C-Control Pro
Mega 32

Inhoud

Hoofdstuk 1

1 Belangrijke aanwijzingen

1.1 Introductie

Het C-Control Pro systeem is gebaseerd op de Atmel Mega 32 RISC micro –controller. Deze micro –controller 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 ingangs-condities de desbetreffende schakelsignalen afgeven, b.v. 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 b.v. 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/NE/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 gebruiks-aanwijzing 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 svp 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 daar geen acht op slaat, kan dat leiden tot foutief functioneren of tot beschadigingen.

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

Lees, voor u dit apparaat in gebruik neemt, 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 geen acht slaan op 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 onderdelen. 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, b.v. door het aanraken van een geaard, geleidend voorwerp (b.v. een radiator). Vermijd het aanraken van de aansluitpins van de C-Control Pro unit.

1.4 Gebruik waarvoor het apparaat bedoeld is

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 strooibits. Het gebruik van desbetreffende maatregelen (b.v. 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 svp uit de buurt van spatwater en condensatievocht. Let op het toelaatbare temperatuurbereik in de technische specificaties in het aanhangsel.

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 beveiligende 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 aansprakelijkheidsvoorwaarden, maak dan gebruik van onze 14-daagse geld-terug-garantie. Stuur ons dan svp 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. 0044-9604 / 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). Als u geïnteresseerd bent in een klantspecifieke software –oplossing, doen wij u gaarne een desbetreffend aanbod. Dieper gaande 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 5.08	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 Original 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.

Hoofdstuk 2

2 Installatie

2.1 Software

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

➔ Voor de periode van de software installatie en de installatie van de USB –drivers moet de gebruiker zich als administrator aangemeld hebben. Bij het normale werken met C-Control pro is dit niet nodig.

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 het standaardpad geïnstalleerd moet worden, of dat u een eigen doelbestand wilt aangeven. Aan het eind van de installatie wordt u nog gevraagd, of er iconen op uw desktop geïnstalleerd moeten worden.

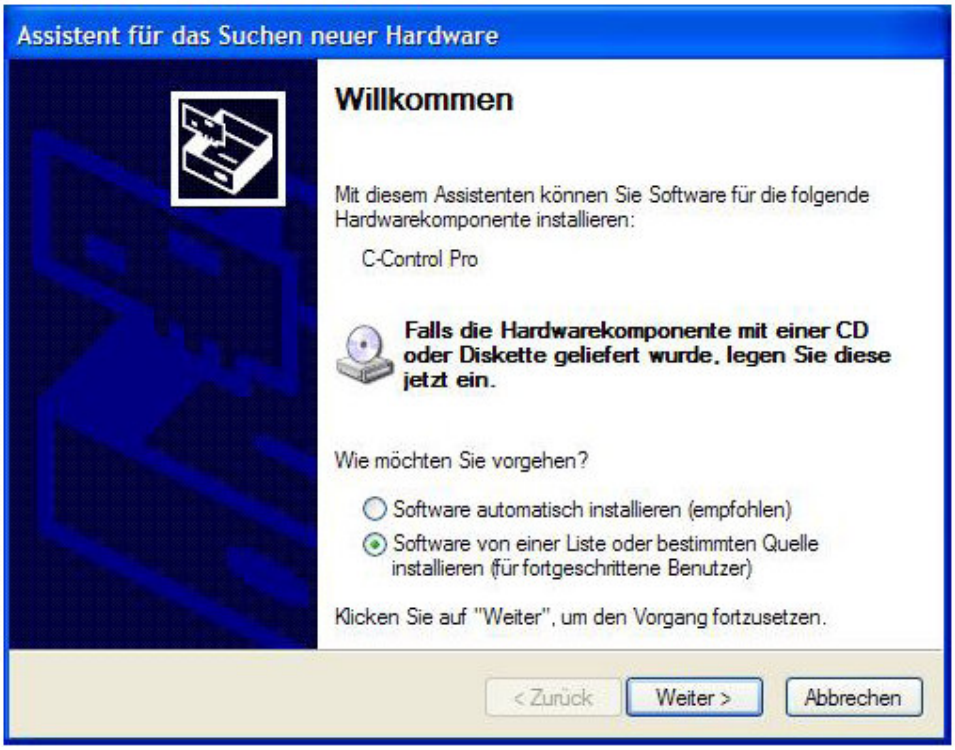
Als de installatieprocedure afgesloten is, kunt u zich als u wilt direct het “ReadMe” laten tonen, of de C-Control Pro ontwikkelingsomgeving starten.

2.2 Application Board

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 de verdere bedrading kan het later noodzakelijk zijn een netvoeding met een grotere 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 voor 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 het eerst aangesloten is, is er geen driver voor de FTDI chip aanwezig. Onder Windows XP wordt dan het volgende venster getoond:



U dient hier dan “Software uit een lijst of een bepaalde bron installeren” te kiezen en op “Verder” te klikken.

Daarna dient u het pad voor de index 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”.

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.

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.

Hoofdstuk 3

3 Hardware

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 commandoregel –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 unit module opgebouwd moet worden, dan dient u bij het inschakelen van de module de toets SW1 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.

Interpreter

De interpreter bestaat uit meerdere componenten:

- Bytecode interpreter
- Multithreading ondersteuning
- Interrupt -verwerking
- Toepassingsfuncties
- RAM en EEPROM interface

In hoofdzaak werkt de interpreter de bytecode af, die door de compiler gegenereerd is. Verder zijn de meeste bibliotheekfuncties er in geïntegreerd, opdat het bytecode –programma b.v. toegang kan krijgen tot hardware –ports. 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.

➔ Als er geen USB interface is aangesloten, en u heeft bij het inschakelen niet op SW1 gedrukt om in de seriële bootloader modus 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

3.2.1 Microcontroller

Mega32 overzicht

De microcontroller ATmega32 komt uit de AVR –familie van ATMEL. Het gaat om een low-power microcontroller met Advanced RISC Archtecture. Hier volgt een korte samenstelling

van de hardware sources:

- **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**

3.2.2 Module

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

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

ADC Referentiespanning –opwekking
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 kunnen worden. De referentiespanning voor de ondergrens is het GND –level, dus 0V. De referentiespanning voor de bovengrens kan door de gebruiker gekozen worden:

* 5V voedingsspanning (VCC)
* Interne referentiespanning
* Externe referentiespanning b.v. 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 klokpuls
Het genereren van de klokpuls gebeurt door een 14,7456MHz kwarts –oscillator. Alle tijdprocedures van de controller zijn van deze klokpuls -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, b.v. 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.

Digitalports (PortA, PortB, PortC, PortD)

De C-Control Pro module beschikt over vier digitale ports met elk 8 pins. Op de digitale ports kunnen b.v. toetsen met pull -up weerstanden, digital -ICs, opto –koppelingen of driverschakelingen voor relais aangesloten worden. De ports kunnen apart, d.w.z. pinwijze of bytewijze aangesproken worden. Elke pin kan of uitgang of ingang zijn.

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

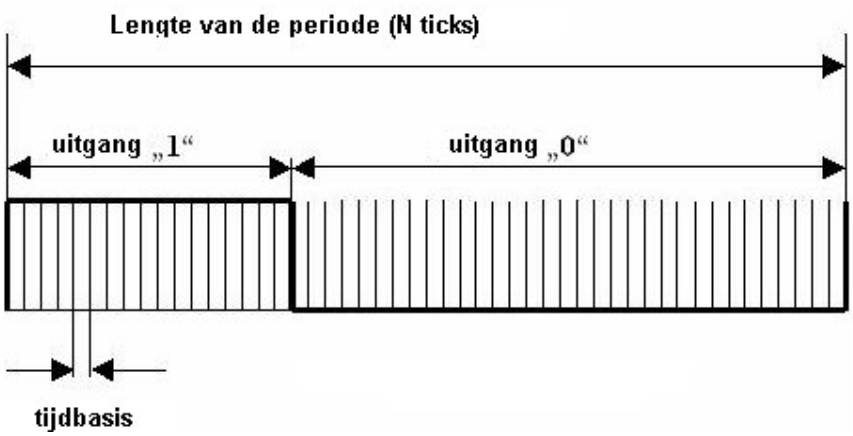
Digitale ingangspins zijn hoogohmig of met een interne pull –up weerstand geschakeld en zetten een aanliggend spanningssignaal om in een logische waarde. Voorwaarde daarvoor is, dat het spanningsignaal zich binnen de voor TTL – resp. CMOS –ICs gedefinieerde bereiken voor Low - of High level bevindt.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

➔ Let op de maximaal toelaatbare laststroom voor een aparte port en voor alle ports 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 port als ingangsport geconfigureerd. Via bepaalde commando’s kan de datarichting omgeschakeld worden.

➔ Het is belangrijk om vóór de programmering de pintoewijzing te bestuderen, aangezien belangrijke functies van de programma –ontwikkeling (b.v. de USB –interface van het Application board) op bepaalde ports liggen. Als deze ports 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 pins verbonden.

PLM -ports

Er staan twee timers ter beschikking 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 servo-motoren 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 –port op X stelt, dan houdt deze gedurende X ticks van een periode high level 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 afgiftewaarde veranderd. De PLM –ports zijn vanwege hun elektrische eigenschappen digitale ports. Let op de technische randvoorwaarden voor digitale ports (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 pins 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
Ports	
Max. toelaatbare stroom uit digitale ports	± 20mA
Toelaatbare totaal van de stromen op digitale ports	200mA
Toelaatbare ingangsspanning op portpins (digitaal en A/D)	-0,5V … 5,5V
Interne pull –up weerstanden (uitschakelbaar)	20 – 50 kOhm

3.2.3 Application Board

USB
Het Application board beschikt over een USB interface voor het laden van programma's en debuggen. 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 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 Port B.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 –fabrikant 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 staan 5 lichtdiodes tot uw beschikking. 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 hfst. USB hierboven). 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 port pin low (GND) is.

Toetsen

Er zijn vier toetsen aangebracht. Met SW3 (RESET1) wordt bij de Mega32 een reset getriggerd 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 PortD.2 verbonden worden en op dezelfde manier SW2 met PortD.3. Er bestaat de mogelijkheid SWS1/2 of tegen GND of

tegen VCC te schakelen. Deze keuzemogelijkheid wordt vastgelegd meet JP! Resp. met JP2. Om bij een open schakelaar eden gedefinieerd niveau op de ingangsport te hebben, moet de desbetreffende pull –up ingeschakeld zijn (zie hoofdstuk Digitalports).

➔ 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. Het 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 poling is het verkeerd er 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)
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 door gebruiker staat een 12-delig toetsenbord (0 … 9, *, #) tot uw 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 schuif-register 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.

