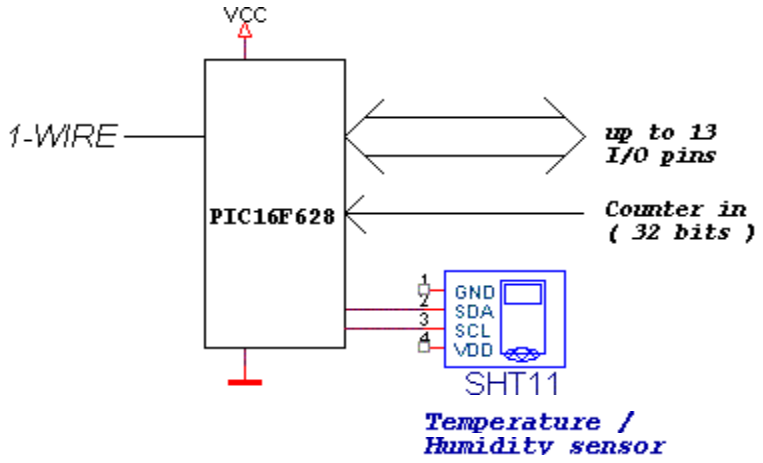
1-Wire Humidity sensor + I/O-module

Around a PIC16F628 microcontroller a circuit is made to connect a SHT-11 (two wire Humidity sensor of Sensirion) to a 1-wire (© Dallas semiconductor) network.
Besides the Humidity connection there are 13 I/O pins for general I/O and a 32 bits counter available.



Block diagram

Features

1. 13** TTL-In/Outputs capable to drive 25mA
2. Connection of a SHT11 Humidity/Temperature Sensor (see:http:/www.sensirion.com/)
3. 32 bits counter (resetable) input
4. 1-Wire input, multidrop
5. 4 x 32 bytes (= 128 bytes) Eeprom
6. Simple low cost one IC design
7. Build in Watchdog circuit, to keep circuit running
8. Low power @3mA without I/O current, @4mA with converting SHT11

** When a SHT11 sensor is used, 11 I/O pins are free

License

To avoid license problems with Dallas, the device must see a valid Dallas chip connected to Pin A0! If not then the device will not work! The cheapest way is to add a 1-wire serial number DS2401.

2. 1-Wire bus

See for a detailed 1-Wire (© Dallas) description the Dallas datasheets.

2.1 1-Wire commands

The circuit can be controlled by means of a 1-Wire bus.
We have several 1-Wire transactions:

1. 1-Wire Reset
2. ROM function commands
3. Memory / Control Commands with data transfer

Every action in the list will be explaned in the paragraphs below.

2.1.1 Reset / ROM-functions

See for a detailed description the Dallas datasheets.
Build in ROM-functions are:

Read ROM	0x33
Match ROM	0x55
Skip ROM	0xCC
Search ROM	0x0F

2.1.2 Memory / Control commands

Command	Value	Description
Write Scratchpad	0x4E	Till 32 bytes can be written to the internal scratch memory
Read Scratchpad	0xBE	Till 32 bytes can be read from the internal scratchpad (no CRC)
Copy Scratchpad	0x48	32 bytes are written to the Eeprom (takes till 50mS) (4 pages)
Recall EEprom	0xB8	32 bytes from Eeprom are copied into the scratchpad (4 pages)
Convert Sensor	0x44	Starts a conversion of the SHT11 sensor (Takes apr. 300mS)
Read Sensor	0x45	Reads the SHT11 conversion results (10 bytes)
Reset Counter	0x01	Resets the 32 bits counter to zero
Read Counter	0x02	Reads the 32 bits counter (message of 6 bytes). Counter-input is PortA_4
Read IO-identification	0x03	Read the IO-module identification
Setup PortA	0xF3	Sets the Port A pins to be Input or Output
Setup PortB	0xF4	Sets the Port B pins to be Input or Output
Read PortA	0xF5	Reads the contents of PortA (inputs PortA_0..PortA_5), 6 bits
Read PortB	0xF6	Reads the contents of PortB (inputs PortB_1..PortB_7), 7 bits
Write PortA	0xF7	Writes the value to PortA, 5 bits
Write PortB	0xF8	Writes the value to PortB, 7 bits

Some of the above commands are original Dallas commands, for explanation see Dallas datasheets.

