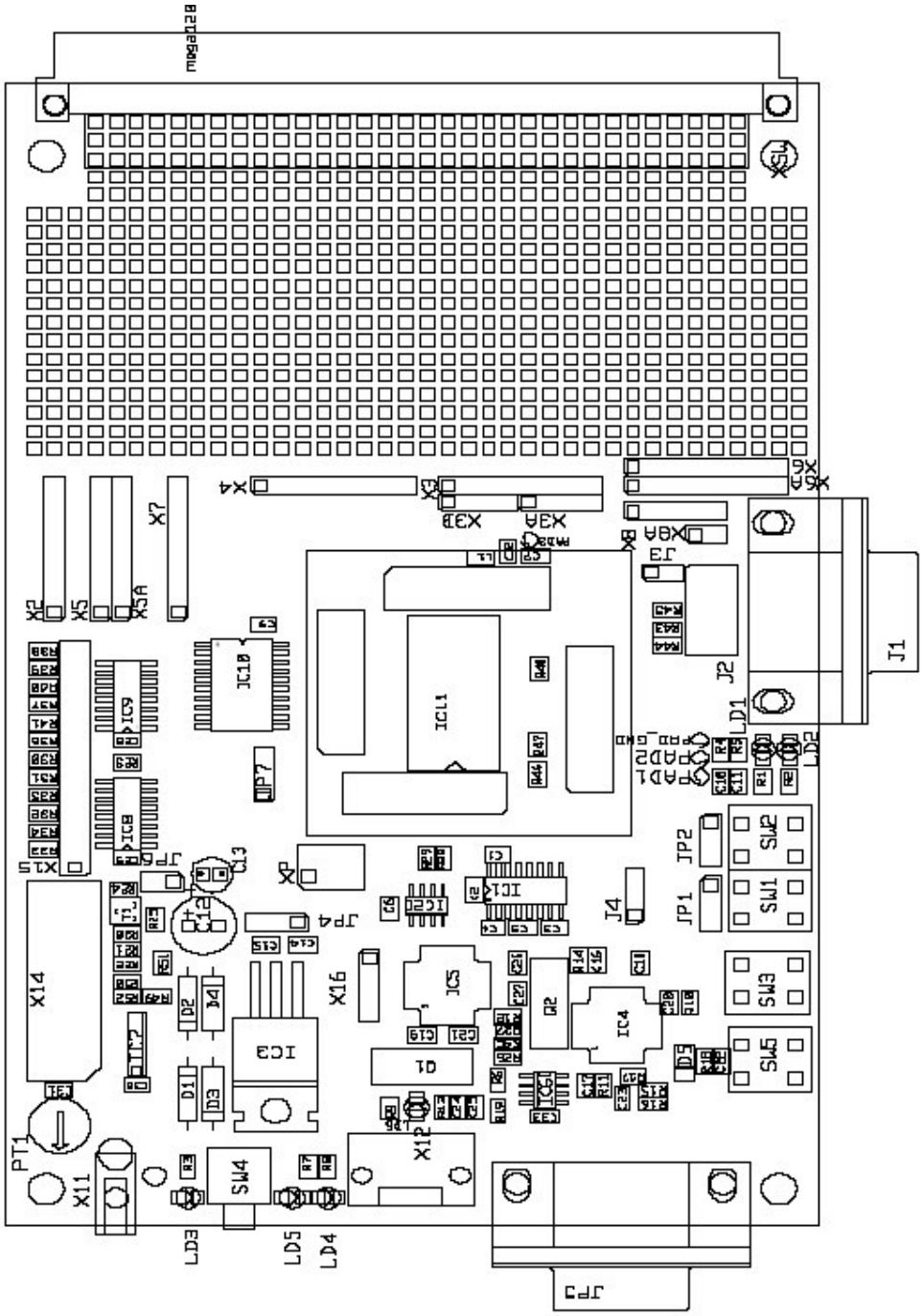



schäffel electronic gmbh
Project: MEGA128app_V2
PCB-Design: MEGA Appl.-Board
Sheet 2 of 4


schäffel electronic gmbh
Project: mega128app_v2
PCB-Design: MEGA Appl.-Board
Sheet 3 of 4


IC11
IC10
R48
JP7
R46
R47
CMOS SRAM
128Kx8
C1
C2
C3
C4
C5
IC1
MAX202
8val
J2
J3
R44
J1
R45
R43
schäffel electronic gmbh
Project: mega128app_v2
PCB-Design: MEGA Appl.-Board
Sheet 4 of 4

3.3.5.3 Opbouwschema

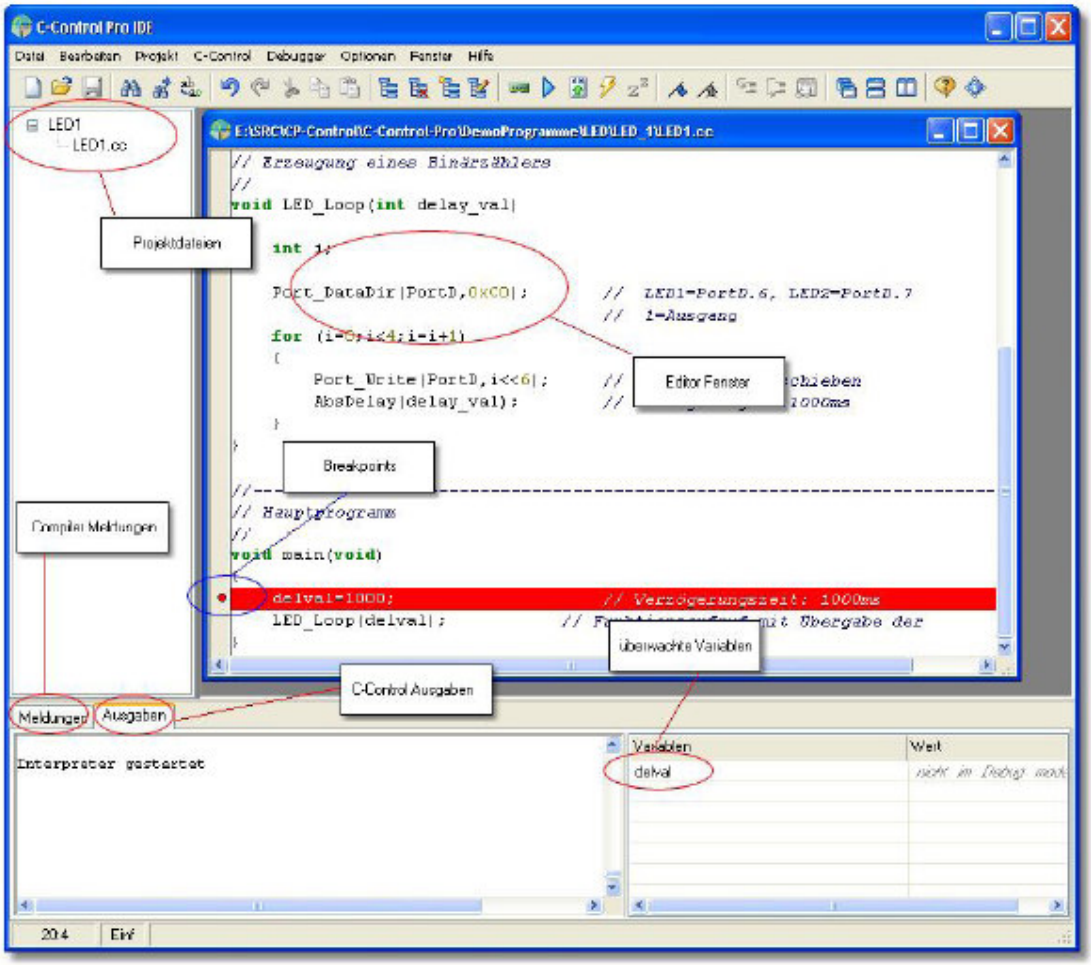


Hoofdstuk

4 IDE

Het C-Control Pro gebruikersoppervlak (IDE) bestaat uit de volgende hoofdelementen:

Sidebar voor [projectbestanden](#) [Editor venster](#)	Meerdere bestanden kunnen hier tot een project geplaatst worden. Er kunnen zo veel editor vensters geopend worden als u maar wilt om bestanden te bewerken.
[Compiler meldingen](#)	Foutmeldingen en algemene compiler informatie worden hier getoond.
[C-Control uitvoeren](#)	Uitvoer van debug berichten van de CompactC programma's.
[Variabelen –venster](#)	Bewaakte variabelen worden hier getoond.



4.1 Projecten

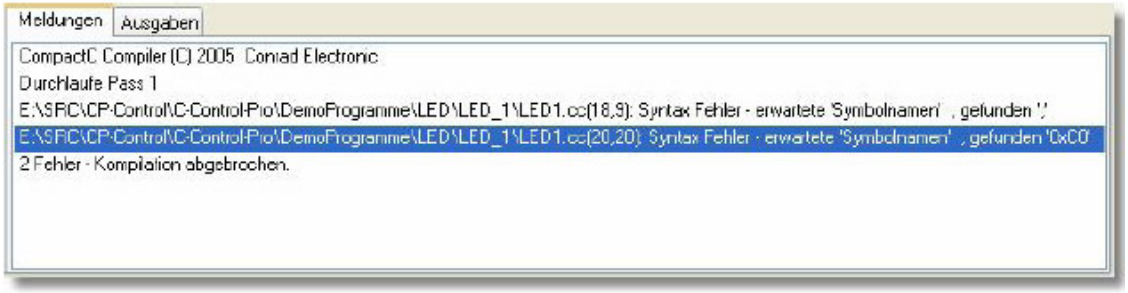
Elk programma voor de C-control module wordt door een project geconfigureerd. In een project staan de gebruikte bronbestanden en bibliotheken. Eveneens zijn hier de instellingen van de compiler. Een project bestaat uit het projectbestand met de extentie “.cprj” en de bijhorende bronbestanden.

4.1.1 Maken van een project

Onder het menu Project kunt u met het oproepen van Nieuw de dialoogbox *Project maken* oproepen. Daar wordt voor het project een projectnaam aangegeven en het project wordt in de sidebar gemaakt.

➔ U dient vooraf te beslissen of u een CompactC of een Basic project wilt maken. In een project kunt u als projectbestanden CompactC en Basic gemengd aanleggen en daaruit een programma maken. De brontekst bestanden in een project bepalen welke programmeertaal van toepassing is. Bestanden met de extensie *.cc lopen in een CompactC context. Bestanden met een extensie *.cbas worden met BASIC vertaald.

4.1.2 Projecten compileren



Onder het menu Project kan met compileren (F9) het actuele project door de compiler vertaald worden.De compiler- berichten worden in een eigen vensterbereik getoond. Als er

fouten bij het compileren optreden, dan wordt per regel de fout beschreven en wel in de vorm van:

```
Bestandsnaam (regel, kolom):     .foutbeschrijving
```

De foutpositie in de brontekst kan via de bevelen Volgende fout (F11) of Vorige fout Shift-F11) gevonden worden. Beide commando's bevinden zich onder het menupunt Project, of er kan door dubbelklikken op een foutbericht van de compiler de cursor bij de foutposities in de editor positioneren.

Bij een succesvolle compilatie wordt de bytecode als bestand met de extensie "*.bc* in het projectregister opgeslagen.

Door met rechts in het bereik van de compilerberichten te klikken kunnen de volgende procedures gestart worden:

- Wissen – verwijdert de lijst met compilerberichten
- Naar het klembord kopiëren – kopieert alle tekstberichten naar het klembord.

4.1 .3 Projectbeheer

Als u met de rechter muistoets op het nieuw gemaakte project in de sidebar klikt, verschijnt er een pop-up menu met de opties

* Neu Hinzufügen - Er wordt een nieuw bestand aangelegd en tegelijkertijd wordt er een editor –venster geopend
* Toevoegen - Een bestaand bestand wordt aan het project toegevoegd
* Andere naam geven - De naam van het project wordt veranderd (dit is niet persé de naam van het projectbestand)
* Kompilieren - De compiler wordt gestart voor het project
* Optionen - De projectopties kunnen veranderd worden

Projectbestanden toevoegen

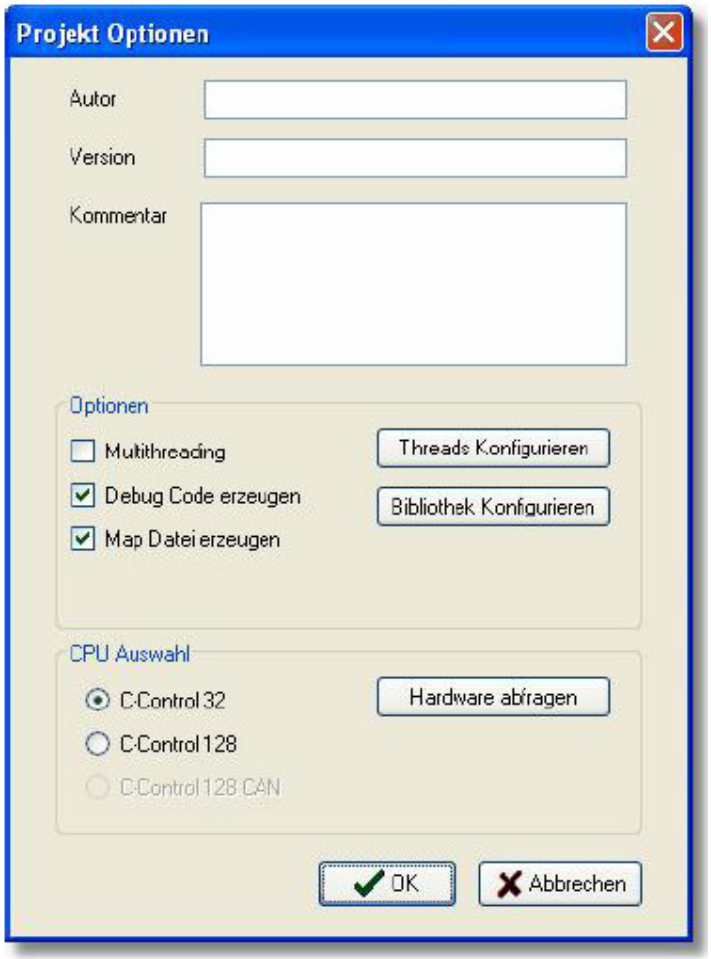
Wordt er op toevoegen van projectbestanden geklikt, verschijnt een bestand- openen-dialoog waarin bestanden geselecteerd kunnen worden die aan het project toegevoegd worden. Er kunnen meerdere bestanden geselecteerd worden.

Projectbestanden

Als bestanden aan het project toegevoegd zijn, kunnen deze geopend worden door dubbelklikken op de bestandsnaam. Door met rechts te klikken verschijnen verdere opties. op de bestandsnaam dubbel dan kunt u de bestanden met een dubbelklik op de bestandsnaam openen. Met een klik op de rechter muisknop verschijnen er nog meer opties:

- Nach oben - Het projectbestand wordt naar boven verschoven (ook met Ctrl – pijl omhoog)
- Nach unten - Het projectbestand wordt naar beneden verschoven (ook met Ctrl – pijl omlaag)
- Umbenennen - De naam van het bestand wordt veranderd
- Entfernen - Het bestand wordt verwijderd uit het project
- Optionen - De projectopties kunnen veranderd worden

4.1.4 Projectopties



Voor elk project kunnen de compilerinstellingen apart veranderd worden.

De invoeren *Autor, Version, Kommentar* kunnen vrij voorzien worden van tekst, ze zijn alleen bedoeld als geheugensteuntje, om zich later beter bijzonderheden van het project te herinneren.

In "CPU Auswahl" legt u het doelplatform van het project vast. Als u op "*Hardware opvragen"* klikt, dan wordt de aangesloten C-Control Pro module uitgelezen en wordt de CPU juist gekozen.

Bij de "Opties" configureert u de multithreading en of er een debug code gemaakt moet worden.

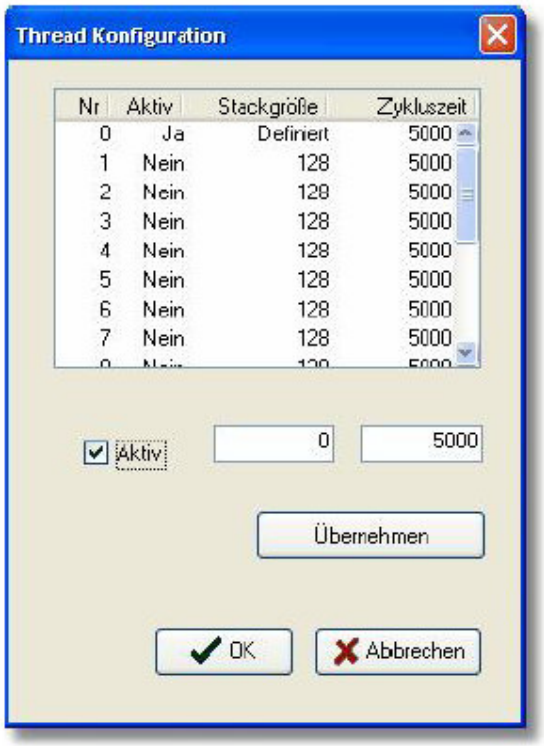
➔ Als er met de debug code gecompileerd wordt, dan wordt de bytecode een klein beetje langer. Per regel in de brontekst die uitvoerbare aanwijzingen bevat, wordt de bytecode een byte groter.

➔ Als er multithreading gebruikt moet worden, dan moet in de project- opties de keuzebox geselecteerd worden en bovendien moeten de threads onder "Threads configureren" apart geparametriseerd worden.

In de opties kan ook gekozen worden of er een Map bestand gemaakt moet worden.

4.1.5 Thread –opties

Om een thread voor de looptijd te kunnen activeren moet hij in deze keuzebox geactiveerd worden en moeten de parameters *stackgrootte* en *cyclustijd* ingesteld worden.



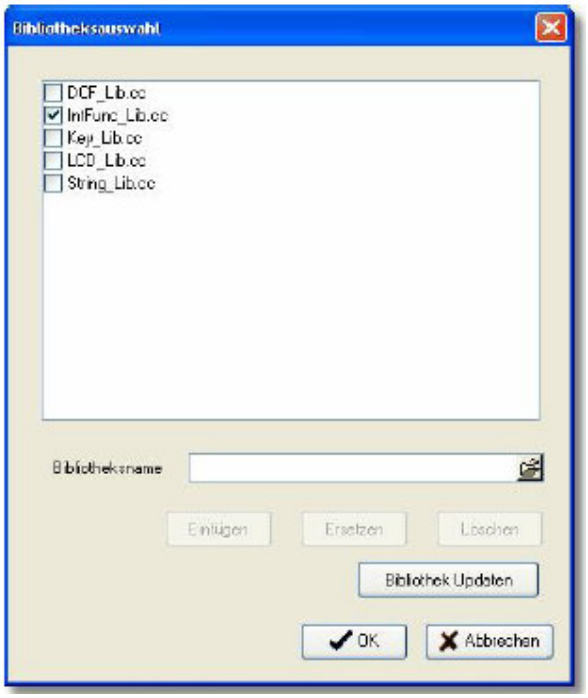
Aan elke extra thread, behalve voor het hoofdprogramma, wordt een plaats op de stack toegewezen, die hij niet mag overschrijden.

➔ Als een thread meer plaats gebruikt dan toegewezen, wordt de geheugenplaats van de andere threads mede beschadigd, en het is zeer waarschijnlijk dat het programma zal crashen.

De cyclustijd is het aantal cycli (bytecode operaties) die een thread mag verwerken tot er omgeschakeld wordt naar een andere thread. Via het aantal cycli tot aan het wisselen van threads wordt ook de prioriteit van de threads gestuurd. Zie ook Threads.

4.1.6 Beheer van de bibliotheek

In het bibliotheekbeheer kunnen de brontekst –bibliotheken gekozen worden die naast de projectbestanden mede gecompileerd worden.



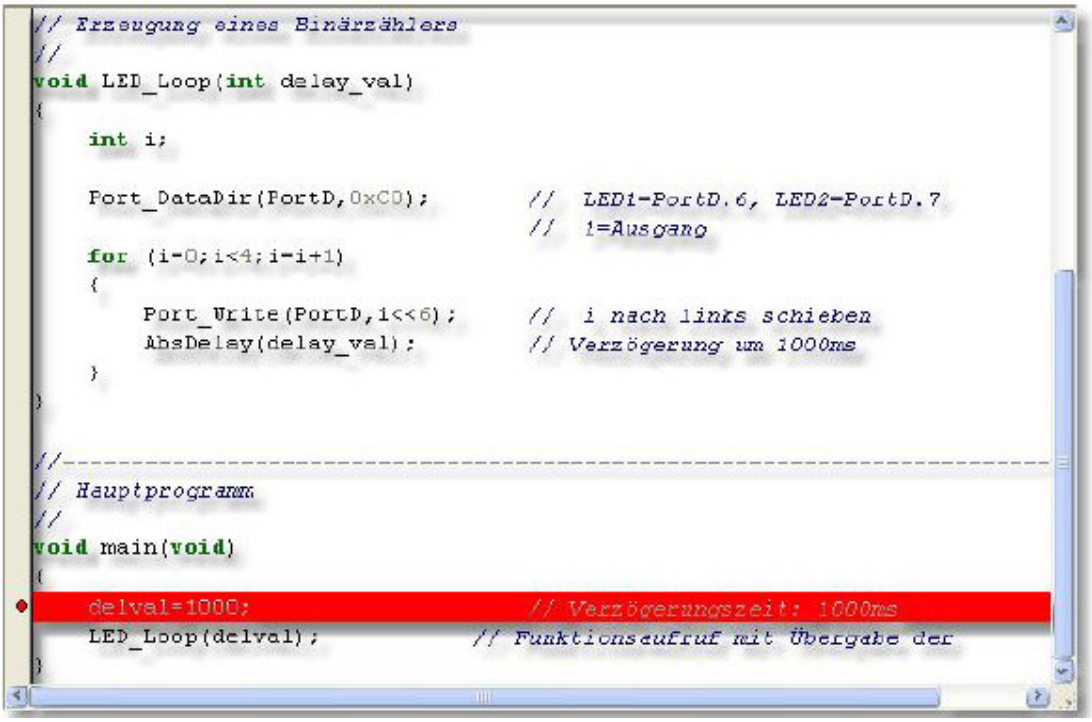
Alleen die bestanden waarvan de checkbox ook geselecteerd werd worden bij het compileren betrokken.

De lijst kan met behulp van het pad tekst -invoerveld “Bibliotheeknaam” en de opties in het dialoogvenster veranderd worden.

* Einfügen - het pad wordt aan de lijst toegevoegd
* Ersetzen - De geselecteerde invoer in de lijst wordt vervangen door de pad –naam
* Löschen - De geselecteerde invoer in de lijst wordt gewist.
* Bibliotheek updaten - Bestanden die in de compiler- voorafinstelling aanwezig zijn, maar niet in deze lijst, worden toegevoegd.

4.2 Editor

U kunt in het C-Control Pro oppervlak meerdere editorvensters openen. Elk venster kan qua grootte en qua getoonde tekstgedeelte veranderd worden. Dubbelklikken op de titelregel maximaliseert het venster.



Een klik op het bereik links naast het begin van de tekst plaatst daar een stop (breakpoint). Daartoe moet eerst de brontekst zonder fouten met *"Debug info"* gecompileerd zijn, en moeten er in de desbetreffende regel daadwerkelijk uitvoerbare programmateksten staan (b.v. geen commentaarregel of dergelijke).

4.2.1 Editorfuncties

Onder het menupunt "bearbeiten" (bewerken) kunt u de belangrijkste editorfuncties vinden:

- "Rückgängig" (Ctrl-Z) voert een Undo operatie uit. Hoeveel dit commando ongedaan maakt hangt ook af van de instelling van " Gruppen rückgängig" (groepen ongedaan).
- "Wiederherstellen" (herstellen) (Ctrl-Y) – herstelt de editortoestand, die eerst door "ongedaan" veranderd werd.
- "Ausschneiden" (knippen) (Ctrl- X) – verwijdert geselecteerde tekst en kopieert deze naar het klembord
- "Kopieren" (Ctrl-C) – kopieert geselecteerde tekst naar het klembord
- "Einfügen" (Crtl-V) – kopieert de inhoud van het klembord naar de cursorpositie
- "Alles markieren" (Ctrl-A) – selecteert de gehele tekst

- “Suchen” (Ctrl-F) – opent de “Zoeken” –dialoog
- “Weitersuchen” (F3) – zoekt verder met dezelfde zoek –criteria
- “Ersetzen” (Ctrl-R) – opent de “Vervangen”-dialoog
- “Gehe zu” (Alt-G) – u kunt naar een bepaalde regel springen

Zoeken / Vervangen dialoog

- Zoektekst – invoerveld voor de te zoeken tekst
- Vervangen door – de vervangende tekst
- Hoofdletters/ kleine letters – onderscheidt hoofd – en kleine letters
- Alleen hele woorden – vindt alleen hele woorden en geen tekenketens
- Reguliere uitdrukkingen – activeert de invoer van reguliere uitdrukkingen in het zoekmasker
- Om bevestiging vragen bij treffers – voor het vervangen wordt de gebruiker om bevestiging gevraagd

Verder kan de zoekrichting bepaald worden, of de gehele tekst of slechts een geselecteerd bereik doorzocht wordt, en of het zoeken bij de plaats van de cursor of aan het begin van de tekst moet beginnen.

4.2.2 Reguliere uitdrukkingen

De zoekfunctie in de editor ondersteunt reguliere uitdrukkingen. Daarmee kunnen tekenketens zeer flexibel gezocht of vervangen worden.

^	Een circumflex aan het begin van een woord vindt het woord aan het begin van de regel
$	Een dollarteken vertegenwoordigt het einde van een regel
.	Een punt symboliseert een willekeurig teken
*	Een sterretje staat voor het meervoudig voorkomen van een patroon. Het aantal mag echter ook nul zijn
+	Een plus staat voor het meerdere keren maar minimaal één keer optreden van een patroon
[ ]	Tekens tussen rechthoekige haakjes staan voor het opduiken van één van de tekens
[^]	Een circumflex tussen rechte haakjes negeert de keuze
[-]	Een minteken tussen rechte haakjes symboliseert een letterbereik
{ }	Accolades groeperen aparte uitdrukkingen. Er mogen maximaal 10 niveaus ingevoegd worden
\	De backslash ontneemt aan het volgende teken de speciale betekenis

Voorbeelden

Voorbeeld	**vindt**
^void	Het woord “void”alleen aan het begin van de regel
;$	De puntkomma alleen aan het eind van de regel
^void$	In de regel mag alleen maar “void” staan
vo.*d	b.v. “vod”, “void”, “vqqd”
vo.+d	b.v. “void”, “vqqd” maar niet “vod”
[qs]	de letters ‘q’of ‘s’
[qs]port	“qport” of “sport”
[^qs]	alle letters behalve ‘q’of ‘s’
[a-g]	alle letters tussen ‘a’ en ‘g’ (inclusief)
{tg}+	b.v. “tg”, “tgtg”, “tgtgtg” enz.
\$	‘$’

4.3 C-Control hardware

Onder het menupunt C-Control kan de hardware relevante functies uitvoeren. Hierbij horen de overdracht en het starten van het programma op de hardware, en eveneens de wacht-woordfunctie.

4.5.1 Programma starten

Programma –overdracht

Als een programma foutloos vertaald is, moet de bytecode eerst overgebracht worden naar de Mega 32 of Mega 128, voor het uitgevoerd kan worden. Dit gebeurt met het commando "Übertragen" (overbrengen – shift-F9) uit het menu C-Control.

➔ Niet alleen de bytecode wordt overgebracht naar de Mega module, maar gelijktijdig wordt ook de nieuwste versie van de interpreter naar de C-Control module gestuurd.

Starten

Door Starten (F10) wordt dan de uitvoering van de bytecode overgebracht naar de Mega 32 of Mega 128 module.

Stoppen

Bij normaal gebruik wordt een programma gestopt door op de toets RESET1 te drukken. Om redenen van performance wordt de uitvoering van het programma op de module bij normaal gebruik niet via de software gestopt. Dit is echter mogelijk met de IDE functie Programma stoppen, als het programma in de debug –modus loopt.

➔ In zeldzame gevallen kan bij USB gebruik het systeem vastlopen als er op de toets RESET1 gedrukt wordt. Gebruik dan de toets RESET2 om ook de Mega8 een reset impuls te geven. De Mega8 houdt zich op het Application Board bezig met de USB interface.

Autostart

Als er geen USB interface is aangesloten, en er werd bij het inschakelen niet op SW1 gedrukt om in de seriële bootloadermodus te komen, dan wordt de bytecode (voor zover aanwezig) in de interpreter gestart. D.w.z. als de module in een hardware applicatie wordt ingebouwd, dan is het aanleggen van de voedingsspanning voldoende om de toepassing automatisch te starten.

➔ Een signaal op INT_0 bij het inschakelen van de C-Control Pro module kan de autostart storen. Volgens de pintoewijzing van M32 en M128 ligt de INT_0 op dezelfde pin als de SW1. Als de SW1 bij het inschakelen ingedrukt wordt, leidt dit tot activering van de seriële bootloader modus en het programma wordt niet automatisch gestart.

4.3.2 Uitvoer

Om debug berichten te tonen is er een “Uitvoer” – venster.

Hier wordt getoond wanneer de bytecode interpreter gestart en beëindigd is, en hoe lang (in milliseconden) de interpreter uitgevoerd werd. De uitvoeringstijd is natuurlijk niet geldig als de interpreter in de debug modus gestopt werd.

In het uitvoervenster kan echter ook de gebruiker zijn eigen debug –berichten laten zien. Voor dit doel bestaan er meerdere debug functies.

Met een klik op de rechter muisknop in het bereik van de debug uitvoer kunt u de volgende commando’s kiezen:

- wissen – wist de lijst met debug berichten
- naar geheugen kopiëren – kopieert alle tekstberichten in het tussengeheugen

4.3.3 PIN functies

De verschillende functies van de interpreter kunnen beveiligd worden met een alfanumerieke PIN. Als een interpreter door een PIN beveiligd is, dan zijn normale operaties verboden. De interpreter kan door een nieuwe overdracht overschreven worden, maar de PIN blijft behouden. Ook het normale starten is niet meer toegestaan, met uitzondering van het Autostart gedrag. Ook het opvragen van de versienummers van hardware en firmware is geblokkeerd.

