HISOFT C++: EEN C-COMPILER

IN ONS VORIGE BLAD HEBBEN WE EEN RECENSIE VAN DE DEVPAC80 ASSEMBLER GEPUBLICEERD. DIT KEER NEMEN WE DE PROGRAMMEERTAAL HISOFT C++ ONDER DE LOEP. C IS EEN TAAL DIE DE LAATSTE TIJD ONTZETTEND AAN POPULARITEIT WINT. ENERZIJDS IS DIT TE DANKEN AAN HET GROTE GEBRUIK IN DE PROFESSIONELE WERELD (ZOALS BIJVOORBEELD BIJ HET OPERATING SYSTEEM 'UNIX'). ANDERZIJDS KOMT DIT DOOR DE GROTE FLEXIBILITEIT DIE DEZE TAAL ONS BIEDT. OOK OP DE MSX IS AL GERUIME TIJD EEN C-COMPILER LEVERBAAR.

C is een compiler. Dat houdt in dat de programmatekst niet direct 'runbaar' is. Deze programmatekst, de 'SOURCE' genoemd, dient eerst door de compiler vertaald te worden, en wordt dan omgezet in machinetaal code. Deze code is direct vanuit DOS uitvoerbaar (het is gewoon een .COM file). Bovendien is de snelheid van het programma erg groot.

Naast compiler programmeertalen zijn er ook interpreters. Zo is de ingebouwde BASIC in de MSX ook een interpreter. Met een interpreter kan een programmatekst wel direkt worden uitgevoerd (voordeel: geen compileertijd nodig), maar wordt helaas veel langzamer uitgevoerd. De computer moet namelijk TIJDENS de programma-uitvoer regel voor regel de programmatekst vertalen. Een compiler heeft dit dus al gedaan tijdens het compileer proces, en de computer heeft daardoor direkt machinecode tot z'n beschikking waarmee razendsnel kan worden gewerkt.

Als een compiler optimaal zou werken, dan zouden hiermee net zo'n snelle programma's kunnen worden geschreven als met direkte machinetaalprogrammering. Met als belangrijk voordeel dat niet met moeilijke machinecode hoeft te worden geprogrammeerd, maar met een 'hogere' programmeertaal. Helaas zijn de meeste -zoniet alle- compilers nooit zo optimaal om ook daadwerkelijk deze maximale snelheid te halen.

ALS EEN COMPILER OPTIMAAL ZOU WERKEN, DAN ZOUDEN HIERMEE NET ZO'N SNELLE PROGRAMMA'S KUNNEN WORDEN GESCHREVEN ALS MET DIREKTE MACHINETAALPROGRAMMERING

DE GELEVERDE WAAR

De C-compiler wordt geleverd op een enkelzijdige schijf. Hierop staat de compiler, een editor om de programmatekst mee aan te maken, enkele bibliotheken en voorbeeldprogramma's.

HET LEUKE VAN C IS DAT JE JE EIGEN BIBLIOTHEEK VAN PROGRAMMA'S AAN KUNT LEGGEN.

Tevens wordt een behoorlijk handboek bijgeleverd. Uit de handleiding blijkt dat het programma oorspronkelijk uit de CP/M wereld stamt, en is omgezet naar MSX.

Bij het doorwerken van de handleiding moet goed worden opgelet dat men de CP/M en MSX hoofdstukken leest, omdat er ook hoofdstukken aanwezig zijn welke speciaal bedoeld zijn voor de Amstrad CPC computers. Zo was ik op een bepaald moment verdiept in de uitleg van een meegeleverde bibliotheek-file, toen bleek dat deze file helemaal niet op de diskette stond, aangezien deze alleen bedoeld was voor Amstrad gebruikers. De bijgeleverde editor is de bekende ED80. Deze wordt ook gebruikt bij de Devpac80 assembler (overigens, bij alle HiSoft-talen voor MSX) en is dus ook behept met dezelfde beperkingen. U kunt hier meer over lezen in ons vorige blad bij de Devpac80 recensie. Afgezien van wat euvels is deze editor verder goed te gebruiken.