Some specific commands are explained below, complete with 1-Wire Master software examples.
(TouchReset() => send 1-Wire reset; TouchByte() => send 1-Wire byte and returns read byte)

2.1.2.1 Convert Sensor (0x44)

After receiving this command, a communication cycle over the I²C-bus to the STH11 will start. Both the temperature and the humidity are measured. An I²C-communication cycle is:

1. Connection Reset,
2. Start sequence (I²C-Start + I²C-Stop),
3. TemperatureConversionCommand,
4. Wait till STH11 finished conversion (14 bits = 220mS)
5. Read 3 times I²C-bytes (last one with a NACK)
6. Connection Reset,
7. Start sequence (I²C-Start + I²C-Stop),
8. HumidityConversionCommand,
9. Wait till STH11 finished conversion (12 bits = 55 mS)
10. Read 3 times I²C-bytes (last one with a NACK)

1-Wire master software example:

```
TouchReset();               /* 1-wire Reset                    */
TouchByte(MatchRom);        /* 0x55                            */
for(i=0;i<8;i++)            /* 64-bit ROM-code                 */
  TouchByte(ident[i]);
TouchByte(ConvertSensor);   /* 0x44, SHT11 Convert command     */
                            /* During conversion (300mS) the PIC16F628 will not respond */
```

2.1.2.2 Read Sensor (0x45)

With this command the results of a Convert Sensor command can be read. Both the temperature and the humidity values are available. The command will return 10 bytes of data:

'T'	**'='**	Temperature Byte1 MSB	Temperature Byte2 LSB	CRC	**'H'**	**'='**	Humidity Byte1 MSB	Humidity Byte2 LSB	CRC

The Temperature Byte1,2; Humidity Byte 1,2 and CRC are direct results of the SHT11 conversion.
See for more detailes the SHT11 datasheet.

The most left character (**'T'**) is first returned to the master. The 'T','=','H' and another '=' are added to the SHT11 results to do some checking. If these characters are not present in the string some went wrong (communication failure, no sensor).

1-Wire master code snipper:

```
TouchReset();                /* 1-Wire Reset               */
TouchByte(SkipRom);          /* 0xCC                       */
TouchByte(ReadSensor);       /* 0x45, read 10 bytes result */
if(TouchByte(0xFF) == 'T')   /* First char = 'T'           */
{
  if(TouchByte(0xFF) == '=')  /* Second char = '='         */
  {
    So = TouchByte(0xFF) << 8;/* Temperature MSB           */
    So |= TouchByte(0xFF);    /* Temperature LSB           */
    TouchByte(0xFF);          /* CRC, can be checked !     */
    Temp = -40.0 + 0.01*So;   /* Calulation formula for the temperature */
    TouchByte(0xFF);          /* char = 'H', can be checked */
    TouchByte(0xFF);          /* char = '='                */
    So = TouchByte(0xFF) << 8;/* Humidity MSB              */
    So |= TouchByte(0xFF);    /* Humidity LSB              */
    TouchByte(0xFF);          /* CRC, can be checked       */
    RH = -4.0 + 0.0405*So - 0.0000028*So*So; /* Humidity calculation    */
    sprintf(buf,"%3.1f °C, %3.1f %",Temp,RH);/* Something to the screen */
  }
  else
  {                           /* Some went wrong */
    sprintf(buf,"No sensor data");
  }
}
```

2.1.2.3 Copy Scratch, 0x48

This commands copies the scratchpad to one of the 4 pages of the EEprom.

1-Wire master software example:

```
TouchReset();             /* 1-wire Reset                    */
TouchByte(MatchRom);      /* 0x55                            */
for(i=0;i<8;i++)          /* 64-bit ROM-code                 */
  TouchByte(ident[i]);
TouchByte(CopyScratch);   /* 0x48, copy 32 bytes of the scratch to eeprom */
TouchByte(Page);          /* 1 (0..31),2 (32..63),3 (64..95) or 4 (96..127) */
TouchReset();             /* 1-wire Reset                    */
```

2.1.2.4 Recall EEprom, 0xB8

This commands copies one of the 4 EEprom pages of 32 bytes to the scratchpad.

