Programmierwettbewerb Echtzeit ’94

Dr.-Ing. Egmont Woitzel
FORTecH Software GmbH
Joachim-Jungius-Straße 9
18059 Rostock
Telefon (03 81) 4 05 94 72
Telefax (03 81) 4 05 94 71

Eigentlich handelt es sich nur um einen Zufall, daß die FORTecH Software GmbH an dem diesjährigen Programmierwettbewerb auf der Echtzeit '94 teilnehmen konnte. Verleitet durch die Mitteilung der Messeveranstalter, daß das Teilnehmerfeld auf 15 Teams vergrößert werden sollte, meldeten wir uns erst in allerletzter Minute an - und konnten nur noch den 2. Reserveplatz buchen. Wir erhielten zwar die Ausschreibung, aber mit einer Teilnahme rechneten wir nicht mehr. Unsere "Wettkampfvorbereitungen" beschränkten sich auf ein Minimum, wir löteten der Ausschreibung entsprechend ein Kabel zusammen und packten einen Rechner ein.

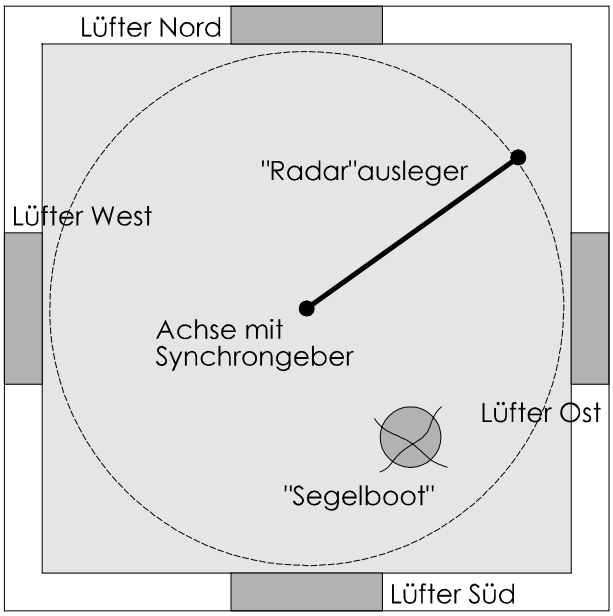
Geheimnisvolle Pusteblume

Ganz konnten wir uns dem Reiz der Programmierwettbewerbs aber auch nicht entziehen. Die in der Ausschreibung enthaltene I-O-Beschreibung und der Name "Pusteblume" des Modells gaben genug Raum für jede Art von Spekulation. Mit zwei Ausgängen und fünf Eingängen (aus Modellsicht) und einer geforderten Abtatsrate von 1 kHz fiel es in die Kategorie der über ein IBM-Printerport zu steuernden Modelle - was von den Ausrichtern vermutlich auch beabsichtigt wurde.

Ganz mysteriös wurde das Modell durch die benötigten "Zutaten". Die Stromversorgung 12 Volt/1,5 A lag im erwrteten Rahmen, aber die Notwendigkeit für eine zusätzliche 9 V-Trockenbatterie und ein Teelicht(!) las sich eher wie ein Plan von Egon Olsen.

Am Mittag des 15. Juni näherte sich unsere Spannung dem Höhepunkt. Nachdem wir von Uli Hoffmann ganz kurz vor Messebeginn davon unterrichtet wurden, daß unser Team auf den Reserveplatz 1 vorrücken konnte, blieb am Wettbewerbstag der 10. Platz tatsächlich leer, so daß es ganz plötzlich richtig ernst wurde. 10 Minuten vor dem Wettbewerbsstart erhielten wir grünes Licht.

Runde Segelboote und stumpfes Wasser



Die "Pusteblume" entpuppte sich als ein ca. 50 cm×50 cm großes Brett, auf dem 4 Lüfter und eine sonderbare drehbare Anlage motiert war (siehe Abb.). Die Lüfter ließen sich über 4 der 5 Eingängen einzeln Ein- und Ausschalten. Mit dem 5. Eingang ließ sich der in der ausgeteilten Beschreibung als Radar bezeichnete Ausleger über einen Antrieb im Uhrzeigersinn drehen. Eine Gabellicht-schranke unter dem Brett (wo auch der Antrieb zu finden war), lieferte pro Umdrehung einen Synchronimpuls. Bei unserem Modell lag dieser Punkt fast genau im Süden. Der Ausleger bestand aus Messingrohr, das zu einer Art drehbarem Tor zusammengelötet war, welches dicht über dem Boden eine Infrarot-Lichtschranke trug.

