MP34

Optical Position Sensor

1. FUNCTION

1.1. Datasheet

1.1.1. Applications

This module is used in conjunction with a MP31 controller module and an optical sensor unit to record the movement of the sensor in relation to a fixed bar code. Onboard software allows via RS232 serial communication the setting of various parameters and the readout of the current position.

In addition the readout of three independent temperature sensors is implemented.

1.1.2. Specifications

Parameter	Value	Dimension
position resolution	> 1	mm
absolute code resolution (software!)	8	bit
absolute code pitch	> 10	mm
temperature range	-45 .. 130	°C
temperature error	< 0.7	°C

1.1.3. Power

Voltage	Current	Power
+7V	100 mA	0.7 W
Total		0.7 W

1.2. Blockdiagram

1.3. Function

All functions are controlled by a central micro controller module. This module (MP31) carries the program in firmware on an EPROM and has a 32 kByte RAM for data. Communication with the unit is done via the serial RS232 link. The commands are described later.

The position sensor unit consists of three optical sensors:
Two sensors on the relative bar code (simply a strip of equally spaced bars) detect the count and the direction of the movement (quadratur decoder). A third detector is able to sense an absolute code to synchronize in case of count errors.

The temperatur sensors (SMT 160-30) carry an onchip ADC which provides a square output signal. The controller counts the duty cycle and calculates the temperature information.

A basic (carrier) module (MP34) combines the functions of power supply for the controller and the sensors.

2. OPERATION

2.1. Connection

Connect a computer terminal with 9600 baud, 8 bits, no parity, 1 or 2 stopbits to the DB25 connector. Reset the unit and see the prompt:

```
POSITION Monitor
(C) vwa210295 V1.0 Physik.Inst. Uni HD
 Type "H" for help!
```

2.2. Commands

All functions are started by simply sending a letter (no CR) to the unit:

H gives a list of all available commands and the status of the modes, e.g.:

```
"H": Help (this screen)
"R": Reset position
"P": Position readout
"C": Continous =OFF
"S": Synchronize =ON
"L": Low resolution (code) =OFF
"B": Bit number =8
"A": Absolute code pitch =50
"G": Group of sensors =1
"1": Temperature sensor 1 readout
"2": Temperature sensor 2 readout
"3": Temperature sensor 3 readout
```

R set position counter to zero (=0)

P send current position counter

C toggle Continuous Mode ON/OFF
ON: whenever the position counter changes (sensor moves) it sends its current value.

S toggle Synchronize Mode ON/OFF
ON: absoulte code will be used to set the current position (AbsCode*AbsCodePitch)

L to improve the readability of the absolute code, it is possible to use a code which is double size (2 mm bars) width.

B the number of bits to be used for the absolute code can be 1..16 (preferable 8 or 10!)

A this determines the distance (in relation to the relative code) of the absolute codes. The reference point is the frame marker next to the MSB of the absolute code.

G it is possible to use a second sensor unit (group) to bridge a gap

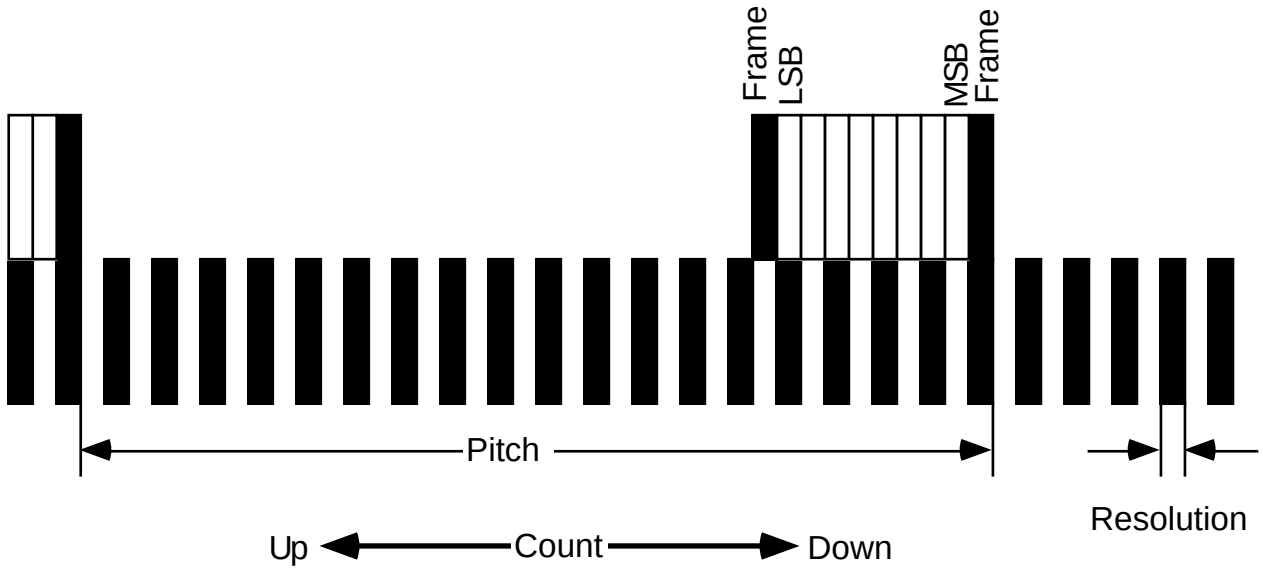
1 readout of temperature sensor 1(presently built in on MP34)

