Vogelaar Electronics

Dorpsstraat 90
3751 ES Bunschoten Netherlands
Telefoon +31 (0)33 2980727
Fax +31 (0)847 115096
E-mail info@vogelaar-electronics.com

DS_intro_alone.doc
28-08-2005

DRAFT

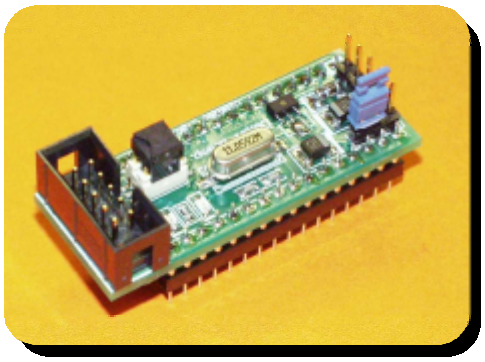
DelphiStamp VE08201

als zelfstandige controller
(introductie)

by Vogelaar Electronics
Bunschoten, Netherlands
28 – augustus - 2005

De DelphiStamp als zelfstandige controller.

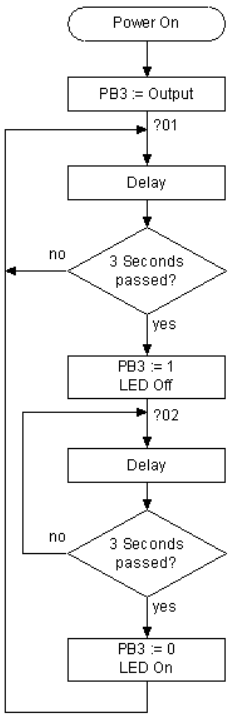
De DelphiStamp is een volledige controller ter grootte van een postzegel en is gemonteerd op een



connector. De controller is gebaseerd op de AtMega128 processor van Atmel en heeft 128 kByte flash geheugen aan boord. Deze controllers worden meestal in de assembly taal geprogrammeerd. Het schrijven en debuggen van assembly programma's is echter een tijdrovende zaak. Hoewel de DelphiStamp zeker kan woden gebruikt als ontwikkel-platform voor assembly programma's maken de meegeleverde software ontwikkeltools het mogelijk de programma's in Delphi te schrijven en te testen waardoor de ontwikkeling van de firmware drastisch wordt vereenvoudigd.

Er worden templates meegeleverd om snel te starten met het schrijven van een programma. Het eerste voorbeeld is het schrijven van een assembly programma. Zoals gebruikelijk is de eerste stap het tekenen van een flowchart die inzicht geeft in de fuctionaliteit van het te schrijven programma. In dit voorbeeld gaat een LED knipperen, 3 seconden aan en dan 3 seconden uit. De LED is via een serie weerstand aangeloten tussen de +5 Volt en PoortB uitgang 3. De controller schakelt de LED aan door de uitgang laag te maken, dit omdat de AtMega128 meer stroom kan sinken dan sourcen, d.w.z. deze manier van aansluiten geeft een geringere warmte onwikkeling in de controller.

Stand alone assembly program for DelphiStamp



De poorten van de microcontroller kunnen zowel als ingang en als uitgang worden gebruikt. Door een 1 op de corresponderende plaats in het Data Direction Register DDRB te schrijven wordt de poort-aansluiting een uitgang. Een 1 of een 0 op de corresponderende plaats in het data register PortB bepaalt of de uitgang hoog of laag wordt.

Het omzetten van de flowchart naar code start met een copy van de template file UMain.pas.

```
Link    $000,$180                       // AtMega128 : Code $0000..$FFFF [W]
                                        //             RAM $0100..$10FF
                                        //             EeRom $000..$7FF

Program Main;
(* DelphiStamp. Template for writing assembly programs.
   Provided by Vogelaar Electronics, Bunschoten Netherlands.
   Rev 0.10 23-08-05 Initial release *)

(* ============================ Start vectors ============================== *)

{#P LLChar ?}
{#P ABS 0}

Begin
   Asm
        SBI     DDRB,3                  ; PB3 := Output
?01     DEC     R29                     ; 3 Sec Delay
        BRNE    ?01
        SBIW    R30,1
        BRNE    ?01
        SBI     PortB,3                 ; PB3 := 1, Led Off
?02     DEC     R29                     ; 3 Sec delay
        BRNE    ?02
        SBIW    R30,1
        BRNE    ?02
        CBI     PortB,3                 ; PB3 := 0, Led On
        RJMP    ?01
   End;
End.
```

In het assembly programma wordt gebruik gemaakt van de instruktieset van de gebruikte microprocessor. De betekenis van de gebruikte instrukties zijn:

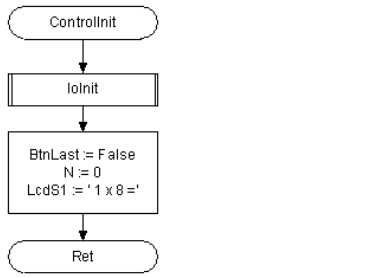
SBI	Set een bit in een I/O register
DEC	Decrement, verminder de waarde in een register met 1
BRNE	Branch if Not Equel, spring als het resultaat van de vorige bewerking ongelijk aan nul
SBIW	Substract Immedite Word, verminder de waarde van twee samengestelde registers
CBI	Clear Bit I/O, reset een bit in een I/O register

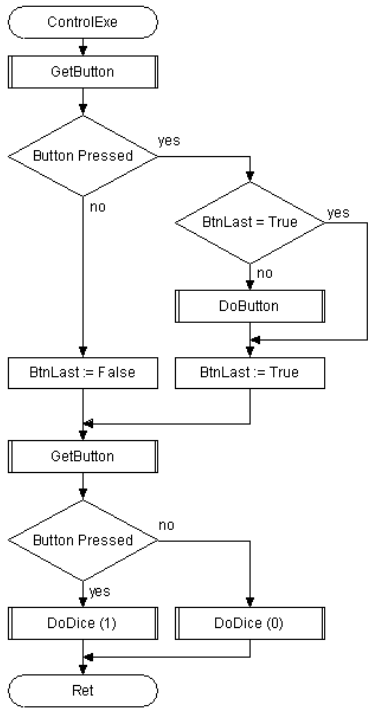
Het programma UMain.pas wordt gecompileerd met de cross-compiler PasAvr en met het monitor programma Mon485 ge-upload en gestart in de DelphiStamp.

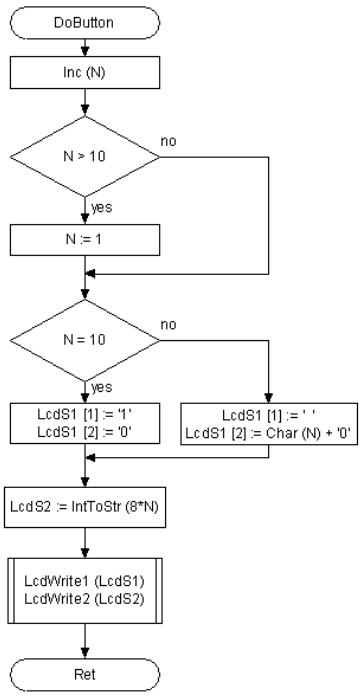
Het tweede voorbeeld beschrijft de ontwikkeling van een programma met gebruik van Delphi. De gewenste functionaliteit staat in onderstaande flowchart:

Demo program DelphiStamp

Multiplication Table







Een controller programma is te verdelen in 5 secties :