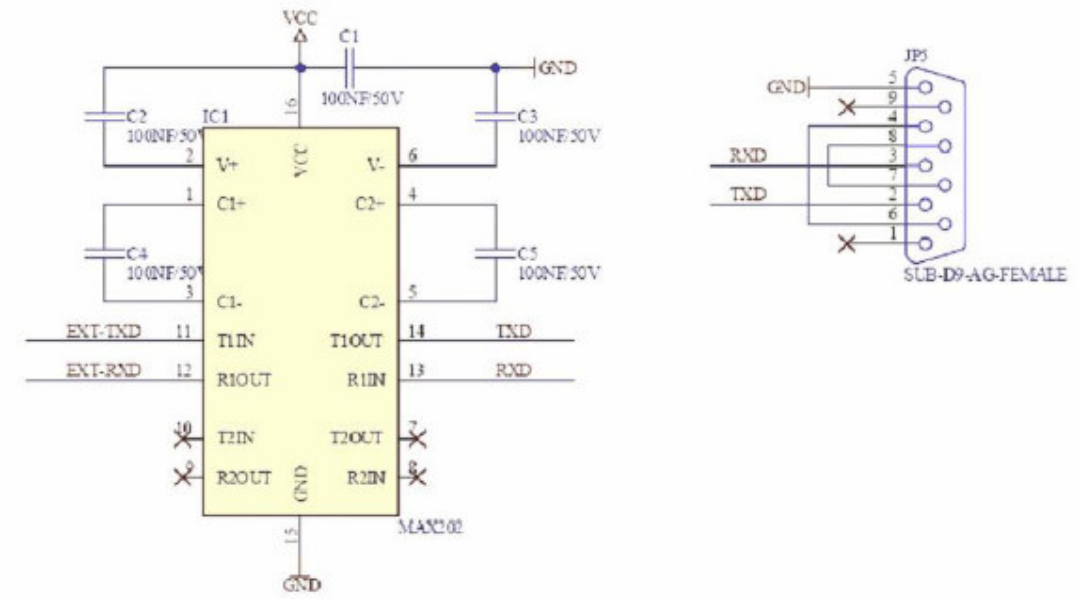
12C-interface
Via deze interface kunnen seriële data met hoge snelheid verzonden worden. Er zijn daar-voor slechts twee signaalleidingen nodig. De overdracht gebeurt via het 12C –protocol. Voor het effectieve gebruik van deze interface worden speciale functies ter beschikking gesteld (zie de softwarebeschrijving 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 verdere bedrading 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 wat te doen bij Reset").

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, b.v. 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 GNBD –aansluiting ligt op pin 5. Er worden voor de seriële dataoverdracht geen handshake –signalen gebruikt.

Een kabelaansluiting met aansluiting aan de NRZ –pins 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 u de pinbezetting kent.

➔ 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 pins) 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 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

3.2.4 Pintoewijzing

PortA tot PortD worden voor directe pin-functies (b.v. Port WriteBit) van 0 tot 31 geteld, zie “Portbit”.

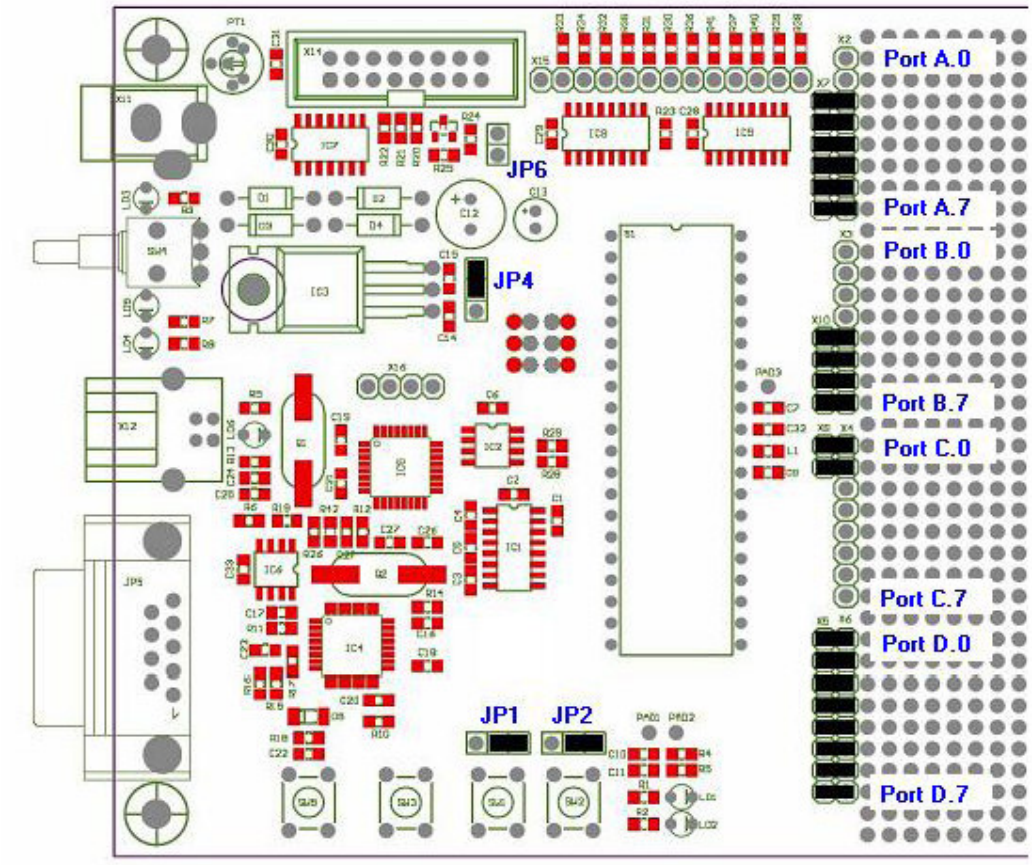
Pinbezetting voor application board Mega32 (pins 1 – 20)

PIN	Port	Port	PortBit	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 timer 0
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

3.2.5 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). Bijvoorbeeld is de seriële interface via de pins PortD.0 en PortD.1 gerealiseerd. Als de seriële interface niet gebruikt wordt, kunnen de desbetreffende jumpers verwijdert worden en deze pins zijn dan voor andere functies beschikbaar. Naast de jumpers voor deze ports zijn er nog extra jumpers, deze worden hierna beschreven.

Jumperposities bij uitlevering



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).

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

3.2.6 Schakelschema's

3.2.6.1 PDF schakelschema's

Om redenen van convertering bevinden de schakelschema's zich niet in het PDF handboek.U kunt ze vinden onder "Manual\Mega32 schakelschema's".

Hoofdstuk 4

4 IDE

4.1 Overzicht

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

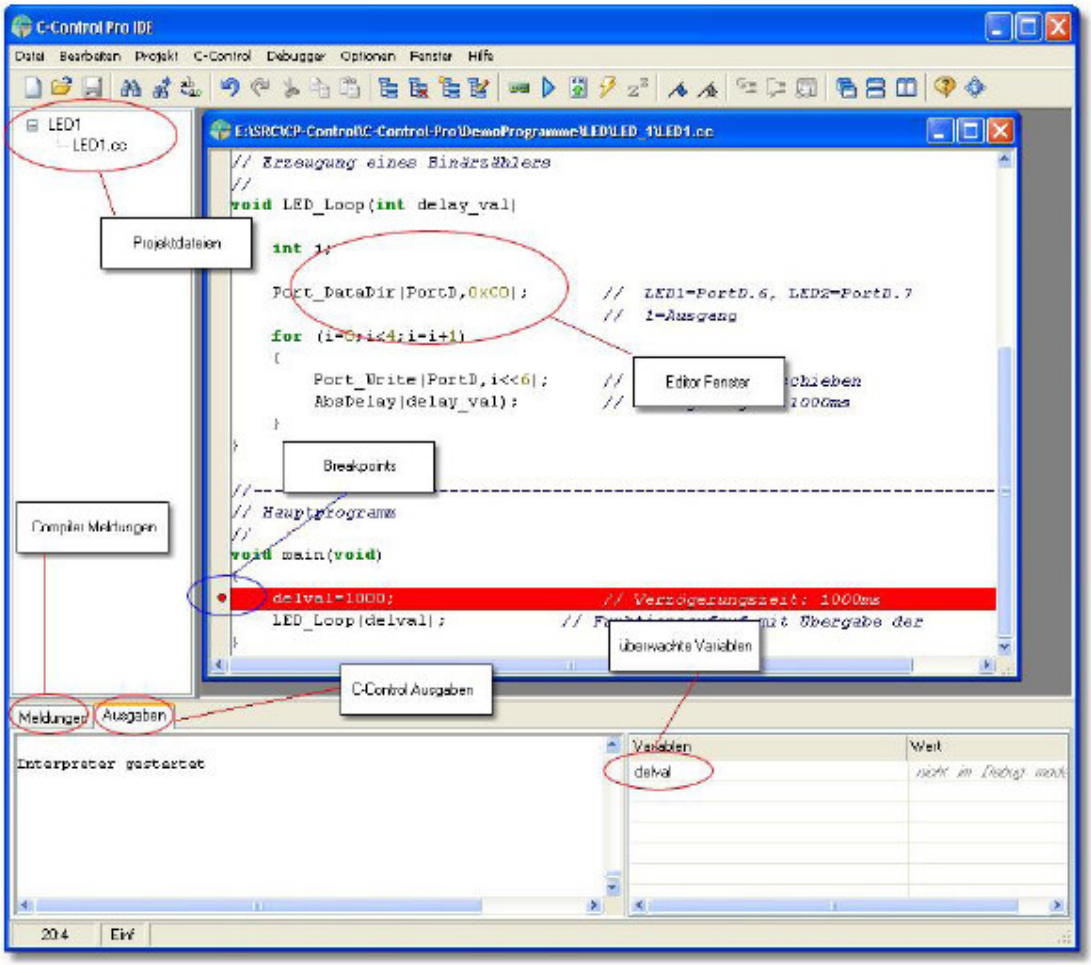
Sidebar voor projectbestanden Meerdere bestanden kunnen hier tot een project afgelegd worden.

Editor venster Er kunnen zo veel editor vensters geopend worden als u maar wilt om bestanden te editen.

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.



4.2 Projecten

4.2.1 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.