BIBLIOTHEEK-FILES

Het leuke van C is dat je je eigen bibliotheek van programma's aan kunt leggen. Je kunt bijvoorbeeld programma-modules schrijven die een aantal grafische truuks uithalen. Een module die een lijn trekt, eentje die vlakken vult en een ander die menubalken laat verschijnen. Je kunt deze modules combineren in een file en ze in ieder ander programma gebruiken, simpelweg door een 'include' commando op te nemen in je programma, welke verwijst naar de gewenste bibliotheek-file. Je kunt op deze manier dus ontzettend structureel werken. Dit afgezien van het feit dat de C taal opzichzelf al zeer structureel in elkaar steekt.

Op de meegeleverde diskette staan drie bibliotheek files: CPM.LIB, BASIC.LIB en de STDIO.LIB. Naast deze drie heeft C zelf ook een aantal ingebouwde functies. Want dat is het leuke: Als je een C-commando gebruikt kan dat er een zijn uit een bibliotheek-file (wel moet deze met 'include' worden aangegeven), of het is er een die al in de taal ingebouwd is. Voor de gebruiker is er in gebruik van het programma geen verschil.

DE BASIC.LIB

De CPM.LIB en de STDIO.LIB, alsmede de ingebouwde C-commando's worden in het kort in het

handboek beschreven. Over de BASIC.LIB wordt echter met geen woord gerept. Nader onderzoek wijst uit dat in deze bibliotheek allerlei routines staan die allerhande zaken in de MSX aanspreken.
Zo is er standaard in C geen voorziening om met real getallen te rekenen. In de BASIC.LIB staan echter routines die dit via aanroep van de interne BASIC-ROM van de MSX toch mogelijk maken. Zo ook allerhande grafische routines zoals lijnen trekken, blokken tekenen enz., en geluid-aansturing, joystickbesturing e.d.. Ook zijn er routines om op een beperkte manier met windows te werken, en copieerroutines van VRAM naar RAM. Allemaal uitermate interessante zaken die niet standaard in de C taal zijn opgenomen, maar zo dus wel mogelijk worden.

ZO IS ER STANDAARD IN C GEEN VOORZIENING OM MET REAL GETALLEN TE REKENEN. IN DE 'BASIC.LIB' STAAN ECHTER ROUTINES DIE DIT VIA AANROEP VAN DE INTERNE BASIC-ROM VAN DE MSX TOCH MOGELIJK MAKEN.

Er zijn echter een aantal problemen met deze bibliotheek. Om te beginnen is er zoals reeds vermeld totaal geen handleiding van aanwezig. Het wordt dus spitten in de 'source'-file om te achterhalen wat er allemaal mogelijk is en hoe een en ander kan worden aangesproken.

Het wordt al erger als blijkt dat alle routines alleen maar de MSX1 standaard ondersteunen. De schermmodes 4 en hoger zijn dus niet te besturen en het zal dus nodig zijn om de bibliotheek zelf op een flink aantal punten aan te passen.

Het ergste is echter dat de bibliotheek niet correct werkt. Het is duidelijk dat bij het overzetten van CP/M naar MSX er even snel een MSX bibliotheek is bijgeschreven, zonder dat hier echt goed over is nagedacht. Dit resulteert in een prima werkend C-programma, maar een helaas slechte MSX bibliotheek (BASIC.LIB dus). Dit is ook de reden dat bijna alle bijgesloten voorbeeldprogramma's bij mij niet werkten.