Als u toch probeert toegang te krijgen tot een verboden functie, dan verschijnt er een dialoog met de tekst “C-Control ist Passwordgeschützt. Operation nicht erlaubt!” (“C-Control is beveiligd met een toegangscode. Operatie niet toegestaan”)

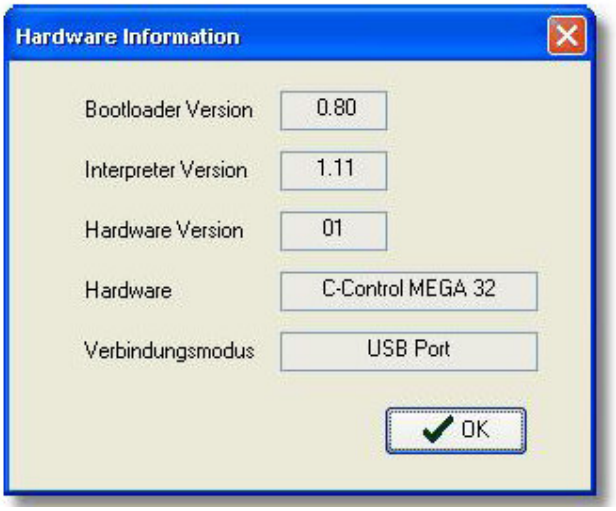
Door het invoeren van de PIN via Pin invoer in het C-Control menu zijn alle operaties toegankelijk.

Om een nieuwe PIN in te voeren of een ingestelde PIN te wissen bestaan er in het C-Control menu de commando’s PIN invoeren en PIN wissen. Als er al een PIN was ingesteld, moet de module natuurlijk eerst door invoer van de oude PIN gedeblokkeerd worden. Een PIN mag maximaal 6 alfanumerieke tekens lang zijn.

➔ Als u de code vergeten bent, is er een functie voor noodgevallen om de module terug te zetten naar de uitgangstoestand. Onder C-Control bestaat de functie Modul zurücksetzen (module terugzetten), waarmee u PIN, interpreter en programma kunt wissen.

4.3.4 Versie controleren

Omdat de C-Control Pro MEGA serie meerdere hardware platformen ondersteunen, is het belangrijk de actuele versienummers van bootloader, interpreter en hardware –versie in het oog te houden. Dit is mogelijk met Hardware versie in C-Control.



4.4 Debugger

Om de debugger te activeren, moet het project eerst zonder fouten met Debug code gecompileerd en naar de module overgebracht zijn. Het bestand met de debug code (*.dbg) moet in het projectregister aanwezig zijn.

In het “Debugger” menu zijn alle debugger commando’s. Met Debug modus (Shift-F 10) wordt de debugger gestart. Is op dat moment geen breakpoint gezet, dan stopt de debugger bij de eerste uitvoerbare aanwijzing.

Als u zich in de Debug modus bevindt, springt u met starten (F10) naar de volgende stop. Als er geen breakpunt gezet is, wordt het programma normaal afgewerkt, met de uitzondering dat het programma gestopt kan worden met Programma stoppen. Dit functioneert echter alleen, als het programma gestart is vanuit de debug modus.

Als de debugger in het programma gestopt is (de groene balk is zichtbaar), dan kunt u het programma stapsgewijs ("singlestep") laten uitvoeren. De commando's Einzelschritt (stap voor stap = Shift-F8) en Prozedurschritt (=procedurestap = F8) voeren steeds de programmacode tot aan de volgende coderegel uit en blijven dan staan. In tegenstelling tot Einzelschritt springt Prozedurschritt niet in functieoproepen, maar loopt overheen.

➔ Als een lus slechts uit één coderegel bestaat, dan voert een enkele stap de hele lus uit, omdat er dan pas naar een nieuwe coderegel vertakt wordt.

Met de aanwijzing Debug modus verlaten wordt de debug modus beëindigd.

➔ Terwijl de debug modus actief is, kan de programmatekst niet veranderd worden. Dit wordt gedaan opdat de regelnummers niet kunnen verschuiven waar breakpoints gezet zijn. De debugger zou anders niet in staat zijn met de bytecode op de C-Control module te synchroniseren.

4.4.1 Breakpoints

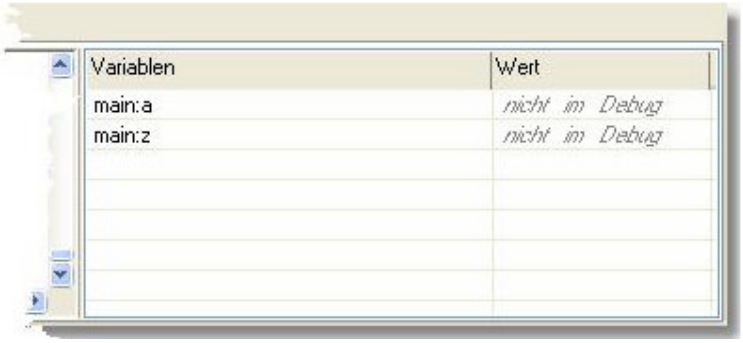
Met de editor is het mogelijk om max. 16 stopposities (breakpoints) te zetten. Een breakpoint wordt ingevoerd door met de linker muistoets, naast de beginposition van een regel, te klikken (zie IDE of Editorvenster).

➔ Het aantal breakpoints is beperkt tot 16, omdat deze informatie bij het lopen van de bytecode interpreter in het RAM meeloopt. Andere debugger zetten stoppunten direct in de programmacode. Dit is hier niet wenselijk, omdat dit de levensduur van het flashgeheugen (ca. 10000 schrijftoegangen) aanzienlijk zou reduceren.

4.4.2 Variabelen -venster

In de debugger kunt u de inhoud van variabelen bekijken. Daartoe wordt de muis boven de variabele geplaatst, en na ca. 2 seconden wordt de inhoud van de variabele als tooltip getoond. De variabele wordt eerst volgens het overeenkomstige bestandstype getoond en dan, door een komma gescheiden, als hexgetal met daarvoor “0x”.

Als u meerdere variabelen wilt controleren, dan kunt u de variabelen in een lijst samenvatten.



Er bestaan twee mogelijkheden om een variabele in te voeren in de lijst van gecontroleerde variabelen. U kunt enerzijds de cursor aan het begin van een variabele in de tekst -editor plaatsen, en dan met een klik op de rechter muisknop variabele invoegen kiezen.

De andere variant gaat via het contextmenu in de lijst van variabelen, dat u eveneens met een klik op de rechter muisknop kunt activeren:

Als u daar variabele invoegen kiest, kunt u de te controleren variabele in de lijst invoeren als tekst. Als het een lokale variabele is, dan wordt daar de functienaam met een dubbele punt vooraan geplaatst (**functienaam : variabele naam**). Met Variabele veranderen kunt u de tekstinvoer in de lijst veranderen, en met variabele verwijderen de variabele uit de lijst verwijderen. Daarbij moet eerst de regel met de te wissen variabele geselecteerd zijn. het commando Alle variabelen verwijderen wist alle invoeren uit de lijst.

➔ Het is niet mogelijk de inhoud van arrays in de debugger te bekijken.

Onder bepaalde omstandigheden wordt in plaats van een waarde in een lijst een foutmelding getoond:

Geen debug code	Er is geen debug code gegenereerd
Foutieve syntax	Bij de tekstinvoer zijn ongeldige tekens voor de variabele ingevoerd
Functie onbekend	De functienaam is onbekend
Variabele onbekend	De variabele –naam is onbekend
Niet in de debug modus	De debug modus is niet geactiveerd
Geen context	Locale variabelen kunnen alleen aangetoond worden, als u zich in de functie bevindt
Niet actueel	De inhoud van de variabele is niet geactualiseerd

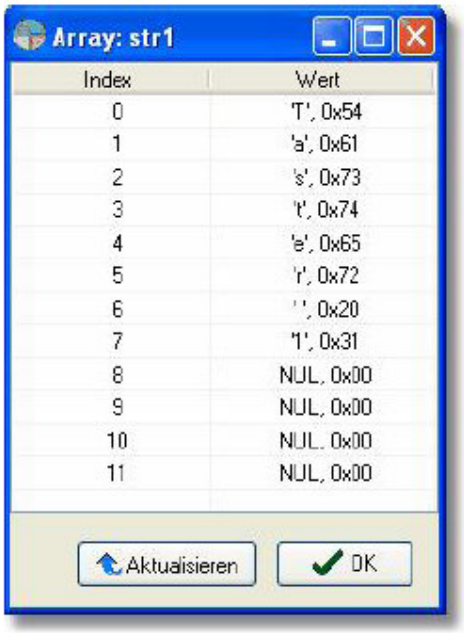
Als er veel variabelen in de controlelijst ingevoerd zijn, dan kan het bij een singlestep lang duren voordat alle inhouden van variabelen uit de module opgevraagd zijn. In dat geval kunt u de optie Auto actualiseren voor aparte variabelen uitschakelen. Dan wordt de inhoud van deze variabelen pas getoond als het commando Variabelen actualiseren uitgevoerd wordt. Op deze manier kunt u snel in de debugger doorgaan met singlestep en de inhouden worden pas indien nodig geactualiseerd.

➔ Variabelen van het type character worden worden als afzonderlijke ASCII tekens weergegeven.

4.4.3 Array venster

Voor het bekijken van de inhoud van array variabelen kan een venster met de inhoud van de array opgeroepen worden. Plaats hiertoe de cursor op de variabelen en klik met rechts op Array weergeven.

In de linker kolom wordt de index van de array getoond en rechts de inhoud. Bij multidimensionele arrays groeit de index in de rechter kolom het snelst.



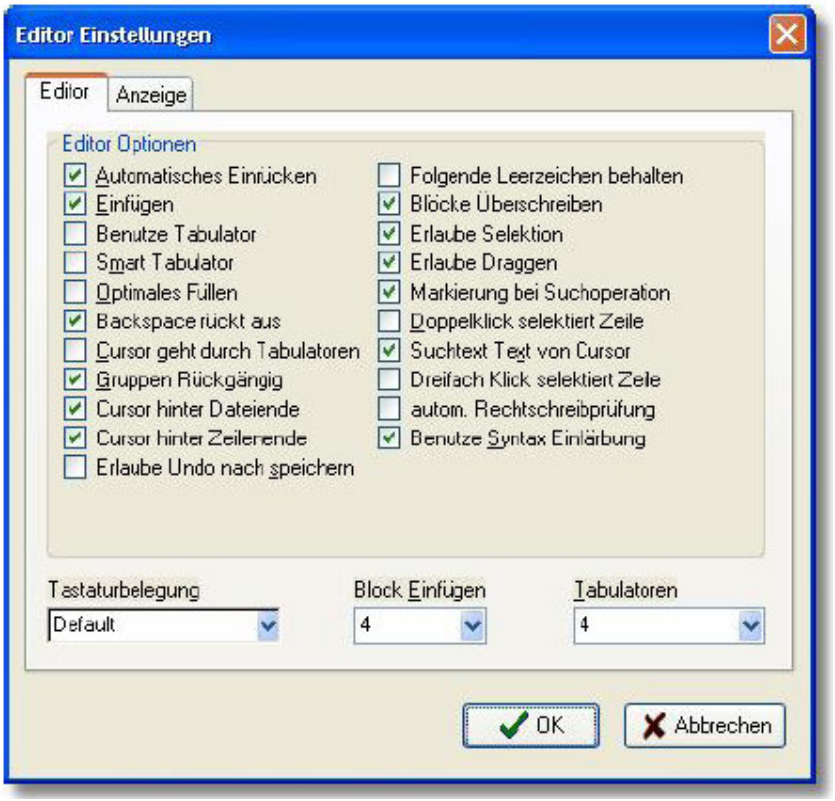
Na elke stop van de debugger of bij een singlestep kan de inhoud van een array venster al niet meer actueel zijn. Als bij elke singlestep in de debugger meerdere array- vensters opnieuw geactualiseerd worden kan het tot vertragingen komen omdat de gegevens van de module geladen moeten worden. Daarom zijn er drie gebruiksmodi:

Automatisch actualiseren	Actualiseren bij singlestep en stoppunten
Bij stoppunt actualiseren	Actualiseren alleen bij stoppunt (breakpoint)
Handmatig actualiseren	Alleen actualiseren na klikken op de schakelaar

4.5 Opties

In het menu opties vindt u de instellingen van de IDE en de standaardinstellingen voor de compiler.

4.5.1 Editorinstellingen



- Automatisch Einrücken – door op Enter te drukken wordt de cursor op de volgende regel ingesprongen op de plaats van de vorige regel.
- Einfügen – Invoegen. is deze optie uit, is “overschrijven” de standaardinstelling.
- Benutze Tabulator – Gebruik tab’s – als deze optie activeert is worden tab’s ingevoegd, anders spaties.
- Smart Tabulator – er wordt met de tabulator een tab ingevoegd tot aan de beginpositie van de vorige regel.
- Optimales Füllen – “Automatisch inspringen” zet eerst tab’s en daarna spaties.
- Backspace rückt aus – met backspace wordt teruggesprongen tot aan de beginpositie van de vorige regel.
- Cursor geht durch Tabulatoren – tab’s worden net als spaties doorlopen.
- Gruppen Rückgang – een undo (ongedaan maken) wordt niet in kleine stappen maar in blokken doorgevoerd.
- Cursor hinter Dateiende – de cursor springt naar het einde van het bestand.
- Cursor hinter Zeilenende – de cursor springt naar het einde van de regel.
- Erlaube Undo nach speichern – de undo- buffer wordt na het opslaan niet leeggemaakt.
- Folgende Leerzeichen behalten – is dit geactiveerd worden spaties op het einde van een regel niet verwijdert.
- Blöcke überschreiben – als een blok geselecteerd is wordt deze door de volgende invoer overschreven.
- Erlaube Selektion – tekst kan geselecteerd worden.
- Erlaube Draggen – tekst kan met de muis geselecteerd en door het vasthouden van de linker muistoets verschoven worden (drag & drop).
- Markierung bei Suchoperation – na het vinden van de gezochte tekst is deze geselecteerd.

- Doppelklick selektiert Zeile – dubbelklikken selecteert een regel, standaard wordt bij het dubbelklikken een woord geselecteerd
- Suchtext von Cursor – de tekst bij het “Zoektekst- invoerveld” wordt door de cursorpositie overgenomen.
- Dreifachklick selektiert Zeile – als een dubbelklik een woord selecteert wordt door drie keer klikken een regel geselecteerd.
- Automatische Rechtschreibprüfung – deze optie schakelt de spellingscontrole voor commentaren in.
- Benutze Syntax Einfärbung – de syntax highlighting voor *.cc en *.cbas- bestanden wordt ingeschakeld.

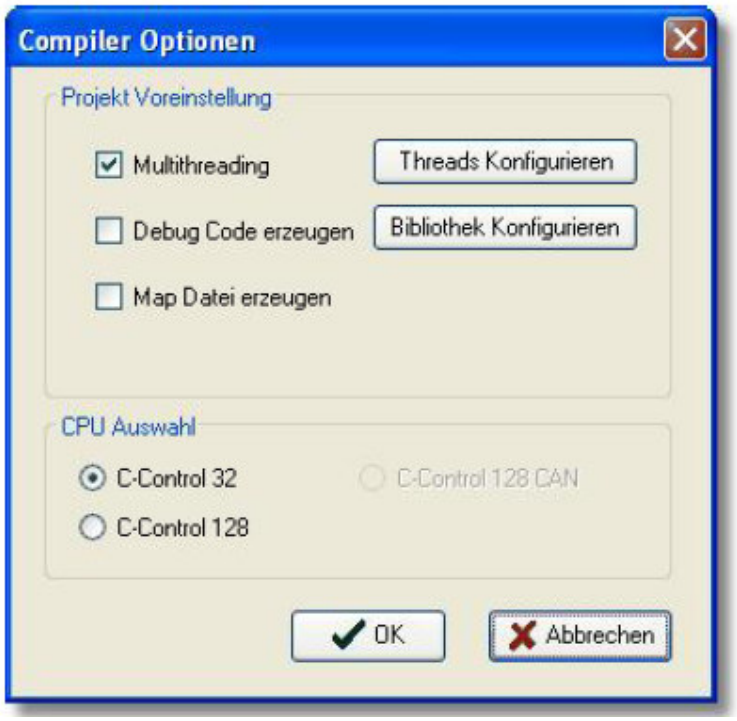
In de keuzebox toetsbezetting kan de toets- layout voor gangbare editor ingesteld worden. Deze emulatie is echter niet compleet omdat het gedrag van de verschillende editor heel complex is. Meestal worden de belangrijkste toetscommando’s ondersteund.

Bij blok invoegen wordt het aantal spaties ingevoegd waarmee de blok geselecteerd werd, met tab of zonder.

Het invoerveld Tabulatoren bepaald hoeveel tekens een tab breed is.

4.5.2 Compiler instellingen vooraf

Bij de compiler instellingen kunnen de standaardwaarden geconfigureerd worden die bij het maken van een nieuwe project opgeslagen worden. De instellingen vooraf zijn onder compiler in het menu optie te vinden.



Een beschrijving van de opties bevindt zich onder projectopties. De keuzeboxen “Threads configureren” en “Bibliotheek configureren” zijn identiek aan de beschrijvingen in hoofdstuk projecten.

4.5.3 IDE instellingen

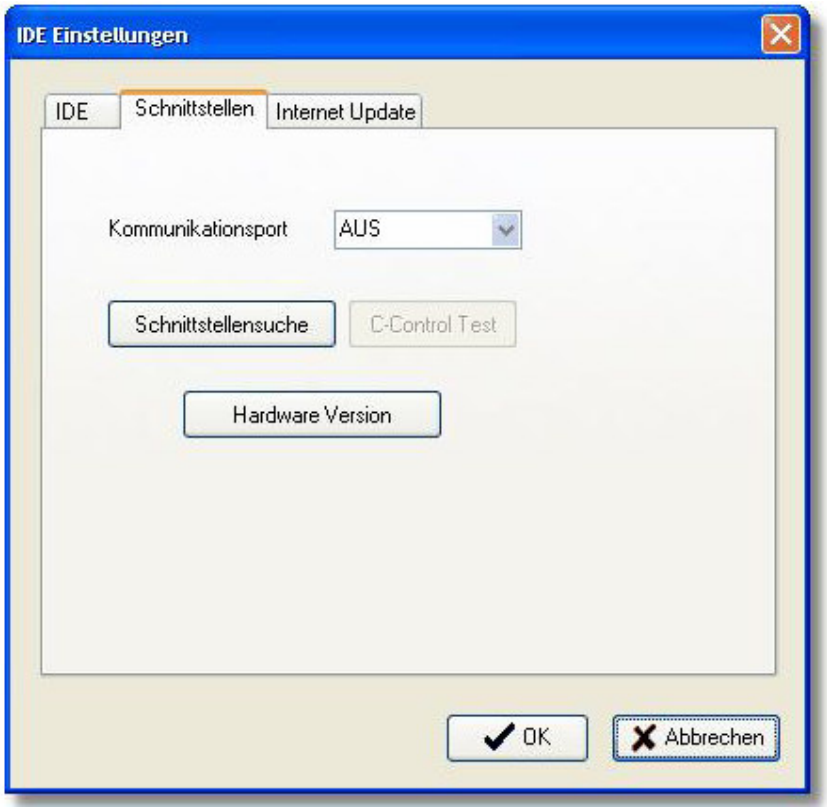
U kunt aparte aspecten van de IDE configureren.

- Übertragung nach Kompilieren Abfrage – Als een programma gecompileerd maar niet naar de C-Control module overgebracht is, wordt de gebruiker gevraagd of het programma gestart moet worden.
- Letztes Projekt wieder öffnen – Het laatste geopende project wordt bij het starten van de C-Control Pro IDE weer geopend.
- Editorvenster maximiert öffnen – Bij het openen van een bestand wordt automatisch het editorvenster op volledige grootte geschakeld.
- Splashscreen nur kurz zeigen – het splashscreen wordt dan alleen tot aan het openen van het hoofdvenster getoond.
- Mehrere Instanzen von C-Control Pro zulassen – Als het C-Control Pro oppervlak meervoudig gestart wordt, kan dat leiden tot conflicten met betrekking tot de USB interface.

Bovendien kunnen hier ook de lijsten van de “laatst geopende projecten”, alsmede de “laatst geopende bestanden” gewist worden.

4.5.3.1 Communicatie

Via een keuzebox kunt u de verbinding met het Application Board instellen. USB verbindingen beginnen met de afkorting “USB” en worden dan doorgenummerd: USB0, USB1 … Seriële interfaces worden op dezelfde manier behandeld. Ze beginnen met de afkorting “COM”: COM 0, COM1 … enz.



Met de toets “Schnittstellensuche” worden alle interfaces doorzocht tot de commandoregels interface van de C-Control Pro reageert. Opdat een Application Board herkend wordt, moet de stroom ingeschakeld zijn en de firmware mag zich niet opgehangen hebben. U kunt het beste voor u gaat zoeken een keer uit – en weer inschakelen.

De knoppen “C-Control Test” en “Hardware Version” maken het mogelijk direct te zien of de gekozen interface ook zinvol kan communiceren met de C-Control Pro module.

4.5.3-2 Internet update

Om te controleren of er door Conrad verbeteringen of correcties van fouten gepubliceerd zijn, kunt u de Internet update activeren. Als u de keuzebox “Alle n dagen controleren op updates” kiest, dan wordt met een interval van n dagen bij het starten van de IDE op internet naar een update gezocht. De parameter n kan ingesteld worden in het invoerveld rechts ernaast.

De knop “Jetzt auf Update prüfen” (“nu op update controleren”) activeert onmiddellijk het zoeken naar updates.

➔ Opdat de internet update zoals voorgeschreven functioneert, mag de MS Internet Explorer niet in de “offline” modus staan.

Als bijv. vanwege een firewall de toegang tot internet beperkt is door een proxy, dan kunnen de proxy instellingen zoals adres, gebruikersnaam en code in deze dialoog aangegeven worden.

➔ Als er in MS Internet Explorer data ingevoerd zijn, dan hebben deze een hogere prioriteit en overschrijven ze de instellingen in deze dialoog.

4.6 Venster

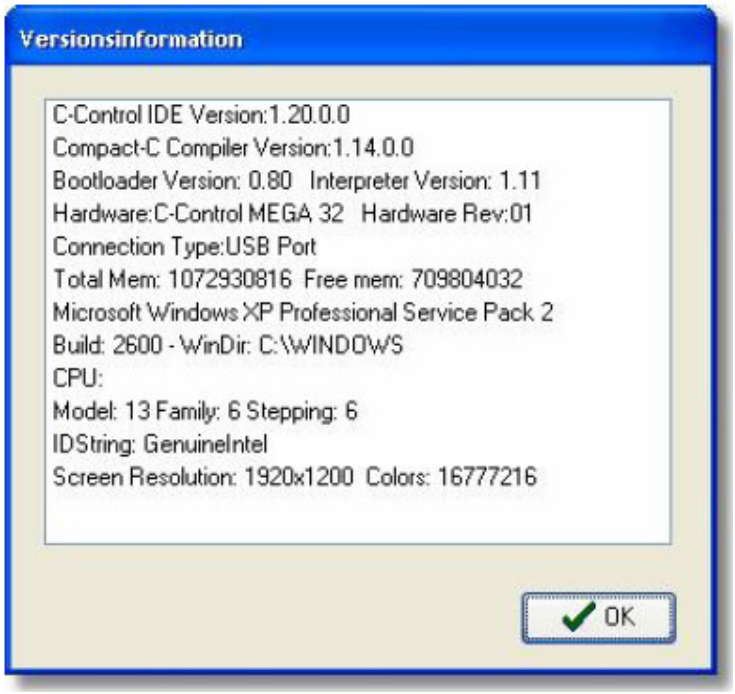
Als er in het editorbereik meerdere vensters geopend zijn, kunt u via de commando’s in het “Venster” menu de editorvensters automatisch laten rangschikken.

- Überlappend – de vensters worden boven elkaar gerangschikt, elk venster is daarbij iets verder naar rechts verschoven dan het vorige.
- Untereinander – de vensters worden verticaal onder elkaar geplaatst
- Nebeneinander – ordent de vensters van links naar rechts naast elkaar
- Alle Minimieren – verkleint de vensters tot symboolgrootte
- Schliessen – sluit het actieve venster

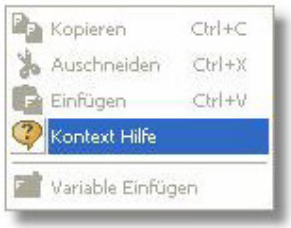
4.7 Hulp

Onder het menupunt “Hilfe” (Hulp) kunt u met Inhalt (Inhoud - toets F1) het helpbestand oproepen.

Het menupunt Programmaversie opent het volgende “Versie –informatie” -venster en kopieert gelijktijdig de inhoud naar het klembord.
Als er een support email naar Conrad geschreven moet worden, dan is deze informatie belangrijk. Omdat u bij het oproepen van Programmaversie ook gelijk in het klembord bent, kunt u deze data makkelijk aan het eind van een email invoegen.



Als u in het hulpbestand naar een bepaald begrip wilt zoeken, dan kan de Kontexthilfe (Hulp bij de context) het zoeken vergemakkelijken. Als u bijv. in de editor met de cursor in het woord “AbsDelay” staat en u zoekt naar de juiste parameters, dan kunt u simpelweg de Kontexthilfe aanklikken. Deze functie neemt het woord waarop de cursor staat als zoekbegrip en toont de resultaten in het Hulpbestand.



Het commando Kontexthilfe kan eveneens opgeroepen worden door een rechtsklik in het editorvenster.

Hoofdstuk

5. Compiler

5.1 Algemene features

Dit bereik geeft informatie over compiler- eigenschappen en features die niet afhankelijk zijn van de gebruikte programmeertaal.

5.1.1 Externe RAM

Op het application board **Mega 128** is een externe RAM aanwezig. Dit RAM wordt door de interpreter automatisch herkent en gebruikt voor het uit te voeren programma. In plaats van ca. 2665 byte zijn dan ca. 63848 byte beschikbaar als programmageheugen. Hiervoor moet het programma niet opnieuw gecompileerd worden.

➔ Als de SRAM niet gebruikt wordt kan dit met JP7 gedeactiveerd worden en deze poorten zijn dan vrij.

5.1.2 Preprocessor

➔ De Gnu Generic preprocessor die hier gebruikt wordt, heeft nog meer functies die onder http://nothingisreal.com/gpp/gpp.html gedocumenteerd zijn. Echter, alleen de hier beschreven functies, ook in samenhang met de C-Control Pro compiler zijn ook uitvoerig getest. Het gebruik van niet hier gedocumenteerde functies geschiedt op eigen risico!

Het C-Control ontwikkelingssysteem beschikt over een volledige C- preprocessor. De preprocessor bewerkt de brontekst voor de compiler gestart wordt. De volgende commando's worden ondersteund:

Definities

Men definieert met het commando "#define" tekstconstanten.

```
#define symbol tekstconstante
```

Omdat de preprocessor voor de compiler loopt, wordt bij elke keer dat symbol in de brontekst opduikt, symbol vervangen door tekstconstante.

Een voorbeeld:

```
#define Pl 3.141
```

➔ Als een project uit meerdere bronnen bestaat, dan is er een #define constante voor alle bronbestanden vanaf het bestand in welke de constante gedefinieerd werd. Daarom is het mogelijk de volgorde van de bronbestanden in een project te wijzigen.

Voorwaardelijke compilering

```
#ifdef  symbol
…
#else  //  optie
…
#endif
```

U kunt controleren welke delen van een brontekst werkelijk gecompileerd worden. Na een #ifdef symbol aanwijzing wordt de volgende tekst alleen gecompileerd als het symbol ook gedefinieerd is door #define symbol.

Als er een optionele #else aanwijzing aangegeven is, dan wordt de brontekst na #else bewerkt wanneer het symbol niet gedefinieerd is.

Invoegen van tekst

```
#include  pad\bestandsnaam
```

Met deze aanwijzing kan een tekstbestand in de broncode ingevoegd worden.

➔ Vanwege een beperking van de preprocessor mag het pad in een #include aanwijzing geen spaties bevatten!

5.1.2.1 Vooraf gedefinieerde symbolen

Om de werkzaamheden met de verschillende uitvoeringen van de C-Control Pro serie te vergemakkelijken, zijn er een serie van definities die in afhankelijkheid van doelsysteem en compiler projectopties gezet worden. Deze constanten kunnen met #ifdef, #ifndef of #if opgevraagd worden.

Symbool	Betekenis
MEGA32	Configuratie voor Mega 32
MEGA128	Configuratie voor Mega 128
MEGA128CAN	Confguratie voor Mega 128 CAN bus
__DEBUG__	Debug bestanden worden gemaakt
__MAPFILE__	Een geheugenlayout wordt berekend

De onderstaande constanten beïnhouden een string. Zinvol is deze in samenhang met tekstuitvoeren te gebruiken.

Symbool	Betekenis
__DATE__	Actuele datum
__TIME__	Tijd van de compilering
__LINE__	Actuele regel in de sourcecode
__FILE__	Naam van het actuele bronbestand
__FUNCTION__	Actuele functienaam

Voorbeeld

Er worden regelnummers, bestandsnaam en functienaam gegeven. Omdat de bestandsnaam lang kan zijn, moet de character array vrij groot gedimensioneert worden:

```
char txt[60];
txt=__LINE__;
Msg_WriteText(txt); // regelnummer aangeven
Msg_WriteChar(13); // LF
txt=__FILE__;
Msg_WriteText(txt); // bestandsnaam aangeven
Msg_WriteChar(13); // LF
txt=__FUNCTION__;
Msg_WriteText(txt); // functienaam aangeven
Msg_WriteChar(13); // LF
```

5.1.3 Pragma aanwijzingen

Met de aanwijzing #pragma kan de uitvoer en het verloop van de compiler gestuurd worden. Volgende commando's zijn toegestaan:

#pragma Error "xyz…"	Een fout wordt gemaakt en de tekst "xyz…" gegeven
#pragma Warning "xyz…"	Een waarschuwing wordt gemaakt en de tekst …"gegeven
#pragma Message "xyz…"	De tekst "xyz…" wordt van de compiler gegeven

Voorbeeld

Deze #pragma aanwijzingen worden vaak in samenwerking met preprocessor commando's en vooraf gedefinieerde constanten toegepast. Een klassiek voorbeeld is de productie van een foutmelding, nadat aan bepaalde hardware- criterias niet voldaan werd:

```
#ifdef MEGA128
#pragma Error "Counter functies niet bij Timer0 en Mega128"
#endif
```

5.1.4 Bestandsmap

Als bij het compileren een map met bestanden aangemaakt wordt kan men daar de geheugengrootte van de gebruikte vaiabelen nakijken.