4.2.2 Beheren van een project

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

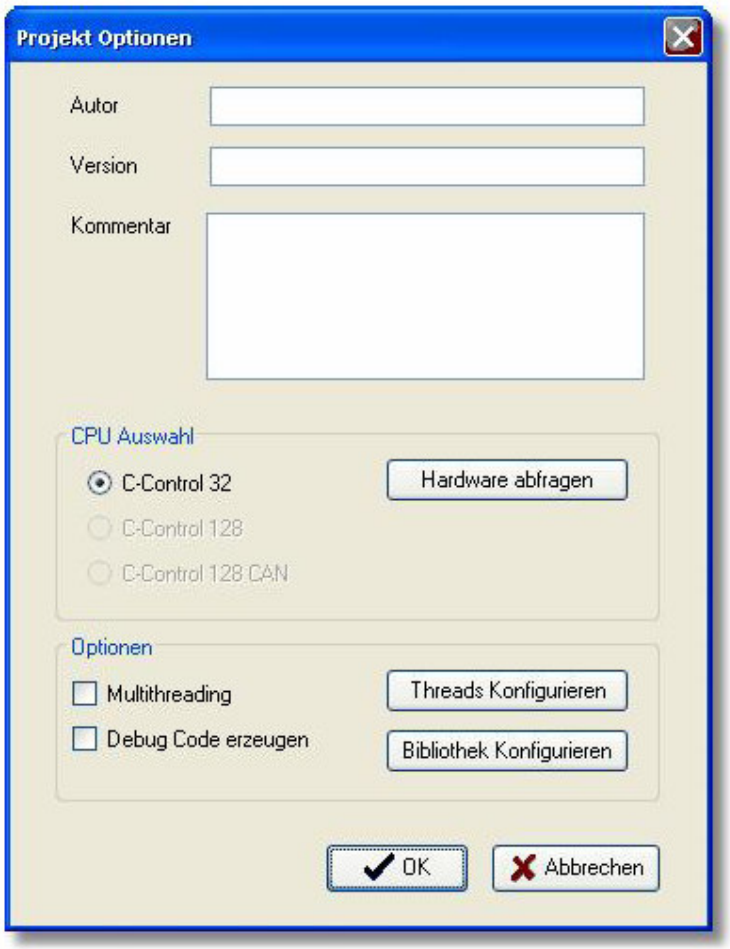
* Neu Hinzufügen - Er wordt een nieuw bestand aangelegd en tegelijkertijd wordt er een editor –venster geopend
* Toevoegen - Een bestaand bestand wordt aan het project toegevoegd
* Andere naam geven - De naam van het project wordt veranderd (dit is niet persé de naam van het projectbestand)
* Kompilieren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

Projectbestanden
Als u bestanden aan het project heeft toegevoegd, dan kunt u de bestanden met een dubbelklik op de bestandsnaam openen. Met een klik op de rechter muisknop verschijnen er nog meer opties:

* Umbenennen - De naam van het bestand wordt veranderd
* Entfernen - Het bestand wordt verwijderd uit het project
* Optionen - De projectopties kunnen veranderd worden

4.2.3 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.

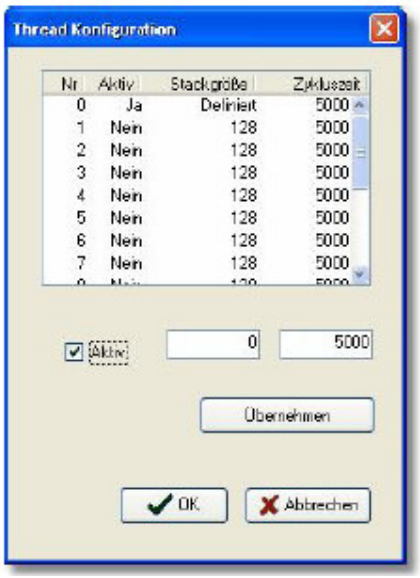
Bij de "Opties" configureert u de multithreading en of er een debug code gemaakt moet worden.

➔ 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 geparameteriseerd worden.

4.2.4 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.



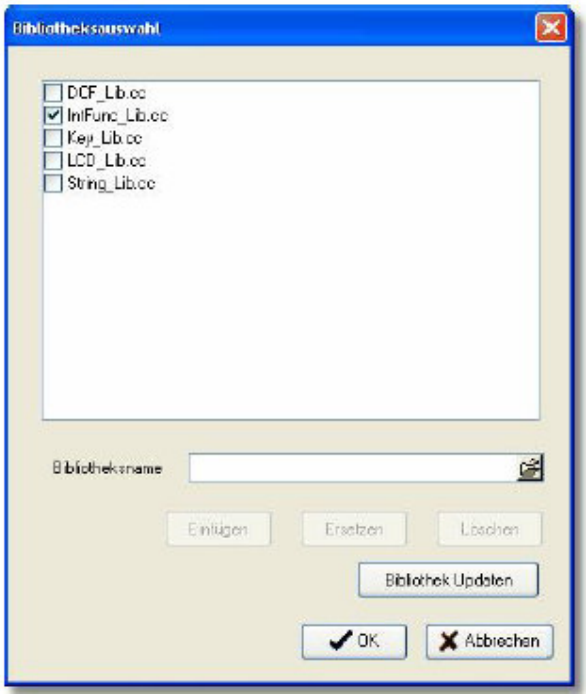
Aan elke extra thread buiten het hoofdprogramma wordt een plaats op de stack toegewezen, die hij niet mag overschrijden.

➔ 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.

4.2.5 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 de compilering betrokken.

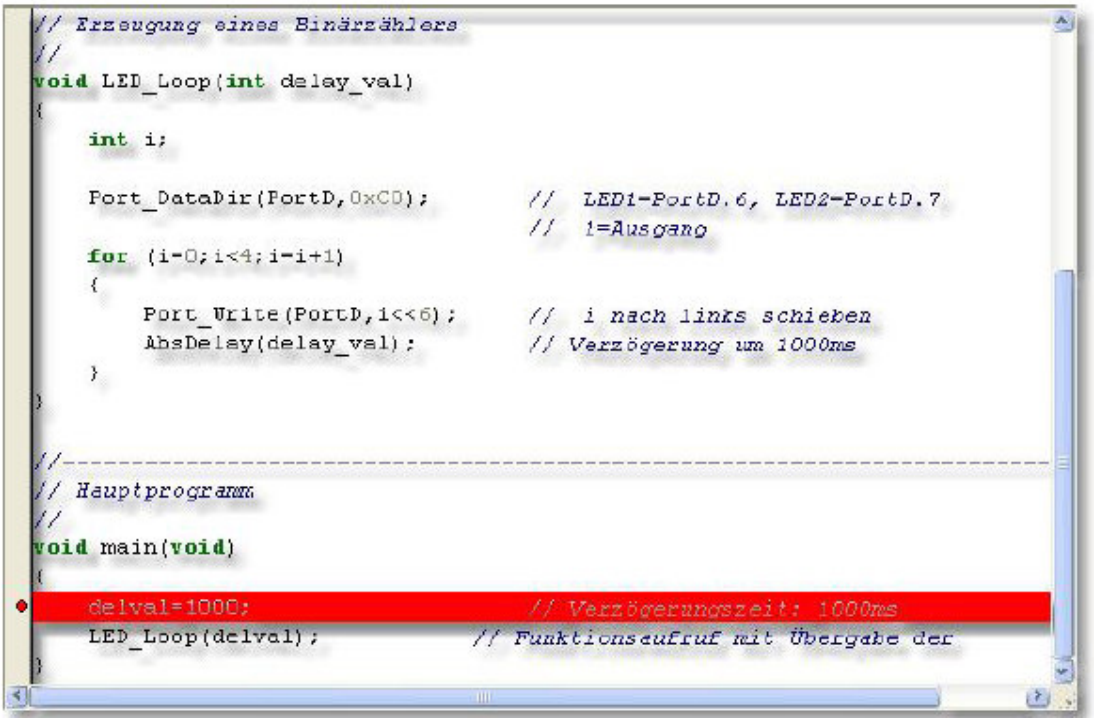
De lijst kan met behulp van het pad tekst -invoerveld “Bibliotheeknaam” en de buttons in de dialoog veranderd worden.

* 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 instelling vooraf van compiler aanwezig zijn, maar niet in deze lijst, worden toegevoegd.

4.3 Editor

4.3.1 Editorvenster

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).

4.3.2 Editorfuncties

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

- "Rückgängig" (Ctrl-Z) voert een Undo operatie uit. Hoeveel dit commando tenietdoet hangt ook af van de instelling van " Gruppen rückgängig" (groepen tenietdoen).
- "Wiederherstellen" (herstellen) (Ctrl-Y) – herstelt de editortoestand weer, die eerst door "Rückgangig" veranderd werd.
- "Ausschneiden" (Ctrl- X) – verwijdert geselecteerde tekst en kopieert deze in de opslag
- "Kopieren" (Ctrl-C) – kopieert geselecteerde tekst naar de opslag
- "Einfügen" (Crtl-V) – kopieert de inhoud van de opslag 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

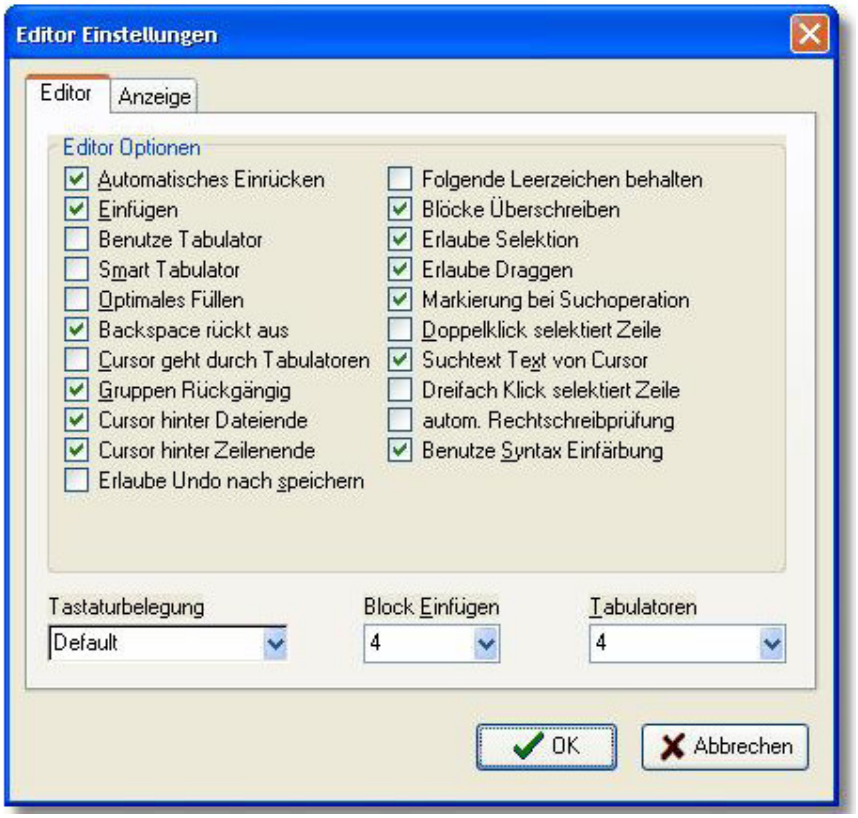
- "Gehe zu" (Alt-G) – u kunt naar een bepaalde regel springen

Zoeken/Vervangen dialoog

- Zoektekst – invoerveld voer de te zoeken tekst
- Vervangen met – de tekst die de te vervangen tekst vervangt
- Hoofdletters/Kleine letters – onderscheidt hoofd – en kleine letters
- Alleen hele woorden – vindt alleen hele woorden en geen deeltekenketens
- 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.