HET PROBLEEM OPGELOST

Het probleem van de BASIC.LIB schuilt hem in het schakelen van de MSX-sloten. Om namelijk allerlei interne BASIC-ROM routines of de MSX-BIOS aan te kunnen spreken, is het nodig om bij zo'n aanroep het juiste MSX-slot te selecteren. Hiervoor wordt in de MSX de zgn. 'CALSLT' routine gebruikt. Er wordt opgeven welke page dient te worden geschakeld en naar welk slot, en de routine doet de rest. In de BASIC.LIB wordt echter geen gebruik gemaakt van deze routine. Wil men een routine uit de BASIC.LIB gebruiken dan dient men eerst procedure 'set_call()' aan te roepen. Er wordt dan een klein 'slot-omschakel'-programmaatje bovenin het geheugen geplaatst. Hier wordt de eerste fout gemaakt: Dit programma wordt geplaatst vanaf adres &hFFDA. Dit kan bij een MSX1, maar bij een MSX2 staan hier enkele variabelen die gebruikt worden voor de VDP. Het gevolg is dat gebruik van deze routine op een MSX2 het zaakje laat vastlopen.
De tweede fout die wordt gemaakt, schuilt in het machinetaal-programmaatje zelf. Hier worden de MSX ROM's ingeschakeld in page 0 en 1 middels direkte aansturing van het primaire slotselectie register op output adres &hA8. Na uitvoering van de gewenste BIOS-routine wordt er weer teruggeschakeld. De fout die wordt gemaakt is dat nu altijd slot 2 wordt geschakeld in page 0 en 1.
Dit is goed als daar ook de RAM van het systeem zit, maar dit is meestal niet het geval. We kunnen beide fouten corrigeren door:
A) Het programma te plaaten op een ander adres dan &hFFDA. De door mij voorgestelde oplossing is adres &h5FFE. Op dit adres bevindt zich een buffer die onder basic wordt gebruikt om direkt ingetypte commando's in op te slaan. Bruikbaar dus, als we niet met BASIC aan het werken zijn.
B) Aanpassing van het slot-schakel programmaatje. We dienen, voordat we de sloten omschakelen, eerst de huidige slotstand te bewaren. Na uitvoer herstellen we deze weer in de orginele staat. Beide voorgestelde oplossingen heb ik uitgewerkt, wat resulteerde in twee aanpassingen in de BASIC.LIB; Routine 'set_call()' en 'calslt()' dienen te worden aangepast. In fig. 1 ziet u de correcte routines.
Na deze aanpassingen werken ook de meegeleverde voorbeeldprogramma's!

**DE C-COMPILER IS BEHOORLIJK SNEL EN VOORAL BIJ GEBRUIK VAN EEN RAMDISK IS DE SNELHEID VAN COMPILEREN BEHOORLIJK.
BIJ EEN KLEIN TESTPROGRAMMA WERDEN 750 REGELS SOURCETEKST IN ONGEVEER 27 SECONDEN GECOMPILEERD. MET GEWONE DRIVE IS DIT EEN FACTOR TWEE LANGZAMER.**

VERDER ALLES OKEE

Na dit op zich minder positieve verhaal moet me van het hart, dat het zaakje verder prima functioneert. De C-compiler is behoorlijk snel en vooral bij het gebruik van een RAM-disk is de snelheid van compileren behoorlijk. Bij een klein testprogramma werden 750 regels sourcetekst in ongeveer 27 seconden gecompileerd (bij gebruik van een RAM disk, met een gewone drive een factor twee langzamer). Een probleem bij te grote sourceteksten is dat er te weinig geheugen kan zijn.
We kunnen dit ten dele voorkomen door zo'n source tekst in delen te compileren (en middels het include commando 'aan elkaar' te koppelen).
Verder is opvallend dat de compiler direkt een .COM file aflevert, en niet zoals meestal het geval is een tussenformaat. We kunnen dan met een 'linker' programma meerdere modulen aan elkaar knopen, eventueel tezamen

met machinetaal, om zo het uiteindelijke programma te verkrijgen. Dit is nu niet mogelijk, maar op zich hoeft dat geen al te groot nadeel te zijn.
We kunnen namelijk in de C source ook direkte 'inline' code opnemen. Dit is pure machinecode in de C source-tekst.

HET IS EEN BEHOORLIJK PAKKET, WAARMEE OP EEN GOEDE MANIER C PROGRAMMA'S KUNNEN WORDEN ONTWIKKELD.