Von dem rätselhaften Teelicht wurde in dem Szenarium nur die Blechhülle und nicht die Kerze benötigt. Die Hülle diente als "Schiffsrumpf" für ein "Segelboot", das dreimal im Kreis um die Mittelachse herum gepustet werden sollte. Die Ausstattung des Bootes mit einem Segel war den Teilnehmern überlassen. Wir benutzten wie die meisten anderen Teilnehmer auch zwei gekreuzte Blätter Papier. Als eines der größten Probleme erwies sich die gegenüber der Gleitreibung sehr hohe Haftreibung des Bootes auf der das Wasser darstellenden blauen Folie, mit der das Brett beklebt war. Man konnte das Boot aus dem Stand nur mit einem "Orkan" in Bewegung bringen, der es dann aber "aus dem Wasser" fegte. Dagegen brachte erst ein Stück Tesa-Pack unter dem Boot Abhilfe.

Software-Wirbelwind

Für die Steuerung des Pusteblume verwendeten wir zum Erstaunen einiger der zuschauenden Echtzeit-Besucher die Beta-Version von comFORTH für Windows, dessen grafische Bedienoberfläche uns das nötige Entwicklungstempo erlaubte - nach etwa 2½...3 h hatten wir die Aufgabe gelöst.

Wenn man das in dieser Zeit geschriebene Programm durchsieht, dürfte es mit Ausnahme des benutzten Windows-Multimedia-Timers eigentlich kaum Verständnisprobleme geben. Die Basis für die ersten Versuche bildeten die Befehle zur Steuerung der Motoren, die durch die Worte `+M` und `-M` voneinander entkoppelt wurden.

Zur Bestimmung der Winkel-Position des Bootes verwendeten wir das einfachste mögliche Verfahren. In einer in 1-ms-Abständen zyklisch aufgerufenen Interruptserviceroutine (um etwas anderes handelt es sich bei dem benutzten Mulitmedia-Timer-Callback im Prinzip nicht) wird die Zählzelle `'W` inkrementiert. Die Positionsbestimmung erfolgt dann durch `POS`

im Pollingverfahren. Am Synchronpunkt wird die Zählzelle auf Null gesetzt, bei der nächsten Detektion des Bootes kann der Winkel dann einfach ausgelesen werden. Nachteilig ist bei diesem Verfahren, daß man im "worst-case" fast zwei Umrundungen des Radars abwarten muß, bis der Meßwert vorliegt, das waren bei unserem Modell etwa 4 s.

An der Implementierung des Callbacks kann man übrigens gut sehen, wie einfach die Bedienung eines 'C'-Interface in Assemblercode ist. Normalerweise erwartet das Multmedia-Interface an dieser Stelle eine 'C'-Funktion. Die bei Start und Stop aufzurufenden Windows-Funktionen sind in den Worten `+W` und `-W` gekapselt worden.

Etwas mehr Probieren erforderte das gezielte Herumpusten des Bootes. Es zeigte sich, daß man für eine zielgerichtete Bewegung des Bootes nicht nur mit einzelnen Lüftern pusten durfte, sondern sowohl mehrere Lüfter kombinieren als auch das nichtlineare Hochlaufverhalten berücksichten mußte. Als Basisschritt führt `PFF` einfachen Mehrlüfter-Puster kann aus. Das "um-die-Kurve-Pusten" besteht aus zwei zeitlich gegeneinander verschobenen "gerade-aus" und "nach-rechts"-Pustern. Die Algorithmen können sicherlich auch viel eleganter mit Hilfe von Tabellen und `CREATE-DOES>`-Konstruktionen implementiert werden, die vierfache Wiederholung fordert das geradezu heraus. In der Hektik des Programmierwettbewerbs, als noch nicht einmal klar war, ob das so funktioniert, war der Weg des Kopierens und Einfügens einfach der schnellere.

