1-WIRE WATER METER APPLICATION NOTE

INTRODUCTION

For automation purpose, it is often required to measure liquids flowing in the pipes. Not only for collecting consumption history but also to allow immediate control of volume delivered.

Water is a valuable and finite resource on our earth. Measuring precisely and getting immediate interaction on the consumption could help us to become aware of and adjust our inappropriate habits. This is good for the earth, this good for the budget.

For example, acquisition of flow data in realtime for both cold and warm water in the bathroom could allow to program smart behaviors like triggering a warning about shower consumption.

THE FLOW SENSOR

The flow sensor is a device installed inline on the pipe that sense the quantity of liquid flowing through the device body.

Many flow sensors are available on the market. For most of them, the sensing is done with a small turbine that produces pulse output. Each pulse represents a fixed volume of liquid traversing the sensor. A flow sensor is characterized by the number of pulse per liter also called K factor.

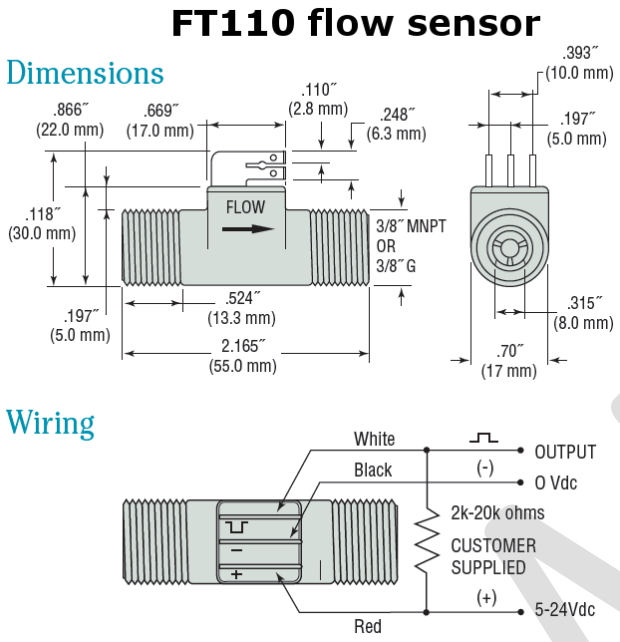
EASE OF USE

Measuring such pulse signals is easily done with the COUNT function from the BAE0910 chip. The CNT input pin automatically increments the COUNT register when the voltage changes on the pin (rising/falling edge).
Here are the registers related to the CNT input:

- COUNT is a 32bit register holding the counted pulses. When connected to a flow sensor, this represents the volume of liquid that has flowed through the sensor device.
- CPS is a 16bit register that provides the instantaneous flow by calculating the pulses occurred in one second interval. (Count Per Second is updated each second)
- MAXCPS is a 16bit register that hold the peak flow by retaining the maximum pulse per second reached since the last time the register was cleared.
- ALCT is a user configurable 32bit setpoint that trigger an alarm bit when the COUNT value exceeds the ALCT value.
- ALCPS is a user configurable 16bit setpoint that trigger an alarm bit when CPS exceeds the set point.
- CNTC configuration register define edge polarity and allows to pause the counting.
- ALARM bits: the alarm bit allows getting attention from the master by responding to conditional search commands.

AN EXAMPLE OF INDUSTRIAL FLOW SENSOR

This flow sensor requires 3 wires for connection to GND, +5V and CNT. The customer supplied pullup resistor is already present on both tinyboards and prototyping boards and is not required.



A direct connection is trivial:

Sensor	**BAE board**
0 Vdc (black)	GND (ground)
OUTPUT (white)	CNT
5-24Vdc (red)	+5V

http://www.gemssensors.com/uploadedFiles/Literature/Spec_Sheets/FT110.pdf

Such flow sensor is capable of measuring 0.5 to 30 liters per minute of water at up to 100°C. It produces very high resolution output (K factor from 1000 to 6900 pulses per liter) with 1% accuracy.

BEVERAGE FLOW SENSOR EXAMPLE

This flow meter is designed specifically for the drinks including pure water, beer, wine and spirits.