Voorbeeld

Het project CNT0.cprj maakt bij het compileren de volgende bestand:

```
Globale Variablen          Lengte     Positie    (RAM begin)
-------------------------------------------------------------
Totale lengte:     0 bytes

Locale variablen           Lengte     Positie    (Stackrelatief)
-------------------------------------------------------------
Funktie Pulse()
Count                       2           4
i                           2           0
```

```
Totale lengte:     4 bytes

Funktie main()
count                          2          2
n                              2          0
Totale lengte: 4 bytes
```

In deze lijst is te zien dat er geen globale variabelen gebruikt worden. Verder zijn er twee functies, "Pulse()" en "main()". Elke functie heeft een geheugenverbruik van 4 Byte aan lokale variabelen.

5.2 CompactC

Voor het programmeren van de C-Control Pro Mega 32 of Mega 128 kan de programmeertaal CompactC toegepast worden. De compiler vertaald de programmeertaal CompactC naar een bytecode, die van de interpreter van de C-Control Pro verwerkt wordt. De taalomvang van CompactC komt ongeveer overeen met ANSI-C maar is op sommige plaatsen gereduceeerd, omdat de firmware rescourcen besparend implementeerd moest worden. Volgende taalconstructies zijn er niet:

- **structs** / **unions**
- **typedef**
- **enum**
- Constanten (**const** aanwijzing)
- Rekenkundige wijzer

Uitgebreide programma voorbeelden vindt u in de map “Demoprogramma’s” die met de ontwikkelaarsomgeving geinstalleerd werd. Daar zijn op bijna alle takengebieden van de C-Control Pro module voorbeeldoplossingen.

5.2.1 Programma

Een programma bestaat uit een hoeveelheid aanwijzingen (zoals bijv. “a=5;”), die over verschillende functies verdeeld zijn. De startfunctie die in elk programma aanwezig moet zijn, is de functie “main()”. Een klein programma dat een getal in het uitvoervenster drukt:

```
void main (void)
{
        Msg_WriteInt(42);  // Het antwoord op alles
}
```

Projecten

Men kan een programma verdelen over meerdere bestanden die in een project (zie projectbeheer) samengevat zijn. Naast deze bestanden kunt u bibliotheken aan een project toevoegen, die functies ter beschikking stellen die door het programma gebruikt worden.

5.2.2 Aanwijzingen

Aanwijzing

Een aanwijzing bestaat uit meerdere gereserveerde commandowoorden, indicatoren en operatoren, die met een puntkomma (';') aan het eind afgesloten wordt. Om verschillende elementen van een aanwijzing te scheiden, bestaat tussen de aparte aanwijzingselementen een tussenruimte, in het Engels ook "*whitespaces*" genoemd.
Onder tussenruimte worden verstaan spaties, tabs en regeldoorvoer ("C/R en LF"). Daarbij maakt het niet uit of de tussenruimte wordt gevormd door één of meerdere *"whitespaces*".

Eenvoudige aanwijzing:

```
a =   5;
```

➔ Een aanwijzing hoeft niet persé compleet in een regel te staan. Omdat ook regeldoorvoeren tot de tussenruimte horen, is het legitiem om een aanwijzing over meerdere regels te verdelen.

```
If(a==5)   // aanwijzing over twee regels
a=a+10;
```

Aanwijzingsblok

Meerdere aanwijzingen kunnen in een blok gegroepeerd worden. Daarbij wordt het blok met een linker accolade ("{") geopend, daarna volgen de aanwijzingen, en aan het eind wordt het blok gesloten met een rechter accolade ("}"). Een blok hoeft niet beëindigd te worden met een puntkomma. Dat betekent, dat als een blok het eind van een aanwijzing vormt, het laatste teken van de aanwijzing de rechter accolade sluiten is.

```
If(a>5)
{
      a=a+1;           // Aanwijzingsblok
      b=a+2;
}
```

Commentaren

Er bestaan twee soorten commentaren, éénregelige en commentaren met meerdere regels. Daarbij wordt de tekst in de commentaren door de compiler genegeerd.

- Eénregelige commentaren beginnen met "//" en stoppen bij het eind van de regel.
- Commentaren met meerdere regels beginnen met "/*" en stoppen met "*/".

```
/* Een
meerregelig
commentaar */

// Een éénregelig commentaar
```

Indicatoren

Indicatoren zijn de namen van functies of variabelen.

- Geldige tekens zijn de letters (A-Z, a-z), de cijfers (0 – 9) en de liggende streep (‘ _’)
- Een indicator begint steeds met een letter
- Er wordt verschil gemaakt tussen hoofd – en klein letters
- Gereserveerde woorden zijn niet toegestaan als indicator
- De lengte van indicatoren is niet beperkt

Rekenkundige termen

Een rekenkundige term is een hoeveelheid waarden, die met operatoren verbonden zijn. Onder waarden worden in deze context verstaan getallen, variabelen en functies.

Een eenvoudig voorbeeld:

2 + 3

Hierbij worden de getallen 2 en 3 gekoppeld d.m.v. de operator “+”. Een rekenkundige term vertegenwoordigt weer een waarde. Hier is de waarde 5.

Andere voorbeelden:

```
a - 3

b + f(5)

2 + 3 * 6
```

Volgens “punt voor streep” wordt hier eerst 3 x 6 uitgerekend en daarna 2 er bij opgeteld. Deze voorrang van operatoren heet bij operatoren precedent. U vindt een opsomming van de prioriteiten in de precedent tabel.

➔ Ook vergelijkingen zijn wiskundige termen. De vergelijkingsoperatoren geven als resultaat een waarheidswaarde van “1”of “0”, afhankelijk van of de vergelijking correct was. De term “3 < 5” geeft de waarde “1” (waar; true).

Constante termen

Een term of delen van een term kan/kunnen constant zijn. Deze deeltermen kunnen al tijdens de compiler –looptijd berekend worden.

Zo wordt bijv.

12 + 123 – 15

door de compiler samengevat tot

120

Soms moeten termen constant zijn opdat ze geldig zijn. Zie bijv. declarering van array variabelen.

5.2.3 Datatypes

Waarden bezitten steeds een bepaalde datatype. De integerwaarden (gehele getallen waarden) hebben in CompactC een 8 of 16 bit breed datatype, getallen met floating point zijn altijd 4 byte lang.

Datatypee	Voorteken	Waardebereik	Bit
char	Ja	-128 … +127	8
unsigned char	Nee	0 … 2555 8	8
byte	Nee	0 … 2555 8	8
int	Ja	-32768 … +32767	16
unsigned int	Nee	0 … 65535	16
word	Nee	0 … 65535	16
float	Ja	±1.175^{e}-38 to ±3.402^{e}38	32

Zoals te zien is, zijn de datatypes **"unsigned char"** en **"byte"** identiek, net als **"unsigned int"** en **"word".**

Strings

Er is geen expliciet "String" datatype. Een string is gebaseerd op een character array. U moet de grootte van de array dusdanig kiezen, dat alle tekens van de string in het character array passen. Bovendien is er ruimte nodig voor een termineringsteken (decimale nul), om het einde van de keten aan te geven.

Type –convertering

Bij wiskundige termen gebeurt het heel vaak dat aparte waarden niet van hetzelfde type zijn. Zo zijn de datatypes in de volgende term gemengd (**a** is integer variabele).

```
a + 5.5
```

In dit geval wordt **a** eerst geconverteerd naar het datatype **float** en daarna wordt er 5.5 bij opgeteld. Het datatype van het resultaat is eveneens **float**. Bij de typeconvertering gelden de volgende regels:

- Als bij de verbinding van twee 8 bit of 16 bit integere waarden één van beide datatypes van een voorteken is voorzien ("**signed**"), dan is ook het resultaat van de term van een voorteken voorzien. D.w.z. de operatie wordt **"signed**" uitgevoerd.

- Als één van beide operandi van het type **float** is, dan is het resultaat eveneens van het type **float**. Als één van de beide operandi een 8 bit of 16 bit datatype heeft, dan wordt deze voor de operatie omgevormd tot een **float** datatype.

5.2.4 Variabelen

Variabelen kunnen verschillende waarden aannemen, afhankelijk van het datatype waarmee ze gedefinieerd zijn. Een variabele –definitie ziet er als volgt uit:

```
Type  variabelennaam;
```

Als u meerdere variabelen van hetzelfde type wilt definiëren, kunt u meerdere variabelennamen door een komma gescheiden aangeven:

```
Type  naam1, naam2, naam3, …;
```

Als type zijn toegestaan: **char, unsigned char, byte, int, unsigned int, word, float**

Voorbeelden:

```
int  a;

int  i, j;

float  xyz;
```

Aan integere variabelen kunnen getalwaarden decimaal of als hexgetal toegewezen worden. Voor een hexgetal worden voor het getal de letters **"0x"** gezet. Bij variabelen met een van voorteken voorzien datatype kunnen negatieve decimale getallen toegewezen worden door een minteken voor het getal te plaatsen.

Voorbeelden:

```
word a;
int  i,j;

a=0x3ff;
i=15;
j=-22;
```

Getallen met zwevende komma (datatype **float**) mogen een decimale komma en een exponent bevatten:

```
float  x,y;

x=5.70;
y=2.3e+2;
x=-5.33e-1;
```

sizeof Operator

Met de operator **sizeof()** kan het aantal bytes bepaald worden die een variabele in het geheugen inneemt.

Voorbeeld:

```
int  s;
float  f:

s=sizeof(f);      //  de waarde van s = 4
```

➔ Bij arrays wordt ook alleen de bytelengte van het basis –datatype als uitkomst gegeven. U moet de waarde met het aantal elementen vermenigvuldigen om het geheugenverbruik van de array te berekenen.

Array variabelen

Als u achter de naam bij de variabelen –definitie tussen rechte haakjes een getalswaarde schrijft, dan heeft u een array gedefinieerd. Een array legt de plaats voor de gedefinieerde

variabele meervoudig in het geheugen vast. Bij de voorbeelddefinitie:

```
int  x[10 ];
```

wordt voor de variabele x de 10-voudige geheugenplaats vastgelegd. De eerste geheugenplaats kan aangesproken worden met x[0], de tweede met x[1], de derde met x[2], …tot x[9]. U mag bij de definitie natuurlijk ook andere indexgroottes kiezen. De beperking is alleen de RAM geheugenplaats van de C-Control Pro.
U kunt ook meerdimensionale arrays declareren, waarin nog meer rechte haakjes bij de variabelen –definitie toegevoegd worden:

```
int  x[3] [4];        // array met 3*4 invoeren
int  y[2] [2] [2];          // array met 2*2*2 invoeren
```

➔ Arrays mogen in Compact-C maximaal 16 indices (dimensies) hebben. De maximale waarde voor een index is 65535. De indices van de arrays zijn altijd op nul gebaseerd, d.w.z. elke index begint met 0.

➔ Er vindt tijdens het lopen van het programma geen controle plaats of de gedefinieerde indexgrens van een array is overschreden. Als de index tijdens de programmabewerking te groot wordt, neemt het programma zijn toevlucht tot vreemde variabelen en is de kans groot dat het programma ‘crasht’.

Strings

Er is geen specifieke “String” datatype. Een string is gebaseerd op een array van het datatype **char**. U moet de grootte van de array zo kiezen, dat alle tekens van de string in de character array passen. Bovendien is er plaats nodig voor een termineringsteken (decimale nul), om het eind van de tekenketen aan te geven.

Voorbeeld van een tekenketen met maximaal 20 tekens:

```
char  str1[21];
```

Als uitzondering mag men aan **char** arrays tekenketens toewijzen. Daarbij wordt de tekenketen tussen aanhalingstekens gezet.

```
Str1=”Hallo wereld!”;
```

➔ Er kan geen String aan grotere **char** arrays toegewezen worden. Echter zijn er trucs voor ontwikkelaars:

```
char str_array[3][40];
char single_str[40];
single_str="A String";
Str_StrCopy(str_array,single_str,40); // kopieert single_str in de tweede
String
```

Dit functioneert omdat met een afstand van 40 tekens achter de string in str_array de ruimte voor de tweede string ligt.

Zichtbaarheid van variabelen

Als variabelen buiten de functies gedeclareerd worden, hebben ze een globale zichtbaarheid. D.w.z., ze zijn vanuit elke functie aanspreekbaar. Declaraties van variabelen binnen

functies produceren locale variabelen. Locale variabelen kunnen alleen binnen de functie bereikt worden. Een voorbeeld:

```
int  a,b;
void   func1 (void)
{
      int  a,x,y;
      // globale b is toegankelijk
      // globale a is niet toegankelijk, deze is door locale a afgedekt
      // locale x,y zijn toegankelijk
      // u is niet toegankelijk omdat deze lokaal hoort tot functie main
}
void   main (void)
{
      int u;
      // globale a, u zijn toegankelijk
      // locale u is toegankelijk
      //x,y niet toegankelijk omdat deze lokaal hoort  tot functie func1
}
```

Globale variabelen hebben een gedefinieerd geheugenbereik dat tijdens de totale programmaduur ter beschikking staat.

➔ Bij de start van het programma worden de globale variabelen met nul geïnitialiseerd.

Locale variabelen worden tijdens de berekening van een functie door de variabelen in het stack aangelegd. Dat betekent dat locale variabelen alleen in het geheugen bestaan tijdens de tijd dat de functie verwerkt wordt.

Als bij locale variabelen dezelfde naam gekozen wordt als bij een globale variabele, dan verbergt de locale variabele de globale variabele. Zolang zich het programma dan ophoudt in de functie waar de locale variabele met dezelfde naam gedefinieerd is, kan de globale variabele niet aangesproken worden.

Static variabelen

Bij locale variabelen kan de eigenschap **static** voor het datatype gezet worden.

```
void func1 (void)
{
      static int  a;
}
```

Static variabelen behouden in tegenstelling tot normale variabelen hun waarde ook als de functie verlaten wordt. Bij een volgende oproep van de functie heeft de statische variabele dezelfde inhoud als bij het verlaten van de functie. Omdat de inhoud van een **static** variabele bij de eerste toegang gedefinieerd is, worden statische variabelen net als globale ook bij de start van het programma met nul geïnitialiseerd.

5.2.5 Operatoren

Prioriteit van operatoren

Operatoren verdelen wiskundige termen in deeltermen. De operatoren worden dan in de volgorde van hun prioriteit (precedentie) geëvalueerd. Termen met operatoren van dezelfde prioriteit worden van links naar rechts berekend. Voorbeeld:

```
i= 2+3*4-5;      // resultaat 9 => eerst 3*4, dan +2, daarna -5
```

U kunt de volgorde van de bewerking beïnvloeden door haakjes te plaatsen. Haakjes hebben de grootste prioriteit. Als u het laatste voorbeeld strikt van links naar rechts wilt evalueren:

```
i= (2+3)*4-5;    // resultaat 15 => eerst 2+3, dan *4, daarna -5
```

Een opstelling van de prioriteiten vindt u in de pecedentietabel.

5.2.5.1 Rekenkundige operatoren

Alle rekenkundige operatoren met uitzondering van “modulo” zijn gedefinieerd voor integer en zwevende komma datatypes. Alleen modulo is beperkt tot één integer -datatype.

➔ U dient er op te letten dat in een term aan het cijfer 7 een integer datatype toegewezen wordt. Als u persé een getal van het datatype **float** wilt maken, dient u een decimale punt toe te voegen: 7.0.

Operator	Uitleg	Voorbeeld	Resultaat
+	**Optellen**	2+1 3.2+4	3 7.2
-	**Aftrekken**	2 – 3 22 - 1.1^{e}1	-1 11
*	**Vermenigvuldigen**	5*4	20
/	**Delen**	7 / 2 7.0 / 2	3 3.5
%	**Modulo**	15%4 17%2	3 1
-	**Neg. voorteken**	-(2+2)	-4

5.2.5.2 Bit –operatoren

Bit –operatoren zijn alleen toegestaan voor integer –datatypes.

Operator	Verklaring	Voorbeeld	Resultaat
&	**En**	0x0f & 3 0xf0 & 0x0f	3 0
|	**Of**	1 | 3 0xf0 | 0x0f	3 0xff
^	**exclusieve of**	0xff ^ 0x0f 0xf0 ^ 0x0f	0xf0 0xff
~	**Bit -invertering**	~0xff ~0xf0	0 0x0f

5.2.5.3 Bitschuif operatoren

Bitschuif operatoren zijn alleen toegestaan voor Integer datatypes. Bij een Bit-Shift operatie wordt er steeds aan het einde een 0 tussen geschoven.

Operator	Verklaring	Voorbeeld	Resultaat
<<	**Één bit naar links schuiven**	1 << 2 3 << 3	4 24
>>	**Één bit naar rechts schuiven**	0xff >> 6 16 >> 2	3 4

5.2.5.4 In –Decrement operatoren

Increment (toename) en decrement (afname) operatoren zijn alleen toegestaan voor variabelen met Integer datatypes.

Operator	Verklaring	Voorbeeld	Resultaat
variabele++	Waarde der variabelen, daarna variabele met één verhoogd (post-increment)	a++	a
Variabele - -	Waarde der variabelen, daarna variabele met één verlaagd (post-decrement)	a - -	a
++variabele	Waarde der variabelen met één verhoogd (pré –increment)	a++	a+1
- - variabele	Waarde der variabelen met één verlaagd (pré –decrement)	a - -	a-1

5.2.5.5 Vergelijkingsoperatoren

Vergelijkingsoperatoren zijn toegestaan voor **float** en integer datatypes.

Operator	Verklaring	Voorbeeld	Resultaat
<	Kleiner dan	1 < 2 2 < 1 2 < 2	1 0 0
>	Groter dan	-3 > 2 3 > 2	0 1
<=	Kleiner dan of gelijk	2 <= 2 3 <= 2	1 0
>=	Groter dan of gelijk	2 >= 3 3 >= 2	0 1
==	Gelijk	5 == 5 1 == 2	1 0
!=	Ongelijk	2 != 2 2 != 5	0 1

5.2.5.6 Logische operatoren

Logische operatoren zijn alleen toegestaan voor Integer datatypes. Elke waarde ongelijk aan **nul** geldt als logisch 1. De **nul** geldt als logisch 0.

Operator	Verklaring	Voorbeeld	Resultaat
&&	**Logisch En**	1 && 1 5 && 0	1 0
II	**Logisch Of**	0 II 0 1 II 0	0 1
!	**Logisch Niet**	!2 !0	0 1

5.2.6 Controlestructuren

Controlestructuren laten het toe om het programmaverloop in afhankelijkheid van termen, variabelen of invloeden te wijzigen.

5.2.6.1 Voorwaardelijke evaluatie

Met een voorwaardelijke evaluatie kunnen termen gemaakt worden die voorwaardelijk berekend worden. De formule is:

```
(  term1  )   ?  term2 :  term3
```

Het resultaat van deze term is term2 als *term1* niet gelijk aan 0 berekend is, anders is het resultaat term3.

Voorbeelden:

```
a  =  (i>5)  ?  i  : 0;

a  =  (i>b*2)  ?  i-5  :  b+1;

while(1>  ((x>y)  ?  x  :  y)  )  i++;
```

5.2.6.2 do .. while

Met een **do .. while** constructie kunnen, afhankelijk van een voorwaarde, aanwijzingen in een lus herhaald worden:

```
do aanwijzing while(  term  );
```

De aanwijzing of het aanwijzingsblok wordt uitgevoerd. Aan het eind wordt de term geëvalueerd. Als het resultaat niet gelijk is aan 0, leidt dit tot de herhaalde uitvoering van de aanwijzing. De hele procedure wordt herhaald tot de *term* de waarde 0 aanneemt.

Voorbeelden:

```
do
a=a+2;
while(a<10);
```

```
do
{
      a=a*;
      x=a;
}  while(a);
```

➔ Het wezenlijke verschil tussen de **do .. while** lus en de normale **while** lus is de omstandigheid dat in de **do .. while** lus de aanwijzing tenminste éénmaal uitgevoerd wordt.

break aanwijzing

Een **break** aanwijzing verlaat de lus, en de uitvoering van het programma start met de volgende aanwijzing na de **do .. while** lus.

continue aanwijzing

Bij de uitvoering van **continue** binnen een lus volgt er onmiddellijk een nieuwe berekening van de *term*. Afhankelijk van het resultaat wordt bij niet gelijk aan 0 de lus herhaald. Een uitkomst 0 breekt de lus af.

Voorbeeld:

```
do
{
      a++;
      if(a>10)  break;  //  breekt lus af
} while(1) // eindeloze lus
```

5.2.6.3 for

Een for lus wordt normaalgesproken gebruikt om een bepaald aantal lusdoorlopen te programmeren.

```
for(aanwijzing1;  term;  aanwijzing2)  aanwijzing3;
```

Als eerste wordt aanwijzing1 uitgevoerd, die normaalgesproken een initialisering bevat. Daarna volgt de evaluatie van de *term*. Als de *term* niet gelijk is aan 0 worden aanwijzing2 en aanwijzing3 uitgevoerd, en de lus wordt herhaald. Als de *term* een waarde heeft van 0, wordt de lus afgebroken. Net als bij andere lustypes kan bij aanwijzing3 in plaats van een aanwijzing ook een aanwijzingsblok gebruikt worden.

```
for (i=0;i<10;i++)
{
      if(i>a)  a=i;
      a - -;
}
```

➔ U dient er aan te denken dat de variabele i binnen de lus de waarden van 0 tot 9 doorloopt, en niet van 1 tot 10!

Als u een lus wilt programmeren die een andere stappenbreedte heeft, dient u aanwijzing2 overeenkomstig aan te passen:

```
for(i=0;i<100;i=i+3);      // de variabele i neemt nu toe in
                              drievoudige stappen
{
      a=5*i;
}
```

break aanwijzing

Een **break** aanwijzing verlaat de lus, en de uitvoering van het programma start met de volgende aanwijzing na de **for** lus.

continue aanwijzing

continue zorgt voor de directe nieuwe berekening van de *term*. Afhankelijk van het resultaat wordt bij niet gelijk aan 0 aanwijzing2 uitgevoerd en de lus wordt herhaald. Een resultaat van 0 breekt de lus af.

Voorbeeld:

```
for(i=0;i<10;i++)
{
      if(i==5)  continue;
}
```

5.2.6.4 goto

Ook wanneer het binnen een gestructureerde programmeertaal vermeden zal worden is het toch mogelijk om binnen een procedure met **goto** naar een label te springen:

```
// for lus met goto realiseert
void main(void)
{
      int a;
```

```
      a=0,
label0:
      a++;
      if(a<10) goto label0;
}
```

5.2.6.5 if .. else

Een **if** aanwijzing heeft de volgende syntax:

```
if(  term  )  Aanwijzing1;
else  Aanwijzing2;
```

Achter de **if** aanwijzing volgt tussen haakjes een wiskundige term. Als deze *term* bepaald als niet gelijk aan 0, dan wordt aanwijzing 1 uitgevoerd. U kunt met behulp van het **else** commandowoord een alternatieve aanwijzing2 definiëren, die dan uitgevoerd wordt, als de *term* als 0 berekend is. Het toevoegen van een **else** aanwijzing is een optie en hoeft niet te gebeuren.

Voorbeelden:

```
if(a==2)  b++;

if(x==y)  a=a+2;
else  a=a-2;
```

In plaats van een enkele aanwijzing kan ook een aanwijzingsblok gedefinieerd worden.

Voorbeelden:

```
if(x<y)
{
      c++;
      if(c==10)  c=0;
}
else  d - -;

if(x>y)
{
      a=b*5;
      b - -;
}
else
{
      a=b*4;
      y++;
}
```

5.2.6.6 switch

Als er, afhankelijk van de waarde van een term, verschillende commando's uitgevoerd moeten worden, dan is een **switch** aanwijzing zeer elegant:

```
Switch(  term  )
{
```

```
        case  constante_1;
                 aanwijzing_1;
        break;
        case  constante_2;
              aanwijzing_2;
        break;
        .
        .
        case  constante_n;
              aanwijzing_n;
        break;
        default:        //  default is optioneel
              aanwijzing_0;
};
```

De waarde van de *term* wordt berekend. Daarna springt de uitvoering van het programma naar de constante die overeenkomt met de waarde van de *term* en gaat daar verder met het programma. Als er geen constante overeenkomt met de waarde van de term, dan wordt de **switch** constructie verlaten.
Als er in een **switch** aanwijzing een **default** gedefinieerd is, dan worden de aanwijzingen na **default** uitgevoerd, als er geen constante is gevonden die met de waarde van de *term* overeenkomt.

Voorbeeld:

```
switch(a+2);
{
   case  1:
        b=b+2;
   break;

   case  5*5:
         b=b+2;
   break;

   case  100&0xf:
         b=b/c;
   break;

   default:
         b=b+2;
}
```

break aanwijzing

Een **break** verlaat de switch aanwijzing. Als u voor **case** de **break** weglaat, dan worden de aanwijzingen ook uitgevoerd als er naar de vorige **case** gesprongen wordt:

```
switch(a)
{
    case 1:
        a++;

    case 2:
        a++;       // wordt ook uitgevoerd bij een waarde van a==1
```

```
        case 3:
            a++;        // wordt ook uitgevoerd bij een waarde van a==1 of a==2
}
```

In dit voorbeeld worden alle drie “a++” aanwijzingen uitgevoerd als a gelijk is aan 1.

5.1.6.2 while

Met een **wile** aanwijzing kunnen afhankelijk van een voorwaarde aanwijzingen in een lus herhaald worden.

```
while(  term )  aanwijzing;
```

Eerst wordt de *term* bepaald. Als het resultaat niet gelijk is aan 0, dan wordt de aanwijzing uitgevoerd. Daarna vindt weer de berekening van de *term* plaats en de hele procedure wordt net zo lang herhaald tot de *term* de waarde 0 aanneemt. In plats van een enkele aanwijzing kan ook een aanwijzingsblok gedefinieerd worden.

Voorbeelden:

```
while(a<10)  a=a+2;

while(a)
{
        a=a*2;
        x=a;
}
```

break aanwijzing
Als er binnen de lus een **break** uitgevoerd wordt, dan wordt de lus verlaten en de uitvoering van het programma start met de volgende aanwijzing achter de **while** lus.

continue aanwijzing
Bij uitvoering van **continue** binnen een lus volgt er onmiddellijk een nieuwe berekening van de *term*. Afhankelijk van het resultaat wordt bij niet gelijk aan 0 de lus herhaald. Een uitkomst 0 breekt de lus af.

Voorbeeld:

```
while(1)        // eindeloze lus
{
        a++;
        if(a>10)  break;  //  breekt lus af
}
```

5.2.7 Functies

Om grotere programma’s te structureren worden ze in meerdere subfuncties verdeeld. Dit verhoogt niet alleen de leesbaarheid, maar maakt het tevens mogelijk programma – aanwijzingen die meervoudig voorkomen in functies samen te vatten.
Een programma bestaat steeds uit de functie “main” die als allereerste gestart wordt.

Daarna kunnen vanuit main andere functies oproepen worden. Een eenvoudig voorbeeld:

```
void  (func1 (void)
{
     //  aanwijzingen in functie func1
           .
     .
}
void  (main(void)
{
     //  de functie func1 wordt twee keer opgeroepen
     funct1();
     funct1();
}
```

Parameteroverdracht
Opdat functies flexibel gebruikt kunnen worden, kunt u ze parameteriseren. Hiervoor worden in de haakjes na de functienaam de parameters voor de functie gescheiden door komma's doorgegeven. U geeft net als in de variabelendeclaratie eerst het datatype en daarna de parameternaam aan. Als u geen parameters wilt doorgeven, dan schrijft u **void** tussen de ronde haakjes. Een voorbeeld:

```
void  funct1 (word param1, float  param2)
{
     Msg_WriteHex(param1); // de eerste parameter aangeven
     Msg_WriteFloat(param2);      // de tweede parameter aangeven
}
```

➔ Net als bij locale variabelen zijn overgenomen parameters alleen in de functie zelf zichtbaar.

Om de functie func1 met de parameters op te roepen schrijft u bij het oproepen de parameters in dezelfde volgorde als waarin ze bij func1 gedefinieerd zijn. Als de functie geen parameters krijgt, laat u de haakjes leeg.

```
void  main (void)
{
          word  a;
          float  f;
          funct1(128,12.0);      // u kunt numerieke constanten doorgeven …
          a=100;
          f=12.0;

          func1(a+28,f);  // of ook variabelen en zelfs numerieke termen
}
```

➔ U moet bij het oproepen van een functie steeds alle parameters aangeven. De volgende oproepen zouden ongeldig zijn:

```
func1();              // func1 krijgt 2 parameters!
func1(128);           // func1 krijgt 2 parameters!
```

Retourparameters
Het is niet alleen mogelijk parameters door te geven, een functie kan ook een retourwaarde hebben. Het datatype van deze waarde wordt bij de functiedefinitie voor de naam van de

functie aangegeven. Als u geen waarde wilt laten teruglopen, dan gebruikt u **void** als datatype.

```
int  func1 (int  a)
{
     return a-10;
}
```

De retourwaarde wordt binnen de functie met de aanwijzing **"return** *term*" aangegeven. Als u een functie van het type **void** heeft, kunt u de **return** aanwijzing ook zonder parameter gebruiken om de functie te verlaten.

Referenties

Omdat het niet mogelijk is arrays als parameter door te geven, kunt u uw toevlucht nemen tot arrays via referenties. Daarvoor schrijft u in de parameterdeclaratie van een functie een paar rechte haakjes achter de parameternaam:

```
int  Stringlength (char  str[ ])
{
          int  i;

          i=0;
          while(str[i])  i++; // herhaal zolang het teken niet nul is
          return(i);
}

void  main(void)
{
         int  len;
         char  text[15];

         text="hallo wereld";
         len=Stringlength(text);
}
```

In main wordt de referentie van tekst als parameter doorgegeven aan de functie Stringlength. Als u in een functie een normale parameter verandert, is deze verandering buiten deze functie niet zichtbaar. Bij referenties is dat anders. Via de parameter str kunt u in Stringlength de inhoud van tekst veranderen, omdat str slechts een referentie (aanwijzer) naar de array Variabele tekst is.