Beim Probieren zeigte sich dann, daß alle Lüfter verschieden starken Wind produzierten und vermutlich auch die Pusteblume nicht ganz genau waagerecht stand. Deswegen mußte jeder der 12 `Pxx-VALUE`'s unter erheblichem Probieren einzeln eingestellt werden.

Noch eine kurze Bemerkung zur Programmstruktur. Das entstandene Programm ist kein wirkliches Windows-Programm. Es ist nur bei seinem Aufruf aus dem comFORTH-Workspace heraus lauffähig. Bei seiner Programmierung als eigenes Applikations-Fenster wäre ein größerer Aufwand nötig gewesen. Da die Worte KEY? und MS nicht in Message-Handlern benutzt werden dürfen, hätte vor allem die Ablaufsteuerung radikal geändert werden müssen. Aber das war ja nicht unbedingt nötig...

PS: Das Quellprogramm wurde der besseren Verständlichkeit wegen für den Abdruck noch einmal nachbearbeitet. In der Hektik des Programmierwettbewerbs verzichteten wir fast völlig auf Kommentare. Die verwendeten Wortnamen und ihre Codierung wurden jedoch nicht geändert.

```
\ ========================================================
\ Programmierwettbewerb Echtzeit ’94
\ Die "Pusteblume" von Andreas Dobbertin & Uli Hoffmann
\
\ Egmont Woitzel und Udo Schütz
\ FORTecH Software GmbH
\
\ Rechner und Software:
\ PC-AT 486DX33 mit MS-DOS® 6.0 und Windows® 3.1
\ comFORTH für Windows (Beta II)
\ ========================================================

\ --- Utilities

\NEEDS CODE @:ASM86 1 INCLUDE


\ --- Ports

HEX
: PB@ ( ps: ===> 8b )( Pusteblume auslesen )
         379 CP@ ;

: PB! ( ps: 8b ===> )( Pusteblume beschreiben )
         378 CP! ;
DECIMAL


\ --- Inputs

HEX
: RADAR? ( ps: ===> ? )( ?t, wenn "Segelboot" detektiert )
         PB@ 40 AND 0= ;

: SYNC? ( ps: ===> ? )( ?t, wenn Synchronposition erreicht )
         PB@ 40 AND 0<> ;
DECIMAL


\ --- Outputs

HEX
 2 CONSTANT OST   \ Bitmaske für Windmaschine im Osten
 4 CONSTANT SUED  \ Bitmaske für Windmaschine im Süden
 8 CONSTANT WEST  \ Bitmaske für Windmaschine im Westen
10 CONSTANT NORD  \ Bitmaske für Windmaschine im Norden
DECIMAL

VARIABLE O ( ps: ===> addr )( enthält aktuellen Output )
         O OFF    \ initialisieren

: +M ( ps: mask ===> )( schaltet die Motoren der Maske dazu )
         O SET  O @ PB! ;
```

```
: -M ( ps: mask ===> )( schaltet die Motoren der Maske ab )
        O RESET  O @ PB! ;

: +R ( ps: ===> )( schaltet den Radarmotor dazu )
        1 +M ;

: -R ( ps: ===> )( schaltet den Radarmotor ab )
        1 -M ;


\ --- Winkelmessung

VARIABLE 'W ( ps: ===> addr )( enthält aktuellen Radarwinkel )

LABEL WCB ( ps: ===> addr )( enthält Code für den Windows- )
        ( Multimedia-Timer-Callback, liefert eigenen Offset )
        ( im Callback wird einfach nur 'W inkrementiert )
        ( auf dem Stapel liegen folgende Parameter: )
        ( wTimerID-16b wMsg-16b dwUser-32b dw1-32b dw2-32b )
        ( diese müssen mit der Rückkehr weggepopt werden )

        \ CPU in Ordnung bringen
        AX PUSH,                    \ Register retten
        DS PUSH,
        CS: 'DSEG #) AX MOV,        \ Selektor des Forth-DSEG
        AX DS MOV,                  \ als DS verwenden

        \ die eigentliche Callback-Aktion
        'W #) INC,

        \ CPU wieder aufräumen
        DS POP,                     \ Register zurückholen
        AX POP,
        16 # FAR RET,               \ Stack-Frame abräumen
        END-CODE

0 VALUE hEvent