LAATSTE KANTTEKENING

Er rest me nog een kanttekening, waarna niets het onbelemmerd werken met deze prima compiler meer in de weg staat. Het is namelijk zo, dat de compiler bovenin het geheugen een tabel aanlegt waarin de programma-variabelen worden bijgehouden. Deze tabel wordt zo hoog mogelijk geplaatst. Het kan echter zijn, dat op een andere computer een lager high-mem-adres aanwezig is.
Hierdoor kan het zijn dat een programma wel op het ene, maar niet op het andere systeem draait. We kunnen dit voorkomen, door deze variabelentabel lager te plaatsen (liefst zo laag mogelijk). We kunnen de compiler hiertoe dwingen middels het commando '#DATA'. Nadat het programma is gecompileerd meldt de compiler: '#DATA', gevolgd door een adres. We plaatsen nu vooraan in de source-file het commando '#DATA', gevolgd door het gegeven adres, en compileren het programma opnieuw;
En klaar is kees!

CONCLUSIE

Afgezien van een handleiding die hier en daar tekort schiet, en een bibliotheek-file die duidelijk te wensen over laat, kunnen we toch niet zeggen dat we met een slecht programma worden opgescheept. Het is een behoorlijk pakket, waarmee op een goede manier C programma's kunnen worden ontwikkeld. Als de in dit verhaal aangegeven wijzigingen worden overgenomen, dan is er alsnog goed te werken met de BASIC.LIB, en zijn dus de interne MSX routines goed bereikbaar (helaas wel alleen MSX1, voor MSX2 programma's zal er hier en daar wat moeten worden aangepast).
C is geen echt makkelijke programmeertaal, maar mensen die al enige ervaring met BASIC hebben opgedaan zullen -weliswaar met behulp van een leerboek- deze taal toch wel onder de knie kunnen krijgen.

Als men echt structureel wil programmeren op 'hoog' niveau en tegelijkertijd toch ook op 'laag' niveau wil kunnen werken, (BIOS, machinetaal, etc.) dan heeft men met deze HiSoft C++ een erg aantrekkelijke taal, die bovendien snel werkende programma's oplevert. **AvZ**

**ALLE HISOFT TALENPAKKETTEN (C++, PASCAL80, DEVPAC80 EN COBOL) ZIJN VERKRIJGBAAR BIJ DE MCM LEZERSSERVICE EN STICHTING GREEN/ MSX-INTERACTIEF. DAARNAAST ZULLEN DE VIER PAKKETTEN OP DE HCC-DAGEN WAARSCHIJNLIJK OOK LEVERBAAR ZIJN BIJ DE STAND VAN STG. CODE.
MOCHT U VRAGEN HEBBEN OVER DIT PAKKET C++, DAN KUNT U DIE METEEN DAAR STELLEN AAN ONZE RECENSENT. TOT DAN!!!**

```
/* volgende twee routines aanpassen in de file BASIC.LIB */
/* Put set_call() before any function which calls ROM using calslt()*/
/* usually near start of program */
void set_call()
/* Loads code for calslt() */
{ blt(0xf55E,"\365\333\250\62\156\365\346\360\323\250\361\315\356\21\365\76\0\323\250\361\311",21);
/* push af : inp a,(0xa8) : ld (0xF56E),a : and 0xF0 : out (0xa8),a : pop af : call ####: push af : ld a,#:
   out(0xa8),a : pop af : ret */ }
/*****************************************************************************************************/
/* Function to call MSX ROM */
calslt(a,bc,de,hl,addr)
 char a;
 unsigned bc,de,hl,addr;
  /* Call address in ROM (0-0x7FFF) */
  /* put 0 in unused parameters */
{
 reg_a=a; reg_bc=bc; reg_de=de; reg_hl=addr;
 inline(LD_HL_from, &reg_hl, LD_HL_into, 0xF56A);
   reg_hl=hl;
   inline(PUSH_IX,
        LD_A_from, &reg_a, LD_BC_from, &reg_bc,
         LD_DE_from, &reg_de, LD_HL_from, &reg_hl,
          CALL, 0xF55E, LD_A_into, &reg_a,
          LD_BC_into, &reg_bc, LD_DE_into, &reg_de,
          POP_IX);
/* results now in HL and BC,DE,A */}
```

FIGUUR 1; DE BEIDE AANGEPASTE ROUTINES