5.2.8 Tabellen

5.2.8.1 Operator voorrang

Rang	Operator
13	( )
12	++ -- ! ~ - (negatief voorteken)
11	* / %
10	+ -
9	<< >>
8	< <= > >=
7	== !=

6	&
5	^
4	|
3	&&
2	||
1	? :

5.2.8.2 Operatoren

	Wiskundige operatoren
+	Optellen
-	Aftrekken
*	Vermenigvuldigen
/	Delen
%	Modulo
-	Negatief voorteken

	Vergelijkende operatoren
<	Kleiner dan
>	Groter dan
<=	Kleiner dan of gelijk
>=	Groter dan of gelijk
==	Gelijk
!=	Ongelijk

	Bitschuifoperatoren
<<	Één bit naar links schuiven
>>	Eén bit naar rechts schuiven

	Toename-/Afname -operatoren
++	Post/Pre -toename
--	Post/Pre –afname

	Logische operatoren
&&	Logisch en
||	Logisch of
!	Logisch niet

	Bitoperatoren
&	En
|	Of
^	Exclusief of
~	Bit –invertering

5.2.8.3 Gereserveerde woorden

De volgende woorden zijn **gereserveerd** en kunnen niet als naam voor kenmerken gebruikt worden:

break	**byte**	**case**	**char**	**continue**
default	**do**	**else**	**false**	**float**
for	**goto**	**if**	**int**	**return**
signed	**static**	**switch**	**true**	**unsigned**
void	**while**	**word**		

5.3 BASIC

De tweede programmeertaal voor de C-Control Pro Mega module is BASIC. De compiler vertaald de programmeertaal BASIC commando's naar een bytecode, die van de interpreter van de C-Control Pro verwerkt wordt. De taalomvang, van het hier genomde BASIC dialect, komt voor het grootste gedeelte overeen met de industriestandaard van grote anbieders van software. Volgende taalconstructies zijn er niet:

- Objectorienteerde progrmmering
- Structuren
- Constanten

Uitgebreide programma- voorbeelden vindt u in de map "Demoprogramma's" die met de ontwikkelomgeving geinstalleerd werd. Daar zijn op bijna alle takengebieden van de C-Control Pro module voorbeeldoplossingen.

De volgende paragrafen bevatten een systematische kennismaking in de syntax en semantik van C-Control Pro BASIC.

5.3.1 Programma

Een programma bestaat uit een hoeveelheid aanwijzingen (zoals bijv. "a=5;"), die over verschillende functies verdeeld zijn. De startfunctie die in elk programma aanwezig moet zijn, is de functie "main()". Een klein programma dat een getal in het uitvoervenster drukt:

```
Sub main ()
      Msg_WriteInt(42);  //  Het antwoord op alles
End Sub
```

Projecten

Men kan een programma verdelen over meerdere bestanden die in een project (zie projectbeheer) samengevat zijn. Naast deze bestanden kunt u bibliotheken aan een project toevoegen, die functies ter beschikking stellen die door het programma gebruikt worden.

5.3.2 Aanwijzingen

Aanwijzing

Een aanwijzing bestaat uit meerdere gereserveerde commandowoorden, indicatoren en operatoren, die door het einde van de regel afgesloten wordt. Om verschillende elementen van een aanwijzing te scheiden, bestaat tussen de aparte aanwijzingselementen een tussenruimte, in het Engels ook "*whitespaces*" genoemd.
Onder tussenruimte worden verstaan spaties, tabs en regeldoorvoer ("C/R en LF"). Daarbij maakt het niet uit of de tussenruimte wordt gevormd door één of meerdere *"whitespaces*".

Eenvoudige aanwijzing:

```
a =    5;
```

➔ Een aanwijzing hoeft niet persé compleet in een regel te staan. Omdat ook regeldoorvoeren tot de tussenruimte horen, is het legitiem om een aanwijzing over meerdere regels te verdelen.

```
If a=5 _ ' Aanwijzing over 2 regels
a=a+10
```

➔ Er kunnen ook meer dan één aanwijzing in een regel geplaatst worden. Het teken ":" (dubbele punt) scheidt dan de enkele aanwijzingen. Echter, in verband met de leesbaarheid, moet men deze optie niet vaak gebruiken.

```
a=1 : b=2 : c=3
```

Commentaren
Een commentaar wordt met een enkele aanhalingsteken begonnen en met het regeleinde afgesloten. De tekst in de commentaren wordt door de compiler genegeerd.

```
' Een commentaarregel
```

Indicatoren

Indicatoren zijn de namen van functies of variabelen.

- Geldige tekens zijn de letters (A-Z, a-z), de cijfers (0 – 9) en de liggende streep (' _')
- Een indicator begint steeds met een letter
- Er wordt verschil gemaakt tussen hoofd – en kleine letters
- Gereserveerde woorden zijn niet toegestaan als indicator
- De lengte van indicatoren is niet beperkt

Rekenkundige termen

Een rekenkundige term is een hoeveelheid waarden, die met operatoren verbonden zijn. Waarden betekenen in deze samenhang getallen, variabelen en functies.

Een eenvoudig voorbeeld:

```
2 + 3
```

Hierbij worden de getallen 2 en 3 gekoppeld d.m.v. de operator "+". Een rekenkundige term vertegenwoordigt weer een waarde. Hier is de waarde 5.

Andere voorbeelden:

```
a - 3

b + f(5)

2 + 3 * 6
```

Volgens "punt voor streep" wordt hier eerst 3 x 6 uitgerekend en daarna 2 er bij opgeteld. Deze voorrang van operatoren heet bij operatoren precedent. U vindt een opsomming van de prioriteiten in de precedent tabel.

➔ Ook vergelijkingen zijn wiskundige termen. De vergelijkingsoperatoren geven als resultaat een waarheidswaarde van “1”of “0”, afhankelijk van of de vergelijking correct was. De term “3 < 5” geeft de waarde “1” (waar; true).

Constante termen

Een term of delen van een term kan/kunnen constant zijn. Deze deeltermen kunnen al tijdens de compiler –looptijd berekend worden.

Zo wordt bijv.

```
12 + 123 – 15
```

door de compiler samengevat tot

```
120
```

Soms moeten termen constant zijn opdat ze geldig zijn. Zie bijv. declarering van array variabelen.

5.3.3 Datatypes

Waarden bezitten steeds een bepaald datatype. De integerwaarden (gehele getallen waarden) hebben in BASIC een 8 of 16 bit breed datatype, getallen met floating point zijn altijd 4 byte lang.

Datatypee	Voorteken	Waardebereik	Bit
Char	Ja	-128 … +127	8
Byte	Nee	0 … 255 8	8
Integer	Ja	-32768 … +32767	16
Word	Nee	0 … 65535	16
Single	Ja	$\pm 1.175^{e}$-38 to $\pm 3.402^{e}$38	32

Strings

Er is geen expliciet “String” datatype. Een string is gebaseerd op een character array. U moet de grootte van de array dusdanig kiezen, zodat alle tekens van de string in het character array passen. Bovendien is er ruimte nodig voor een termineringsteken (decimale nul), om het einde van de tekenketen aan te geven.

Type –convertering

Bij wiskundige termen gebeurt het heel vaak dat aparte waarden niet van hetzelfde type zijn. Zo zijn de datatypes in de volgende term gemengd (**a** is integer variabele).

```
a + 5.5
```

In dit geval wordt **a** eerst geconverteerd naar het datatype **Single** en daarna wordt er 5.5 bij opgeteld. Het datatype van het resultaat is eveneens **Single**. Bij de typeconvertering gelden

de volgende regels:

- Als bij de verbinding van twee 8 bit of 16 bit integer- waarden één van beide datatypes van een voorteken is voorzien, dan is ook het resultaat van de term van een voorteken voorzien.

- Als één van beide operandi van het type **Single** is, dan is het resultaat eveneens van het type **Single**. Als één van de beide operandi een 8 bit of 16 bit datatype heeft, dan wordt deze voor de operatie omgevormd tot een single datatype.

5.3.4 Variabelen

Variabelen kunnen verschillende waarden aannemen, afhankelijk van het datatype waarmee ze gedefinieerd zijn. Een variabele –definitie ziet er als volgt uit:

```
Dim variabelennaam As type
```

Als u meerdere variabelen van hetzelfde type wilt definiëren, kunt u meerdere variabelen-namen door een komma gescheiden aangeven:

```
Dim naam1, naam2, naam3, A Integer
```

Als type zijn toegestaan: **Char, Byte, Integer, Word, Single**

Voorbeelden:

```
Dim a As Integer
Dim i,j As Integer
Dim xyz As Single
```

Aan integer- variabelen kunnen getalwaarden decimaal of als hexgetal toegewezen worden. Voor een hexgetal worden voor het getal de letters **"&H"** gezet. Bovendien mag men zoals bij C hexadecimale getallen met een prefix "**0x**" beginnen.Bij variabelen met een van voorteken voorzien datatype kunnen negatieve decimale getallen toegewezen worden door een minteken voor het getal te plaatsen.

Voorbeelden:

```
Dim a As Word
Dim i,j As Integer
a=&H3ff
i=15
j=-22
a=0x3ff
```

Getallen met zwevende komma (datatype **Single**) mogen een decimale punt en een exponent bevatten:

```
Dim x,y As Single
x=5.70
y=2.3e+2
x=-5.33e-1
```

sizeof Operator

Met de operator **sizeof()** kan het aantal bytes bepaald worden die een variabele in het geheugen inneemt.

Voorbeeld:

```
Dim s As Integer
Dim f As Single
s=SizeOf(f) ' de waarde van s is 4
```

➔ Bij arrays wordt ook alleen de bytelengte van het basis –datatype als uitkomst gegeven. U moet de waarde met het aantal elementen vermenigvuldigen om het geheugenverbruik van de array te berekenen.

Array variabelen

Als u achter de naam bij de variabelen –definitie tussen ronde haakjes een getalswaarde schrijft, dan heeft u een array gedefinieerd. Een array legt de plaats voor de gedefinieerde variabele meervoudig in het geheugen vast. Bij de voorbeelddefinitie:

```
Dim x(10) As Integer
```

wordt voor de variabele x de 10-voudige geheugenplaats vastgelegd. De eerste geheugenplaats kan aangesproken worden met x(0)], de tweede met x(1)], de derde met x(2), …tot x(9). U mag bij de definitie natuurlijk ook andere indexgroottes kiezen. De beperking is alleen de RAM geheugenplaats van de C-Control Pro.
U kunt ook meerdimensionale arrays declareren, waarin nog meer rechte haakjes bij de variabelen –definitie toegevoegd worden:

```
Dim x(3,4) As Integer ' Array met 3*4 invoeren
Dim y(2,2,2) As Integer ' Array met 2*2*2 invoeren
```

➔ Arrays mogen in BASIC maximaal 16 indices (dimensies) hebben. De maximale waarde voor een index is 65535. De indices van de arrays zijn altijd op nul gebaseerd, d.w.z. elke index begint met 0.

➔ Er vindt tijdens het lopen van het programma geen controle plaats of de gedefinieerde indexgrens van een array is overschreden. Als de index tijdens de programmabewerking te groot wordt, neemt het programma zijn toevlucht tot vreemde variabelen en is de kans groot dat het programma ‘crasht’.

Strings

Er is geen specifieke “String” datatype. Een string is gebaseerd op een array van het datatype **Char**. U moet de grootte van de array zo kiezen, dat alle tekens van de string in de character array passen. Bovendien is er plaats nodig voor een termineringsteken (decimale nul), om het eind van de tekenketen aan te geven.

Voorbeeld van een tekenketen met maximaal 20 tekens:

```
Dim str1(21) As Char
```

Als uitzondering mag men aan **Char** arrays tekenketens toewijzen. Daarbij wordt de tekenketen tussen aanhalingstekens gezet.

```
str1="Hallo wereld!"
```

➔ Er kan geen String aan grotere **Char** array toegewezen worden. Echter zijn er trucs voor gevorderden:

```
Dim str_array(3,40) As Char
Dim Single_str(40) As Char

Single_str="A String"
Str_StrCopy(str_array,Single_str,40); // kopieert Single_str in de
tweede String
```

Dit functioneert omdat met een afstand van 40 tekens achter de string in str_array de ruimte voor de tweede string ligt.

Zichtbaarheid van variabelen

Als variabelen buiten de functies gedeclareerd worden, hebben ze een globale zichtbaarheid. D.w.z., ze zijn vanuit elke functie aanspreekbaar. Declaraties van variabelen binnen functies produceren locale variabelen. Locale variabelen kunnen alleen binnen de functie bereikt worden. Een voorbeeld:

```
Dim a,b As Integer

Sub func1()
    Dim a,x,y As Integer
    // globale b is toegankelijk
    // globale a is niet toegankelijkbar, deze is door locale a afgedekt
    // locale x,y zijn toegankelijk
    // u is niet toegankelijk omdat deze locaal hoort tot functie main
End Sub

Sub main()
    Dim u As Integer
    // globale a,b zijn toegankelijk
    // locale u is toegankelijk
    // x,y niet toegankelijk omdat deze locaal hoort tot func1
End Sub
```

Globale variabelen hebben een gedefinieerd geheugenbereik dat tijdens de totale programmaduur ter beschikking staat.

➔ Bij de start van het programma worden de globale variabelen met nul geïnitialiseerd.

Locale variabelen worden tijdens de berekening van een functie door de variabelen in het stack aangelegd. Dat betekent dat locale variabelen alleen in het geheugen bestaan tijdens de tijd dat de functie verwerkt wordt.

Als bij locale variabelen dezelfde naam gekozen wordt als bij een globale variabele, dan verbergt de locale variabele de globale variabele. Zolang zich het programma dan ophoudt in de functie waar de locale variabele met dezelfde naam gedefinieerd is, kan de globale variabele niet aangesproken worden.

Static variabelen

Bij locale variabelen kan de eigenschap **static** voor het datatype gezet worden.

```
Sub func1()
    Static a As Integer
End Sub
```

Static variabelen behouden in tegenstelling tot normale variabelen hun waarde ook als de functie verlaten wordt. Bij een volgende oproep van de functie heeft de statische variabele dezelfde inhoud als bij het verlaten van de functie. Omdat de inhoud van een **Static** variabele bij de eerste toegang gedefinieerd is, worden statische variabelen net als globale ook bij de start van het programma met nul geïnitialiseerd.

5.3.5 Operatoren

Prioriteiten van operatoren

Operatoren verdelen wiskundige termen in deeltermen. De operatoren worden dan in de volgorde van hun prioriteit (precedentie) bepaald. Termen met operatoren van dezelfde prioriteit worden van links naar rechts berekend. Voorbeeld:

```
i= 2+3*4-5;        // resultaat 9 => eerst 3*4, dan +2, daarna -5
```

U kunt de volgorde van de bewerking beïnvloeden door haakjes te plaatsen. Haakjes hebben de grootste prioriteit. Als u het laatste voorbeeld strikt van links naar rechts wilt evalueren:

```
i= (2+3)*4-5;      // resultaat 15 => eerst 2+3, dan *4, daarna -5
```

Een opstelling van de prioriteiten vindt u in de pecedentietabel.

5.3.5.1 Rekenkundige operatoren

Alle rekenkundige operatoren met uitzondering van “modulo” zijn gedefinieerd voor integer en zwevende komma datatypes. Alleen modulo is beperkt tot één integer -datatype.

➔ U dient er op te letten dat in een term aan het cijfer 7 een integer datatype toegewezen wordt. Als u persé een getal van het datatype **Single** wilt maken, dient u een decimale punt toe te voegen: 7.0.

Operator	Verklaring	Voorbeeld	Resultaat
+	**Optellen**	2+1 3.2+4	3 7.2
-	**Aftrekken**	2 – 3 22 - 1.1^{e}1	-1 11
*	**Vermenigvuldigen**	5*4	20
/	**Delen**	7 / 2 7.0 / 2	3 3.5
Mod	**Modulo**	15 Mod 4 17 Mod 2	3 1
-	**Neg. voorteken**	-(2+2)	-4

5.3.5.2 Bit –operatoren

Bit –operatoren zijn alleen toegestaan voor integer –datatypes.

Operator	Verklaring	Voorbeeld	Resultaat
And	**En**	&H0f And 3 &Hf0 And &H0f	3 0
Or	**Of**	1 Or 3 &Hf0 Or &H0f	3 &Hff
Xor	**exclusieve of**	&Hff Xor &H0f &Hf0 Xor &H0f	&Hf0 &Hff
Not	**Bit -invertering**	Not &Hff Not &Hf0	0 &H0f

5.3.5.3 Bitschuif operatoren

Bitschuif operatoren zijn alleen toegestaan voor Integer datatypes. Bij een Bit-Shift operatie wordt er steeds aan het einde een 0 tussen geschoven.

Operator	Verklaring	Voorbeeld	Resultaat
<<	**Één bit naar links schuiven**	1 << 2 3 << 3	4 24
>>	**Één bit naar rechts schuiven**	&Hff >> 6 16 >> 2	3 4

5.2.5.5 Vergelijkingsoperatoren

Vergelijkingsoperatoren zijn toegestaan voor **Single** en integer datatypes.

Operator	Verklaring	Voorbeeld	Resultaat
<	Kleiner dan	1 < 2 2 < 1 2 < 2	1 0 0
>	Groter dan	-3 > 2 3 > 2	0 1
<=	Kleiner dan of gelijk	2 <= 2 3 <= 2	1 0
>=	Groter dan of gelijk	2 >= 3 3 >= 2	0 1
=	Gelijk	5 = 5 1 = 2	1 0
<>	Ongelijk	2 <> 2 2 <> 5	0 1

5.3.6 Controlestructuren

Controlestructuren laten het toe om het programmaverloop in afhankelijkheid van termen, variabelen of invloeden te wijzigen.

5.3.6.1 Do Loop While

Met een **Do… Loop While** constructie kunnen, afhankelijk van een voorwaarde, aanwijzingen in een lus herhaald worden:

```
Do
       aanwijzing
Loop While  term
```

De aanwijzing wordt uitgevoerd. Aan het eind wordt de term geëvalueerd. Als het resultaat niet gelijk is aan 0, leidt dit tot de herhaalde uitvoering van de aanwijzing. De hele procedure wordt herhaald tot de *term* de waarde 0 aanneemt.

Voorbeelden:

```
Do
     a=a+2;
Loop While a<10
```

```
Do
      a=a*2
      x=a
Loop While a
```

➔ Het wezenlijke verschil tussen de **Do Loop while** lus en de normale **Do While** lus is de omstandigheid dat in de **Do Loop While** lus de aanwijzing tenminste éénmaal uitgevoerd wordt.

Exit aanwijzing

Een **Exit** aanwijzing verlaat de lus, en de uitvoering van het programma start met de volgende aanwijzing na de **Do Loop While** lus.

Voorbeeld:

```
Do
     a=a+1
     If a>10  Then
         Exit   ' breekt lus af
     End If
Loop While 1    ' eindeloze lus
```

5.3.6.2 Do While

Met een **while** aanwijzing kunnen, afhankelijk van een voorwaarde, aanwijzingen in een lus herhaald worden:

```
Do While term
      Aanwijzingen
End While
```

Eerst wordt de *term* geëvalueerd. Als het resultaat niet gelijk is aan 0, leidt dit tot de uitvoering van de aanwijzing. Daarna wordt de berekening van de *term* opnieuw uitgevoerd en de hele procedure wordt herhaald tot de *term* de waarde 0 aanneemt.

Voorbeelden:

```
Do While a<10
    a=a+2
End While

Do While a
    a=a*2
    x=a
End While
```

Exit aanwijzing

Als er binnen een lus **Exit** uitgevoerd wordt, dan wordt de lus verlaten en de uitvoering van het programma start met de volgende aanwijzing naa de **While** lus.

Voorbeeld:

```
Do While 1 ' eindeloze lus
      a=a+1
      If a>10 Then
          Exit ' breekt lus af
      End If
End While
```

5.3.6.3 For Next

Een **For Next** lus wordt normaalgesproken gebruikt om een bepaald aantal lusdoorlopen te programmeren.

```
For Tellervariable=Startwaarde To Eindwaarde Step stapbreedte
    Aanwijzingen
Next
```

De tellervariabele wordt op de startwaarde gezet en daarna worden de aanwijzingen zo vaak herhaald tot de eindwaarde bereikt is. Bij elke lussendoorloop verhoogt zich de waarde van de tellervariabele met de stapbreedte, deze mag ook negatief zijn. Het aangeven van de stapbreedte achter de eindwaarde is optioneel. Als de stapbreedte niet wordt aangegeven dan heeft deze de waarde 1.

➔ Omdat bij de **For Next** lus de extreme waarde bepaald wordt moet de tellervariabele van het type integer zijn.

Voorbeelden

```
For i=1 To 10
      If i>a Then
            a=i
      End If
      a=a-1
Next

For i=1 To 10 Step 3 ' verhoog i in 3-voudige stappen
      If i>3 Then
            a=i
      End If
      a=a-1
Next
```

➔ Op deze plaats nogmaals de opmerking, arrays zijn steeds op nul gebaseerd. Een **For Next** lus moet daarom bij een array toegang eerst van 0 to 9 lopen.

Exit aanwijzing

Een **Exit** aanwijzing verlaat de lus, en de uitvoering van het programma start met de volgende aanwijzing na de **For** lus.

Voorbeeld:

```
For i=1 To 10
      If i=6 Then
            Exit
      End If
Next
```

5.3.6.4 Goto

Ook wanneer het binnen een gestructureerde programmeertaal vermeden zal worden is het toch mogelijk om binnen een procedure met **Goto** naar een label te springen. Om een label te kenmerken wordt het commando **Lab** voor de labelnaam gezet.

```
' For lus met Goto maakt
Sub main()
      Dim a As Integer
      a=0
Lab label1
      a=a+1
      If a<10 Then
            Goto label1
      End If
End Sub
```

5.3.6.5 If .. Else

Een **If** aanwijzing heeft de volgende syntax:

```
If term1 Then
      aanwijzingen1
ElseIf term2 Then
      aanwijzingen2
Else
      Aanwijzingen3
End If
```

Achter de **if** aanwijzing volgt een rekenkundige term. Als deze *term* bepaald wordt als niet gelijk aan 0, dan wordt aanwijzingen1 uitgevoerd. U kunt met behulp van het **Else** commandowoord een alternatieve aanwijzingen2 definiëren, die dan uitgevoerd wordt, als de *term* als 0 berekend is. Het toevoegen van een **Else** aanwijzing is een optie en hoeft niet te gebeuren.

Als in de **Else**- tak direct weer een **If**- aanwijzing staat is het mogelijk met **ElseIf** direct weer een **If** te activeren. Hierbij hoeft de nieuwe **If** niet in de **Else**- blok staan, en de brontekst blijft overzichtelijk.

Voorbeelden:

```
If a=2 Then
      b=b+1
End If

If x=y Then
      a=a+2
Else
      a=a-2
End If

If a<5 Then
      a=a-2
ElseIf a<10 Then
      a=a-1
Else
      a=a+1
End If
```

5.3.6.6 Select Case

Als afhankelijk van de waarde van een term verschillende commando's uitgevoerd moeten worden is een Select Case aanwijzing heel geschikt:

```
Select Case term
      Case constante_1
            Aanwijzingen_1
      Case constante_2
            Aanwijzingen_2
      .
      .
      Case constante_n
            Aanwijzingen_n
      Else ' Else is optioneel
            Aanwijzingen
End Case
```

De waarde van *term* wordt berekend. Daarna sprint de programmauitvoering naar constante, die overeenkomt met de waarde van de term, en laat het programma daar verder lopen. Als er geen constante overeenkomt met de termwaarde, dan wordt de **Select Case** constructie verlaten.

Als in een **Select Case** aanwijzing een **Else** gedefinieerd is, dan worden de aanwijzingen na **Else** uitgevoerd, wanneer geen constante gevonden wordt, die overeenkomt met de waarde van de *term*.

Voorbeeld:

```
Select Case a+2
      Case 1
            b=b*2
      Case 5*5
            b=b+2
      Case 100 And &Hf
            b=b/c
      Else
            b=b+2
End Case
```

➔ In CompactC worden de aanwijzingen na de **case**- aanwijzing verder uitgevoerd tot er een **break** komt of de **switch** aanwijzing verlaten wordt. Dit is in BASIC anders: hier wordt de verwerking van de commando's na **Case** afgebroken zodra een **Case** aanwijzing bereikt wordt.

5.3.7 Functies

Om grotere programma's te structureren worden ze in meerdere subfuncties verdeeld. Dit verhoogt niet alleen de leesbaarheid, maar maakt het tevens mogelijk programma – aanwijzingen die meervoudig voorkomen in functies samen te vatten.
Een programma bestaat steeds uit de functie "main" die als allereerste gestart wordt.
Daarna kunt u vanuit main andere functies oproepen. Een eenvoudig voorbeeld:

```
Sub func1()
    ' Aanwijzingen in functie func1
```

```
End Sub

Sub main()
      ' de functie func1 wordttwee keer opgeroepen
      func1()
      func1()
End Sub
```

Parameteroverdracht

Opdat functies flexibel gebruikt kunnen worden, kunt u ze parameteriseren. Hiervoor worden in de haakjes na de functienaam de parameters voor de functie gescheiden door komma's doorgegeven. U geeft net als in de variabelendeclaratie eerst de parameternaam aan en daarna het datatype. Als u geen parameters wilt doorgeven, dan blijven de haakjes leeg. Een voorbeeld:

```
Sub func1(param1 As Word, param2 As Single)
      Msg_WriteHex(param1) ' de eerste parameter aangeven
      Msg_WriteFloat(param2) ' de tweede parameter aangeven
End Sub
```

➔ Net als bij locale variabelen zijn ingevoerde parameters alleen in de functie zelf zichtbaar.

Om de functie func1 met de parameters op te roepen schrijft u bij het oproepen de parameters in dezelfde volgorde zoals deze bij func1 gedefinieerd zijn. Als de functie geen parameters krijgt, laat u de haakjes leeg.

```
Sub main()
      Dim a As Word
      Dim f As Single

      func1(128,12.0) ' men kan numerieke constanten doorgeven ...
      a=100
      f=12.0
      func1(a+28,f) ' of ook variablen en zelfs numerieke termen
End Sub
```

➔ U moet bij het oproepen van een functie steeds alle parameters aangeven. De volgende oproepen zouden ongeldig zijn:

```
func1()          ' func1 krijgt 2 parameter!
func1(128)       ' func1 krijgt 2 parameter!
```

Return parameters

Het is niet alleen mogelijk parameters door te geven, een functie kan ook een retourwaarde hebben. Het datatype van deze waarde wordt bij de functiedefinitie na de parameterlijst van de functie aangegeven.

```
Sub func1(a As Integer) As Integer
      Return a-10
End Sub
```

De retourwaarde wordt binnen de functie met de aanwijzing "**Return** *term*" aangegeven. Als een functie geen retourwaarde heeft, kunt u de **Return** aanwijzing ook zonder parameter gebruiken om de functie te verlaten.

Referenties

Omdat het niet mogelijk is, arrays als parameter door te geven, kunt u uw toevlucht nemen tot arrays via referenties. Daarvoor schrijft u in de parameterdeclaratie van een functie "**ByRef**" voor de parameternaam:

```
Sub StringLength(ByRef str As Char) As Integer
      Dim i As Integer

      i=0
      Do While str(i)
            i=i+1 ' herhaal zolang het teken nu is
      End While
      Return i
      End Sub

Sub main()
      Dim Len As Integer
      Dim Text(15) As Char

      Text="hallo wereld"
      Len=StringLength(Text)
End Sub
```

In main wordt de referentie van tekst als parameter doorgegeven aan de functie Stringlength. Als u in een functie een normale parameter verandert, is deze verandering buiten deze functie niet zichtbaar. Bij referenties is dit anders. Via de parameter *str* kunt u in Stringlength de inhoud van *text* veranderen, omdat *str* slechts een referentie (aanwijzer) naar de Array Variabele *text* is.

5.3.8 Tabellen

5.3.8.1 Operator precedentie (voorrang)

Rang	Operator
10	( )
9	- (negatief voorteken)
8	* /
7	Mod
6	+ -
5	<< >>
4	= <> < <= > >=
3	Not
2	And
1	Or Xor

5.3.8.2 Operatoren

	Wiskundige operatoren
+	Optellen
-	Aftrekken
*	Vermenigvuldigen
/	Delen
Mod	Modulo
-	Negatief voorteken

	Vergelijkende operatoren
<	Kleiner dan
>	Groter dan
<=	Kleiner dan of gelijk
>=	Groter dan of gelijk
=	Gelijk
<>	Ongelijk

	Bitschuifoperatoren
<<	Één bit naar links schuiven
>>	Eén bit naar rechts schuiven

	Bitoperatoren
And	En
Or	Of
Xor	Exclusieve of
Not	Bit –invertering

5.3.8.3 Gereserveerde woorden

De volgende woorden zijn **gereserveerd** en kunnen niet als naam voor kenmerken gebruikt worden:

And	As	ByRef	Byte	Case
Char	Dim	Do	Else	ElseIf
End	Exit	False	For	Goto
If	Integer	Lab	Loop	Mod
Next	Not	Opc	Or	Return
Select	Single	SizeOf	Static	Step
Sub	Then	To	True	While
Word	Xor			

5.4 Bibliotheken

In dit gedeelte van het handboek worden alle bijgeleverde hulpfuncties beschreven, waarmee het voor de gebruiker mogelijk is om toegang te krijgen op de harware. In het begin wordt voor elke functie de syntax voor CompactC en BASIC getoond. Daarna volgt de beschrijving van de functie en de gebruikte parameters.

5.4.1 Interne functies

Opdat de compiler de interne functies die in de interpreter aanwezig zijn kan herkennen, moeten deze functies in de bibliotheek "IntFunc_Lib.cc" gedefinieerd zijn. Als deze bibliotheek niet ingepakt is, dan kunnen er geen uitvoeren van het programma gedaan worden. Een typische invoer in "IntFunc_Lib.cc" ziet er bijv. zo uit:

```
void  Msg_WriteHex$Opc(0x23) (Word  val);
```

Deze definitie zegt, dat de functie ("Msg_WriteHex") in de interpreter met een sprongvector van 0x23 opgeroepen wordt, en er als parameter een word naar de stack doorgegeven moet worden.