: +W ( ps: ===> )( Winkelmessung aktivieren )
        1 timeBeginPeriod DROP              \ ms-Timer starten
        1 1 CSEG WCB 0, 1 timeSetEvent      \ WCB zuordnen
        TO hEvent ;                         \ Handle merken

: -W ( ps: ===> )( Winkelmessung deaktivieren )
        hEvent timeKillEvent DROP           \ WCB abmelden
        1 timeEndPeriod DROP ;              \ Timer stoppen
```

```
\ --- Positionsbestimmung

: POS ( ps: ===> +n )( Winkel des "Segelboots" bestimmen )
        ( Das Radar muß laufen und das Segelboot sehen! )
        BEGIN  SYNC?  UNTIL       \ warten bis Synchronpunkt
        'W OFF                    \ dort ist Winkel Null
        BEGIN  RADAR?  UNTIL      \ warten auf "Segelboot"
        'W @ ;                    \ Winkel auslesen

\ die in den nachfolgenden Worten benutzten Zahlen wurden
\ online "handvermessen", indem die Lüfter ausgeschaltet
\ wurden, das Seegelboot an die Grenzpositionen geschoben
\ und POS ausgeführt wurde - keine weiteren Umrechnungen
HEX
: SUED? ( ps: +n ===> ? )( ?t, wenn Segelboot im Südwasser )
        DUP 1B4 < SWAP 626 > OR ;

: WEST? ( ps: +n ===> ? )( ?t, wenn Segelboot im Westwasser )
        3FE < ;

: NORD? ( ps: +n ===> ? )( ?t, wenn Segelboot im Nordwasser )
        1B5 626 WITHIN? ;

: OST? ( ps: +n ===> ? )( ?t, wenn Segelboot im Ostwasser )
        3FE > ;
DECIMAL


\ --- Drehende Winde

\ Puste-Zeitkonstanten für jede einzelne Richtung und Phase
\ leider lagen die originalen Werte Hand-eingestellt nur
\ im RAM: diese Zahlen sind daher nur grobe Richtwerte,
\ die Werte der einzelnen Phasen waren alle verschieden

500 VALUE PO1     \ Puste nach Osten
300 VALUE PO2
250 VALUE PO3

500 VALUE PW1     \ Puste nach Westen
300 VALUE PW2
250 VALUE PW3

500 VALUE PN1     \ Puste nach Norden
300 VALUE PN2
250 VALUE PN3

500 VALUE PS1     \ Puste nach Süden
300 VALUE PS2
250 VALUE PS3

: PFF ( ps: mask u ===> )( einen u-ms-Puster loslassen )
        OVER +M           \ Lüfter dazuschalten
        MS                \ warten
```

```
         -M ;              \ und Lüfter wieder wegschalten

: >NORD ( ps: ===> )( Segelboot von West nach Nord pusten )
         BEGIN    SUED              PN1 PFF
                  SUED WEST OR      PN2 PFF
                  WEST              PN3 PFF
                  POS NORD?  KEY? OR
         UNTIL ;

: >WEST ( ps: ===> )( Segelboot von Süd nach West pusten )
         BEGIN    OST               PW1 PFF
                  OST SUED OR       PW2 PFF
                  SUED              PW3 PFF
                  POS WEST?  KEY? OR
         UNTIL ;

: >SUED ( ps: ===> )( Segelboot von Ost nach Süd pusten )
         BEGIN    NORD              PS1 PFF
                  NORD OST OR       PS2 PFF
                  OST               PS3 PFF
                  POS SUED?  KEY? OR
         UNTIL ;

: >OST ( ps: ===> )( Segelboot von Nord nach Ost pusten )
         BEGIN    WEST              PO1 PFF
                  WEST NORD OR      PO2 PFF
                  NORD              PO3 PFF
                  POS OST?  KEY? OR
         UNTIL ;


\ --- Herumpusten

: T ( ps: ===> )( eigentlich nur als Test gedacht, )
         ( tat diese Schleife schon alles nötige )
         TRUE -M
         +R +W
         BEGIN    \ einmal im Kreis herumpusten
                  >NORD >OST >SUED >WEST
                  KEY?
         UNTIL
         KEY DROP
         -R -W ;
```