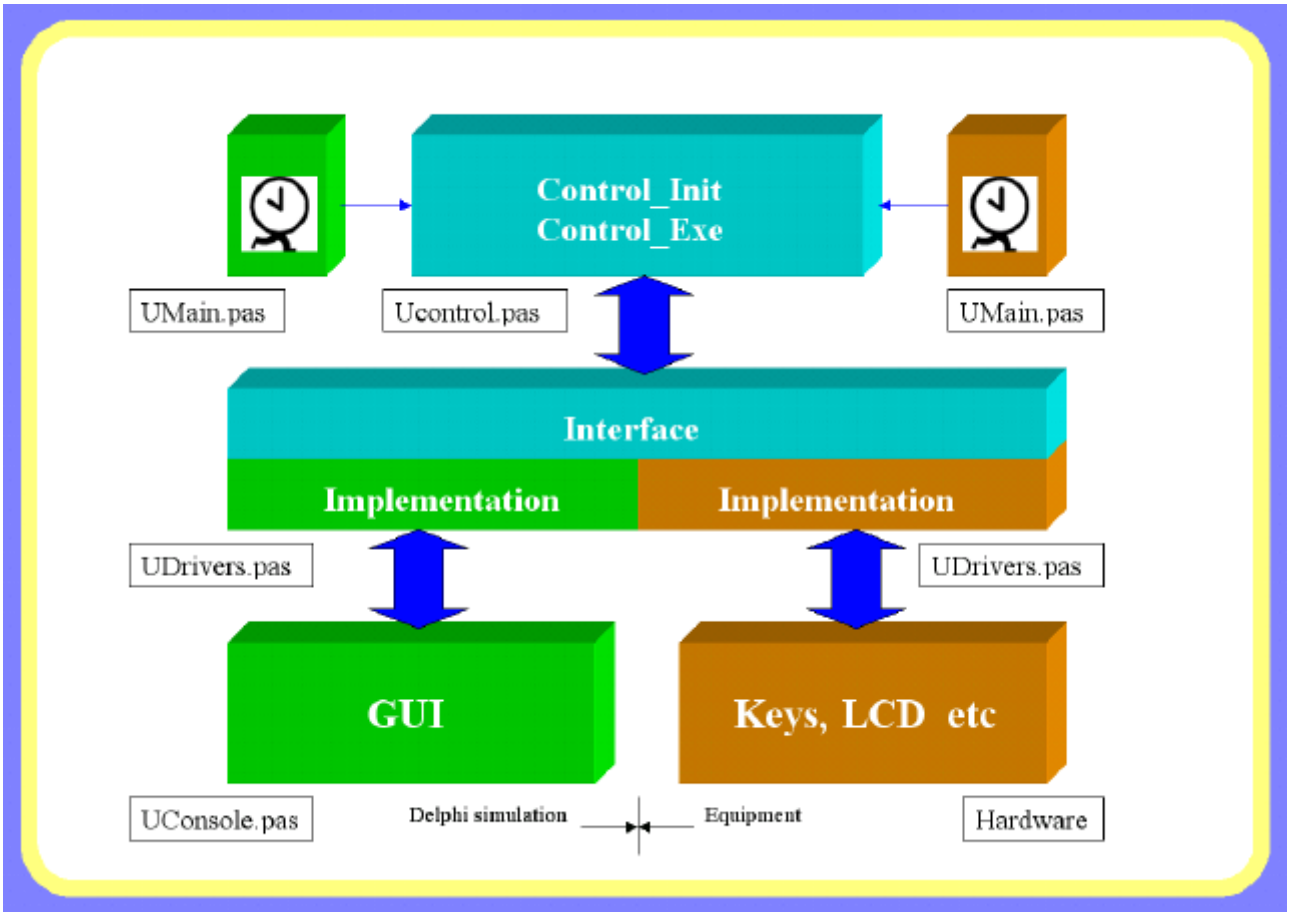
1. **ControlInit.** In deze procedure staan de eenmalige initializaties welke bij het inschakelen worden uitgevoerd.
2. **ControlExe.** Deze procedure wordt periodiek, bijvoorbeeld iedere 100 mSec, aangeroepen en bevat het besturingsprogramma.
3. **Interrupts.** Tijd critische programma delen die geen 100 mSec kunnen wachten om in ControlExe te worden uitgevoerd. Deze interrupts zijn meestal een onderdeel van de drivers.
4. **Drivers.** Procedures en functies voor het aansturen van hardware. In de BIOS van de DelphiStamp zijn drivers aanwezig voor standaard hardware.
5. **ControlIdle.** In deze procedure kunnen niet tijd critische taken worden geschreven omdat ControlIdle wordt uitgevoerd als alle overige taken zijn afgehandeld.

De taak van het hiernaast afgebeelde demonstratie programma is het laten zien van de tafel van 8 op een LCD display. Iedere keer dat op de knop wordt gedrukt verschijnt op het LCD de volgende regel van de tafel. Een LED geeft licht als de knop is ingedrukt.

GetButton, LcdWrite en DoDice zijn in de BIOS aanwezige drivers. DoDice is een driver voor het aansturen van zeven LED's in de configuratie van een dobbelsteen. Met DoDice (1) en DoDice (0) gaat de middelste LED aan en uit.

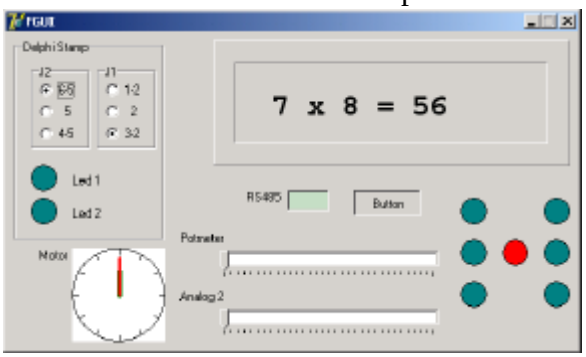
Het LCD is van het type 1 regel met 16 karakters. LcdWrite1 schrijft een string naar de linker 8 karakters, LcdWrite2 naar de rechter 8 karakters.

Door programma's voor de DelphiStamp in een gedefinieerde struktuur te ontwikkelen kan er gebruik worden gemaakt van templates. Deze struktuur ziet er als volgt uit:



Het controller programma wordt eerst in Delphi geschreven en getest. Een basis project is beschikbaar als uitgangspunt. Dit project bevat vier units t.w.