4.3.3 Editor –instellingen



- Automatisch einrücken – als u op Enter drukt, wordt de cursor op de volgende regel tot aan het begin van de vorige regel geplaatst
  - Einfügen – Als deze functie uitgeschakeld is, is overschrijven ingesteld als standaard
  - Benutzte Tabulator – als deze geactiveerd is, worden er tab tekens ingevoegd, anders worden er spaties gebruikt
  - Smart Tabulator - met "Tabulator" springt u naar de plaats waar de tekens van de vorige regel beginnen
  - Optimaal vullen – "Automatisch einrücken" vult eerst met tabs en de rest met spaties

- Backspace rückt aus – met “Backspace” springt u naar de plaats waar de tekens van de vorige regel beginnen
- Cursor geht durch Tabulatoren – u gaat door tabs zoals door spaties
- Gruppen rückgängig – een Undo operatie wordt niet in kleine stappen, maar in blokken uitgevoerd
- Cursor hinter Dateiende – u kunt de cursor aan het einde van het bestand plaatsen
- Cursor hinter Zeilenende – u kunt de cursor naar het eind van de regel bewegen
- Erlaube Undo nach speichern – de “Undo”-buffer wordt na het opslaan niet geleegd
- Folgende Leerzeichen behalten – als deze functie geactiveerd is, dan worden spaties aan het eind van een regel niet gewist
- Blöcke überschreiben – als een blok geselecteerd is, dan vervangt de volgende invoer het blok
- Erlaube Selektion – er kan tekst geselecteerd worden
- Erlaube Draggen – geselecteerde tekst kan met de muis “gedragged” (bij ingedrukte linker muistoets verschoven) worden
- Markierung bei Suchoption – als een gezochte tekst gevonden is, is het resultaat geselecteerd
- Doppelklick selektiert Zeile – normaalgesproken selecteert dubbel klikken een woord
- Suche Text von Cursor - de tekst bij het “zoektekst” –invoerveld wordt door de cursorpositie overgenomen
- Dreifachklick selektiert Zeile – als dubbel klikken een woord selecteert, dan kan met deze optie met drie keer klikken een regel geselecteerd worden
- Automatische Rechtschreibprüfung – deze optie schakelt de spellingcontrole in de commentaren in
- Benutzte Syntax Einfärbung – de syntax Highlighting voor *.cc en *cbas bestanden wordt ingeschakeld

In de keuzebox toetsbezetting kunt u de toetsen –lay-out van gangbare editors instellen. Deze emulatie is echter slechts onvolledig, omdat het gedrag van de verschillende editors zeer complex is. De belangrijkste toetsfuncties worden echter meestal ondersteund.

Bij Block einfügen wordt het aantal spaties ingevoerd waarmee een geselecteerd blok met de tabulatortoets ingevoegd resp. verwijderd wordt.

Het invoerveld Tabulatoren bepaalt hoeveel tekens een tabulator breed is.

4.3.4 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 eind 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.
\$	‘$’

4.4 Compiler

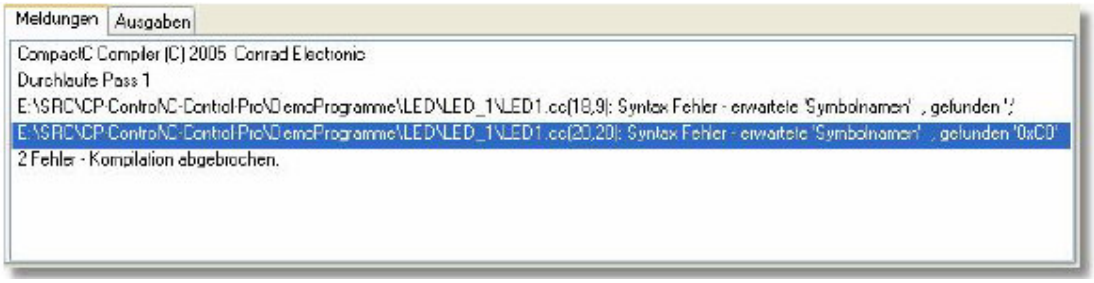
4.4.1 Instellingen vooraf van de compiler

In de instellingen vooraf kunnen de standaardwaarden geconfigureerd worden, die bij het maken van een nieuw project opgeslagen worden.

De keuzeboxen "Threads configureren" en "Bibliotheek configureren" zijn identiek aan de instellingsparameters bij Projectopties.

4.4.2 Compileren

Onder het menupunt Project kan met Compileren (F9) het actuele project door de compiler vertaald worden. De compilermeldingen worden in een eigen vensterbereik getoond.



Als er bij het compileren fouten optreden, dan wordt er per regel een fout beschreven. De vorm is:

Bestandsnaam(regel,tussenruimte) : foutbeschrijving

U kunt de positie van de fout in de brontekst vinden via de commando's Volgende fout (F11) of Vorige fout (Shift-F11). Beide commando's bevinden zich in het menupunt Project. Als alternatief kunt u ook met dubbel klikken op een foutmelding van de compiler de cursor bij de fout in de editor plaatsen.

Bij een geslaagde compilering wordt de bytecode als bestand met de uitgang “*.bc” in het projectregister afgelegd.

Met een klik op de rechter muisknop in het bereik van de compilermeldingen kunnen de volgende procedures getriggerd worden:

- wissen – wist de lijst van de compilermeldingen
- in geheugen kopiëren – kopieert alle tekstberichten in het tussengeheugen

4.5 C-Control hardware

4.5.1 Programma starten

Programma –overdracht

Als een programma foutloos vertaald is, moet de bytecode eerst overgebracht worden naar de Mega 32, 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 32 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 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 het gebruikersprogramma automatisch te starten.

➔ Een signaal op INT_0 bij het inschakelen van de C-Control Pro module kan de autostart storen. Volgens de pintoewijzing 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.

4.5.2 Uitvoer

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

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

4.5.3 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”)

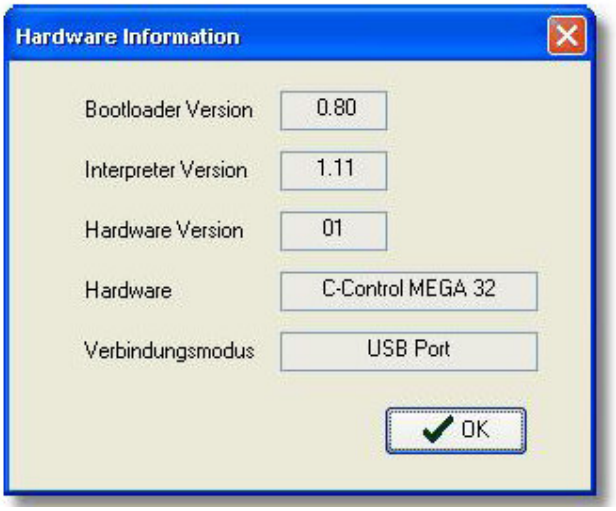
Door het invoeren van de PIN via Pin Eingeben in C-Control kunt u alle operaties vrijschakelen.

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.

4.5.4 Versie controleren

Omdat er gepland is om na de C-Control Pro MEGA 32 nog andere hardware platformen te 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.



4.6 Debugger

4.6.1 Breakpoints

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.

Bij de editor is het mogelijk maximaal 16 stops (breakpoints) in te stellen. Een breakpoint wordt overgebracht door links naast het begin van een regel met de muis te klikken (zie "Übersicht" - overzicht – of "Editorfenster").

➔ Het aantal breakpoints is beperkt tot 16, omdat deze informatie bij het lopen van de bytecode interpreter in RAM meegenomen wordt. Andere debuggers plaatsen de stops direct in de programmacode. Dat is hier niet wenselijk, omdat het de levensduur van het flashgeheugen (ca. 10.000 schrijfmogelijkheden) drastisch zou reduceren.

In het Debugger menu vindt u alle debugger commando's. Met Debug modus (Shift-F10) start u de debugger. Als er op dat moment geen breakpoint gezet is, 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 gaat daaraan voorbij.

➔ 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.

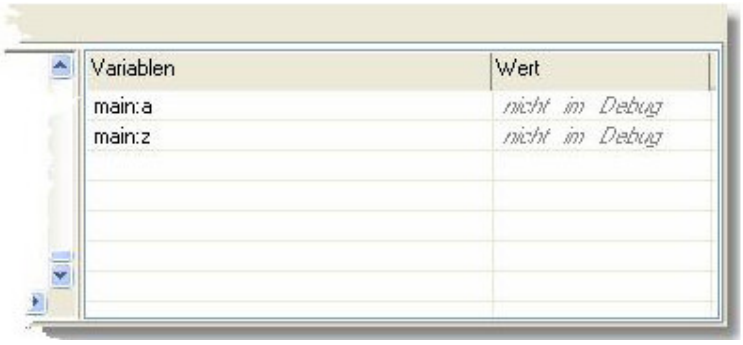
Met de aanwijzing Debug modus verlaten wordt de debug modus beëindigd.

➔ 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.

4.6.2 Variabelen

In de debugger kunt u de inhoud van variabelen bekijken. Daarvoor plaatst u de muis boven de variabele, en na ca. 2 seconden wordt de inhoud van de variabele als tooltip getoond.

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 context –menu 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 locale variabele is, dan wordt daar de functienaam met een dubbele punt voorgeplaatst (**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.

U kunt de **waarden van de variabelen** als decimaal getal of als hexgetal bekijken. Variabelen van het type **char** worden in de decimale modus als ASCII tekens weergegeven.