Technical Specifications	
Flow	0.5 – 15 L/min
Supply voltage	4.5 to 24Vdc
Maximum working pressure	10 bar
K factor	1120 pulses per litre
Temperature range	0 to +100°C
Output low	100mV max.
Connection	3/8" push on

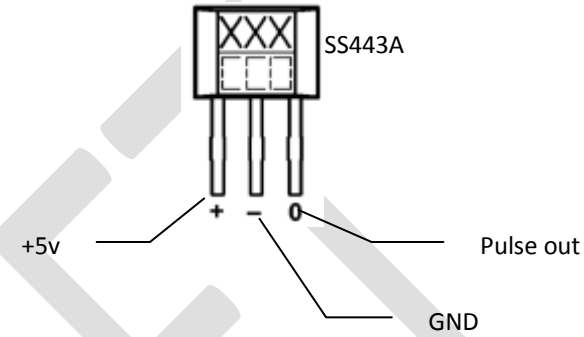
Such sensor also produces pulse output which the count is proportional to the volume flowing through the device.

MODIFIED WATER METER EXAMPLE *(NOT YOUR OFFICIAL ONE)*

Most of water counter found in DIY stores are designed with a rotating magnet inducting movement to the gears
A typical mechanical water meter has two parts:

1. a sealed copper body where the water flowing activate a small propeller
2. a plastic head containing mechanical reduction gears that increments the digits.

The movement from the turbine is transmitted to the gears via a magnetic field. It is possible to measure the rotation of the magnetic field with a very small hall-effect sensor (ex. the SS443A measure 4.1 x 2.8 x 1.6 mm).
As you may note, the connection of the hall sensor is directly compatible with the CNT input header from tinyboards. Each revolution of the propeller generates a pulse readable captured by the BAE0910 chip.



The easiest method is to remove the plastic part and stick the sensor in the center of the copper body (red arrow).
It is also possible keep a fully functional display if you carefully install the sensor in the plastic body.

The model pictured here produce 38 pulses / liter.

Some calibration is needed to discover the K factor. The easiest way is to consume water during some days, and simply divide pulses accumulated in the COUNT register by the volume shown on the mechanical counter.

Some water meters already have a dedicated hole to receive a reed switch that provides an open/close contact transition ten times per liter or so. This is even easier to interface them with BAE chips.