➔ Veranderingen in de bibliotheek "IntFunc_Lib.cc" kunnen er toe leiden, dat de daar gedeclareerde functies niet meer correct uitgevoerd kunnen worden!

5.4.2 AbsDelay

Algemene functies .

Syntax

```
void AbsDelay(word ms);
Sub AbsDelay(ms As Word);
```

Beschrijving

De functie AbsDelay() wacht een bepaald aantal milliseconden.

➔ De functie werkt weliswaar heel nauwkeurig, maar onderbreekt niet alleen de bewerking van de actuele thread, maar laat de Bytecode interpreter in zijn geheel wachten. Interrupts worden weliswaar geregistreerd, maar de interruptroutines worden in deze tijd niet verwerkt, omdat ook daarvoor de Bytecode interpreter nodig is.

➔ Bij het werken met threads moet steeds Thread Delay en niet AbsDelay gebruikt worden. Als er toch bijv. een AbsDelay(1000) gebruikt wordt, leidt dit tot het volgende effect: opdat de thread pas na 5000 cycli (default waarde) naar de volgende thread wisselt, zou de thread 5000*1000ms (5000sec.) lopen tot de volgende thread zou kunnen werken.

Parameter

ms	Wachttijd in ms

5.4.3 Analoge comparator

De analoge comparator maakt het mogelijk twee signalen te vergelijken. Het resultaat van dieze vergelijking wordt of als “0” of “1” teruggegeven.

5.4.3.1 Acomp

Acomp Functies Voorbeeld

Syntax

```
void AComp(byte mode);
Sub AComp(mode As Byte);
```

Beschrijving

De analoge comparator maakt het mogelijk twee analoge signalen te vergelijken. Het resultaat van deze vergelijking wordt of als “0” of als “1” teruggegeven (uitgang van de comparator). De negatieve ingang is **Mega32**: AIN1 (Poort B.3), **Mega128**: AIN1 (PoortE.3).. De positieve ingang kan of **Mega32**: AIN0 (Poort B.22, **Mega128**: AIN0 (PoortE.2) zijn, of een interne referentie-spanning van 1,22V.

Parameter

`mode` Werkmodus

Moduswaarden:

0x00	Externe ingangen (+)AIN0 en (-)AIN1 worden toegepast
0x40	Externe ingang (-)AIN1 en interne referentiespanning worden toegepast
0x80	Analoge comparator wordt uitgeschakeld

5.4.3.2 AComp voorbeeld

Voorbeeld: gebruik van de analoge comparator

```
// AComp: analoge comparator
// Mega32:  ingang (+) PB2 (PortB.2) resp. band gap reference 1,22V
//          ingang (-) PB3 (PortB.3)
// Mega128: ingang (+) PE2 (PortE.2) resp. band gap reference 1,22V
//          ingang (-) PE3 (PortE.3)
// noodzakelijke Library: IntFunc_Lib.cc
// De functie AComp geeft de waarde van de comparator terug.
// Is de spanning op de ingang PB2/PE2 groter dan op de ingang PB3/PE3
// heeft de functie AComp de waarde 1.
// Mode:
// 0x00 externe ingangen (+)AIN0 en (-)AIN1 worden toegepast
// 0x40 externe ingang (-)AIN1 en interne Referentiespanning worden
// toegepast
// 0x80 analoge comparator wordt uitgeschakeld
// De oproep kan met de parameter 0 (beide ingangen worden toegepast)
// of 0x40 (interne referentiepanning op (+) ingang, externe ingang
// PB3/PE3) gebeuren.


//-------------------------------------------------------------------------
// Hoofdprogramma
//
void main(void)
{
      while (true)
{
            if (AComp(0x40)==1)      // ingang (+) band gap reference 1,22V
            {
                  Msg_WriteChar('1'); // uitvoer: 1
            }
            else
            {
                  Msg_WriteChar('0'); // uitvoer: 0
            }
            // De comparator wordt alle 500ms gelezen en uitgegeven
            AbsDelay(500);
       }
}
```

5.4.4 Analoge- digitale- omvormer

De microcontroller beschikt over een analoog-digitaal-omvormer met een resolutie van 10 bit. Dat betekent dat gemeten spanningen als gehele getallen van 0 tot 1023 weergegeven worden. De referentiespanning voor de ondergrens is het GND-niveau, dus 0V. De referentiespanning voor de bovengrens kan gekozen worden.

- externe referentiespanning
- AVCC met condensator op AREF
- Interne spanningsreferentie 2,56V met condensator op AREF

Analoge ingangen ADC0 … ADC7, ADC, BG, ADC GND

Als ingangen voor de ADC staan de ingangen ADC0 … ADC7 (poort A.0 tot A.7 bij de **Mega32**, poort F.0 tot F.7 bij de **Mega128**), een interne band gap (1,22V) of GND (0V) ter beschikking. ADC_BG en ADC_GND kunnen gebruikt worden voor het controleren van de ADC.

Als x een digitale meetwaarde is, dan wordt de desbetreffende spanningswaarde u als volgt berekend:

*u = x * referentiespanning / 1024*

Als de externe referentiespanning 4,096V bedraagt, bijv. opgewekt door een referentiespanning –IC, dan komt een verschil van één bit van de gedigitaliseerde meetwaarde overeen met een spanningsverschil van 4mV of:

*u = x * 0,004V*

Verschil -ingangen

ADC22x10	Verschil –ingangen ADC2, ADC2, versterking 10	; offsetmeting
ADC23x10	Verschil –ingangen ADC2, ADC3, versterking 10	
ADC22x200	Verschil –ingangen ADC2, ADC2, versterking 200	; offsetmeting
ADC23x200	Verschil –ingangen ADC2, ADC3, versterking 200	
ADC20x1	Verschil –ingangen ADC2, ADC0, versterking 1	
ADC21x1	Verschil –ingangen ADC2, ADC1, versterking 1	
ADC22x1	Verschil –ingangen ADC2, ADC2, versterking 1	; offsetmeting
ADC23x1	Verschil –ingangen ADC2, ADC3, versterking 1	
ADC24x1	Verschil –ingangen ADC2, ADC4, versterking 1	
ADC25x1	Verschil –ingangen ADC2, ADC5, versterking 1	

ADC2 is de negatieve ingang.

De ADC kan ook verschilmetingen uitvoeren. Het resultaat kan positief of negatief zijn. De resolutie bedraagt bij differentiemetingen +/- 9 bit en wordt weergegeven als two's complement. Bij de verschilwerking staat een versterker ter beschikking met de versterkingen V: x1, x10, x200. Als x een digitale meetwaarde is, dan wordt de desbetreffende spannings-waarde als volgt berekend:

*u = x * referentiespanning / 512 / V*

5.4.4.1 ADC_Disable

ADC functies

Syntax

```
void ADC_Disable(void);
```

```
Sub ADC_Disable()
```

Beschrijving:

De functie ADC_Disable schakelt de A/D –omvormer uit om het stroomverbruik te verminderen.

Parameters
Geen

5.4.4.2 ADC_Read

ADC functies

Syntax

```
word ADC_Read(void);
```

```
Sub ADC_Read() As Word
```

Beschrijving:

De functie ADC_Read levert de gedigitaliseerde meetwaarde van één van de 8 ADC-poorten. Het nummer van de poort (0 …7) werd bij het oproepen van ADC Set() als parameter doorgegeven. Het resultaat ligt binnen het bereik van 0 tot 1023 – hetgeen overeenkomt met de 10-bit resolutie van de A/D –omvormer. Er kunnen de analoge ingangen ADC0 tot ADC7 tegen GND gemeten worden of verschilmetingen met de versterkingsfactoren 1/10/200.

Returnwaarde

gemeten waarde van de ADC –poort .

5.4.4.3 ADC_ReadInt

ADC –functies

Syntax

```
word ADC_ReadInt(void);
```

```
Sub ADC_ReadInt() As Word
```

Beschrijving

Deze functie wordt toegepast om na een ADC-interrupt de meetwaarde te lezen. De ADC-interrupt wordt geactiveerd nadat de AD_omvorming afgesloten is en hiermede een nieuwe meting beschikbaar is. Zie ook ADC_SetInt en ADC_StartInt. De functie ADC_Read levert de gedigitaliseerde meetwaarde van één van de 8 ADC poorten. Het nummer van de poort (0 … 7) werd bij het oproepen van ADC_SetInt als parameter doorgegeven. Het resultaat ligt binnen het bereik van 0 tot 1023 – hetgeen overeenkomt met de 10-bit resolutie van de A/D – omvormer. U kunt de analoge ingangen ADC0 tot ADC7 tegen GND meten of verschilmetingen met de versterkingsfactoren 1/10/200.

Returnwaarde

gemeten waarde van de ADC –poort .

5.4.4. ADC_Set

ADC –functies

Syntax

```
word ADC_Set(byte v_ref,byte channel);

Sub ADC_Set(v_ref As Byte,channel As Byte) As Word
```

Beschrijving

De functie ADC_Set initialiseert de analoge- digitale-_omvormer. De referentiespanning en het meetkanaal worden gekozen en de A/D omvormer wordt voorbereid voor de metingen. De meetwaarde wordt daarna met ADC Read() uitgelezen.

Parameter

channel Poortnummer (0 … 7) van de ADC (poort A.0 tot A.7 bij de **Mega32**, poort F.0 tot F.7 bij de **Mega128**)
v_ref Referentiespanning (zie tabel)

Naam	Waarde	Beschrijving
ADC_VREF_BG	0xC0	2,56V interne referentiespanning
ADC_VREF_VCC	0x40	Voedingsspanning (5V)
ADC_VREF_EXT	0x00	Externe referentiespanning op PAD3

5.4.4.5 ADC_SetInt

ADC functies

Syntax

```
word  ADC_SetInt(byte  v_ref,byte  channel );

Sub  ADC_SetInt(v_ref, As Byte,  channel  As Byte) As Word
```

Beschrijving

De functie ADC_SetInt initialiseert de analoog-digitaal_omvormer voor de interrupt -functie. De referentiespanning en het meetkanaal worden gekozen en de A/D omvormer wordt voorbereid voor de metingen. De interrupt-service-routine voor de ADC moet gedefinieerd zijn. Nadat de interrupt heeft plaatsgevonden kan de meetwaarde ADC ReadInt() uitgelezen worden.

Parameter

channel Poortnummer (0..7) van de ADC (poort A.0 tot A.7 bij de **Mega32**, poort F.0 tot F.7 bij de **Mega128)**
v_ref Referentiespanning (zie tabel)

Naam	Waarde	Beschrijving
ADC_VREF_BG	0xC0	2,56V interne referentiespanning
ADC_VREF_VCC	0x40	Voedingsspanning (5V)
ADC_VREF_EXT	0x00	Externe referentiespanning op PAD3

Voor de positie van PAD3 zie jumper Application board M32 of M128.

5.4.4.6 ADC_StartInt

ADC functies

Syntax

```
void ADC_StartInt(void);

Sub ADC_StartInt()
```

Beschrijving

De meting wordt gestart, als eerst de A/D omvormer met behulp van ADC SetInt() op interrupt geïnitialiseerd is. Als het meetresultaat klaarligt, wordt er een ADC_Interrupt geactiveerd.

Parameters

Geen

5.4.5 DCF 77

Alle DCF –routines zijn in de bibliotheek “LCD_Lib.cc” gerealiseerd. Voor het gebruik van deze functies dient u de bibliotheek “DCF_Lib.cc” in het project te integreren.

RTC met DCF77 tijdsynchronisatie

Het DCF77 signaal

De logische informatie (de tijdinformatie) wordt samen met de normale frequentie (de draag-frequentie van de zender, dus 77,5 kHz) verzonden. Dit gebeurt door negatieve modulatie van het signaal (verlaging van de draagamplitude tot 25%). Het begin van de verlaging ligt steeds op het begin van de seconden 0 … 58 binnen een minuut. In de 59^{e} seconde vindt er geen verlaging plaats, waardoor het volgende secondekenmerk het begin van een minuut aangeeft, en de ontvanger gesynchroniseerd kan worden. De logische waarde van de tekens volgt uit de duur ervan: 100ms is de “0”, 200ms is de “1”. Daardoor staat er binnen een minuut 59 bit voor infor-matie ter beschikking. Daarvan worden de secondekenmerken 1 tot 14 gebruikt voor gebruiks-informatie, die niet voor de DCF77 –gebruiker bedoeld zijn. De secondekenmerken 15 tot 19 kenmerken de zendantenne, de tijdzone en kondigen tijdomschakelingen aan:

Van de 20^{e} tot de 58^{e} seconde wordt de tijdinformatie voor de daaropvolgende minuut serieel in de vorm van BCD –getallen overgedragen, waarbij steeds begonnen wordt met de bit met de laagste waarde:

Bits	Betekenis
20	Startbit (is altijd “1”)
21-27	Minuut
28	Pariteit minuut
29-34	Uur
35	Pariteit uur
36-41	Dag van de maand
42-44	Dag van de week
45-49	Maand
50-57	Jaar
58	Pariteit datum

Dit betekent, dat de ontvangst minimaal een volle minuut moet lopen, voor de tijdinformatie ter beschikking kan staan. De binnen deze minuut gedecodeerde informatie is slechts beveiligd door drie pariteitbits, daardoor leiden al twee foutief ontvangen bits tot een op deze manier niet te herkennen overdrachtfout. Bij hogere eisen kunnen extra testmechanismen gebruikt worden, b.v. plausibiliteitcontrole (bevindt de ontvangen tijd zich binnen de toelaatbare grenzen), of meerdere keren lezen van de DCF77- tijdinformatie en vergelijking van de data. Een andere mogelijkheid zou zijn de DCF-tijd te vergelijken met de actuele tijd van de RTC en alleen een bepaalde afwijking toe te staan. Deze procedure geldt niet dan nadat het programma gestart is, omdat de RTC eerst ingesteld moet worden.

Beschrijving van het voorbeeldprogramma “DCF_RTC.cc”

Het programma “DCF_RTC.cc” is een klok, die via DCF77 gesynchroniseerd wordt. De tijd en de datum worden op een LCD –display getoond. De synchronisatie vindt plaats na het starten van het programma en dan dagelijks op een in het programma vastgelegde tijd (Update_uren, Update_minuten). Er worden twee bibliotheken gebruikt: DCF77_Lib.cc en LCD_Lib.cc. Voor de zendontvangst van het tijdsignaal is een DCF77 –ontvanger noodzakelijk. De uitgang van de DCF77 –ontvanger wordt aangesloten op de ingangspoort (**Mega32**: poortD.7 – **M128**: poortF.0). Eerst moet het begin van een tijdinformatie gevonden worden. Er wordt gesynchroniseerd op het puls –hiaat (59^e bit). Daarna worden de bits in het ritme per seconde opgenomen. Er vindt een pariteitcontrole plaats na de informatie betreffende minuten en seconden en eveneens aan het eind van de overdracht. Het resultaat van de pariteitcontrole wordt opgeslagen in de DCF_ARRAY[6]. Voor de overdracht van de tijdinformatie wordt de DCF_ARRAY[0..6] gebruikt. Na de ontvangst van de tijdinformatie wordt de RTC ingesteld met de nieuwe tijd en loopt daarna zelfstandig verder. Zowel de RTC als de DCF77 –decodering worden via een 10ms interrupt gestuurd. Deze tijdbasis is afgeleid van de kwartsfrequentie van de controller. DCF_mode stuurt het verloop voor de DCF77 –tijdopname.

Tabel DCF -modi

DCF-Mode	Beschrijving
0	Geen DCF77 –functie
1	Puls zoeken
2	Synchroniseren op begin frame
3	Data decoderen en opslaan, pariteitcontrole

RTC (Real Time Clock)

De RTC wordt via een 10ms interrupt gestuurd en loopt op de achtergrond onafhankelijk van het gebruikersprogramma. Elke seconde wordt de weergave op het LC-display getoond. Het weergave -formaat is
1e regel: uur : minuut : seconde
2e regel: dag . maand . jaar

LED1 knippert éénmaal per seconde.
Na het starten van het programma begint de RTC met de vastgelegde tijd. De datum is op nul gezet en geeft aan dat er nog geen DCF –tijdcompensatie heeft plaatsgevonden. Na de ontvangst van de DCF –tijd wordt de RTC geactualiseerd met de actuele data. De RTC is niet gebufferd met een batterij, d.w.z. de tijd loopt niet door zonder spanningsvoeding van de controller.

5.4.5.1 DCF_FRAME

DCF -functies

Syntax

```
void DCF_FRAME(void);
Sub DCF_FRAME()
```

Beschrijving

Schakel de **DCF Mode** op 3 (“Data decoderen en opslaan, pariteitcontrole”).

Parameter

Geen

5.4.5.2 DCF_INIT

DCF -functies

Syntax

```
void DCF_INIT(void);
Sub DCF_INIT()
```

Beschrijving

DCF_INIT bereidt de DCF -functie voor. De ingang voor het DCF –signaal wordt ingesteld.
DCF Mode = 0.

Parameter

Geen

5.4.5.3 DCF_PULS

DCF -functies

Syntax

```
void DCF_PULS(void);

Sub DCF_PULS()
```

Beschrijving

DCF Mode op 1 schakelen (“Puls zoeken”).

Parameter

Geen

5.4.5.4 DCF_START

DCF -functies

Syntax

```
void DCF_START(void);

Sub DCF_START()
```

Beschrijving

DCF_START initialiseert alle gebruikte variabelen en zet DCF Mode op 1. De DCF – tijdregistratie loopt nu automatisch.

Parameter

Geen

5.4.5.5 DCF_SYNC

DCF -functies

Syntax

```
void DCF_SYNC(void);
Sub DCF_SYNC()
```

Beschrijving

DCF Mode op 2 schakelen (“synchronisatie op begin frame”).

Parameter

Geen

5.4.6 Debug

De Debug Message functies maken het mogelijk een geformatteerde tekst naar het uitvoer-venster van de IDE te zenden. Deze functies worden interrupt –aangestuurd met een buffer van maximaal 128 Byte. D.w.z. er kunnen maximaal 128 Bytes via de debug afgezet worden zonder dat de Mega32 of Mega128 module moet wachten op de voltooiing van de uitvoer. De overdracht van de aparte tekens gebeurt op de achtergrond. Als er geprobeerd wordt meer dan 128 te verzenden, dan moet de Mega Risc CPU wachten tot alle tekens die niet meer in de buffer passen verzonden zijn.

5.4.6.1 Msg_WriteChar

Debug Message functies

Syntax

```
void Msg_WriteChar(char c);
Sub Msg_WriteChar(c As Char);
```

Beschrijving

Er wordt een teken naar het uitvoervenster gestuurd. Een C/R (Carriage Return – waarde 13) activeert een sprong naar het begin van de volgende regel.

Parameter

c het uit te voeren teken

5.4.6.2 Msg_WriteFloat

Debug Message functies

Syntax

```
void Msg_WriteFloat(float val);

Sub Msg_WriteFloat(val As Single)
```

Beschrijving

Het doorgegeven floating point getal wordt met voorteken weergegeven in het uitvoervenster.

Parameter

val float waarde

5.4.6.3 Msg_WriteHex

Debug Message functies

Syntax

```
void Msg_WriteHex(word val);
Sub Msg_WriteHex(val As Word)
```

Beschrijving

De doorgegeven 16bit waarde wordt weergegeven in het uitvoervenster. De uitvoer wordt als hexgetal met 4 cijfers geformatteerd. Als het getal kleiner is dan vier cijfers, worden de eerste posities opgevuld met nullen.

Parameter

val 16bit waarde

5.4.6.4 Msg_WriteInt

Debug Message functies

Syntax

```
void Msg_WriteInt(int val);
Sub Msg_WriteInt(val As Integer)
```

Beschrijving

De doorgegeven integere wordt weergegeven in het uitvoervenster. Bij negatieve waarden wordt er een minteken voor geplaatst.

Parameter

val 16bit integer waarde

5.4.6.5 Msg_WriteText

Debug Message functies

Syntax

```
void Msg_WriteText(char text[]);

Sub Msg_WriteText(ByRef text As Char)
```

Beschrijving

Alle tekens van de char array worden uitgevoerd tot aan de nul aan het eind.

Parameter

text cursor op char array

5.4.6.6 Msg_WriteWord

Debug Message functies

Syntax

```
void Msg_WriteWord(word val);

Sub Msg_WriteWord(val As Word)
```

Beschrijving

De parameter val wordt als getal zonder voorteken in het uitvoervenster geschreven.

Parameter

val 16bit unsigned integer waarde

5.4.7 EEPROM

Op de C-Control Por module zijn M32: 1KB M128: 4KB EEPROM geintegreerd. Deze bibliotheeksfuncties maakt de toegang tzot de EEPROM van de integer mogelijk. 32 byte van het EEPROM bereik worden voor interne doelen gebruikt, en zijn daarom niet toegankelijk.

5.4.7.1 EEPROM_Read

EEPROM functies

Syntax

```
byte EEPROM_Read(word pos);

Sub EEPROM_Read(pos As Word) As Byte
```

Beschrijving

Leest een byte van positie pos uit de EEPROM. De eerste 32 bytes zijn gereserveerd voor de C-Control Pro OS. Een waarde voor pos van 0 en groter heeft daarom betrekking op byte 32 en hoger in de EEPROM.

Parameter

pos positie in de EEPROM

Returnwaarde

De waarde van de byte op positie pos in de EEPROM.

5.4.7.2 EEPROM_WriteWord

EEPROM functies

Syntax

```
word EEPROM_ReadWord(word pos);

Sub EEPROM_ReadWord(pos As Word) As Word
```

Beschrijving

Leest een byte van positie pos uit de EEPROM. De eerste 32 bytes zijn gereserveerd voor de C-Control Pro OS. Een waarde voor pos van 0 en groter gaat daarom naar de byte 32 en hoger in de EEPROM. De waarde van pos is een byte positie in de EEPROM. Hierop moet gelet worden bij word- en zwevende komma-toegang.

Parameter

pos Byte positie in de EEPROM

Returnwaarde

De waarde van word op positie pos in de EEPROM.

5.4.7.3 EEPROM_ReadFloat

EEPROM functies

Syntax

```
float EEPROM_ReadFloat(word pos);

Sub EEPROM_ReadFloat(pos As Word) As Single
```

Beschrijving

Leest een zwevende komma-waarde van positie pos uit de EEPROM. De eerste 32 bytes zijn gereserveerd voor de C-Control Pro OS. Een waarde voor pos van 0 en groter grijpt daarom naar byte 32 en hoger in de EEPROM. De waarde van pos is een byte positie in de EEPROM. Hierop moet gelet worden bij word- en zwevende komma-toegang.

Parameter

pos Byte positie in de EEPROM

Returnwaarde

De zwevende kommawaarde op positie pos in de EEPROM.

5.4.7.4 EEPROM_Write

EEPROM functies

Syntax

```
void EEPROM_Write(word pos,byte val);

Sub EEPROM_Write(pos As Word,val As Byte)
```

Beschrijving

Schrijft een byte op positie pos in de EEPROM. De eerste 32 bytes zijn gereserveerd voor de C-Control Pro OS. Een waarde voor pos van 0 en groter grijpt daarom naar byte 32 en hoger in de EEPROM.

Parameter

pos positie in de EEPROM
val de in de EEPROM te schrijven waarde

5.4.7.5 EEPROM_WriteWord

EEPROM functies

Syntax

```
void EEPROM_WriteWord(word pos,word val);

Sub EEPROM_WriteWord(pos As Word,val As Word)
```

Beschrijving

Schrijft een word op positie pos in de EEPROM. De eerste 32 bytes zijn gereserveerd voor de C-Control Pro OS. Een waarde voor pos van 0 en groter grijpt daarom naar byte 32 en hoger in de EEPROM. De waarde van pos is een byte positie in de EEPROM. Hierop moet gelet worden bij word- en zwevende komma-toegang.

Parameter

pos byte positie in de EEPROM
val de in de EEPROM te schrijven waarde

5.4.7.6 EEPROM_WriteFloat

EEPROM functies

Syntax

```
void EEPROM_WriteFloat(word pos,float val);

Sub EEPROM_WriteFloat(pos As Word,val As Single)
```

Beschrijving

Schrijft een zwevende komma waarde op positie pos in de EEPROM. De eerste 32 bytes zijn gereserveerd voor de C-Control Pro OS. Een waarde voor pos van 0 en groter grijpt daarom naar byte 32 en hoger in de EEPROM. De waarde van pos is een byte positie in de EEPROM. Hierop moet gelet worden bij word- en zwevende komma-toegang.

Parameter

pos byte positie in de EEPROM
val de in de EEPROM te schrijven waarde

5.4.8 I2C

De Controller beschikt over een I2C logica, die een effectieve communicatie mogelijk maakt. De Controller werkt als I2C –Master (single master systeem). Werking als Slave is mogelijk, maar in de huidige versie niet geïmplementeerd.

5.4.8.1 I2C_Init

I2C functies **Voorbeeld**

Syntax

```
void I2C_Init(byte I2C_BR);

Sub I2C_Init(I2C_BR As Byte)
```

Beschrijving

Deze functie initialiseert de I2C –interface.

Parameters

I2C_BR geeft de bitrate aan. De volgende waarden zijn al vooraf gedefinieerd:

```
I2C_100kHz
I2C_400kHz
```

5.4.8.2 I2C_Read_ACK

I2C functies

Syntax

```
byte I2C_Read_ACK(void);

Sub I2C_Read_ACK() As Byte
```

Beschrijving

Deze functie ontvangt een byte en bevestigt met ACK. Daarna kan met I2C_Status de status van de interface opgevraagd worden.

Returnwaarde

Gelezen waarde van de I2C bus

5.4.8.3 I2C_Read_NACK

I2C functies Voorbeeld

Syntax

```
byte I2C_Read_NACK(void);

Sub I2C_Read_NACK() As Byte
```

Beschrijving

Deze functie ontvangt een byte en bevestigt met NACK. Daarna kan met I2C_Status de status van de interface opgevraagd worden.

Returnwaarde

Gelezen waarde van de I2C bus

5.4.8.4 I2C_Start

I2C functies Voorbeeld

Syntax

```
void I2C_Start(void);

Sub I2C_Start()
```

Beschrijving

Deze functie start de communicatie met een startsequentie. Daarna kan met I2C_Status de status van de interface opgevraagd worden.

Parameter

Geen

5.4.8.5 I2C_Status

I2C functies

Syntax

```
byte I2C_Status(void);

Sub I2C_Status()
```

Beschrijving

Met I2C_Status kan de status van de interface opgevraagd worden. De betekenis van de statusinformatie is weergegeven in de tabel.

Returnwaarde

actuele I2C status

5.4.8.6 I2C_Stop

I2C functies Voorbeeld

Syntax

```
void I2C_Stop(void);

Sub I2C_Stop()
```

Beschrijving

Deze functie beëindigt de communicatie met een stopsequentie. Daarna kan met I2C_Status de status van de interface opgevraagd worden.

Parameters

Geen

5.4.8.7 I2C_Write

I2C functies Voorbeeld

Syntax

```
void I2C_Write(byte data);

Sub I2C_Write(data As Byte)
```

Beschrijving

Deze functie zendt een byte. Daarna kan met I2C_Status de status van de interface opgevraagd worden.

Parameter

data databyte

5.4.8.8 I2C Status Codes

Tabel: **Status Codes Master Transmitter Mode**

Status Code	Beschrijving
0x08	Er is een START sequentie verzonden
0x10	Er is een “repeated” START sequentie verzonden
0x18	Er is SLA+W verzonden, er werd ACK ontvangen

0x20	Er is SLA+W verzonden, er werd NACK ontvangen
0x28	Er is een data byte verzonden, er werd ACK ontvangen
0x30	Er is een data byte verzonden, er werd NACK ontvangen
0x38	Conflict in SLA+W of data bytes

Tabel: **Status Codes Master Receiver Mode**

Status Code	**Beschrijving**
0x08	Er is een START sequentie verzonden
0x10	Er is een “repeated” START sequentie verzonden
0x38	Conflict in SLA+R of data bytes
0x40	Er is SLA+R verzonden, er werd ACK ontvangen
0x48	Er is een SLA+R verzonden, er werd NACK ontvangen
0x50	Er is een data byte ontvangen, er werd ACK verzonden
0x58	Er is een data byte ontvangen, er werd NACK verzonden

5.4.8.9 I2C voorbeeld

Voorbeeld: EEPROM 24C64 lezen en schrijven zonder I2C_status opvragen

```
// I2C Initialization, Bit Rate 100kHz

main(void)
{
      word address;
      byte data,EEPROM_data;

      address=0x20;
      data=0x42;

      I2C_Init(I2C_100kHz );
      // write data to 24C64 (8k x 8) EEPROM
      I2C_Start();
      I2C_Write(0xA0);                    // DEVICE ADDRESS : A0
      I2C_Write(address>>8);              // HIGH WORD ADDRESS
      I2C_Write(address);                 // LOW WORD ADDRESS
      I2C_Write(data);                    // write Data
      I2C_Stop();
      AbsDelay(5);                        // delay for EEPROM Write Cycle

      // read data from 24C64 (8k x 8) EEPROM
      I2C_Start();
      I2C_Write(0xA0);                    // DEVICE ADDRESS : A0
      I2C_Write(address>>8);              // HIGH WORD ADDRESS
      I2C_Write(address);                 // LOW WORD ADDRESS
      I2C_Start();                        // RESTART
      I2C_Write(0xA1);                    // DEVICE ADDRESS : A1
      EEPROM_data=I2C_Read_NACK();
      I2C_Stop();
      Msg_WriteHex(EEPROM_data);
}
```

5.4.9 Interrupt

De Controller stelt een veelvoud aan interrupts ter beschikking. Sommige daarvan worden gebruikt voor systeemfuncties en staan niet ter beschikking van de gebruiker. De volgende interrupts kunnen door de gebruiker benut worden:

Tabel interrupts

Interrupt naam	Beschrijving
INT_0	Externe interrupt0
INT_1	Externe interrupt1
INT_2	Externe interrupt2
INT_3	Externe interrupt3 (alleen Mega128)
INT_4	Externe interrupt4 (alleen Mega128)
INT_5	Externe interrupt5 (alleen Mega128)
INT_6	Externe interrupt6 (alleen Mega128)
INT_7	Externe interrupt7 (alleen Mega128)
INT_TIM1CAPT	Timer1 Capture
INT_TIM1CMPA	Timer1 CompareA
INT_TIM1CMPB	Timer1 CompareB

INT_TIM1OVF	Timer1 Overflow
INT_TIM0COMP	Timer0 Compare
INT_TIM0OVF	Timer0 Overflow
INT_ANA_COMP	Analoge comparator
INT_ADC	ADC
INT_TIM2COMP	Timer2 Compare
INT_TIM2OVF	Timer2 Overflow
INT_TIM3CAPT	Timer3 Capture (alleen Mega128)
INT_TIM3CMPA	Timer3 CompareA (alleen Mega128)
INT_TIM3CMPB	Timer3 CompareB (alleen Mega128)
INT_TIM3CMPC	Timer3 CompareC (alleen Mega128)
INT_TIM3OVF	Timer3 Overflow (alleen Mega128)

De desbetreffende interrupt moet in een Interrupt Service Routine (ISR) de overeenkomende aanwijzingen ontvangen en de interrupt moet vrijgegeven zijn. Zie Voorbeeld. Tijdens de bewerking van een interrupt –routine wordt de Multithreading uitgezet.