1. **UConsole.pas of UGUI.pas.** Dit is een TForm class en bevat de visuele componenten voor de bediening zoals Led's, schakelaars, meters, LCD's, osciloscopes, schrijvers e.d. Deze unit vervangt de echte hardware zoals aangesloten op de DelphiStamp. Hierdoor wordt het mogelijk de applicatie in simulatie mode op de PC te runnen. De UGUI.pas template bevat standaard hardware presentaties waarvoor de BIOS drivers bevat.



2. **UMain.pas.** Dit is het operating system en roept bij het opstarten ControlInit aan en iedere 100 mSec de procedure ControlExe. In de DelphiStamp voorziet het ook in twee servers n.l. de M485Server dat voor real-time monitoring en het het programmeren van de DelphiStamp wordt gebruikt alsmede de InterfaceServer (zie het vorige artikel).
3. **UDrivers.Pas.** Voor deze unit zijn er twee versies met een identiek interface deel. Een Delphi unit voor het aansturen van de gesimuleerde GUI hardware en een PasAvr unit voor

het aansturen van de echte hardware zoals aangesloten op de DelphiStamp. Deze hardware gerelateerde units worden 1 maal geschreven en zijn steeds herbruikbaar.

4. **UControl.pas.** Deze unit bevat de eigenlijke besturing t.w. de procedures ControlInit, ControlExe en ControlIdle. Nadat deze unit binnen Delphi is getest en in orde bevonden wordt het zonder wijzigingen door PasAvr gecompileerd en levert re ROM code voor de DelphiStamp.

Het schrijven van de applicatie voor het laten zien van de tafel van 8 begint met een copy van de meegeleverde template in Delphi en het editen van de file Control.pas zoals hieronder weergegeven. Het project wordt in Delphi gecompileerd, getest en eventuele bugs kunnen worden verwijderd. Zodra het project in orde is bevonden wordt een copy gemaakt van het PasAvr template. De Control.pas in het PasAvr project wordt vervangen door de Control.pas unit uit het Delphi project. Het PasAvr project wordt met de PasAvr cross-compiler gecompileerd en met het monitor programma Mon485 weggeschreven in de DelphiStamp en kan het programma worden gestart.

Meer informatie over de DelphiStamp kunt u vinden op de websites www.vogelaar-electronics.com en www.learningdelphi.info

```
Unit UControl;
(* Demo Control-unit DelphiStamp. Multiplication table and LED.
   Unit containing control algorithm
   Provided by Vogelaar Electronics, Bunschoten Netherlands
   Rev 0.10  15-04-05  Initial release *)

(* ========================== Interface ======================================= *)
Interface

{*} Uses    UDrivers, SysUtils; {*}            // For Delphi
{*! Uses    UDrivers, IntLib;   !*}            // For PasAvr

Procedure ControlInit;          // Called at start of program
Procedure ControlExe;           // Called at T = 100 mSec
Procedure ControlIdle;          // Called during spare time

(* ========================== Implementation ================================== *)
Implementation

Var     BtnLast : Boolean;
        LcdS1   : TStr20;       // Left  8 char's of LCD
        LcdS2   : TStr20;       // Right 8 char's of LCD
        N       : Byte;         // Counter for multiplication table

(* ================================ Local ============================================ *)

Procedure DoButton;
(* Executed on down press of button connected to PD0 *)

Var     X       : Byte;

Begin
   Inc (N); If N > 10 Then N := 1;
   If N = 10 Then                               // Fast calculation of LcdS1
   Begin
      LcdS1 [1] := '1'; LcdS1 [2] := '0'
   End
   Else Begin
      LcdS1 [1] := ' ';
      X := N + Ord ('0'); LcdS1 [2] := Char (X)
   End;
   LcdS2 := ' ' + IntToStr (8 * N);             // Slow calculation of LcdS2
   LcdWrite1 (LcdS1);
   LcdWrite2 (LcdS2);
End;

(* ================================ Global =========================================== *)

Procedure ControlInit;
(* Called once at start of program *)

Begin
   IoInit;
   BtnLast := False; N := 0;
   LcdS1 := ' 1 x 8 =';
End;


Procedure ControlExe;
(* Called at T = 100 mSec *)

Begin
   If   GetButton Then
   Begin
      If Not BtnLast Then DoButton;                     // DoButton on Press
      BtnLast := True
   End Else BtnLast := False;
   If GetButton Then DoDice (1) Else DoDice (0)         // Led monitors Button
End;


Procedure ControlIdle;
(* Called during spare time *)

Begin
End;

(* ================================ End ============================================== *)

End.
```