2 readout of temperature sensor 1

3 readout of temperature sensor 1

3. BAR CODE

3.1. Layout



4. PROGRAM

4.1. Listing

```
(*$M 0 0*)
program Position;
(* vwa200495 V1.1 *)

 var
  ch: char;
  position: Integer;
  syncPos: Integer;
  zeroCount: Integer;
  absCode: Integer;
  pitch: Integer;
  newPos: Boolean;
  tempCount: Word;
  continous: Boolean;
  synchronize: Boolean;
  group1: Boolean;
  dirBit: Boolean;
  codeBit: Boolean;
  lowRes: Boolean;
  outside: Boolean;
  theBit: Integer;
  nBits: Integer;
  firstBit: Integer;

 procedure interrupt IEX3;

 procedure CountUp;
 begin
  position := position + 1;
  if synchronize and not lowRes then
  begin
   if outside then
    begin
     if CodeBit then
      begin
       outside:=false;
       absCode := 0;
       syncPos := position;
       theBit := nBits;
       firstBit := nBits;
      end;
    end
   else
    begin
     if theBit>0 then
      begin
       theBit := theBit-1;
       if CodeBit then
```

```
        absCode := absCode OR (1 SHL theBit);
      end
     else if theBit=0 then
      begin
       theBit:=-1;
       {may check for right marker}
       if firstBit=nBits then
         position := absCode * pitch + position - syncPos;
      end
     else
       outside:=true;
    end;
  if CodeBit then
   zeroCount := 0
  else
   zeroCount := zeroCount + 1;
  end;
end;

procedure CountDown;
begin
 position := position - 1;
 if synchronize and not lowRes then
 begin
  if outside then
   begin
    if CodeBit then
     begin
      outside := false;
      absCode := 0;
      theBit := -1;
      firstBit := -1;
     end;
   end
  else
   begin
    theBit := theBit+1;
    if theBit<nBits then
     begin
      if CodeBit then
       absCode := absCode OR (1 SHL theBit);
     end
    else if theBit=nBits then
     begin
      {may check for left marker}
      if firstBit=-1 then
        position := absCode * pitch;
     end
    else
     outside := true;
   end;
  if CodeBit then
   zeroCount := 0
  else
   zeroCount := ZeroCount - 1;
```

```
 end;
end;

begin
 DirBit := not P1.6;
 CodeBit := not P1.5;
 if DirBit then
  CountDown
 else
  CountUp;
 if Abs(zeroCount) > nBits then
  begin
   outSide:=true;
   zeroCount := 0;
  end;
 newPos := true;
end;

procedure interrupt IEX2;

procedure CountUp;
begin
 position := position + 1;
 if synchronize and not lowRes then
 begin
  if outside then
   begin
    if CodeBit then
     begin
      outside:=false;
      absCode := 0;
      syncPos := position;
      theBit := nBits;
      firstBit := nBits;
     end;
   end
  else
   begin
    if theBit>0 then
     begin
      theBit := theBit-1;
      if CodeBit then
        absCode := absCode OR (1 SHL theBit);
     end
    else if theBit=0 then
     begin
      theBit:=-1;
      {may check for right marker}
      if firstBit=nBits then
        position := absCode * pitch + position - syncPos;
     end
    else
      outside:=true;
   end;
 if CodeBit then
```

```
  zeroCount := 0
 else
  zeroCount := zeroCount + 1;
 end;
end;

procedure CountDown;
begin
 position := position - 1;
 if synchronize and not lowRes then
 begin
  if outside then
   begin
    if CodeBit then
     begin
      outside := false;
      absCode := 0;
      theBit := -1;
      firstBit := -1;
     end;
   end
  else
   begin
    theBit := theBit+1;
    if theBit<nBits then
     begin
      if CodeBit then
       absCode := absCode OR (1 SHL theBit);
     end
    else if theBit=nBits then
     begin
      {may check for left marker}
      if firstBit=-1 then