4.7 IDE instellingen

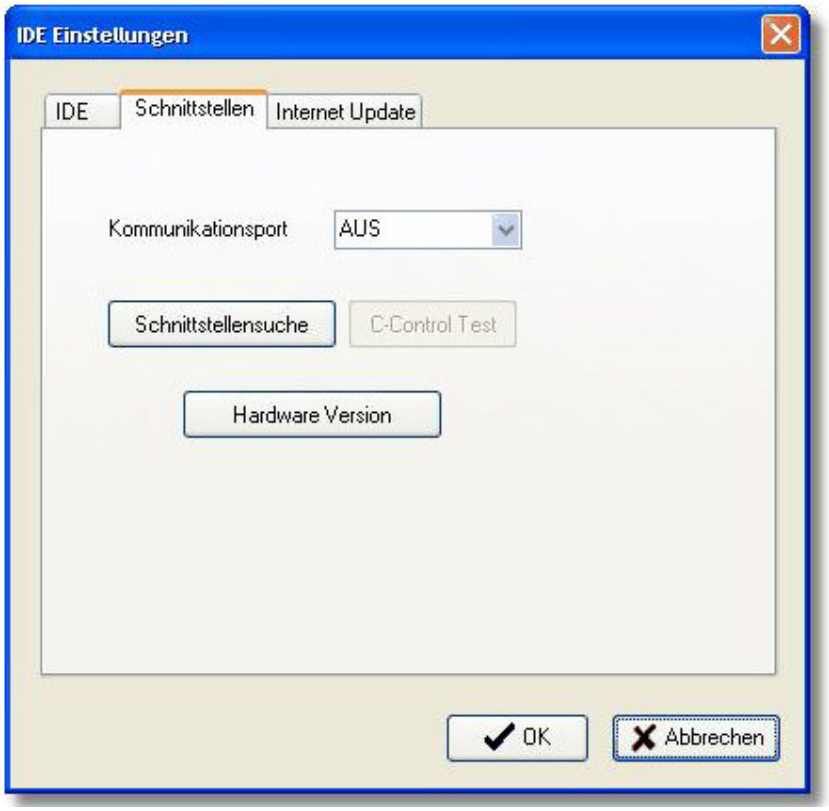
U kunt aparte aspecten van de IDE configureren.

- 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.

4.7.1 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.

4.7.2 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 b.v. 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.

4.8 Vensters

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

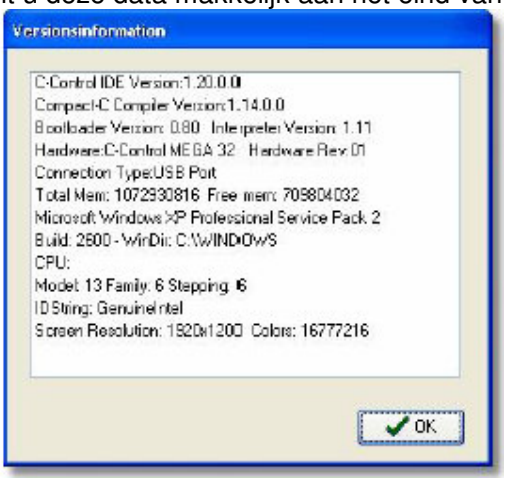
- Überlappend – de vensters worden boven elkaar gerangschikt, elk venster is daarbij iets verder naar rechts verschoven dan het vorige.
- Untereinander – de vensters worden verticaal onder elkaar geplaatst

- Nebeneinander – ordent de vensters van links naar rechts naast elkaar
- Alle Minimieren – verkleint de vensters tot symboolgrootte
- Schliessen – sluit het actieve venster

4.9 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 geheugen.
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 geheugen 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 b.v. 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 staat ook tot uw beschikking met een druk op de rechter muisknop in het editorvenster.

5. Compiler

5.1 Compact C

5.1.1 Programma

Een programma bestaat uit een hoeveelheid aanwijzingen (zoals b.v. “a=5”), die over verschillende functies verdeeld zijn. De startfunctie die in elk programma aanwezig moet zijn, is de functie “main()”. Een minimalistisch 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.

5.1.2 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;
```

[Afb.] 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 voorrang. U vindt een opsomming van de prioriteiten in de voorrang 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" levert 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 b.v.

12 + 123 – 15

door de compiler samengevat tot

120

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

5.1.3 Datatypes

Waarden hebben altijd een bepaald datatype. De waarden van gehele getallen hebben in CompactC een 8 of 16 bit breed datatype, getallen met floating point zijn altijd 4 byte lang.

Datatype	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	$\pm 1.175^{e}$-38 to $\pm 3.402^{e}$38	32

Zoals u ziet, zijn de datatypes **“unsigned char”** en **“byte”** identiek, net als **“unsigned int”** en **“word”.**

Strings
Er bestaat 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 zeer 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 type –convertering 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.

5.1.2 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 geheugen-plaats 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, wordt neemt het programma zijn toevlucht tot vreemde variabelen en is de kans groot dat het programma ‘crasht’.

Strings
Er bestaat geen specifiek “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!”;
```

Zichtbaarheid van variabelen

Als variabelen behalve functies gedeclareerd worden, hebben ze een globale zichtbaarheid. Dat betekent, dan u ze vanuit elke functie kunt aanspreken. 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 omdat deze door locale a afgedekt is
        // 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 stapelgeheugen aangelegd. Dat betekent dat locale variabelen alleen in het geheugen bestaan tijdens de tijd dat de functie verwerkt wordt.
Als bij lokale 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. Opdat 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.

5.1.5 Operatoren

Prioriteit van operatoren

Operatoren verdelen wiskundige termen in deeltermen. De operatoren worden dan in de volgorde van hun prioriteit (voorrang) 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 prioriteitstabel.

5.1.5.1 Wiskundige operatoren

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

[Afb.] 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 komma 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

5.1.5.2 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

5.1.5.3 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

5.1.5.4 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

5.1.2.1 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

5.1.5.6 Bit –operatoren

Bitschuif –operatoren zijn alleen toegestaan voor integer –datatypes.

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

5.1.6 Controlestructuren

5.1.6.1 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* geëvalueerd wordt 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++;
}
```

5.1.6.2 while

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

While(*term*) *aanwijzing;*

Eerst wordt de *term* geëvalueerd. 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 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:

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

5.1.6.3 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 minimaal 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 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
}
```

5.1.6.4 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 stapbreedte 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;
}
```

5.1.6.5 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.

5.1.6.6 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++;
```

5.1.7 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:

```
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);          // voer de eerste parameter in
        Msg_WriteFloat(param2);        // voer de tweede parameter in
}
```

➔ 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 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 invoeren …
       a=100;
       f=12.0;               // maar 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!
```

Terugloopparameters
Het is niet alleen mogelijk parameters door te geven, een functie kan ook een terugloopwaarde 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 terugloopwaarde 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.

5.1.8 Tabellen

5.1.8.1 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**		

5.1.8.1 Operator voorrang

Rang	Operator
13	( )
12	++ -- ! ~ - (negatief voorteken)
11	* / %
10	+ -
9	<< >>
8	< <= > >=
7	== !=
6	&
5	^
4	|
3	&&
2	||
1	? :

5.1.8.3 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
II	Logisch of
I	Logisch niet

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

5.2 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 PI 3.141

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 als optie een #else aanwijzing aangegeven is, dan wordt de brontekst na #else bewerkt wanneer het symbol niet gedefinieerd is.

Invoegen van tekst

#include pfad\dateiname (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!

5.3 Bibliotheken

5.3.1 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 b.v. 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!

5.3.2 AbsDelay

Algemene functies

Syntax

void AbsDelay (word ms);

Beschrijving

De functie AbsDelay() wacht een bepaald aantal milliseconden.

[Afb.] De functie werkt weliswaar zeer 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.

Als alleen de actuele thread moet wachten, moet u de functie Thread Delay gebruiken.

Parameter
ms Wachttijd in ms

5.3.3 Analoge comparator

5.3.3.1 Acomp

Acomp Functies **Voorbeeld**

Syntax

Void Acomp (**byte mode);**

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 AIN1 (Poort B.3). De positieve ingang kan of AIN0 (Poort B.2) of een interne referentiespanning van 1,22V zijn.

Parameters

Mode Werkmodus

Moduswaarden:

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

5.3.3.2 AComp voorbeeld

Voorbeeld: gebruik van de analoge comparator

```
// Acomp:  analoge comparator
//  Ingang (+)  PB2 resp. band gap reference 1,22V
///  Ingang (-)  PB3
// De functie Acomp geeft als resultaat de waarde van de comparator
// De oproep kan met de parameter 0 (beide ingangen worden gebruikt) of
// of 0x40 (interne referentiespanning op de (+) ingang, externe ingang PB3.