1-Wire master software example:

```
TouchReset();             /* 1-wire Reset                    */
TouchByte(MatchRom);      /* 0x55                            */
for(i=0;i<8;i++)          /* 64-bit ROM-code                 */
  TouchByte(ident[i]);
TouchByte(RecallEEprom);  /* 0xB8, copy 32 bytes of the eeprom to scratch */
TouchByte(Page);          /* 1 (0..31),2 (32..63),3 (64..95) or 4 (96..127) */
TouchReset();             /* 1-wire Reset                    */
```

2.1.2.5 Reset Counter, 0x01

This commands resets the 32 bits counter to zero.

1-Wire master software example:

```
TouchReset();             /* 1-wire Reset                    */
TouchByte(MatchRom);      /* 0x55                            */
for(i=0;i<8;i++)          /* 64-bit ROM-code                 */
  TouchByte(ident[i]);
TouchByte(ResetCounter);  /* 0x01, reset the 32 bits counter  */
TouchReset();             /* 1-wire Reset                    */
```

2.1.2.6 Read Counter, 0x02

This commands reads the contents of the 32 bits counter.
6 bytes are returned:

'C'	**'='**	Counter Byte1 LSB	Counter Byte2	Counter Byte3	Counter Byte4 MSB

1-Wire master software example:

long int counter;

```
TouchReset();                          /* 1-wire Reset                    */
TouchByte(MatchRom);                   /* 0x55                            */
for(i=0;i<8;i++)                       /* 64-bit ROM-code                 */
  TouchByte(ident[i]);
TouchByte(ReadCounter);                /* 0x02, read the 32 bits counter   */
if(TouchByte(0xFF) == 'C')             /* First char must be 'C'          */
{
  if(TouchByte(0xFF) == '=')           /* Second char == '='              */
  {
    counter  = TouchByte(0xFF);        /* LSB */
    counter |= (TouchByte(0xFF)) << 8;
    counter |= (TouchByte(0xFF)) << 16;
    counter |= (TouchByte(0xFF)) << 24; /* MSB */
    sprintf(buf,"counter = %ld",counter);
```

```
  }
  else
  {                                   /* Check char error */
    sprintf(buf,"counter = Error!");
  }
}
else
{                                     /* Check char error */
  sprintf(buf,"counter = Error!");
}
TouchReset();                         /* 1-wire Reset     */
```

2.1.2.7 Read IO-identification, 0x03

This reads the device code of the IO-module. Because the number of a DS2401 is used to identify the device to the 1-Wire bus, with this command it's possible to determine the sort of IO-module.
An IO-module based on a 16F628 will return 0x01 at this command.
An original DS2401 will return a 0xFF (unknown command)

1-Wire master software example:

```
TouchReset();             /* 1-wire Reset                    */
TouchByte(MatchRom);      /* 0x55                            */
for(i=0;i<8;i++)          /* 64-bit ROM-code                 */
  TouchByte(ident[i]);
TouchByte(0x03);          /* Read IO-identification          */
IO_device = TouchByte(0xFF); /* == 0x01 when 16F628, 0xFF when a DS2401 */
TouchReset();             /* 1-wire Reset                    */
```

2.1.2.8 Setup PortA, 0xF3; Setup PortB, 0xF4

These commands sets the 13 I/O-pins to be input or output. 6 bits are available for PortA and 7 bits for PortB. Note that input A5 is input only!

Writing an ONE to the specific bit position sets the I/O pin to be INPUT.
Writing a ZERO to the specific bit position sets the I/O pin to be OUTPUT.

PortA setup byte definition:

PortA_0	Bit0 = '0'	Output	Push/Pull 25mA
	Bit0 = '1'	Input	No internal pullup resistor
PortA_1	Bit1 = '0'	Output	Push/Pull 25mA
	Bit1 = '1'	Input	No internal pullup resistor
PortA_2	Bit2 = '0'	Output	Push/Pull 25mA
	Bit2 = '1'	Input	No internal pullup resistor
PortA_3	Bit3 = '0'	Output	Push/Pull 25mA
	Bit3 = '1'	Input	No internal pullup resistor
PortA_4	Bit4 = '0'	Output	Open collector 25mA
	Bit4 = '1'	Input	No internal pullup resistor; also counter input rising edge

Bit5..7 of PortA setup have no effect

PortB setup byte definition:

PortB_1	Bit1 = '0'	Output	Push/Pull 25mA
	Bit1 = '1'	Input	Internal pullup resistor
PortB_2	Bit2 = '0'	Output	Push/Pull 25mA
	Bit2 = '1'	Input	Internal pullup resistor
PortB_3	Bit3 = '0'	Output	Push/Pull 25mA
	Bit3 = '1'	Input	Internal pullup resistor
PortB_4	Bit4 = '0'	Output	Push/Pull 25mA
	Bit4 = '1'	Input	Internal pullup resistor
PortB_5	Bit5 = '0'	Output	Open collector 25mA
	Bit5 = '1'	Input	Internal pullup resistor
PortB_6	Bit6 = '0'	Output	Push/Pull 25mA
	Bit6 = '1'	Input	Internal pullup resistor
PortB_7	Bit7 = '0'	Output	Push/Pull 25mA
	Bit7 = '1'	Input	Internal pullup resistor

Bit 0 is used for the 1-Wire input

Code example:

```
TouchReset();               /* Reset 1-Wire                          */
TouchByte(SkipRom);         /* 0xCC                                  */
TouchByte(SetupPortA);      /* 0xF3,                                 */
TouchByte(0x03);            /* PortA_0 and PortA_1 input, rest output */
TouchReset();               /* Reset 1-Wire                          */
TouchByte(SkipRom);         /* 0xCC                                  */
TouchByte(SetupPortB);      /* 0xF4,                                 */
TouchByte(0x81);            /* PortB_7 input, rest output            */
TouchReset();               /* Reset 1-Wire                          */
```

2.1.2.9 Read PortA, 0xF5; Read PortB, 0xF6

With these commands the value of the specific I/O-port is read.
The I/O-mapping of the read value is shown in the table below:

Bit	Port A (0xF5)	Port B (0xF6)
0 ; 0x01	PortA_0 *(DS2401)*	n.a. (1-wire input)
1 ; 0x02	PortA_1	PortB_1
2 ; 0x04	PortA_2	PortB_2
3 ; 0x08	PortA_3	PortB_3
4 ; 0x10	PortA_4	PortB_4
5 ; 0x20	PortA_5	PortB_5
6 ; 0x40	n.a	PortB_6
7 ; 0x80	n.a	PortB_7

After the Read command the port can be read till a 1-Wire reset.
This feature makes it possible to poll for an I/O-bit.

Code example:

```
TouchReset();                 /* Reset 1-Wire                */
TouchByte(SkipRom);           /* 0xCC                        */
TouchByte(ReadPortA);         /* 0xF5,                       */
while(TouchByte(0xFF) & 0x02);/* Wait till PortA_1 gets low  */
TouchReset();                 /* Reset 1-Wire                */
```

2.1.2.10 Write PortA, 0xF7; Write PortB, 0xF8

With these commands a value can be written to an I/O-port.
The I/O-mapping for the write value is shown in the table below:

Bit	Port A (0xF7)	Port B (0xF8)
0 ; 0x01	PortA_0	n.a. (1-wire input)
1 ; 0x02	PortA_1	PortB_1
2 ; 0x04	PortA_2	PortB_2
3 ; 0x08	PortA_3	PortB_3
4 ; 0x10	PortA_4	PortB_4
5 ; 0x20	Input only	PortB_5
6 ; 0x40	n.a	PortB_6
7 ; 0x80	n.a	PortB_7

After the Write command more values can be written to the port.
This makes a relative fast bytetransfer possible to a port (~1mS per byte).

Code example:

```
TouchReset();                 /* Reset 1-Wire                    */
TouchByte(SkipRom);           /* 0xCC                            */
TouchByte(WritePortB);        /* 0xF8,                           */
for(i=0;i<128;i++)
  TouchByte(i<<1);            /* Write some values to PortB      */
TouchReset();                 /* Reset 1-Wire                    */
```

3 Usage of the scratch memory

The PIC16F628 has a build in scratch memory of 32 bytes. This memory is used by the following processes:

1. Write scratch (till 32 bytes)
2. Read scratch (till 32 bytes)
3. Copy scratch (4 pages of 32 bytes to Eeprom)
4. Recall Eeprom (4 pages of 32 bytes from the Eeprom)