        position := absCode * pitch;
     end
    else
     outside := true;
   end;
 if CodeBit then
  zeroCount := 0
 else
  zeroCount := ZeroCount - 1;
 end;
end;

begin
 DirBit := P1.6;
 CodeBit := not P1.5;
 if DirBit then
  CountDown
 else
  CountUp;
 if Abs(zeroCount) > nBits then
  begin
   outSide:=true;
```

```
    zeroCount := 0;
   end;
  newPos := true;
 end;

function GetTemp(port:Byte):Real;
 var
  High,Low,n:Word DATA;
begin
 High:=0;
 Low:=0;
 case port of
  1: for n:=1 to 60000 do
      if P1.1 then High:=High+1 else Low:=Low+1;
  2: for n:=1 to 60000 do
      if P1.2 then High:=High+1 else Low:=Low+1;
  3: for n:=1 to 60000 do
      if P1.3 then High:=High+1 else Low:=Low+1;
  end;
 GetTemp:=212.77*High/60000-68.085;
end;

 function AnyChar: Boolean;
 begin
  AnyChar := ri;
 end;

begin
 reset(serial);
 TH1 := 250;  {9600 Baud}
 EX2 := true;   {enable external INT2}
 I2FR := false;   {enable neg edge interrupt}
 EX3 := true;   {enable external INT3}
 I3FR := true;   {enable pos edge interrupt}
 EA := true;    {enable all interrupts}
 Writeln(' ');
 Writeln('POSITION Monitor');
 Writeln('(C) vwa200495 V1.1 Physik.Inst. Uni HD');
 Writeln(' Type "H" for help!');
 newPos := false;
 synchronize := true;
 continous := false;
 nBits := 8;
 position := 0;
 pitch:=50;
 group1:=true;
 P3.2:=false;
 P3.3:=true;
 lowRes := false;
 tempCount := 60000;
 repeat
  if AnyChar then
  begin
   Readln(ch);
   case ch of
```

```
 'h', 'H':
begin
 Writeln('POSITION Monitor');
 Writeln('(C) vwa200495 V1.1 Physik.Inst. Uni HD');
 WriteLn('Commands:');
 WriteLn(' "H": Help (this screen)');
 WriteLn(' "R": Reset position');
 WriteLn(' "P": Position readout');
 Write(' "C": Continous =');
 if continous then Writeln('ON') else Writeln('OFF');
 Write(' "S": Synchronize =');
 if synchronize then Writeln('ON') else Writeln('OFF');
 Write(' "L": Low resolution (code) =');
 if lowRes then Writeln('ON') else Writeln('OFF');
 WriteLn(' "B": Bit number =',nBits);
 WriteLn(' "A": Absolute code pitch =',pitch);
 Write(' "G": Group of sensors =');
 if group1 then Writeln('1') else Writeln('2');
 WriteLn(' "1": Temperature sensor 1 readout');
 WriteLn(' "2": Temperature sensor 2 readout');
 WriteLn(' "3": Temperature sensor 3 readout');
end;
 'r', 'R':
Position := 0;
 'p', 'P':
Writeln(Position : 8);
'1':
Writeln ('T1/degCels = ',GetTemp(1):6:2);
'2':
Writeln ('T2/degCels = ',GetTemp(2):6:2);
'3':
Writeln ('T3/degCels = ',GetTemp(3):6:2);
 'g', 'G':
 if group1 then
  begin
   Writeln('G=2');
   group1:=false;
   P3.2:=true;
   P3.3:=false;
  end
 else
  begin
   Writeln('G=1');
   group1:=true;
   P3.2:=false;
   P3.3:=true;
  end;
'c', 'C':
 if continous then
  begin
   Writeln('C=OFF');
   continous:=false;
  end
 else
  begin
```

```
        Writeln('C=ON');
       continous:=true;
      end;
     's',  'S':
     if synchronize then
      begin
       Writeln('S=OFF');
       synchronize:=false;
      end
     else
      begin
       Writeln('S=ON');
       synchronize:=true;
      end;
     'l',  'L':
     if lowRes then
      begin
       Writeln('L=OFF');
       lowRes:=false;
      end
     else
      begin
       Writeln('L=ON');
       lowRes:=true;
      end;
     'a','A':
     begin
       Write('Abs pitch=');
       ReadLn (pitch);
     end;
     'b','B':
     begin
       Write('Bit number=');
       ReadLn (nBits);
     end;
   end;
  end;
  if continous and newPos then
   begin
    Writeln(Position : 8);
    newPos:=false;
   end;
 until false;

end.
```