void  main (void)
{
    //  De comparator wordt elke 50ms uitgelezen en uitgegeven
    while  (1)
    {
      if  (Acomp(0x40)==1)              // Ingang (+) interne band gap reference 1,22V
     {
        Msg_WriteChar('1');
      }
      else
      {
         Msg_WriteChar('0');
      }
      AbsDelay (500);
    }
}
```

5.3.4 Analoog-Digitaal-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, 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 als volgt berekend:

*U = x * referentiespanning / 1024*

Als de externe referentiespanning 4,096V bedraagt, b.v. 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 differentiemetingen staat een versterker ter beschikking met de versterkingen V: x1, x10, x200. Als x een digitale meetwaarde is, dan wordt de desbetreffende spanningswaarde als volgt berekend:

*U = x * referentiespanning / 512 / V*

5.3.4.1 ADC_Disable

ADC functies

Syntax

void ADC_Disable(void);

Beschrijving:

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

Parameters
Geen

5.3.4.2 ADC_Read

ADC functies

Syntax

word ADC_Read(void);

Beschrijving:

De functie ADC_Read levert de gedigitaliseerde meetwaarde van één van de 8 ADC-poort s. 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.
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 .

5.3.4.3 ADC_ReadInt

ADC –functies

Syntax

word ADC_ReadInt(void);

Beschrijving

Deze functie wordt gebruikt om na een ADC-Interrupt de meetwaarde te lezen. De ADC-Interrupt wordt getriggerd, als de AD_omvorming afgesloten is en er dus een nieuwe meting tot uw beschik-king staat. 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 op-roepen 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 .

5.3.4.4 ADC_Set

ADC –functies

Syntax

word ADC_Set(byte v_ref,byte channel);

Beschrijving

De functie ADC_Set initialiseert de analoog-digitaal_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.

Parameters

channel Poortnummer (0 … 7) van de ADC
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

5.3.4.5 ADC_SetInt

ADC functies

Syntax

word ADC_Int(byte v_ref,byte channel);

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.

Parameters

channel Poort nummer (0 … 7) van de ADC
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

5.3.4.6 ADC_StartInt

ADC functies

Syntax

void ADC_StartInt(void);

Beschrijving

De meting wordt gestart, als eerst de A/D 0mvormer met behulp van ADC SetInt() op interrupt geïnitialiseerd is. Als het meetresultaat klaarligt, wordt er een ADC_Interrupt getriggerd.

Parameters

Geen

5.3.5 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 Poort D7. 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 LCD –display getoond. Het weergave -formaat is
1^{e} regel: uur : minuut : seconde
2^{e} 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.

5.3.5.1 DCF_FRAME

DCF -functies

Syntax

Void DCF_FRAME (void);

Beschrijving

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

Parameters

Geen

5.3.5.2 DCF_INIT

DCF -functies

Syntax

Void DCF_INIT(void)

Beschrijving

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

Parameters

Geen

5.3.5.3 DCF_PULS

DCF -functies

Syntax

Void DCF_PULS(void);

Beschrijving

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

Parameters

Geen

5.3.5.4 DCF_START

DCF -functies

Syntax

Void DCF_START(void);

Beschrijving

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

Parameters

Geen

5.3.5.5 DCF_SYNC

DCF -functies

Syntax

Void DCF_SYNC(void);

Beschrijving

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

Parameters

Geen

5.3.6 Debug

De Debug Message functies maken het mogelijk een geformatteerde tekst naar het uitvoervenster 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 moet wachten op de voltooiing van de opgave. 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.

5.3.6.1 Msg_WriteChar

Debug Message functies

Syntax

void (Msg_WriteChar (char c);

Beschrijving

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

Parameters

c het uit te voeren teken

5.3.6.2 Msg_WriteFloat

Debug Message functies

Syntax

void (Msg_WriteFloat (float val);

Beschrijving

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

Parameters

val float waarde

5.3.6.3 Msg_WriteHex

Debug Message functies

Syntax

void (Msg_WriteHex (word val);

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.

Parameters

val 16bit waarde

5.3.6.4 Msg_WriteInt

Debug Message functies

Syntax

void (Msg_WriteInt (int val);

Beschrijving

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

Parameters

val 16bit integere waarde

5.3.6.5 Msg_WriteText

Debug Message functies

Syntax

void (Msg_WriteText (int char text[ ]);

Beschrijving

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

Parameters

text cursor op char array

5.3.6.6 Msg_WriteWord

Debug Message functies

Syntax

void (Msg_WriteWord (word val);

Beschrijving

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

Parameters

val 16bit unsigned integere waarde

5.3.7 EEPROM

5.3.7.1 EEPROM_Read

EEPROM functies

Syntax

byte EEPROM_Read(word pos);

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.

Parameters

pos positie in de EEPROM

Returnwaarde

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

5.3.7.2 EEPROM_Write

EEPROM functies

Syntax

void EEPROM_Write(word pos, byte_val);

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 heeft daarom betrekking op byte 32 en hoger in de EEPROM.

Parameters

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

5.3.8 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.

5.3.8.1 I2C_Int

I2C functies **Voorbeeld**

Syntax

void I2C_Init (byte I2C_BR);

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

5.3.8.2 I2C_Read_ACK

I2C functies

Syntax

byte I2C_Read_ACK(void);

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

5.3.8.3 I2C_Read_NACK

I2C functies Voorbeeld

Syntax

byte I2C_Read_NACK(void);

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

5.3.8.4 I2C_Start

I2C functies Voorbeeld

Syntax

void I2C_Start(void);

Beschrijving

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

Parameters

Geen

5.3.8.5 I2C_Status

I2C functies

Syntax

byte I2C_Status(void);

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

5.3.8.6 I2C_Stop

I2C functies **Voorbeeld**

Syntax

void I2C_Stop(void);

Beschrijving

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

Parameters

Geen

5.3.8.7 I2C_Write

I2C functies Voorbeeld

Syntax

void I2C_Write(void);

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

Parameters
data databyte

5.3.8.8 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

5.3.8.9 I2C voorbeeld

Voorbeeld: EEPROM 24C64 lezen en schrijven zonder I2C_status opvragen

```
// I2C initialisatie, bit rate 100kHz

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

        address=0x20;
        data=0x42;

        I2C_Init(12C_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();
        Mag_WriteHex(EEPROM_data);
```

5.3.9 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, ingang Poort D.2
INT_1	Externe interrupt1, ingang Poort D3
INT_2	Externe interrupt2, ingang Poort B.2
INT_TIM1CAPT	Timer1 Capture, ingang Poort D.6
INT_TIM1CMPA	Timer1 CompareA
INT_TIM1CMPB	Timer1 CompareB
INT_1TIM1OVF	Timer1 Overflow
INT_1TIM0COMP	Timer0 Compare
INT_TIM0OVF	Timer0 Overflow
INT_ANA_COMP	Analoge comparator
INT_ADC	ADC
INT_TIM2COMP	Timer2 Compare
INT_TIM2OVF	Timer2 Overflow

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 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.

5.3.9.1 Ext_Int0

Interrupt functies

Syntax

void Ext_Int0 (byte Mode);

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.

Parameters

Mode Parameter

1: een low niveau triggert een interrupt
2: elke flankwisseling triggert een interrupt
3: een vallende flank triggert een interrupt

4: een stijgende flank triggert een interrupt

5.3.9.2 Ext_Int0Disable

Interrupt functies

Syntax

void Ext_Int0Disable(void);

Beschrijving

De externe interrupt 0 wordt geblokkeerd.

Parameters

Geen

5.3.9.3 Ext_Int1

Interrupt functies

Syntax

void Ext_Int1(byte Mode);

Beschrijving

Deze functie schakelt de externe interrupt 1 vrij. De parameter Mode legt vast, wanneer een interrupt gemaakt moet worden.

Parameters

Mode Parameter

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

5.9.3.4 Ext_Int1Disable

Interrupt functies

Syntax

void Ext_Int1Disable(void);

Beschrijving

De externe interrupt 1 wordt geblokkeerd.

Parameters

Geen

5.3.9.5 Ext_Int2

Interrupt functies

Syntax

void Ext_Int2(byte Mode);

Beschrijving

Deze functie schakelt de externe interrupt 0 vrij. De parameter Mode legt vast, wanneer er een interrupt gemaakt moet worden.

Parameters

Mode Parameter

0: een vallende flank triggert een interrupt
1: een stijgende flank triggert een interrupt

5.3.9.6 Ext_Int2Disable

Interrupt functies

Syntax

void Ext_Int2Disable(void);

Beschrijving
De externe interrupt 2 wordt geblokkeerd.

Parameters

Geen

5.3.9.7 Irq_GetCount

Interrupt functies **Voorbeeld**

Syntax

byte Irq_GetCount(void);

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() getriggerd is. Een waarde groter dan 1 kan optreden als de hardware sneller interrupts genereert dan de interpreter de interrupt routine kan verwerken.

5.3.9.8 Irq_SetVect

Interrupt functies **Voorbeeld**

Syntax

Void Irq_SetVect(byte irqnr, word vetc);

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.

Parameters

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

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

5.3.9.9 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_GetCountg(INT_TIM2COMP);
}

void main(void)
{
    cnt=0;

    Irq_SetVect(INT_TIM2COMP,ISR);
    while (true); // Eindeloze lus
}
```

5.3.10 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.

5.3.10.1 Key_Init

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

Syntax

```
Void Key_Init(void);
```

Beschrijving

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

Parameters

Geen

5.3.10.2 Key_Scan

Keyboard functies

Syntax

```
word  Key_Scan(void);
```

Beschrijving
Key_Scan zoekt op volgorde de invoerpins 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.

5.3.10.3 Key_TranslateKey

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

Syntax

```
char  Key_TranslateKey(word  keys);
```

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

Parameters

keys bitveld dat door Key_Scan() teruggeleverd wordt.

Returnwaarde

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

5.3.11 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.

5.3.11.1 LCD_ClearLCD

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

Syntax

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

Beschrijving
Wist het display en schakelt de cursor in.

Parameters

Geen

5.3.11.2 LCD_CursorOff

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

Syntax

```
void LCD_CursorOff(void);
```

Beschrijving
Schakelt de cursor van het display uit.

Parameters

Geen

5.3.11.3 LCD_CursorOn

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

Syntax

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

Beschrijving

Schakelt de cursor van het display in.

Parameter

Geen

5.3.11.4 LCD_CursorPos

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

Syntax

void LCD_CursorPos(byte pos);

Beschrijving

Plaatst de cursor in de positie pos.

Parameter

pos cursorpositie

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

5.3.11.5 LCD_Init

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

Syntax

void LCD_Init(void);

Beschrijving

“High Level” initialisering van het LCD display. Roept als eerste LCD InitDisplay() op.

Parameters

Geen

5.3.11.6 LCD_SubInit

LCD functies

Syntax

void LCD_SubInit(void);

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.

Parameters

Geen

5.3.11.7 LCD_WriteChar

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

Syntax

void LCD_WriteChar(char c);

Beschrijving

Schrijft een teken op de cursorpositie op het LCD display.

Parameters

c ASCII waarde van het teken

5.3.11.8 LCD_TestBusy

LCD functies

Syntax

void LCD_TestBusy(void);

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

5.3.11.9 LCD_WriteCTRRegister

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

Syntax

void LCD_WriteCTRRegister(byte cmd);

Beschrijving

Stuurt een commando naar de Display Controller.

Parameter

cmd Commando in byte –vorm

5.3.11.10 LCD_WriteDataRegister

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

Syntax

void LCD_WriteDataRegister(char x);

Beschrijving

Stuurt een databyte naar de Display Controller.

Parameter

x databyte

5.3.11.11 LCD_WriteNibble

LCD functies

Syntax

void LCD_WriteNibble(byte x, byte cmd);

Beschrijving

Stuurt een 4-bit commando naar de Display Controller. x bevat daarbij een "1"voor de nibble met hogere waarde en een "2" voor de nibble met lagere waarde. Het commando "Nibble" staat in bit 4 tot 7 van byte cmd. Voor verdere details dient u het datablad van het display te bestuderen.

Parameters

x datanibble
cmd commandonibble

5.3.11.12 LCD_WriteRegister

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

Syntax

void LCD_WriteRegister(byte y, byte x);

Beschrijving

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

Parameters

y databyte
cmd commandonibble

5.3.11.13 LCD_WriteText

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

Syntax

void LCD_WriteText(char text[ ]);

Beschrijving

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

Parameters

text char array

5.3.12 Poort

De Atmel Mega 32 heeft 4 in - /uitgangspoorten van elk 8 bit. Elke bit van een aparte poort kan als ingang of als uitgang geconfigureerd worden. Omdat echter het aantal pins van de Mega 32 Risc CPU beperkt is, zijn extra functies aan aparte poorten toegewezen. U vindt hier een tabel van de pintoewijzing.