[Afb.] Een signaal op INT_0 bij het inschakelen van de C-Control Pro module kan de autostartprocedure storen. Volgens de pintoewijzing van M32 en M128 ligt de INT_0 op dezelfde pin als SW1. Als de SW1 bij het inschakelen van de module ingedrukt wordt, leidt dit tot activering van de seriële bootloader modus en het programma wordt niet automatisch gestart.

5.4.9.1 Ext_IntEnable

Interrupt functies

Syntax

```
void Ext_Int0(byte IRQ,byte Mode);

Sub Ext_Int0(IRQ As Byte,Mode As Byte)
```

Beschrijving

Deze functie schakelt de externe interrupt 0 vrij. De parameter Mode legt vast, wanneer een interrupt gemaakt moet worden. Een signaal op INT_0 kan leiden tot Autostart problemen.

Parameter

IRQ Nummer van de te blokkeren interrupt
Mode Parameter:

1: een low niveau activeert een interrupt
2: elke flankwisseling activeert een interrupt
3: een vallende flank activeert een interrupt
4: een stijgende flank activeert een interrupt

5.4.9.2 Ext_IntDisable

Interrupt functies

Syntax

```
void Ext_Int0Disable(byte IRQ);

Sub Ext_Int0Disable(IRQ As Byte)
```

Beschrijving

De externe interrupt IRQ wordt geblokkeerd.

Parameter

Geen

5.4.9.3 Irq_GetCount

Interrupt functies Voorbeeld

Syntax

```
byte Irq_GetCount(void);

Sub Irq_GetCount() As Byte
```

Beschrijving

Signaleert dat de interrupt verwerkt is (interrupt acknowledge). Als de functie niet aan het eind van een interrupt routine wordt opgeroepen, wordt er ononderbroken in de interrupt gesprongen.

Returnwaarde

Geeft aan, hoe vaak de interrupt vanaf de hardware tot aan het oproepen van Irq_GetCount() geactiveerd werd. Een waarde groter dan 1 kan optreden als de hardware sneller interrupts genereert dan de interpreter de interruptroutine kan verwerken.

5.4.9.4 Irq_SetVect

Interrupt functies Voorbeeld

Syntax

```
void Irq_SetVect(byte irqnr,float vect);

Sub Irq_SetVect(irqnr As Byte,vect As Single)
```

Beschrijving

Zet de op te roepen interrupt functie voor een bepaalde interrupt. Aan het eind van de interrupt routine moet de functie Irq_GetCount() opgeroepen worden, anders wordt er ononderbroken in de interrupt functie gesprongen.

Parameter

irqnr specificeert het type van de interrupt (zie tabel)
vect is de naam van de op te roepen interrupt functie

Opmerking
De zwevende komma- datatype mag ongepast verschijnen maar hij wordt intern als 4 byte waarde behandeld. Een functie- aanwijzer moet sinds de Mega128 ondersteuning meer dan 18 bit lang zijn.

Tabel interrupt vectoren:

Nr	Interrupt naam	Beschrijving
0	INT_0	Externe interrupt0
1	INT_1	Externe interrupt1
2	INT_2	Externe interrupt2
3	INT_TIM1CAPT	Timer1 Capture
4	INT_TIM1CMPA	Timer1 CompareA
5	INT_TIM1CMPB	Timer1 CompareB
6	INT_TIM1OVF	Timer1 Overflow
7	INT_TIM0COMP	Timer0 Compare
8	INT_TIM0OVF	Timer0 Overflow
9	INT_ANA_COMP	Analoge comparator
10	INT_ADC	ADC
11	INT_TIM2COMP	Timer2 Compare
12	INT_TIM2OVF	Timer2 Overflow

5.4.9.5 IRQ voorbeeld

Voorbeeld: gebruik van interrupt routines

```
// Timer 2 loopt normaalgesproken in de 10msa maat. In dit
// voorbeeld wordt daarom de variabele cnt elke 10ms met 1 verhoogd

int cnt;

void ISR(void)
{
      int irqcnt;

      cnt=cnt+1;
      irqcnt=Irq_GetCount(INT_TIM2COMP);
}

void main(void)
{
      cnt=0;
      Irq_SetVect(INT_TIM2COMP,ISR);
      while(true); // Eondeloze lus
}
```

5.4.10 Keyboard

Een deel van deze routines is in de interpreter geïmplementeerd, een ander deel wordt opgeroepen door het toevoegen van de bibliotheek “Key_Lib.cc”. Omdat de functies in “Key_Lib.cc” door bytecodes gerealiseerd worden, zijn ze langzamer in de verwerking. Bibliotheekfuncties hebben echter het voordeel dat als u ze niet gebruikt, deze functies door weglaten van de bibliotheek uit het project gehaald worden. Directe interpreter –functies zijn steeds aanwezig, maar kosten flashgeheugen.

5.4.10.1 Key_Init

Keyboard functies (bibliotheek ”*Key_Lib.cc*”)

Syntax

```
void Key_Init(void);

Sub Key_Init()
```

Beschrijving
De globale array keymap wordt geïnitialiseerd met de ASCII waarden van het toetsenbord.

Parameter

Geen

5.4.10.2 Key_Scan

Keyboard functies

Syntax

```
word Key_Scan(void);

Sub Key_Scan() As Word
```

Beschrijving
Key_Scan zoekt op volgorde de invoerpinnen van het aangesloten toetsenbord door en geeft het resultaat retour als bitveld. De “1” bits vertegenwoordigen de toetsen die tot aan het tijdstip van de scan ingedrukt zijn.

Returnwaarde
16 bits die de aparte invoerleidingen van het toetsenbord vertegenwoordigen.

5.4.10.3 Key_TranslateKey

Keyboard functies (bibliotheek ”*Key_Lib.cc*”)

Syntax

```
char Key_TranslateKey(word keys);

Sub Key_TranslateKey(keys As Word) As Char
```

Beschrijving
Deze hulpfunctie levert het teken terug dat overeenkomt met het eerste opduiken van een “1” in het bitveld van de invoerparameter.

Parameter

keys bitveld dat door Key_Scan() teruggeleverd wordt.

Returnwaarde

ASCII waarde van de herkende toets
-1 als er geen toets ingedrukt is

5.4.11 LCD

Een deel van deze routines is geïmplementeerd in de interpreter, een ander deel kan opgeroepen worden door het toevoegen van de bibliotheek “LCD_Lib.cc”. Omdat de functies in “LCD_Lib.cc” gerealiseerd worden door bytecodes, zijn ze langzamer in de verwerking. Bibliotheekfuncties hebben echter het voordeel dat als u ze niet gebruikt, deze functies door weglaten van de bibliotheek uit het project gehaald worden. Directe interpreter –functies zijn steeds aanwezig, maar kosten flashgeheugen.

5.4.11.1 LCD_ClearLCD

LCD functies (bibliotheek ”*LCD_Lib.cc*”)

Syntax

```
void LCD_ClearLCD(void);
Sub LCD_ClearLCD()
```

Beschrijving
Wist het display en schakelt de cursor in.

Parameter
Geen

5.4.11.2 LCD_CursorOff

LCD functies (bibliotheek ”*LCD_Lib.cc*”)

Syntax

```
void LCD_CursorOff(void);

Sub LCD_CursorOff()
```

Beschrijving
Schakelt de cursor van het display uit.

Parameter

Geen

5.4.11.3 LCD_CursorOn

LCD functies (bibliotheek "*LCD_Lib.cc*")

Syntax

```
void LCD_CursorOn(void);
```

```
Sub LCD_CursorOn()
```

Beschrijving

Schakelt de cursor van het display in.

Parameter

Geen

5.4.11.4 LCD_CursorPos

LCD functies (bibliotheek "*LCD_Lib.cc*")

Syntax

```
void LCD_CursorPos(byte pos);
```

```
Sub LCD_CursorPos(pos As Byte)
```

Beschrijving

Plaatst de cursor in de positie pos.

Parameter

pos cursorpositie

Waarde van pos	**Positie op het display**
0x00-0x07	0 – 7 in de 1e regel
0x40-0x47	0 – 7 in de 2e regel

Bij een display met meer dan 2 regels en max. 32 tekens per regel geldt het volgende schema:

Waarde van pos	**Positie op het display**
0x00-0x1f	0 – 31 in de 1e regel
0x40-0x5f	0 – 31 in de 2e regel
0x20-0x3f	0 – 31 in de 3e regel
0x60-0x6f	0 – 31 in de 4e regel

5.4.11.5 LCD_Init

LCD functies (bibliotheek "*LCD_Lib.cc*")

Syntax

```
void LCD_Init(void);

Sub LCD_Init()
```

Beschrijving

"High Level" initialisering van het LC- display. Roept als eerste LCD InitDisplay() op.

Parameter

Geen

5.4.11.6 LCD_SubInit

LCD functies

Syntax

```
void LCD_SubInit(void);

Sub LCD_SubInit()
```

Beschrijving

Initialiseert de poort s voor de displaybesturing op assembler –niveau. Moet als eerste routine voor alle andere LCD uitvoerfuncties opgeroepen worden. Wordt gebruikt als eerste commando van LCD_Init() gebruikt.

Parameter

Geen

5.4.11.7 LCD_TestBusy

LCD functies

Syntax

```
void LCD_TestBusy(void);

Sub LCD_TestBusy()
```

Beschrijving

De functie wacht tot de display controller niet meer “*busy*” is. Als u eerst naar de controller gaat, wordt de data –opbouw op het display gestoord.

Parameter

Geen

5.4.11.8 LCD_WriteChar

LCD functies (bibliotheek ”*LCD_Lib.cc*”)

Syntax

```
void LCD_WriteChar(char c);

Sub LCD_WriteChar(c As Char)
```

Beschrijving

Schrijft een teken bij de cursorpositie op het LC- display.

Parameter

c ASCII waarde van het teken

5.4.11.9 LCD_WriteCTRRegister

LCD functies (bibliotheek ”*LCD_Lib.cc*”)

Syntax

```
void LCD_WriteCTRRegister(byte cmd);

Sub LCD_WriteCTRRegister(cmd As Byte)
```

Beschrijving

Stuurt een commando naar de display controller.

Parameter

cmd Commando in byte –vorm

5.4.11.10 LCD_WriteDataRegister

LCD functies (bibliotheek ”*LCD_Lib.cc*”)

Syntax

```
void LCD_WriteDataRegister(char x);

Sub LCD_WriteDataRegister(x As Char)
```

Beschrijving

Stuurt een databyte naar de display controller.

Parameter

x databyte

5.4.11.11 LCD_WriteRegister

LCD functies (bibliotheek "*LCD_Lib.cc*")

Syntax

```
void LCD_WriteRegister(byte y,byte x);
```

```
Sub LCD_WriteRegister(y As Byte,x As Byte)
```

Beschrijving

LCD_WriteRegister splitst databyte y in twee nibbles en stuurt ze naar de display controller.

Parameter

y databyte
x commandonibble

5.4.11.12 LCD_WriteText

LCD functies (bibliotheek "*LCD_Lib.cc*")

Syntax

```
void LCD_WriteText(char text[]);
```

```
Sub LCD_WriteText(ByRef Text As Char)
```

Beschrijving

Alle tekens van de char array tot aan een bepaalde nul worden uitgegeven.

Parameter

text char array

5.4.12 Poort

De Atmel Mega 32 heeft 4 in - /uitgangspoorten van elk 8 bit. De Atmel Mega 128 heeft 6 in - /uitgangspoorten van elk 8 bit en 1 in-/ uitgangspoort van 5 bit. Elke bit van een afzonderlijke poort kan als ingang of als uitgang geconfigureerd worden. Omdat echter het aantal pinnen van de Mega 32 Risc CPU beperkt is, zijn extra functies aan aparte poorten toegewezen. U vindt een tabel van de pintoewijzing in de documentatie van de M32 en M128..

➔ Het is belangrijk om de pintoewijzing te bestuderen voor u met de programmering begint, omdat belangrijke functies van de programma –ontwikkeling (bijv. de USB interface van het Application Board) op bepaalde poorten liggen. Als deze poorten omgeprogrammeerd worden of de bijbehorende jumpers op het Application Board zijn niet meer geplaatst, kan het gebeuren dat de ontwikkelingsomgeving geen programma's meer naar de C-Control Pro kan overbrengen.

➔ De datarichting (ingang/uitgang) kan met de functie Port _DataDir of Port _DataDirBit vastgelegd worden. Als een pin als ingang geconfigureerd is, dan kan deze pin of hoogohmig ("floatend") of met een interne pull-up gebruikt worden. Als met Port_Write of Port_WriteBit een "1" schrijft op een ingang, dan wordt de pull-up weerstand (referentieniveau VCC) geactiveerd en de ingang is gedefinieerd.

5.4.12.1 Port _DataDir

Poort functies voorbeeld

Syntax

```
void Port_DataDir(byte port,byte val);

Sub Port_DataDir(port As Byte,val As Byte)
```

Beschrijving

De functie Port _DataDir configureert de bits van de poort voor in - of uitvoer. Als de bit "1" is, dan wordt de pin van de desbetreffende bitpositie op uitgang geschakeld. Een voorbeeld: als port =PortB en val = 0x02, dan wordt pin 2 van de Atmel Mega (komt overeen met Poort B.1 – zie pintoewijzing van M32 en M128) op uitgang geconfigureerd.

Parameter

port poort nummer (zie tabel)
val uitvoer byte

Tabel poortnummers

Definitie	Waarde
Port A	0
Port B	1
Port C	2
Port D	3
PortE (Mega128)	4
PortF (Mega128)	5
PortG (Mega128)	6

5.4.12.2 Port _DataDirBit

Poortfuncties

Syntax

```
void Port_DataDirBit(byte portbit,byte val);

Sub Port_DataDirBit(portbit As Byte,val As Byte)
```

Beschrijving

De functie Port _DataDirBit configureert een bit (pin) van een poort voor in - of uitvoer. Als de bit “1” is, dan wordt de pin op uitgang geschakeld. Een voorbeeld: als port bit = 9 en val = 0, dan wordt pin 2 van de Atmel Mega (komt overeen met PortB.1 – zie pintoewijzing) op ingang geconfigureerd.

Parameters

port bit bitnummer van de poort (zie tabel)
val 0- = ingang, 1 = uitgang

Tabel poort bits

Definitie	Poort bit
PortA.0	0
…	…
PortA.7	7
PortB.0	8
…	…
PortB.7	15
PortC.0	16
…	…
PortC.7	23
PortD.0	24
…	…
PortD.7	31
Vanaf hier alleen Mega128	
PortE.0	32
…	…
PortE.7	39
PortF.0	40
…	…
PortF.7	47
PortG.0	48
…	…
PortG.4	52

5.4.12.3 Port _Read

Poortfuncties

Syntax

```
byte Port_Read(byte port);

Sub Port_Read(port As Byte) As Byte
```

Beschrijving

Leest een byte van een specifieke poort . Alleen de pinnen van de poort die geschakeld zijn op ingang, leveren een geldige waarde op de desbetreffende bitpositie in de gelezen byte terug. (Voor de afbeelding tussen poortbits en de pinnen van de Atmel Mega chips zie pintoewijzing van M32 en M128).

Parameter

port poortnummer (zie tabel)

Returnwaarde

Waarde van de poort

Tabel poortnummers

Definitie	Waarde
PortA	0
PortB	1
PortC	2
PortD	3
PortE (Mega128)	4
PortF (Mega128)	5
PortG (Mega128)	6

5.4.12.4 Port _ReadBit

Poortfuncties

Syntax

```
byte Port_ReadBit(byte port);

Sub Port_ReadBit(port As Byte) As Byte
```

Beschrijving

Leest een bitwaarde van de gespecificeerde poort. De desbetreffende pin van de poort moet op ingang zijn geschakeld. (Voor de afbeelding tussen poortbits en de pinnen van de Atmel Mega Chips zie pintoewijzing van M32 en M128).

Parameter

portbit bitnummer van de poort (zie tabel)

Returnwaarde

Bitwaarde van de poort (0 of 1)

Poortbits tabel

Definitie	Poortbit
PortA.0	0
…	…
PortA.7	7
PortB.0	8
…	…
PortB.7	15
PortC.0	16
…	…
PortC.7	23
PortD.0	24
…	…
PortD.7	31
Vanaf hier alleen Mega128	
PortE.0	32
…	…
PortE.7	39
PortF.0	40
…	…
PortF.7	47
PortG.0	48
…	…
PortG.4	52

5.4.12.5 Port _Write

Poortfuncties Voorbeeld

Syntax

```
void Port_Write(byte port,byte val);
```

```
Sub Port_Write(port As Byte,val As Byte)
```

Beschrijving

Schrijft een byte op de gespecificeerde poort . Alleen de pinnen van de poort die op uitgang geschakeld zijn, nemen de bitwaarden van de doorgegeven parameters over.
Als een pin op ingang geschakeld is, kan de interne pull-up weerstand ingeschakeld (1) of uitgeschakeld (0) worden. (Voor de afbeelding tussen poortbits en de pinnen van de Atmel Mega chips zie pintoewijzing van M32 en M128).

Parameters

port poortnummer (zie tabel)
val uitvoerbyte

Tabel poortnummers

Definitie	Waarde
PortA	0
PortB	1
PortC	2
PortD	3
PortE (Mega128)	4
PortF (Mega128)	5
PortG (Mega128)	6

5.4.12.6 Port _WriteBit

Poortfuncties

Syntax

```
void Port_WriteBit(byte portbit,byte val);

Sub Port_WriteBit(portbit As Byte,val As Byte)
```

Beschrijving

De functie Port_WriteBit zet de waarde van een pin die op uitgang geschakeld is. Als een pin op ingang geschakeld is, dan kan de interne pull-up weerstand ingeschakeld (1) of uitgeschakeld (0) worden.
(Voor de afbeelding tussen portbits en de pins van de Atmel Mega chips zie pintoewijzing van M32 en M128).

Parameter

portbit bitnummer van de poort (zie tabel)
val mag 0 of 1 zijn

Poortbits tabel

Definitie	Poortbit
PortA.0	0
…	…
PortA.7	7
PortB.0	8
…	…
PortB.7	15
PortC.0	16
…	…
PortC.7	23

PortD.0	24
…	…
PortD.7	31
Vanaf hier alleen Mega128	
PortE.0	32
…	…
PortE.7	39
PortF.0	40
…	…
PortF.7	47
PortG.0	48
…	…
PortG.4	52

5.4.12.7 Poort voorbeeld

```
// Programma laat afwisselend de beide LED's op het
// Application Board knipperen in een 1 seconde – ritme

void main(void)
{
      Port_DataDirBit(PORT_LED1,PORT_OUT);
      Port_DataDirBit(PORT_LED2,PORT_OUT);

      while(true)    // Eindelozw lus
      {
            Port_WriteBit(PORT_LED1,PORT_ON);
            Port_WriteBit(PORT_LED2,PORT_OFF);
            AbsDelay(1000);
            Port_WriteBit(PORT_LED1,PORT_OFF);
            Port_WriteBit(PORT_LED2,PORT_ON);
            AbsDelay(1000);
      }
}
```

5.4.13 Math

Hieronder volgen de wiskundige functies die de C-Control Pro 128 in eenvoudige zwevende komma precisie (32 bit) beheerst. Deze functies zijn niet in de bibliotheek van de C-Control Pro 32, anders is er te weinig flash-geheugen voor de gebruikerprogramma's.

5.4.13.1 acos

Wiskundige functies

Syntax

```
float acos(float val);

Sub acos(val As Single) As Single
```

Beschrijving

De arcus cosinus wordt berekend. De hoek wordt in radiant aangegeven. In- en uitvoerwaarden liggen tussen –pi en +pi.

Parameter

val Waarde van welke de functie berekend wordt

Returnwaarde

Arcus cosinus van de invoerwaarde

5.4.13.2 asin

Wiskundige functies

Syntax

```
float asin(float val);

Sub asin(val As Single) As Single
```

Beschrijving

De arcus sinus wordt berekend. De hoek wordt in radiant aangegeven. In- en uitvoerwaarden liggen tussen –pi en +pi.

Parameter

val Waarde van welke de functie berekend wordt

Returnwaarde

Arcus sinus van de invoerwaarde

5.4.13.3 atan

Wiskundige functies

Syntax

```
float atan(float val);

Sub atan(val As Single) As Single
```

Beschrijving

De arcus tangens wordt berekend. De hoek wordt in radiant aangegeven. In- en uitvoerwaarden liggen tussen –pi en +pi.

Parameter

val Waarde van welke de functie berekend wordt

Returnwaarde

Arcus tangens van de invoerwaarde

5.4.13.4 ceil

Wiskundige functies

Syntax

```
float ceil(float val);

Sub ceil(val As Single) As Single
```

Beschrijving

Het eerst volgende grotere integergetal met betrekking tot het zwevende-kommagetal val wordt berekend. De hoek wordt in radiant aangegeven. In- en uitvoerwaarden liggen tussen –pi en +pi.

Parameter

val Waarde van welke de integer berekend wordt

Returnwaarde

Het resultaat van de functie

5.4.13.5 cos

Wiskundige functies

Syntax

```
float cos(float val);

Sub cos(val As Single) As Single
```

Beschrijving

De cosinus wordt berekend. De hoek wordt in radiant aangegeven. In- en uitvoerwaarden liggen tussen –pi en +pi.

Parameter

val Waarde van welke de functie berekend wordt

Returnwaarde

Cosinus van de invoerwaarde

5.4.13.6 exp

Wiskundige functies

Syntax

```
float exp(float val);

Sub exp(val As Single) As Single
```

Beschrijving

De functie e ^ val wordt berekend.

Parameter

val Exponent

Returnwaarde

Resultaat van de functie

5.4.13.7 fabs

Wiskundige functies

Syntax

```
float fabs(float val);

Sub fabs(val As Single) As Single
```

Beschrijving

De absolute waarde van het zwevende kommagetal wordt berekend.

Parameter

val invoerwaarde

Returnwaarde

Resultaat van de functie

5.4.13.8 floor

Wiskundige functies

Syntax

```
float floor(float val);

Sub floor(val As Single) As Single
```

Beschrijving

Het eerst volgende kleinere integergetal met betrekking tot het zwevende-kommagetal val wordt berekend.

Parameter

val waarde waarvan de integer berekend wordt

Returnwaarde

Resultaat van de functie

5.4.13.9 floor

Wiskundige functies

Syntax

```
float ldexp(float val,int expn);

Sub ldexp(val As Single,expn As Integer) As Single
```

Beschrijving

De functie val *2^ expn wordt berekend

Parameter

val Multiplicator
expn Exponent

Returnwaarde

Resultaat van de functie

5.4.13.10 ln

Wiskundige functies

Syntax

```
float ln(float val);

Sub ln(val As Single) As Single
```

Beschrijving

De natuurlijke logaritme wordt berekend

Parameter

val Invoerwaarde

Returnwaarde

Resultaat van de functie

5.4.13.11 log

Wiskundige functies

Syntax

```
float log(float val);

Sub log(val As Single) As Single
```

Beschrijving

De logaritme voor basis 10 wordt berekend

Parameter

val Invoerwaarde

Returnwaarde

Resultaat van de functie

5.4.13.12 pow

Wiskundige functies

Syntax

```
float pow(float x,float y);

Sub pow(x As Single,y As Single) As Single
```

Beschrijving

Machtfunctie. De functie x ^ y wordt berekend

Parameter

x Basis
y Exponent

Returnwaarde

Resultaat van de functie

5.4.13.13 sin

Wiskundige functies

Syntax

```
float sin(float val);

Sub sin(val As Single) As Single
```

Beschrijving

De sinus wordt berekend. De hoek wordt in radiant aangegeven. In- en uitvoerwaarden liggen tussen –pi en +pi.

Parameter

val Waarde van welke de functie berekend wordt

Returnwaarde

sinus van de invoerwaarde

5.4.13.14 sqrt

Wiskundige functies

Syntax

```
float sqrt(float val);

Sub sqrt(val As Single) As Single
```

Beschrijving

De vierkantswortel wordt berekend.

Parameter

val Waarde qaarvan de vierkantswortel wordt berekend

5.4.13.15 tan

Wiskundige functies

Syntax

```
float tan(float val);

Sub tan(val As Single) As Single
```

Beschrijving

De tangens wordt berekend. De hoek wordt in radiant aangegeven. In- en uitvoerwaarden liggen tussen –pi en +pi.

Parameter

val Waarde van welke de functie berekend wordt

Returnwaarde

Tangens van de invoerwaarde

5.4.14 RS232

In tegenstelling tot de Debug Message functies werken alle seriële routines niet met interrupt maar “pollend”. Dit betekent dat de functies pas terugkeren als het teken of de tekst geschreven of gelezen is. De seriële interface kan met snelheden tot 230.4 kbaud gebruikt worden. Bij de functies voor de seriële interface geedt de eerste parameter het poortnummer aan (0 of 1). Bij de Mega32 staat alleen één seriële interface ter beschikking (0), voor de Mega128 twee (0, 1).

5.4.14.1 Serial_Disable

Seriële functies

Syntax

```
void Serial_Disable(byte serport);

Sub Serial_Disable(serport As Byte)
```

Beschrijving
De seriële interface wordt uitgeschakeld en de bijhorende poorten kunnen anders toegepast worden.

Parameter

serport interfacenummer (0 = 1e seriële, 1 = 2e seriële enz.)

5.4.14.2 Serial_Init

Seriële functies Voorbeeld

Syntax

```
void Serial_Init(byte serport,byte par,byte divider);

Sub Serial_Init(serport As Byte,par As Byte,divider As Byte)
```

Beschrijving
De seriële interface wordt geïnitialiseerd. De waarde par wordt door optellen van de vooraf gedefinieerde bitwaarden samengesteld. Men telt eerst de tekenlengte, dan het aantal stopbits en dan de pariteit, bijv. “SR_7BIT│SR_2STOP │ SR_EVEN_PAR” voor 7 bits per teken, 2 stopbits en even pariteit (zie ook voorbeeld). Deze waarden zouden in BASIC Syntax als volgt uitzien: “SR_7BIT Or SR_2STOP Or SR_EVEN_PAR” De baudrate wordt als verdelerwaarde, zoals ook in de tabel gespecificeerd wordt.

Parameter

serport Interfacenummer (0 = 1e seriële, 1 = 2e seriële enz.)
par Interface –parameter (zie tabel)
divider Baudrate –initialisering d.m.v. verdeler (zie tabel)

Tabel par definities:

Definitie	Functie
SR_5BIT	5 bit tekenlengte
SR_6BIT	6 bit tekenlengte
SR_7BIT	7 bit tekenlengte
SR_8BIT	8 bit tekenlengte
SR_1STOP	1 stop bit
SR_2STOP	2 stop bit
SR_NO_PAR	no parity
SR_EVEN_PAR	even parity
SR_ODD_PAR	odd parity

Tabel divider definities

Divider	Definitie	Baudrate
383	SR_BD2400	2400bps
191	SR_BD4800	4800bps
95	SR_BD9600	9600bps
63	SR_BD14400	14400bps
47	SR_BD19200	19200bps
31	SR_BD28800	28800bps
23	SR_BD38400	38400bps
15	SR_BD57600	57600bps
11	SR_BD76800	76800bps
7	SR_BD115200	115200bps
3	SR_BD230400	230400bps

5.4.14.3 Serial_Init_IRQ

Seriële functies Voorbeeld

Syntax

```
void Serial_Init_IRQ(byte serport,byte ramaddr[],byte recvlen,byte
sendlen,byte par,

Sub Serial_Init_IRQ(serport As Byte,ByRef ramaddr As Byte,recvlen As
Byte,sendlen As par As Byte,div As Byte)
```

Beschrijving

De seriële interface wordt voor de toepassing in de interrupt mode geïnitialiseerd. De gebruiker moet een **globale** variabele als buffer beschikbaar stellen. In deze buffer worden de ontvangen data alsook de te versturen data bewaard. De **grootte van de buffer moet de grootte van de ontvangstbuffer plus de grootte van de zendbuffer plus 6** zijn (zie ook voorbeeld). De zend- en de ontvangstbuffer kan maximaal 255 tekens opnemen.

Voor de waarde par wordt door optellen van de vooraf gedefinieerde bitwaarden samengesteld. Men telt eerst de tekenlengte, dan het aantal stopbits en dan de pariteit, bijv. “SR_7BIT│SR_2STOP │ SR_EVEN_PAR” voor 7 bits per teken, 2 stopbits en even pariteit. Deze waarden zouden in BASIC Syntax als volgt uitzien: “SR_7BIT Or SR_2STOP Or SR_EVEN_PAR” De baudrate wordt als verdelerwaarde, zoals ook in de tabel gespecificeerd wordt.