➔ Het is belangrijk om de pintoewijzing te bestuderen voor u met de programmering begint, omdat belangrijke functies van de programma –ontwikkeling (b.v. de USB interface van het Application Board) op bepaalde pins liggen. Als deze poort s 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 u met poort _Write of poort WriteBit een "1" op een ingang schrijft, dan wordt de pull-up weerstand (referentieniveau VCC) geactiveerd en de ingang is gedefinieerd.

5.3.12.1 Port _DataDir

Poort functies voorbeeld

Syntax

void Poort _DataDir(byte port ,byte val);

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 =Poort B en val = 0x02, dan wordt pin 2 van de Atmel Mega (komt overeen met Poort B.1 – zie pintoewijzing) op uitgang geconfigureerd.

Parameters

port poort nummer (zie tabel)
val uitvoer byte

Tabel poortnummers

Definitie	Waarde
Poort A	0
Poort B	1
Poort C	2
Poort D	3

5.3.12.2 Port _DataDirBit

Poortfuncties

Syntax

void Port _DataDirBit(byte port bit,byte val);

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

Definitie	Poort bit
PortC.0	16
…	…
PortC.7	23
PortD.0	24
…	…
PortD.7	31

5.3.12.3 Port _Read

Poortfuncties

Syntax

byte Port_Read(byte port);

Beschrijving

Leest een byte van een specifieke poort . Alleen de pins van de poort die op ingang zijn geschakeld, leveren een geldige waarde naar de desbetreffende bitpositie in de gelezen byte terug. (Voor de afbeelding tussen portbits en de pins van de Atmel Mega Chips zie Pintoewijzing).

Parameters

port poortnummer (zie tabel)

Returnwaarde

Waarde van de poort

Tabel poortnummers

Definitie	Waarde
PortA	0
PortB	1
PortC	2
PortD	3

5.3.12.4 Port _ReadBit

Poortfuncties

Syntax

byte Poort _ReadBit(byte port);

Beschrijving

Leest een bitwaarde van de gespecificeerde poort. De desbetreffende pin van de poort moet op ingang zijn geschakeld. (Voor de afbeelding tussen portbits en de pins van de Atmel Mega Chips zie Pintoewijzing).

Parameters

portbit bitnummer van de poort (zie tabel)

Returnwaarde

Bitwaarde van de poort (0 of 1)

Poortbits tabel

Definitie	Poort bit
PortA.0	0
…	…
PortA.7	7
PortB.0	8
…	…
PortB.7	15

Definitie	Poort bit
PortC.0	16
…	…
PortC.7	23
PortD.0	24
…	…
PortD.7	31

5.3.12.5 Port _Write

Poortfuncties Voorbeeld

Syntax

void Port _Write(byte port ,byte val);

Beschrijving

Schrijft een byte op de gespecificeerde poort . Alleen de pins 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 portbits en de pins van de Atmel Mega Chips zie Pintoewijzing).

Parameters

port poortnummer (zie tabel)
val uitvoerbyte

Tabel poortnummers

Definitie	Waarde
PortA	0
PortB	1
PortC	2
PortD	3

5.3.12.6 Port _WriteBit

Poortfuncties

Syntax

void Poort _WriteBit(byte poort bit,byte val);

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).

Parameters

portbit bitnummer van de poort (zie tabel)
val mag 0 of 1 zijn

Tabel poortnummers

Definitie	Poortbit
PortA.0	0
…	…
PortA.7	7
PortB.0	8
…	…
PortB.7	15

Definitie	Poortbit
Poort C.0	16
…	…
Poort C.7	23
Poort D.0	24
…	…
Poort D.7	31

5.3.12.7 Poort voorbeeld

```
// Programma laat afwisselend de beide LEDs op het
// Application Board knipperen in een 1 seconde – ritme

void main(void)
{
      Port _DataDir(Poort D,0xc0);        // de bovenste beide bits van poort D worden op
      uitgang geschakeld

      while(true)          // eindeloze lus
      {
            Port _Write(Poort D,0x40);
            AbsDelay(1000);
            Port _Write(Poort D,0x80);
            AbsDelay(1000);
      }
}
```

5.3.13 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 resp. gelezen is. De seriële interface kan met snelheden tot 230.4kbaud gebruikt worden.

5.3.13.1 Serial_Init

Seriële functies **Voorbeeld**

Syntax

void Serial_Init(byte par, byte divider);

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, b.v. “SR_7BIT | SR_2STOPSR_EVEN_PAR” voor 7 bits per teken, 2 stopbits en even pariteit (zie ook voorbeeld). De baudrate wordt als verdelerwaarde, zoals ook in de tabel gespecificeerd wordt.

Parameters

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
SR1_STOP	1 stop bit
SR2_STOP	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

5.3.13.2 Serial_Read

Seriële functies

Syntax

byte Serial_Read(void);

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.

Returnwaarde

Ontvangen byte uit de seriële interface

5.3.13.3 Serial_ReadExt

Seriële functies

Syntax

word Serial_ReadExt(void);

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.

Returnwaarde

ontvangen byte uit de seriële interface

5.3.13.4 Serial_Write

Seriële functies **voorbeeld**

Syntax

void Serial_Write(byte val);

Beschrijving

Een byte wordt naar de seriële interface gestuurd.

Parameters

val de uit te voeren byte waarde

5.3.13.5 Serial_WriteText

Seriële functies

Syntax

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

Beschrijving
Alle tekens van de char array tot aan de laatste nul worden doorgegeven naar de seriële.

Parameters

Text char array

5.3.13.6 Serial voorbeeld

```
// Stringuitgifte naar de seriële interface
void  main(void)
{
        int  i;
        char  str[10];

        str=”test“;
        i=0;
        // Initialiseer interface met 19200Baud, 8 bit, 1 stop bit, geen pariteit
        Serial_Init(SR_8BIT|SR_1STOP|SR_NO_PART,SR_BD19200);

        while(str[i])  Serial_Write(str[i++]);    // Voer de string uit
}
```

5.3.14 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, dat u bij het niet gebruiken deze functies door het weglaten van de bibliotheek uit het project wegneemt. 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 tekens -ketting (char array) aan te geven.

5.3.14.1 Str_Comp

String functies

Syntax
```
char  Str_Comp(char  str1[ ],char  str2[ ]);
```
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

5.3.14.2 Str_Copy

String functies

Syntax

```
void Str_Copy(char destination[ ],char source[ ],word offset);
```

Beschrijving
De bronstring (source) wordt gekopieerd naar de doelstring (destination). Bij de kopieeractie wordt echter in elk geval het string –eindteken van de brontekens -ketting (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.

5.3.14.3 Str_Fill

String functies (bibliotheek *"String_Lib.cc"*)

Syntax

```
void Str_Fill(char dest[ ],char c,word len);
```

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

5.3.14.4 Str_Isalnum

String functies (bibliotheek *"String_Lib.cc"*)

Syntax

byte Str_Isalnum(char c);

Beschrijving
Een teken c wordt er op gecontroleerd, of het uit het alfabet stamt of een cijfer is.

Parameters

c	het te controleren teken

Returnwaarde

1	als het teken numeriek of alfabetisch is (zowel hoofd – als kleine letters)
0	overig

5.3.14.5 Str_Isalpha

String functies (bibliotheek *"String_Lib.cc"*)

Syntax

byte Str_Isalpha(char c);

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

5.3.14.6 Str_Len

String functies

Syntax

word Str_Len(char str[ ]);

Beschrijving

De lengte van de tekens –ketting (van de char array) wordt teruggegeven.

Parameters
str cursor op string

Returnwaarde

Aantal tekens in string (zonder de nul aan het eind).

5.3.14.7 Str_Substr

String functies (bibliotheek *"String_Lib.cc"*)

Syntax

```
int Str_Substr(char source[ ],char search[ ]);
```

Beschrijving
Een string search wordt gezocht in de string source. Als de gezochte tekens –ketting (char array) gevonden wordt, dan wordt de positie ervan teruggegeven.

Parameters

source string die doorzocht wordt
search tekens –ketting (char array)

Returnwaarde
Positie van de gezochte string in de onderzochte tekens –ketting (char array).
-1 overig

5.3.14.8 Str_WriteFloat

String functies

Syntax

```
void Str_WriteFloat(float n,int decimal,char text[ ]),word offset);
```

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.

5.3.14.9 Str_Writeint

String functies

Syntax

void Str_WriteFloat(int n,char text[ ]),word offset);

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.

5.3.14.10 Str_WriteWord

String functies

Syntax

void Str_WriteWord(word n,byte base,char text[ ],word offset,byte minwidth);

Beschrijving
Het woord n wordt geconverteerd in 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 base 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 b.v. 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.

5.3.15 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 15 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, b.v. 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.

Thread synchronisatie

Soms is het nodig dat een thread op de andere wacht. Dit kan b.v. 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

5.3.15.1 Thread_Cycles

Thread functies

Syntax

void Thread_Cycles(byte thread, word cycles);

Beschrijving

Zet het aantal bytecode instructies tot aan de volgende thread -wisseling op cycles.

➔ Als een thread nieuw 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 (0-15) nummer van de thread waarvan de cyclus veranderd moet worden
cycles aantal cycli tot aan het wisselen van de thread

5.3.15.2 Thread_Delay

Thread functies **Voorbeeld**

Syntax

void Thread_Delay(word delay);

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

5.3.15.3 Thread_Kill

Thread functies

Syntax

void Thread_Kill(byte thread);

Beschrijving

Beëindigt de verwerking van een thread. Wanneer als threadnummer 0 wordt doorgegeven, dan wordt het hoofdprogramma en daarmee de gehele interpreter -loop gestopt.

Parameters

thread (0_-15) nummer van de thread

5.3.15.4 Thread_Lock

Thread functies

Syntax

void Thread_Lock(byte lock);

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.

Parameters

lock bij 1 wordt de thread -wisseling verhinderd, bij 0 weer toegelaten.

5.3.15.5 Thread_Resume

Thread functies

Syntax

void Thread_Resume(byte thread);

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.

Parameters

thread (0-15) nummer van de thread

5.3.15.6 Thread_Signal

Thread functies

Syntax

void Thread_Signal(byte signal);

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

5.3.15.7 Thread_Start

Thread functies Voorbeeld

Syntax

```
void Thread_Start(byte thread, word func);
```

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	(0-15) nummer van de thread
func	naam van de functie waarin de nieuwe thread gestart wordt

5.3.15.8 Thread_Wait

Syntax

```
void Thread_Wait(byte signal);
```

Beschrijving

De thread krijgt de status "*wachtend*". D.m.v. Thread Resume() of Thread Signal() kan de thread weer in een inactieve toestand terechtkomen.