Parameter

serport Interfacenummer (0 = 1e seriële, 1 = 2e seriële enz.)
ramadr Adres van de buffer
recvlen Grootte van de ontvangstbuffer
senlen Grootte van de zendbuffer
par Interface –parameter (zie tabel)
div Baudrate –initialisering d.m.v. verdeler (zie tabel)

Tabel par definities:

Definitie	Functie
SR_5BIT	5 bit tekenlengte
SR_6BIT	6 bit tekenlengte
SR_7BIT	7 bit tekenlengte
SR_8BIT	8 bit tekenlengte
SR_1STOP	1 stop bit
SR_2STOP	2 stop bit
SR_NO_PAR	no parity
SR_EVEN_PAR	even parity
SR_ODD_PAR	odd parity

Tabel divider definities

Divider	Definitie	Baudrate
383	SR_BD2400	2400bps
191	SR_BD4800	4800bps
95	SR_BD9600	9600bps
63	SR_BD14400	14400bps
47	SR_BD19200	19200bps
31	SR_BD28800	28800bps
23	SR_BD38400	38400bps
15	SR_BD57600	57600bps
11	SR_BD76800	76800bps
7	SR_BD115200	115200bps
3	SR_BD230400	230400bps

5.4.14.4 Serial_IRQ_Info

Seriële functies

Syntax

```
byte Serial_IRQ_Info(byte serport, byte info);
Sub Serial_IRQ_Info(serport As Byte, info As Byte) As Byte
```

Beschrijving

Afhankelijk van de parameter info wordt teruggegeven hoeveel bytes in de zend- of ontvangstbuffer nog plaats hebben.

Parameter

serport Interfacenummer (0 = 1e seriële, 1 = 2e seriële enz.)

Info waarden:

RS232_FIFO_RECV (0) Plaats in de ontvangstbuffer
RS232_FIFO_Send (1) Plaats in de zendbuffer

Returnwaarde

Plaats in de buffer van de seriële interface in bytes

5.4.14.5 Serial_Read

Seriële functies

Syntax

```
byte Serial_Read(byte serport);

Sub Serial_Read(serport As Byte) As Byte
```

Beschrijving

Een byte wordt door de seriële interface gelezen. Als er zich geen byte in de seriële buffer bevindt, keert de functie pas terug als er een teken ontvangen is.

Parameter

serport Interfacenummer (0 = 1e seriële, 1 = 2e seriële enz.)

Returnwaarde

Ontvangen byte uit de seriële interface

5.4.14.6 Serial_ReadExt

Seriële functies

Syntax

```
word Serial_ReadExt(byte serport);

Sub Serial_ReadExt(serport As Byte) As Word
```

Beschrijving

Een byte wordt door de seriële interface gelezen. In tegenstelling tot Serial Read(), keert de functie ook dan direct terug als er zich geen teken in de seriële interface bevindt. In dit geval wordt dan de waarde 256(0x100) geretourneerd.

Parameter

serport Interfacenummer (0 = 1e seriële, 1 = 2e seriële enz.)

Returnwaarde

ontvangen byte uit de seriële interface

5.4.14.7 Serial_Write

Seriële functies voorbeeld

Syntax

```
void Serial_Write(byte val);

Sub Serial_Write(val As Byte)
```

Beschrijving

Een byte wordt naar de seriële interface gestuurd.

Parameter

serport Interfacenummer (0 = 1e seriële, 1 = 2e seriële enz.)
val de uit te voeren byte waarde

5.4.14.8 Serial_WriteText

Seriële functies

Syntax

```
void Serial_WriteText(byte serport,char text[]);

Sub Serial_WriteText(serport As Byte,ByRef Text As Char)
```

Beschrijving

Alle tekens van de char array tot aan de laatste nul worden doorgegeven naar de seriële interface.

Parameter

serport Interfacenummer (0 = 1e seriële, 1 = 2e seriële enz.)
Text char array

5.4.14.9 Serial voorbeeld

```
// Stringuitvoer op de seriële interface
void main(void)
{
      int i;
      char str[10];

      str="test";
      i=0;
      // Initialiseer interface met 19200baud, 8 bit, 1 stopbit, geen
      pariteit
      Serial_Init(0,SR_8BIT|SR_1STOP|SR_NO_PAR,SR_BD19200);

      while(str[i]) Serial_Write(0,str[i++]); // Geef de string aan
}
```

5.4.1410 Serial voorbeeld (IRQ)

```
// 35 byte zend + ontvangstbuffer + 6 byte interne FIFO beheer
byte buffer[41]; // Array gedeclareerd

// Stringuitvoer op de seriële interface
void main(void)
{
      int i;
      char str[10];

      str="test";
      i=0;
      // Initialiseer interface met 19200baud, 8 bit, 1 stopbit, geen
      pariteit
      // 20 byte ontvangstbuffer - 15 byte zendbuffer
      Serial_Init_IRQ(0,buffer,20,15,SR_8BIT|SR_1STOP|SR_NO_PAR,
      SR_BD19200);
while(str[i]) Serial_Write(0,str[i++]); // Geef de string aan
}
```

5.4.15 SPI

De SPI interface wordt momenteel alleen gebruikt om op het Application board de USB gegevens van de Mega8 controller te ontvangen. In de toekomst worden meer functies de communicatie met andere apparaten via SPI ondersteunen.

5.4.15.1 SPI_Disable
SPI functies

Syntax

```
void SPI_Disable(void);

Sub SPI_Disable()
```

Beschrijving

De SPI interface wordt uitgeschakeld en de bijhorende poorten kunnen anders toegepast worden.

Parameter

Geen

5.4.16 Strings

Een deel van deze stringroutines is in de interpreter geïmplementeerd, een ander deel kan door toevoegen van de bibliotheek “String_Lib.cc” opgeroepen worden. Omdat de functies in “String_Lib.cc” door bytecodes gerealiseerd worden, zijn ze langzamer in de verwerking. Bibliotheekfuncties hebben echter als voordeel, wanneer de functies niet gebruikt worden door het weglaten van de bibliotheek deze uit het project neemt. Directe interpreter –functies zijn steeds aanwezig maar kosten flash –geheugen.

Er bestaat geen expliciet “String”-datatype. Een string is gebaseerd op een character array. U dient de grootte van de array zo kiezen dat alle tekens van de string in de character array passen. Bovendien is er ruimte nodig voor een eindteken (decimale nul), om het einde van de tekenketen (char array) aan te geven.

5.4.16.1 Str_Comp

String functies

Syntax

```
char Str_Comp(char str1[],char str2[]);

Sub Str_Comp(ByRef str1 As Char,ByRef str2 As Char) As Char
```

Beschrijving

Twee strings worden met elkaar vergeleken.

Parameters

str1 Cursor op char array 1
str2 Cursor op char array 2

Returnwaarde

0 als beide strings gelijk zijn
<0 als op het onderscheidingspunt de 1e string kleiner is
>0 als op het onderscheidingspunt de 1e string groter is

5.4.16.2 Str_Copy

String functies

Syntax

```
void Str_Copy(char destination[],char source[],word offset);

Sub Str_Copy(ByRef destination As Char,ByRef source As Char,offset As
Word)
```

Beschrijving

De bronstring (source) wordt gekopieerd naar de doelstring (destination). Bij de kopieeractie wordt echter in elk geval het string –eindteken van de brontekensketen (char array) mee gekopieerd.

Parameters

destination cursor op de doelstring
source cursor op de bronstring
offset aantal tekens waarmee de bronstring verschoven naar de doelstring gekopieerd wordt.

Als offset de waarde **STR_APPEND**(0xffff) heeft, dan wordt als offset de lengte van de doelstring aangenomen. In dit geval wordt de bronstring achter de doelstring gekopieerd.

5.4.16.3 Str_Fill

String functies (bibliotheek *"String_Lib.cc"*)

Syntax

```
void Str_Fill(char dest[],char c,word len);

Sub Str_Fill(ByRef dest As Char,c As Char,len As Word)
```

Beschrijving

De string dest wordt opgevuld met het teken c.

Parameters

dest	cursor op de doelstring
c	het teken dat herhaald in de string gekopieerd wordt
len	aantal keren hoe vaak c in de doelstring geschreven wordt

5.4.16.4 Str_Isalnum

String functies (bibliotheek *"String_Lib.cc"*)

Syntax

```
byte Str_Isalnum(char c);

Sub Str_Isalnum(c As Char) As Byte
```

Beschrijving

Een teken c wordt er op gecontroleerd, of het uit het alfabet stamt of een cijfer is.

Parameter

c	het te controleren teken

Returnwaarde

1	als het teken numeriek of alfabetisch is (zowel hoofd – als kleine letters)
0	overig

5.4.16.5 Str_Isalpha

String functies (bibliotheek *"String_Lib.cc"*)

Syntax

```
byte Str_Isalpha(char c);

Sub Str_Isalpha(c As Char) As Byte
```

Beschrijving

Een teken c wordt er op gecontroleerd, of het uit het alfabet stamt.

Parameters

c het te controleren teken

Returnwaarde

1 als het teken alfabetisch is (zowel hoofd – als kleine letters)
0 overig

5.4.16.6 Str_Len

String functies

Syntax

```
word Str_Len(char str[]);

Sub Str_Len(ByRef str As Char) As Word
```

Beschrijving

De lengte van de tekenketen (van de character array) wordt teruggegeven.

Parameters
str cursor op string

Returnwaarde

Aantal tekens in string (zonder de nul aan het eind).

5.4.16.7 Str_Substr

String functies (bibliotheek *"String_Lib.cc"*)

Syntax

```
int Str_SubStr(char source[],char search[]);

Sub Str_SubStr(ByRef source As Char,ByRef search As Char) As Integer
```

Beschrijving

Een string search wordt gezocht in de string source. Als de gezochte tekenketen gevonden wordt, dan wordt de positie ervan teruggegeven.

Parameters

source string die doorzocht wordt
search tekenketen die gezocht wordt

Returnwaarde

Positie van de gezochte string in de onderzochte tekenketen.
-1 overig

5.4.16.8 Str_WriteFloat

String functies

Syntax

```
void Str_WriteFloat(float n, byte decimal, char text[], word offset);

Sub Str_WriteFloat(n As Single,decimal As Byte,ByRef text As Char,offset
As Word)
```

Beschrijving

Het float getal n wordt in een ASCII String met decimal decimale posities geconverteerd. Het resultaat wordt in de string text met een opvulling van offset opgeslagen.

Parameters

n	float getal
decimal	aantal decimale posities waarin n geconverteerd wordt
text	cursor op de doelstring
offset	aantal tekens waarmee de ASCII weergave van het float getal verschoven in de string gekopieerd wordt.

Als offset de waarde **STR_APPEND**(0xffff) heeft, dan wordt als offset de lengte van de doelstring aangenomen. In dit geval wordt het float getal aan de textstring gehangen.

5.4.16.9 Str_WriteInt

String functies

Syntax

```
void Str_WriteInt(int n, char text[], word offset);

Sub Str_WriteInt(n As Integer,ByRef text As Char,offset As Word)
```

Beschrijving

Het integere getal n wordt geconverteerd in een ASCII string met voorteken. Het resultaat wordt opgeslagen in de string text met een opvulling van offset.

Parameters

n	integer getal
text	cursor op de doelstring
offset	aantal tekens waarmee de ASCII weergave van het getal verschoven in de tekststring gekopieerd wordt

Als offset de waarde **STR_APPEND**(0xffff) heeft, dan wordt als offset de lengte van de doelstring aangenomen. In dit geval wordt het integere getal aan de textstring gehangen.

5.4.16.10 Str_WriteWord

String functies

Syntax

```
void Str_WriteWord(word n,byte base,char text[],word offset,byte
minwidth);

Sub Str_WriteWord(n As Word,base As Byte,ByRef text As Char,offset As
Word, minwidth As Byte)
```

Beschrijving

Het woord n wordt geconverteerd naar een ASCII string. Het resultaat wordt opgeslagen in de string text met een opvulling van offset. U kunt voor de uitvoer een willekeurige basis aangeven. Met een basis van 2 krijgt u binaire getallen, met 8 achttallige getallen en bij 16 worden er hexgetallen uitgegeven, etc. Als de basis groter is dan 16, dan worden er letters van het alfabet toegevoegd. Als bijv. de basis 18 is, dan heeft het getal de cijfers 0 – 9, en 'A' – 'H'. Als de ASCII string korter is dan minwidth, dan wordt het begin van de string opgevuld met nullen.

Parameters

n	16 Bit woord
base	Basis van het talstelsel
text	Cursor op de doelstring
offset	Aantal tekens waarmee de ASCII weergave het getal verschoven in de tekststring gekopieerd wordt
minwidth	minimale breedte van de string

Als offset de waarde **STR_APPEND**(0xffff) heeft, dan wordt als offset de lengte van de doelstring aangenomen. In dit geval wordt het integere getal aan de textstring gehangen.

5.4.17 Threads

Multithreading

Onder multithreading verstaat men het quasi parallel verwerken van meerdere processen in een programma. Eén van deze procedures wordt thread (= draad) genoemd. Bij multithreading wordt in snelle afstanden tussen de verschillende threads gewisseld, zodat bij de gebruiker de indruk van gelijktijdigheid ontstaat.

De C-Control Pro firmware ondersteunt behalve het hoofdprogramma (thread "0") maximaal 13 extra threads. Bij multithreading wordt na een bepaald aantal verwerkte byte instructies de actuele thread in de status "*Inactief*" gezet en wordt de volgende uitvoerbare thread gezocht. Daarna start de bewerking van de nieuwe thread. De nieuwe thread kan weer dezelfde als de vorige zijn, afhankelijk van hoeveel threads er geactiveerd zijn of voor een uitvoering klaar zijn. Het hoofdprogramma geldt als eerste thread. Daarom is thread "0" steeds actief, ook als er expliciet geen threads gestart zijn.

De prioriteit van een thread kan beïnvloed worden als u verandert hoeveel bytecodes een thread tot aan de volgende thread wisseling uit mag voeren (zie threadopties). Hoe kleiner het aantal cycli tot aan de wisseling, hoe geringer de prioriteit van de thread. De uitvoeringstijd van een bytecode is gemiddeld

7 – 9 µsec. Bij enkele bytecode commando's duurt het echter langer, bijv. Floating Point operaties.

➔ Ook interne interpreter –functies gelden als een cyclus. Omdat b.v. Serial Read wacht tot een teken van de seriële interface aankomt, kan in uitzonderingsgevallen een cyclus zeer lang duren.

Een thread krijgt voor zijn locale variabelen zoveel plaats als hem in de threadopties van het project toegewezen is. Een uitzondering is thread "0" (het hoofdprogramma). Deze thread krijgt de overige geheugenruimte, die de andere threads overlaten. U moet daarom vooraf plannen hoeveel geheugenruimte elke extra thread werkelijk nodig heeft.

➔ Opdat extra threads gestart kunnen worden, moet *"multithreading*" in de projectopties ingeschakeld worden, en moeten de parameters voor de andere threads in de threadopties op de correcte waarde gezet worden.

➔ Bij het werken met threads moet steeds Thread Delay en niet AbsDelay gebruikt worden. Als er toch bijv. een AbsDelay(1000) gebruikt wordt, leidt dit tot het volgende effect: opdat de thread pas na 5000 cycli (default waarde) naar de volgende thread wisselt, zou de thread 5000*1000ms (5000sec.) lopen tot de volgende thread zou kunnen werken.

Thread synchronisatie

Soms is het nodig dat een thread op de andere wacht. Dit kan bijv. een gemeenschappelijke hardware bron zijn, die alleen één thread kan bewerken. Of soms definieert men een kritisch programmabereik, dat slechts één thread mag betreden. Deze functies worden gerealiseerd door de aanwijzingen Thread Wait en Thread Signal.

Een thread die moet wachten, voert de aanwijzing Thread_Wait uit met een signaalnummer. De toestand van de thread wordt op *wachtend* gezet. Dit betekent dat deze thread bij een mogelijke wisseling van thread overgeslagen wordt. Als de andere thread zijn kritische werk beëindigd heeft, geeft hij het commando Thread_Signal met hetzelfde signaalnummer dat de andere thread voor Thread_Wait gebruikt heeft. De thread –toestand van de wachtende thread verandert dan van *wachtend* in *inactief*. Nu wordt hij bij een mogelijke thread wisseling weer "meegenomen".

Deadlocks

Als alle threads zich in een wachttoestand begeven met Thread Wait, dan is er geen thread meer die de andere threads uit de wachtende toestand zou kunnen bevrijden. Deze constellaties dient u bij de programmering te vermijden.

Tabel thread –toestanden

Toestand	**Betekenis**
actief	De thread wordt op dit moment bewerkt
inactief	Kan na een thread wisseling weer geactiveerd worden
slapend	Wordt na een aantal ticks weer op "inactief" gezet
wachtend	De thread wacht op een signaal

5.4.17.1 Thread_Cycles

Thread functies

Syntax

```
void Thread_Cycles(byte thread,word cycles);

Sub Thread_Cycles(thread As Byte,cycles As Word)
```

Beschrijving

Zet het aantal bytecode instructies tot aan de volgende thread -wisseling op cycles.

➔ Als een thread opnieuw gestart wordt, krijgt hij steeds het aantal cycli toegewezen die in de projectopties gedefinieerd zijn. Het heeft dus alleen maar zin om Thread_Cycles() op te roepen nadat een thread gestart is.

Parameters

thread	(0-13) nummer van de thread waarvan de cyclus veranderd moet worden
cycles	aantal cycli tot aan het wisselen van de thread

5.4.17.2 Thread_Delay

Thread functies Voorbeeld

Syntax

```
void Thread_Delay(word delay);

Sub Thread_Delay(delay As Word)
```

Beschrijving

Hiermee wordt een thread voor een bepaalde tijd op *"slapend"* geschakeld. Na de aangegeven periode is hij weer klaar voor de verwerking. De periode wordt aangegeven in ticks, die door timer 2 geproduceerd worden. Als timer 2 uitgeschakeld wordt of voor een ander doel wordt gebruikt, is de functiewijze van Thread_Delay() ongedefinieerd.

➔ Ook als Thread_Delay() normaalgesproken als een wachtfunctie werkt, moet u er toch aan denken dat na de wachttijd de thread niet steeds automatisch weer uitgevoerd wordt. Hij is dan weliswaar klaar voor gebruik, maar moet eerst door een wisseling van thread weer tijd voor uitvoering krijgen.

Parameters

delay	aantal van 10ms ticks dat gewacht moet worden

5.4.17.3 Thread_Info

Thread functies

Syntax

```
word Thread_Info(byte info);

Sub Thread_Info(info As Byte) As Word
```

Beschreibung

Levert informatie over de thread, die de functie Thread_Info oproept. De info parameter bepaald welke informatie teruggegeven wordt,

Parameter

info waarden:

TI_THREADNUM	Nummer van de op te roepen thread
TI_STACKSIZE	Gedefinieerde stackgrootte
TI_CYCLES	Aantal van de uit te voeren cycli voor een threadwisseling

Returnwaarde

Aangevraagde parameter

5.4.17.4 Thread_Kill

Thread functies

Syntax

```
void Thread_Kill(byte thread);

Sub Thread_Kill(thread As Byte)
```

Beschrijving

Beëindigt de verwerking van een thread. Wanneer als threadnummer 0 wordt doorgegeven, dan wordt het hoofdprogramma en daarmee de gehele interpreter -loop gestopt.

Parameter

thread (0-13) nummer van de thread

5.4.17.5 Thread_Lock

Thread functies

Syntax

```
void Thread_Lock(byte lock);

Sub Thread_Lock(lock As Byte)
```

Beschrijving

Met deze functie kan een thread zijn thread -wisseling verhinderen. Dit is zinvol als bij een serie poort - uitvoeren of andere hardware commando's de tijdelijke scheiding door een thread -wisseling vermeden moet worden.

➔ Als er vergeten wordt het “Lock” (de vergrendeling) weer uit te schakelen, vindt er geen multithreading meer plaats.

Parameter

lock bij 1 wordt de thread -wisseling verhinderd, bij 0 weer toegelaten.

5.4.17.6 Thread_MemFree

Thread functies

Syntax

```
word Thread_MemFree(void);

Sub Thread_MemFree() As Word
```

Beschrijving

Geeft het vrije geheugen aan die voor de thread nog beschikbaar is.

Parameter

Geen

Returnwaarde

Vrij geheugen in bytes

5.4.17.7 Thread_Resume

Thread functies

Syntax

```
void Thread_Resume(byte thread);

Sub Thread_Resume(thread As Byte)
```

Beschrijving

Als een thread zich in de toestand “*wachtend*” bevindt, kan hij hiermee weer op ”*inactief*” gezet worden. De status “inactief” betekent, dat de thread klaar is om bij een thread – wisseling weer geactiveerd te worden.

Parameter

thread (0-13) nummer van de thread

5.4.17.8 Thread_Signal

Thread functies

Syntax

```
void Thread_Signal(byte signal);

Sub Thread_Signal(signal As Byte)
```

Beschrijving
Als een thread d.m.v. Thread Wait() op "*wachtend*" is gezet, dan kan de toestand met behulp van Thread_Signal weer in "*inactief*" veranderd worden. De parameter signal moet dezelfde waarde hebben die bij Thread Wait() gebruikt is.

Parameter

signal waarde van het signaal

5.4.17.9 Thread_Start

Thread functies Voorbeeld

Syntax

```
void Thread_Start(byte thread,float func);

Sub Thread_Start(Byte thread As Byte,func As Single)
```

Beschrijving

Er wordt een nieuwe thread gestart. Als startfunctie voor de thread kan een willekeurige functie gebruikt worden.

➔ Als er een functie uitgezocht wordt die overdracht –parameters bevat, dan is bij de start van de thread de inhoud van deze parameters niet gedefinieerd!

Parameters

thread (0-13) nummer van de thread die gestart moet worden
func naam van de functie waarin de nieuwe thread gestart wordt

Opmerking
De zwevende komma- datatype kan niet gepast verschijnen, maar hij wordt intern als 4 byte waarde behandelt. Een functie- aanwijzer moet vanaf de Mega128 ondersteuning meer zijn dan 16 bit.

5.4.17.10 Thread_Wait

Thread functies

Syntax

```
void Thread_Wait(byte signal);
Sub Thread_Wait(signal As Byte)
```

Beschrijving

De thread krijgt de status "*wachtend*". D.m.v. Thread Resume() of Thread Signal() kan de thread weer in een inactieve toestand terechtkomen.

Parameter

signal waarde van het signaal

5.4.17.11 Thread voorbeeld

```
// Demoprogramma voor multithreading
// het programma is niet gedempt, het kort indrukken van een toets leidt daarom tot
// meervoudige invoer van de string

void thread1(void)
{
      while(true) // eindeloze lus
      {
            if(!Port_ReadBit(PORT_SW2)) Msg_WriteText(str2); // SW2 werd
            ingedrukt
      }
}
char str1[12],str2[12];

void main(void)
{
      str1="Taster 1";
      str2="Taster 2";

      Port_DataDirBit(PORT_SW1, PORT_IN); // Pin op ingang
      Port_DataDirBit(PORT_SW2, PORT_IN); // Pin op ingang
      Port_WriteBit(PORT_SW1, 1); // Pullup zetten
      Port_WriteBit(PORT_SW1, 1); // Pullup zetten

      Thread_Start(1,thread1); // Thread 1 starten

      while(true) // eindeloze lus
      {
            if(!Port_ReadBit(PORT_SW1)) Msg_WriteText(str1); // SW1 werd
            ingedrukt
      }
}
```

5.4.17.12 Thread voorbeeld 2

```
// multithread2: Multithreading met Thread_Delay
// benodigde Library: IntFunc_Lib.cc

void thread1(void)
{
      while(true)
      {
            Msg_WriteText(str2); Thread_Delay(200);
      }                                 // "Thread2" wordt gegeven.
      }                                 // Daarna "slaapt"de thread
                                        // voor 200ms.
char str1[12],str2[12];                 // globale variablendeclaratie

//-------------------------------------------------------------------------
// Hoofdprogramma
//
void main(void)
{
      str1="Thread1";                   // Variablendeclaratie
      str2="Thread2";                   // Variablendeclaratie
      Thread_Start(1,thread1);          // Functie- oproep met aangeven van de
                                       // threadnummer.

      while(true)                       // Eindeloze lus
      {
            Thread_Delay(100); Msg_WriteText(str1);
      }                                 // De thread os "slapend" voor 100ms.
}                                       // Daarna wordt "Thread1" uitgevoerd.
```

5.4.18 Timer

Er staan in de C-Control Pro Mega 32 twee , Mega128 drie onafhankelijke Timer-Counters ter beschikking: *Timer_0* met 8 bit en *Timer_1* met 16 Bit en *Timer_3* met 16 Bit (alleen Mega128). *Timer_2* wordt door de firmware als interne tijdbasis gebruikt, en is vast ingesteld op een 10ms interrupt. U kunt de interne timers voor veelvuldige opgaven inzetten:

- Teller van gebeurtenissen
- Produceren van frequenties
- Pulsbreedte -modulatie
- Timerfuncties
- Puls - & periodemeting
- Frequentiemeting

5.4.18.1 Teller van gebeurtenissen

Hier twee voorbeelden hoe de timer als teller van gebeurtenissen gebruikt worden:

Timer0 (8 Bit)

```
//  Voorbeeld: pulstelling met CNT0
Timer_T0CNT();
```

```
pulse(n);                  //   n pulsen genereren
count=Timer_T0GetCNT();
```

➔ Bij de Mega128 kan vanwege de hardware Timer_0 niet als teller gebruikt worden.

Timer1(16 Bit)

```
//  Voorbeeld: pulstelling met CNT1
Timer_T1CNT();
pulse(n);                  //   n pulsen genereren
count=Timer_T1GetCNT();
```

5.4.18.2 Produceren van frequenties

Voor het produceren van frequenties kunnen *Timer_0* en *Timer_1* als volgt gebruikt worden:

Timer0 (8 Bit)

1e voorbeeld:

```
Timer_T0FRQ(10,  ps_8)       // Rechthoeksignaal met 10*1,085  µs = 10,85 µs periodeduur
```

2e voorbeeld: gepulste frequentieblokken (project FRQ0)

```
void main(void)
{
      int delval;                    // Variable v.d. in-/uitschakeltijd
      delval=200;                    // Waarde toewijzing variablen delval
      Timer_T0FRQ(100,PS0_1024);     // De timer wordt op de frequentie
                                     // Periode=138,9 µs*100=13,9 ms,
                                     Frequenz= 72Hz
      while (1)
      {
            AbsDelay(delval);        // tijdvertraging 200ms
            Timer_T0Stop();          // De timer wordt gestopt.
            AbsDelay(delval);        // Tijdvertraging 200ms
            Timer_T0Start(PS0_1024);// De timer wordt met de timer
                                     // Prescaler PS0_1024 eingeschaltet.
      }
}
```

➔ Het programma kan op de **Mega128** niet in de USB- modus werken omdat de uitgang PB4 in samenhang met de USB interface op het apllication board gebruikt wordt.

Timer1 (16 Bit)

1e voorbeeld: produceren van frequenties met 125*4,34µs = 1085µs periode

```
Timer_T1FRQ(125,ps_64);
```

2e voorbeeld: produceren van frequenties met 10*1,085 µs = 10,85 µs periode en 2*1,085µs = 2,17 µs faseverschuiving

```
Timer_T1FRQX(10,2,ps_8);
```

5.4.18.3 Frequentiemeting

Voor het direct meten van een frequentie kan de Timer1(16Bit) gebruikt worden. De pulsen binnen een seconde worden geteld en het resultaat is dan in Hertz. De maximale meetfrequentie is 64kHz en wordt geleverd door de 16Bit teller. Een voorbeeld van deze manier van frequentiemeting vindt u onder “Demo programma’s/Frequentiemeting”. Door het verkorten van de meettijd kunnen ook hogere frequenties gemeten worden. Het resultaat moet dan dienovereenkomstig omgerekend worden.

5.4.18.4 Pulsbreedte –modulatie

Er staan voor de pulsbreedte –modulatie twee timers tot uw beschikking: *Timer_0* met 8 bit en *Timer_1* met 16 Bit. Met een pulsbreedte –modulatie kunt u heel gemakkelijk een digitaal –analoog –omvormer realiseren.

Timer0 (8 Bit)
Voorbeeld: Pulsbreedte –modulatie met 138,9 µs periode en 5,42 µs pulsbreedte, veranderd naar 10,84 µs pulsbreedte

```
Timer_T0PWM(10,2);     // Puls: 10*542,5 ns  =  5,42 µs,  periode: 256*542,5  ns  = 138,9 µs
Timer_T0PW(20);     // Puls: 20*542,5 ns  = 10,84 µs
```

Timer1 (16 Bit)
Voorbeeld: pulsbreedte –modulatie met 6,4 ms periode en 1,28 ms pulsbreedte kanaal A en 640 µs pulsbreedte kanaal B

```
Timer_T0PWMX(10,20,10,ps_1024);   // Periode:  100*69,44 µs  =  6,94  ms
                                  // PulsA: 20*69,44 µs  =  1,389  ms
                                  // PulsB: 10*69,44 µs  =  694,4  µs
```

5.4.18.5 Puls & periodemeting

Met *Timer_1* kunnen pulsbreedtes of signaalperiodes gemeten worden. Met behulp van de Input Capture functie (speciaal register van de Controller) wordt de tijd tussen twee flanken gemeten. Deze functie gebruikt de Capture -Interrupt (INT_TIM1CAPT). De puls wordt gemeten tussen een stijgende en de volgende vallende flank. De periode wordt gemeten tussen twee stijgende flanken. Door de Input Capture functie worden programma –looptijden niet als onnauwkeurigheid in het meetresultaat ingevoegd. Met de programmeerbare voordeler kan de resolutie van de *Timer_1* vastgelegd worden. Voordeler zie Tabel.

Voorbeeld: pulsbreedtemeting (Project Pmeting) 434 µs (100*4,34 µs, zie Tabel) inschakelen

```
word  PM_waarde;

void  Timer1_ISR(void)
{
     int  irqcnt;

     PM_waarde=Timer_T1GetPM(0);          // Pulsbreedte meten
     irqcnt=Irq_GetCount(INT_TIM1CAPT);
}
```

```
void  main(void)
{
     byte n;

     Irq_SetVect(INT_TIM1CAPT,Timwer1_ISR);   // Interrupt Service routine definiëren
     Timer_T0PWM(100,ps_64);                  // pulsgenerator starten
     // De meting begint hier
     // Output timer0 OC0(Port B.3) verbinden met ICP (Input Capture Pin) (Port D.6)

     PM_waarde=0;
     Timer_T1PM(ps_64);                       // Voordeler voor meting vastleggen

     while(PM_waarde==0);                     // Pulsbreedte of periode meten

     Mag_WriteHex(PM_waarde);                 // Meetwaarde afgeven
}
```

➔Vanwege de overzichtelijkheid is hier een eenvoudige versie aangegeven. Bij de **Mega128** wordt vanwege een conflict op pin B4 de *Timer_0* voor de pulsproductie gebruikt. Het volledige programma vindt u in de map PW_Messung.

5.4.18.6 Timerfuncties

Er staan twee onafhankelijke timers tot uw beschikking: *Timer_0* met 8 bit en *Timer_1* met 16 Bit. De timers beschikken over een programmeerbare voordeler, zie tabel. Met de timer kunt u een tijd vastleggen, nadat een interrupt getriggerd is. In de interrupt -routine kunnen dan verschillende bewerkingsstappen uitgevoerd worden.

Timer_T0Time (8 Bit)
Voorbeeld: Timer0: uitgang met een vertraging van 6,94 ms (100x69,44 µs, zie tabel) inschakelen

```
Void  Timer0_ISR(void)
{
          int  irqcnt;
          Port _WriteBit(0,1);
          Timer_T0Stop();                    //  Timer0 stoppen
          irqcnt=Irq_GetCount(INT_TIM0COMP);
}

void  main(void)
{
     Port _DataDirBit(0,0);                  // PortA.0  uitgang
     Port _WriteBit(0,0);                    // PortA.0  uitgang = 0
     Irq_SetVect(INT_TIM0MP,Timer0_ISR);   // Interrupt service routine
                                             definiëren
     TimerT0Time(100,ps_1024);               // Tijd vastleggen en timer0
                                             starten
     //  verdere verloop programma …
}
```

5.4.18.7 Timer_Disable

Timer functies

Syntax

```
void Timer_Disable(byte timer);

Sub Timer_Disable(timer As Byte)
```

Beschrijving

Deze functie schakelt de geselecteerde timer uit. Timerfuncties bezetten I/O poorten. Als een timer niet meer nodig is en de poorten moeten als normale I/Os gebruikt worden, dan moet de timerfunctie uitgeschakeld worden.

Parameters

0 = *Timer_0*
1 = *Timer_1*
3 = *Timer_3* (alleen Mega128)

5.4.18.8 Timer_T0CNT

Timer functies

Syntax

```
void Timer_T0CNT(void);

Sub Timer_T0CNT()
```

Beschrijving

Deze functie initialiseert de Counter0. De Counter0 wordt bij een positieve signaalflank op de ingang **Mega32**:T0 (PIN1) opgehoogd.

➔Bij de **Mega128** kan vanwege de hardware *Timer_0* niet als teller gebruikt worden.

Parameters

Geen

5.4.18.9 Timer_T0FRQ

Timer functies

Syntax

```
void Timer_T0FRQ(byte period,byte PS);

Sub Timer_T0FRQ(period As Byte,PS As Byte)
```

Beschrijving

Deze functie initialiseert de Timer0 met de aangegeven voordeler en periodeduur, zie tabel.
Het uitgangssignaal verschijnt op Port B.3 (PIN4) = **Mega32,** PortB.4 (X1_4) = **Mega128..**
Het produceren van de frequentie wordt automatisch gestart. De Mega128 beschikt over uitgebreide voordeler- definities, zie tabel.

Parameters

period periodeduur
PS voordeler

Tabel prescaler: Mega 32

Voordeler (prescaler)	Tijdbasis (duur van een tick)
PS0_1 (1)	135,6 ns
PS0_8 (2)	1,085 µs
PS0_64 (3)	8,681 µs
PS0_256 (4)	34,72 µs
PS0_1024 (5)	138,9 µs

Mega128

Voordeler (prescaler)	Tijdbasis (duur van een tick)
PS0_1 (1)	135,6 ns
PS0_8 (2)	1,085 µs
PS0_32(3)	4,340 µs
PS0_64 (4)	8,681 µs
PS0_128 (5)	17,36 µs
PS0_256 (6)	34,72 µs
PS0_1024 (7)	138,9 µs

5.4.18.10 Timer_T0GetCNT

Timer functies

Syntax

```
byte Timer_T0GetCNT(void);

Sub Timer_T0GetCNT() As Byte
```

Beschrijving

De waarde van Counter0 wordt gelezen. Als er een overflow plaats vindt, dan wordt de waarde 0xFF doorgegeven.
➔Bij de **Mega128** kan vanwege de hardware *Timer_0* niet als teller gebruikt worden.

Returnwaarde

De gemeten waarde van de teller

5.4.18.11 TimerT0PW

Timer functies

Syntax

```
void Timer_T0PW(byte PW);

Sub Timer_T0PW(PW As Byte)
```

Beschrijving
Deze functie stelt een nieuwe pulsbreedte voor Timer0 in, zonder de voordeler te veranderen.

Parameters

PW pulsbreedte

5.4.18.12 TimerT0PWM

Timer functies

Syntax

```
void Timer_T0PWM(byte PW,byte PS);

Sub Timer_T0PWM(PW As Byte,PS As Byte)
```

Beschrijving
Deze functie initialiseert deTimer0 met de aangegeven voordeler en pulsbreedte, zie tabel.
Het uitgangssignaal verschijnt bij de **Mega32**:op Port B.3 (PIN4) **Mega128**: PortB.4 (X1_4).
De Mega128 beschikt over uitgebreide voordeler- definities, zie tabel.

Parameters

PW pulsbreedte
PS voordeler

Tabel prescaler: Mega32

Voordeler (prescaler)	Tijdbasis (duur van een tick)
PS_1 (1)	67,8 ns
PS_8 (2)	542,5 ns
PS_64 (3)	4,34 µs
PS_256 (4)	17,36 µs
PS_1024 (5)	69,44 µs

Mega128

Voordeler (prescaler)	Tijdbasis (duur van een tick)
PS0_1 (1)	135,6 ns
PS0_8 (2)	542,5 ns
PS0_32(3)	2,17 µs
PS0_64 (3)	4,34 µs
PS0_128 (5)	8,68 µs
PS0_256 (6)	17,36 µs
PS0_1024 (5)	69,44 µs

5.4.18.13 Timer_T0Start

Timer functies

Syntax

```
void Timer_T0Start(byte prescaler);

Sub Timer_T0Start(prescaler As Byte)
```

Beschrijving

Het produceren van de frequentie wordt met de bovenstaande instelling gestart. De voordeler moet nieuw aangegeven worden.

Parameters

Prescaler voordeler (tabel prescaler)

5.4.18.14 Timer_T0Stop

Timer functies

Syntax

```
void Timer_T0Stop(void);

Sub Timer_T0Stop()
```

Beschrijving

Het produceren van de frequentie wordt gestopt. Het uitgangssignaal kan 0 of 1 zijn, afhankelijk van de laatste toestand. Alleen de klokpuls voor de timer wordt gestopt. De overige instellingen blijven behouden.

Parameters

Geen

5.4.18.15 Timer_T0Time

Timer functies

Syntax

```
void Timer_T0Time(byte Time,byte PS);

Sub Timer_T0Time(Time As Byte,PS As Byte)
```

Beschrijving

Deze functie initialiseert de Timer0 met de aangegeven voordeler en de waarde (8Bit) voor de tijd, zie tabel. Als deze waarde bereikt is, dan wordt de Timer0 Interrupt (INT_TIM0COMP) geactiveerd. De Mega128 beschikt over uitgebreide voordeler- definities, zie tabel.

Parameters

Time tijdwaarde waarbij de interrupt geactiveerd wordt
PS voordeler

Tabel prescaler: Mega32

Voordeler (prescaler)	Tijdbasis (duur van een tick)
PS_1 (1)	67,8 ns
PS_8 (2)	542,5 ns
PS_64 (3)	4,34 µs
PS_256 (4)	17,36 µs
PS_1024 (5)	69,44 µs

Mega128

Voordeler (prescaler)	Tijdbasis (duur van een tick)
PS0_1 (1)	67,8 ns
PS0_8 (2)	542,5 ns
PS0_32(3)	2,17 µs
PS0_64 (3)	4,34 µs
PS0_128 (5)	8,68 µs
PS0_256 (6)	17,36 µs
PS0_1024 (5)	69,44 µs

5.4.18.16 Timer_T1CNT

Timer functies

Syntax

```
void Timer_T1CNT(void);

Sub Timer_T1CNT()
```

Beschrijving

Deze functie initialiseert de Counter1. De Counter1 wordt bij een positieve signaalflank op de ingang bij de **Mega32**:op Port B.1 (PIN2) **Mega128**: PortD.6 (X2_15) opgehoogd.

Parameters

Geen

5.4.18.17 Timer_T1CNT_Int

Timer functies

Syntax

```
void Timer_T1CNT_Int(word limit);

Sub Timer_T1CNT_Int(limit As Word)
```

Beschrijving

Deze functie initialiseert de Counter1. De Counter1 wordt bij een positieve signaalflank op de ingang bij de **Mega32**:op Port B.1 (PIN2) **Mega128**: PortD.6 (X2_15) opgehoogd. Als de limiet bereikt is, wordt een interrupt geactiveerd. De desbetreffende interrupt_Service_Routine moet vooraf gedefinieerd zijn.

Parameters

limit

5.4.18.18 Timer_T1FRQ

Timer functies

Syntax

```
void Timer_T1FRQ(word period,byte PS);

Sub Timer_T1FRQ(period As Word,PS As Byte)
```

Beschrijving

Deze functie initialiseert de Timer1 met de aangegeven voordeler en periodeduur, zie tabel. Het uitgangssignaal verschijnt bij de **Mega32**:op Port D.5 (PIN19) **Mega128**: PortB.5 (X1_3) Het produceren van de frequentie wordt automatisch gestart.

Parameters

period periodeduur
PS voordeler

Tabel prescaler:

Voordeler (prescaler)	Tijdbasis (duur van een tick)
PS_1 (1)	135,6 ns
PS_8 (2)	1,085 µs
PS_64 (3)	8,681 µs
PS_256 (4)	34,72 µs
PS_1024 (5)	138,9 µs

5.4.18.19 Timer_T1FRQX

Timer functies

Syntax

```
void Timer_T1FRQX(word period,word skew,byte PS);

Sub Timer_T1FRQX(period As Word,skew As Word,PS As Byte)
```

Beschrijving
Deze functie initialiseert de Timer1 met de aangegeven voordeler periodeduur en fase-verschuiving van de beide uitgangssignalen, zie tabel. De uitgangssignalen verschijnen op **Mega 32:** PortD.4 (PIN18) en PortD.5(PIN19). **Mega128**: PortB.5(X1_3) en PortB.6 (X1_2). Het produceren van de frequentie wordt automatisch gestart. De waarde voor de faseverschuiving moet kleiner zijn dan de halve periode.

Parameters

period periodeduur
skew faseverschuiving
PS voordeler (tabel prescaler)

5.4.18.20 Timer_T1GetCNT

Timer functies

Syntax

```
word Timer_T1GetCNT(void);

Sub Timer_T1GetCNT() As Word
```

Beschrijving

De waarde van Counter1 wordt gelezen. Als er een overflow plaats vindt, dan wordt de waarde 0xFFFF doorgegeven.

Returnwaarde

De gemeten waarde van de teller

5.4.18.21 Timer_T1GetPM

Timer functies

Syntax

```
word Timer_T1GetPM(byte Mode);

Sub Timer_T1GetPM(Mode As Byte) As Word
```

Beschrijving

Deze functie legt vast of er een pulsbreedte – of periodemeting moet worden uitgevoerd, en levert het meetresultaat op.

Parameter

Mode
0 pulsbreedte -meting
1 periodemeting

Returnwaarde:
Resultaat van de meting

➔ Om het meetresultaat te berekenen wordt de geretourneerde 16bit waarde met de waarde uit de tabel prescaler tabel gemultipliceert, die bij het oproepen van Timer_T1PM aangegeven werd (zie ook voorbeeld).

5.4.18.22 Timer_T1PWA

Timer functies

Syntax

```
void Timer_T1PWA(word PW0);

Sub Timer_T1PWA(PW0 As Word)
```

Beschrijving:

Deze functie stelt een nieuwe pulsbreedte (kanaal_A) in voor Timer1, zonder de voordeler te veranderen.

Parameter

PW0 pulsbreedte

5.4.18.23 Timer_T1PM

Timer functies

Syntax

```
void Timer_T1PM(byte PS);

void Timer_T1PM(PS As Byte)
```

Beschrijving

Deze functie initialiseert Timer_1 voor de meting en stelt de voordeler in.

Parameter

PS voordeler

Tabel prescaler:

Voordeler (prescaler)	Tijdbasis (duur van een tick)
PS_1 (1)	67,8 ns
PS_8 (2)	542,5 ns
PS_64 (3)	4,34 µs
PS_256 (4)	17,36 µs
PS_1024 (5)	69,44 µs

5.4.18.24 Timer_T1PWB

Timer functies

Syntax

```
void Timer_T1PWB(word PW1);

Sub Timer_T1PWB(PW1 As Word)
```

Beschrijving:

Deze functie stelt een nieuwe pulsbreedte (kanaal_B) in voor Timer1, zonder de voordeler te veranderen.

Parameter

PW1 pulsbreedte

5.4.18.25 Timer_T1PWM

Timer functies

Syntax

```
void Timer_T1PWM(word period,word PW0,byte PS);

Sub Timer_T1PWM(period As Word,PW0 As Word,PS As Byte)
```

Beschrijving:

Deze functie initialiseert Timer1 met de aangegeven voordeler, periodeduur en faseverschuiving van beide uitgangssignalen. **Mega32**: PortD.4 (PIN18) en PortD.5 (PIN19). **Mega128**: PortB.5 (X1_3) en PortB.6 (X1_2). Het produceren van de frequentie wordt automatisch gestart. De waarde voor de faseverschuiving moet kleiner zijn dan de halve periode.

Parameters

period periodeduur
PW0 pulsbreedte
PS voordeler

Tabel prescaler:

Voordeler (prescaler)	Tijdbasis (duur van een tick)
PS_1 (1)	67,8 ns
PS_8 (2)	542,5 ns
PS_64 (3)	4,34 µs
PS_256 (4)	17,36 µs
PS_1024 (5)	69,44 µs

5.4.18.26 Timer_T1PWMX

Timer functies

Syntax

```
void Timer_T1PWMX(word period,word PW0,word PW1,byte PS);

Sub Timer_T1PWMX(period As Word,PW0 As Word,PW1 As Word,PS As Byte)
```

Beschrijving:

Deze functie initialiseert Timer1 met de aangegeven voordeler, pulsbreedte voor kanaal A en B en periodeduur, zie tabel. Het uitgangssignaal verschijnt op **Mega32**: PortD.4 (PIN18) en PortD.5(PIN19). **Mega128**: PortB.5 (X1_3) en PortB.6 (X1_2).

Parameters

period periodeduur
PW0 pulsbreedte kanaal A
PW1 pulsbreedte kanaal B
PS voordeler (tabel prescaler)

5.4.18.27 Timer_T1PWMY

Timer functies

Syntax

```
void Timer_T1PWMY(word period,word PW0,word PW1,word PW2,byte PS);

Sub Timer_T1PWMY(period As Word,PW0 As Word,PW1 As Word,PW2 As Word,PS As
Byte)
```

Beschrijving:

Deze functie initialiseert Timer1 met de aangegeven voordeler, pulsbreedte voor kanaal A en B, C en periodeduur, zie tabel. De uitgangssignalen verschijnen op PortB.5 (X1_3), PortB.6 (X1_2) en PortB.7 (X_1).

Parameters

period periodeduur
PW0 pulsbreedte kanaal A
PW1 pulsbreedte kanaal B
PW2 pulsbreedte kanaal C
PS voordeler (tabel prescaler)

5.4.18.28 Timer_T1Start

Timer functies

Syntax

```
void Timer_T1Start(byte prescaler);

Sub Timer_T1Start(prescaler As Byte)
```

Beschrijving

Het produceren van frequenties wordt met bovenstaande instelling gestart. De voordeler moet nieuw aangegeven worden.

Parameters

prescaler voordeler (tabel prescaler)

5.4.18.29 Timer_T1Stop

Timer functies

Syntax

```
void Timer_T1Stop(void);

Sub Timer_T1Stop()
```

Beschrijving

Het produceren van frequenties wordt gestopt. Het uitgangssignaal kan 0 of 1 zijn, afhankelijk van de laatste toestand. Alleen de puls voor de timer wordt gestopt. De overige instellingen blijven behouden.

Parameters

Geen

5.4.18.30 Timer_T1Time

Timer functies

Syntax

```
void Timer_T1Time(word Time,byte PS);

Sub Timer_T1Time(Time As Word,PS As Byte)
```

Beschrijving

Deze functie initialiseert Timer1 met de aangegeven voordeler en de waarde (16Bit) voor de tijd, zie tabel. Als deze waarde wordt bereikt, dan wordt de Timer1-interrupt (INT_TIM1COMP) geactiveerd.

Parameters

Time tijdswaarde waarbij de interrupt getriggerd wordt
PS voordeler

Voordeler (prescaler)	Tijdbasis (duur van een tick)
PS_1 (1)	67,8 ns
PS_8 (2)	542,5 ns
PS_64 (3)	4,34 µs
PS_256 (4)	17,36 µs
PS_1024 (5)	69,44 µs

5.4.18.31 Timer_T3CNT

Timer functies

Syntax

```
void Timer_T3CNT(void);

Sub Timer_T3CNT()
```

Beschrijving

Deze functie initialiseert de Counter3. De Counter3 wordt bij een positieve signaalflank op de ingang PortE.6 (X2_10) opgehoogd.

Parameters

Geen

5.4.18.32 Timer_T3CNT_Int

Timer functies

Syntax

```
void Timer_T3CNT_Int(word limit);

Sub Timer_T3CNT_Int(limit As Word)
```

Beschrijving

Deze functie initialiseert de Counter3. De Counter3 wordt bij een positieve signaalflank op de ingang PortE.6 (X2_10) opgehoogd. Als de limiet bereikt is, wordt een interrupt geactiveerd. De desbetreffende interrupt_Service_Routine moet vooraf gedefinieerd zijn.

Parameters

limit

5.4.18.33 Timer_T3FRQ

Timer functies

Syntax

```
void Timer_T3FRQ(word period,byte PS);

Sub Timer_T3FRQ(period As Word,PS As Byte)
```

Beschrijving

Deze functie initialiseert de Timer3 met de aangegeven voordeler en periodeduur, zie tabel. Het uitgangssignaal verschijnt PortE.3 (X1_13) Het produceren van de frequentie wordt automatisch gestart.

Parameters

period periodeduur
PS voordeler

Tabel prescaler:

Voordeler (prescaler)	Tijdbasis (duur van een tick)
PS_1 (1)	135,6 ns
PS_8 (2)	1,085 µs
PS_64 (3)	8,681 µs
PS_256 (4)	34,72 µs
PS_1024 (5)	138,9 µs

5.4.18.34 Timer_T3FRQX

Timer functies

Syntax

```
void Timer_T3FRQX(word period,word skew,byte PS);

Sub Timer_T3FRQX(period As Word,skew As Word,PS As Byte)
```

Beschrijving

Deze functie initialiseert de Timer3 met de aangegeven voordeler periodeduur en fase-verschuiving van de beide uitgangssignalen, zie tabel. De uitgangssignalen verschijnen op PortE.3 (X1_13) en PortE.4 (X1_12).. Het produceren van de frequentie wordt automatisch gestart. De waarde voor de faseverschuiving moet kleiner zijn dan de halve periode.

Parameters

period periodeduur
skew faseverschuiving
PS voordeler (tabel prescaler)

5.4.18.35 Timer_T3GetCNT

Timer functies

Syntax

```
word Timer_T3GetCNT(void);

Sub Timer_T3GetCNT() As Word
```

Beschrijving

De waarde van Counter3 wordt gelezen. Als er een overflow plaats vindt, dan wordt de waarde 0xFFFF doorgegeven.

Returnwaarde

De gemeten waarde van de teller

5.4.18.36 Timer_T3GetPM

Timer functies

Syntax

```
word Timer_T3GetPM(byte Mode);

Sub Timer_T3GetPM(Mode As Byte) As Word
```

Beschrijving

Deze functie legt vast of er een pulsbreedte – of periodemeting moet worden uitgevoerd, en levert het meetresultaat op.

Parameters

Mode
0 pulsbreedte -meting
1 periodemeting

Returnwaarde:
Resultaat van de meting

➔ Om het meetresultaat te berekenen wordt de geretourneerde 16bit waarde met de waarde uit de tabel prescaler tabel gemultipliceert, die bij het oproepen van Timer_T3PM aangegeven werd (zie ook voorbeeld).

5.4.18.37 Timer_T3PWA

Timer functies

Syntax

```
void Timer_T3PWA(word PW0);

Sub Timer_T3PWA(PW0 As Word)
```

Beschrijving:
Deze functie stelt een nieuwe pulsbreedte (kanaal_A) in voor Timer3, zonder de voordeler te veranderen.

Parameter

PW0 pulsbreedte

5.4.18.38 Timer_T3PM

Timer functies

Syntax

```
void Timer_T3PM(byte PS);

void Timer_T3PM(PS As Byte)
```

Beschrijving

Deze functie initialiseert *Timer_3* voor de meting en stelt de voordeler in.

Parameter

PS voordeler

Tabel prescaler:

Voordeler (prescaler)	Tijdbasis (duur van een tick)
PS_1 (1)	67,8 ns
PS_8 (2)	542,5 ns
PS_64 (3)	4,34 µs
PS_256 (4)	17,36 µs
PS_1024 (5)	69,44 µs

5.4.18.39 Timer_T3PWB

Timer functies

Syntax

```
void Timer_T3PWB(word PW1);

Sub Timer_T3PWB(PW1 As Word)
```

Beschrijving:

Deze functie stelt een nieuwe pulsbreedte (kanaal_B) in voor Timer3, zonder de voordeler te veranderen.

Parameter

PW1 pulsbreedte

5.4.18.40 Timer_T3PWM

Timer functies

Syntax

```
void Timer_T3PWM(word period,word PW0,byte PS);

Sub Timer_T3PWM(period As Word,PW0 As Word,PS As Byte)
```

Beschrijving:

Deze functie initialiseert Timer3 met de aangegeven voordeler, pulsbreedte en periodeduur, Het uitgangssignaal verschijnt PortE.3 (X1_13.

Parameters

period periodeduur
PW0 pulsbreedte
PS voordeler

Tabel prescaler:

Voordeler (prescaler)	Tijdbasis (duur van een tick)
PS_1 (1)	67,8 ns
PS_8 (2)	542,5 ns
PS_64 (3)	4,34 µs
PS_256 (4)	17,36 µs
PS_1024 (5)	69,44 µs

5.4.18.41 Timer_T3PWMX

Timer functies

Syntax

```
void Timer_T3PWMX(word period,word PW0,word PW1,byte PS);

Sub Timer_T3PWMX(period As Word,PW0 As Word,PW1 As Word,PS As Byte)
```

Beschrijving:

Deze functie initialiseert Timer3 met de aangegeven voordeler, pulsbreedte voor kanaal A en B en periodeduur, zie tabel. Het uitgangssignaal verschijnt op PortE.3 (X1_13) en PortE.4 (X1_12).

Parameters

period periodeduur
PW0 pulsbreedte kanaal A
PW1 pulsbreedte kanaal B
PS voordeler (tabel prescaler)

5.4.18.42 Timer_T3PWMY

Timer functies

Syntax

```
void Timer_T3PWMY(word period,word PW0,word PW1,word PW2,byte PS);

Sub Timer_T3PWMY(period As Word,PW0 As Word,PW1 As Word,PW2 As Word,PS As
Byte)
```

Beschrijving:

Deze functie initialiseert Timer3 met de aangegeven voordeler, pulsbreedte voor kanaal A, B en C en periodeduur, zie tabel. De uitgangssignalen verschijnen op PortE.3 (X1_13), PortE.4 (X1_12) en PortE.5 (X_11).

Parameters

period periodeduur
PW0 pulsbreedte kanaal A
PW1 pulsbreedte kanaal B
PW2 pulsbreedte kanaal C
PS voordeler (tabel prescaler)

5.4.18.43 Timer_T3Start

Timer functies

Syntax

```
void Timer_T3Start(byte prescaler);

Sub Timer_T3Start(prescaler As Byte
```

Beschrijving

Het produceren van frequenties wordt met de vorige instelling gestart. De voordeler moet nieuw aangegeven worden.

Parameters

prescaler voordeler (tabel prescaler)

5.4.18.44 Timer_T3Stop

Timer functies

Syntax

```
void Timer_T3Stop(void);

Sub Timer_T3Stop()
```

Beschrijving
Het produceren van frequenties wordt gestopt. Het uitgangssignaal kan 0 of 1 zijn, afhankelijk van de laatste toestand. Alleen de puls voor de timer wordt gestopt. De overige instellingen blijven behouden.

Parameters

Geen

5.4.18.45 Timer_T3Time

Timer functies

Syntax

```
void Timer_T3Time(word Time,byte PS);

Sub Timer_T3Time(Time As Word,PS As Byte)
```

Beschrijving
Deze functie initialiseert Timer3 met de aangegeven voordeler en de waarde (16Bit) voor de tijd, zie tabel. Als deze waarde wordt bereikt, dan wordt de Timer3-interrupt (INT_TIM3COMP) geactiveerd.

Parameters

Time tijdswaarde waarbij de interrupt getriggerd wordt
PS voordeler

Voordeler (prescaler)	Tijdbasis (duur van een tick)
PS_1 (1)	67,8 ns
PS_8 (2)	542,5 ns
PS_64 (3)	4,34 µs
PS_256 (4)	17,36 µs
PS_1024 (5)	69,44 µs

5.4.18.46 Timer_TickCount

Timer functies

Syntax

```
word Timer_TickCount(void);

Sub Timer_TickCount() As Word
```

Beschreibung
Meet de tijd in 10ms Ticks tussen twee oproepen van Timer_TickCount() en geeft de waarde bij de tweede oproep van Timer_TickCount() terug. De returnwaarde bij de eerste oproep kan geignoreert worden.

Parameter
Geen

Returnwaarde

Tijdverschil tussen twee oproepen

Voorbeeld:

```
void main(void)
{
      word time;
      Timer_TickCount();
      AbsDelay(500); // 500 ms wachten
      time=Timer_TickCount(); // de waarde van time moet 50 zijn
}
```

Hoofdstuk

6 FAQ (vaak gestelde vragen)

Problemen

1. Er bestaat geen USB –verbinding met het Application Board.

- Is de FTDI USB driver op de PC geladen? Of verschijnt er misschien bij het insteken van de USB –stekker een “onbekend apparaat” in de Hardware Manager?
- Is in Opties->IDE->Interfaces de juiste communicatiepoort ingesteld?
- Wordt er een Windowsversie van voor Windows 98 SE (“*Second Edition*”) gebruikt? De USB drivers van Microsoft functioneren pas vanaf Win98SE betrouwbaar met USB apparatuur.
- Worden de poorten **Mega32**: B.4-B.7, A.6-A.7 resp. **Mega128**: B.0-B.4, E.5 per ongeluk in de software gebruikt? (Zie pintoewijzing van Mega32 en Mega128). Zijn de jumpers op het Application board bij deze poorten ook gezet?
- Een signaal op **M32**:PortD.2 **M128**:PortE.4 (SW1) activeert bij het starten de seriële Bootloader.
- (alleen Mega128) Is misschien Port.G4 (LED2) bij de Reset op low? Zie SPI uitschakeling in hoofdstuk "Firmware".

2. De seriële interface geeft geen tekens af of ontvangt geen tekens.

- Worden de poorten D.0-D.1 per ongeluk in de software gebruikt? (Zie pintoewijzing van Mega32 en Mega128). Zijn de jumpers op het Application board bij deze poorten ook gezet?

3. Het Application Board reageert niet op commando”s als het serieel aangesloten is.

- Om de bootloader in de seriële modus te krijgen moet bij het inschakelen van het Application Board de toets SW1 ingedrukt worden. (Let op de jumpers voor SW1). Voor de seriële modus kan **M32**:PortD.2 **Mega128:** PortE.4 (SW1) ook vast op GND gelegd worden.

4. De hardware applicatie start niet automatisch (autostart functie)

- Een signaal op de SPI interface tijdens het starten kan de USB- communicatie activeren
- Een signaal op M32:PortD.2 M128:PortE.4 (SW1) bij het starten activeert de seriële Bootloader.

5. De toetsbezetting van de editor “xyz” is ingesteld, maar sommige toetsenbord-commando”s functioneren niet.

- De mogelijkheid om de toetsbezetting van een bepaalde editor in de IDE in te schakelen is slechts een benadering. Soms kost het te veel moeite de desbetreffende functies van de “vreemde” editor te ondersteunen, een andere keer kunnen toetsenbordcommando’s met de Keybord Shortcuts in de IDE botsen.

6. De spellingcontrole functioneert niet.

- Is de spellingcontrole in Opties->Editor ingeschakeld?
- De spellingcontrole toont alleen schrijffouten in de commentaren. Het controleren van andere bereiken zou zinloos zijn.

7. Waar wordt bepaald of het nieuwe project een BASIC of een C project is?

Er wordt geen verschil gemaakt in het type project. De brontekst- bestanden in een project bepalen welke programmeertaal gebruikt wordt. Bestanden met de extentie *.cc lopen in een nieuwe context, bestanden met de extentie *.cbas worden met BASIC vertaald. Eveneen kan een project ook gemengd zijn met C en BASIC.

8. Ik gebruik een andere LC-display dan de meegeleverde maar met dezelfde controller. De cursorpositie functioneert niet juist.

- De controller kan 4 regels met 32 teken weergeven. Het begin van een regel is in het geheugen volgens het volgende schema:

Waarde van pos	**Positie op het display**
0x00-0x1f	0 – 31 in de 1e regel
0x40-0x5f	0 – 31 in de 2e regel
0x20-0x3f	0 – 31 in de 3e regel
0x60-0x6f	0 – 31 in de 4e regel

9. Ik heb onder Optie->Compiler->Bibliotheek configuratie een nieuwe bibliotheek ingevoerd, echter wordt deze niet door het actuele project gebruikt.

- Deze instelling verandert alleen de standaardinstelling voor nieuwe projecten. Reeds bestaande projecten moeten in de Projectopties->Bibliotheek configuratie gewijzigd worden.

10. Waar vindt ik op het Mega128 Application board de 2e seriële interface?

- Zie J4 in hoofdstuk Jumper Application board M128.