Parameters

signal	waarde van het signaal

5.3.15.9 Thread voorbeeld

```
// Demoprogramma voor multithreading – Bit 26 is SW1 en Bit 27 SW2
// het programma is niet ontkoppeld, het kort indrukken van een toets leidt daarom tot
// meervoudige invoer van de string

void thread1 (void)
{
      while(true)       // eindeloze lus
      {
           if (!Port _ReadBit (27) ) Msg_WriteTexrt (str2); // SW2 werd ingedrukt
      }
}
 char str1[12],str2[12];

void main(void)
{
     str1="toets 1";
     str2="toets 2";

     Port _DataDir(PortD,0);  // poort D op ingang
     Port _Write(PortD, 0xff); // Pull-up voor alle ingangen plaatsen
```

```
    Thread_Start(1,thread1); // Thread 1 starten

    while(true)          //  Eindeloze lus
    {
        if ( ! Port _ReadBit (26) )  Mag_WriteText (str1);   // SW1 werd ingedrukt
    }
}
```

5.3.15.10 Thread voorbeeld 2

```
//  Demoprogramma voor multithreading met Thread_Delay
void  thread1(void)
{
     while(true)       //  eindeloze lus
      {
          Mag_WriteText(str2);  Thread_Delay(200);  //  tekst wordt elke 2 sec uitgevoerd
      }
}
char  str1[12],str2[12];

void main(void)
{
      str1=”Thread1“;
      str2=”Thread2“;

      Thread_Start(1,thread1);   //  Thread 1 wordt gestart

      while(true)      //  eindeloze lus
      {
          Thread_Delay(100);  Msg_WriteText(str1);   //  tekst wordt elke seconde uitgevoerd
      }
}
```

5.3.16 Timer

Er staan in de C-Control Pro Mega 32 twee onafhankelijke Timer-Counters tot uw beschikking: *Timer_0* met 8 bit en *Timer_1* met 16 Bit. *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

5.3.16.1 Teller van gebeurtenissen

Hier twee voorbeelden hoe de timers als teller van gebeurtenissen gebruikt worden:

Timer0 (8 Bit)

```
//  Voorbeeld: pulstelling met CNT0
Timer_T0CNT();
pulse(n);                    //   n pulsen genereren
count=Timer_T0GetCNT();
```

Timer1(16 Bit)

```
//  Voorbeeld: pulstelling met CNT1
Timer_T1CNT();
pulse(n);                    //   n pulsen genereren
count=Timer_T1GetCNT();
```

5.3.16.2 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**

```
int  delval;
void main(void)
{
        delval=200;
        Timer0FRQ(10,2);
        while  (1)
        {
              AbsDelay(delval);
              TimerT0Stop();
              AbsDelay(delval);
              Timer_T0Start(2);
        }
}
```

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);
```

5.3.16.3 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 138,9µs 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
```

5.3.16.4 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 …
}
```

5.3.16.5 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: pulsbreedte –meting 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
}
```

5.3.16.6 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.

5.3.16.7 Timer_T0CNT

Timer functies

Syntax

```
void  Timer_T0CNT(void);
```

Beschrijving

Deze functie initialiseert de Counter0. De Counter0 wordt bij een positieve signaalflank op de ingang T0 (PIN1) opgehoogd.

Parameters

Geen

5.3.16.8 Timer_T0Disable

Timer functies

Syntax

```
void  Timer_T0Disable(void);
```

Beschrijving

Deze functie schakelt Timer 0 uit. Timerfuncties liggen op 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

Geen

5.3.16.9 Timer_T0FRQ

Timer functies

Syntax

void Timer_T0FRQ(byte period, byte PS);

Beschrijving

Deze functie initialiseert de Timer0 met de aangegeven voordeler en periodeduur, zie tabel. Het uitgangssignaal verschijnt op Port B.3 (PIN4). 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

5.3.16.10 Timer_T0GetCNT

Timer functies

Syntax

byte Timer_T0GetCNT(void);

Beschrijving

De waarde van Counter0 wordt gelezen. Als er een overflow plaats vindt, dan wordt de waarde 0xFF doorgegeven.

Returnwaarde

De gemeten waarde van de teller

5.3.16.11 TimerT0PW

Timer functies

Syntax

void Timer_T0PW(byte PW);

Beschrijving
Deze functie stelt een nieuwe waarde voor Timer0 in, zonder de voordeler te veranderen.

Parameters

PM pulsbreedte

5.3.16.12 TimerT0PWM

Timer functies

Syntax

void Timer_T0PWM(byte PW, byte PS);

Beschrijving
Deze functie initialiseert deTimer0 met de aangegeven voordeler en pulsbreedte, zie tabel.
Het uitgangssignaal verschijnt op Port B.3 (PIN4).

Parameters

PW 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

5.3.16.13 Timer_T0Start

Timer functies

Syntax

void Timer_T0Start(byte prescaler);

Beschrijving

Het produceren van de frequentie wordt met de bovenstaande instelling gestart. De voordeler moeit nieuw aangegeven worden.

Parameters

Prescaler voordeler (tabel prescaler)

5.3.16.14 Timer_T0Stop

Timer functies

Syntax

```
void Timer_T0Stop(void);
```

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

5.3.16.15 Timer_T0Time

Timer functies

Syntax

```
void Timer_T0Time(byte Time,byte PS);
```

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) getriggerd.

Parameters

Time tijdwaarde waarbij de interrupt getriggerd wordt
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

5.3.16.16 Timer_T1CNT

Timer functies

Syntax

```
void  Timer_T1CNT(void);
```

Beschrijving
Deze functie initialiseert de Counter1. De Counter1 wordt bij een positieve signaalflank op de ingang T1 (PIN2) opgehoogd.

Parameters

Geen

5.3.16.17 Timer_T1CNT_Int

Timer functies

Syntax

```
void  Timer_T1CNT_Int(word  limit);
```

Beschrijving
Deze functie initialiseert de Counter1. De Counter1 wordt bij een positieve signaalflank op de ingang T1 (PIN2) opgehoogd. Als de limiet bereikt is, wordt een interrupt getriggerd. De desbetreffende interrupt_Service_Routine moet vooraf gedefinieerd zijn.

Parameters

limit

5.3.16.18 Timer_T1Disable

Timer functies

Syntax

```
void  Timer_T1Disable(void);
```

Beschrijving
Deze functie schakelt Timer 1 uit. Timerfuncties liggen op I/O poorten. Als een timer niet meer nodig is en de poorten moeten als normale digitale I/Os gebruikt worden, dan moet de timerfunctie uitgeschakeld worden.

Parameters

Geen

5.3.16.19 Timer_T1FRQ

Timer functies

Syntax

void Timer_T1FRQ(word period,byte PS);

Beschrijving
Deze functie initialiseert de Timer1 met de aangegeven voordeler en periodeduur, zie tabel. Het uitgangssignaal verschijnt op PortD.5 (PIN19). 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

5.3.16.20 Timer_T1FRQX

Timer functies

Syntax

void Timer_T1FRQX(word period,word skew,byte PS);

Beschrijving
Deze functie initialiseert de Timer1 met de aangegeven voordeler periodeduur en fase-verschuiving van de beide uitgangssignalen, zie tabel. De uitgangssignalen verschijnen op PoortD.4 (PIN18) en Poort D.5(PIN19). 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)

5.3.16.21 Timer_T1GetCNT

Timer functies

Syntax

```
word  Timer_T1GetCNT(void);
```

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

5.3.16.22 Timer_T1GetPM

Timer functies

Syntax

```
word  Timer_T1GetPM(byte  Mode);
```

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

5.3.16.23 Timer_T1PM

Timer functies

Syntax

void Timer_T1PM(byte PS);

Beschrijving

Deze functie initialiseert Timer_1 voor de meting en stelt de voordeler in.

Parameters

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

5.3.16.24 Timer_T1PWA

Timer functies

Syntax

void Timer_T1PWA(word PW0);

Beschrijving:

Deze functie stelt een nieuwe pulsbreedte (kanaal_A) in voor Timer1, zonder de voordeler te veranderen.

Parameters

PW0 pulsbreedte

5.3.16.25 Timer_1PWB

Timer functies

Syntax

void Timer_T1PWB(word PW1);

Beschrijving:

Deze functie stelt een nieuwe pulsbreedte (kanaal_B) in voor Timer1, zonder de voordeler te veranderen.

Parameters

PW1 pulsbreedte

5.3.16.26 Timer_1PWM

Timer functies

Syntax

void Timer_T1PWM(word period, word PW0, byte PS);

Beschrijving:

Deze functie initialiseert Timer1 met de aangegeven voordeler, pulsbreedte en periodeduur, zie tabel. Het uitgangssignaal verschijnt op PortD.5(PIN19).

Parameters

period periodeduur
PW0 pulsbreedte
PS voordeler (tabel prescaler)

5.3.16.27 Timer_T1PWMX

Timer functies

Syntax

void Timer_T1PWMX(word period, word PW0, word PW1, byte PS);

Beschrijving:

Deze functie initialiseert Timer1 met de aangegeven voordeler, pulsbreedte voor kanaal A en B en periodeduur, zie tabel. Het uitgangssignaal verschijnt op Port D.4(PIN18) en PortD.5(PIN19).

Parameters

period periodeduur
PW0 pulsbreedte kanaal A
PW1 pulsbreedte kanaal B
PS voordeler (tabel prescaler)

5.3.16.28 Timer_T1Stop

Timer functies

Syntax

void Timer_T1Stop(void);
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

5.3.16.29 Timer_T1Time

Timer functies

Syntax

void Timer_T1Time(word Time,byte PS);

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) getriggerd.

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

5.3.16.30 Timer_T1Start

Timer functies

Syntax

void Timer_T1Start(byte prescaler);

Beschrijving
Het produceren van frequenties wordt met bovenstaande instelling gestart. De voordeler moet nieuw aangegeven worden.

Parameters

Prescaler voordeler (tabel prescaler)

Hoofdstuk 6

6 Aanhangsel

6.1 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 B.4-B.7 of A.6-A.7 per ongeluk in de software gebruikt? Zie Pintoewijzing). Zijn de jumpers op het Application board bij deze poorten ook gezet?

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). 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 PortD.2 (SW1) ook vast op GND gelegd worden.

4. De toetsbezetting van de editor “**xyz**” is ingesteld, maar sommige toetsenbordcommando”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.

5. 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