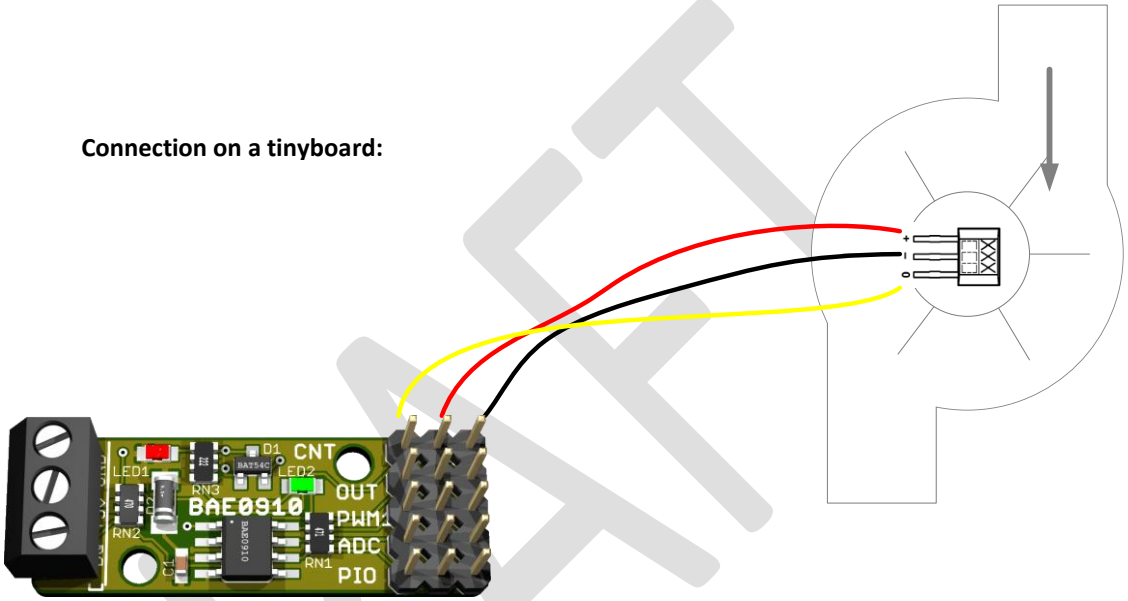
WIRING TO BAE BOARDS

As seen before, pulse sensors have three pin connections: a ground, a 5v supply and the pulse signal.
The pulse output is an open drain signal that requires an external pullup resistor.
This means that the pulses are inverted:

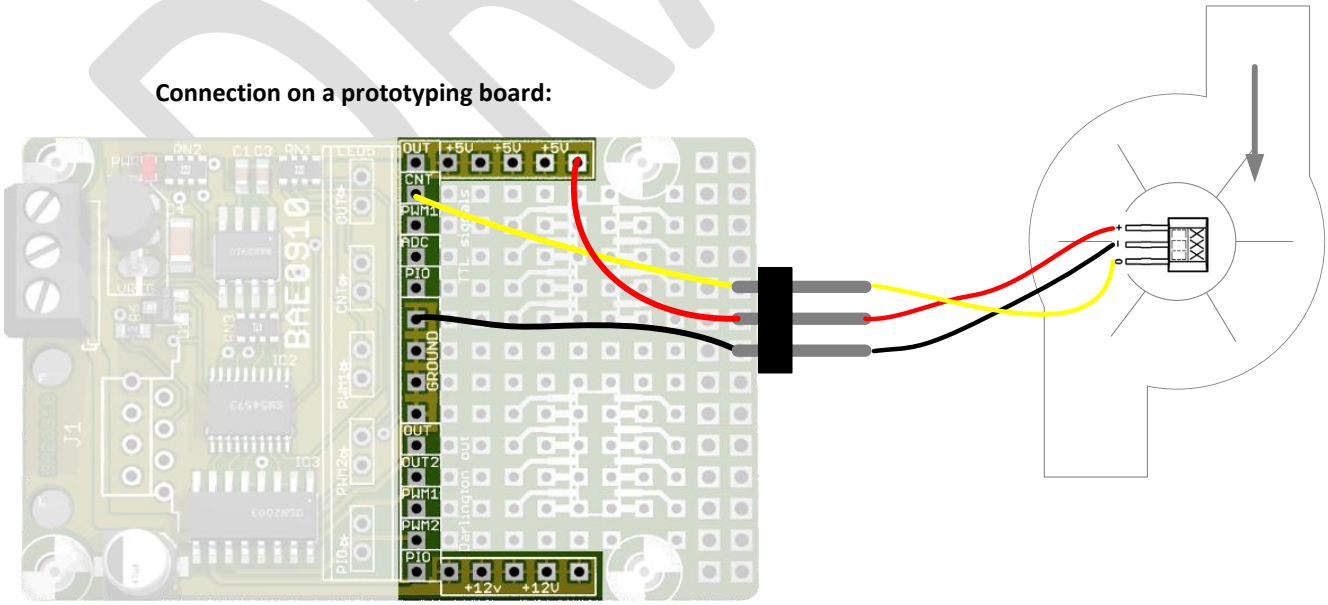
- when idle, the sensor leave the pin in high impedance. The external pullup resistor bring the level to 5v.
- when a pulse occurs, the signal is pulled down to the ground by the sensor.

On bae boards, CNT pin already contains the pullup resistor and does not require additional component for interfacing a flow sensor. This is specifically designed to measure such signals. By default CNTC register is configured to count on falling edge signal transitions which is exactly the case here.

Connection on a tinyboard:



Connection on a prototyping board:



On the BAE boards, connect the sensor to the CNT pin and the supply (GND and VCC) respecting sensor pin out.

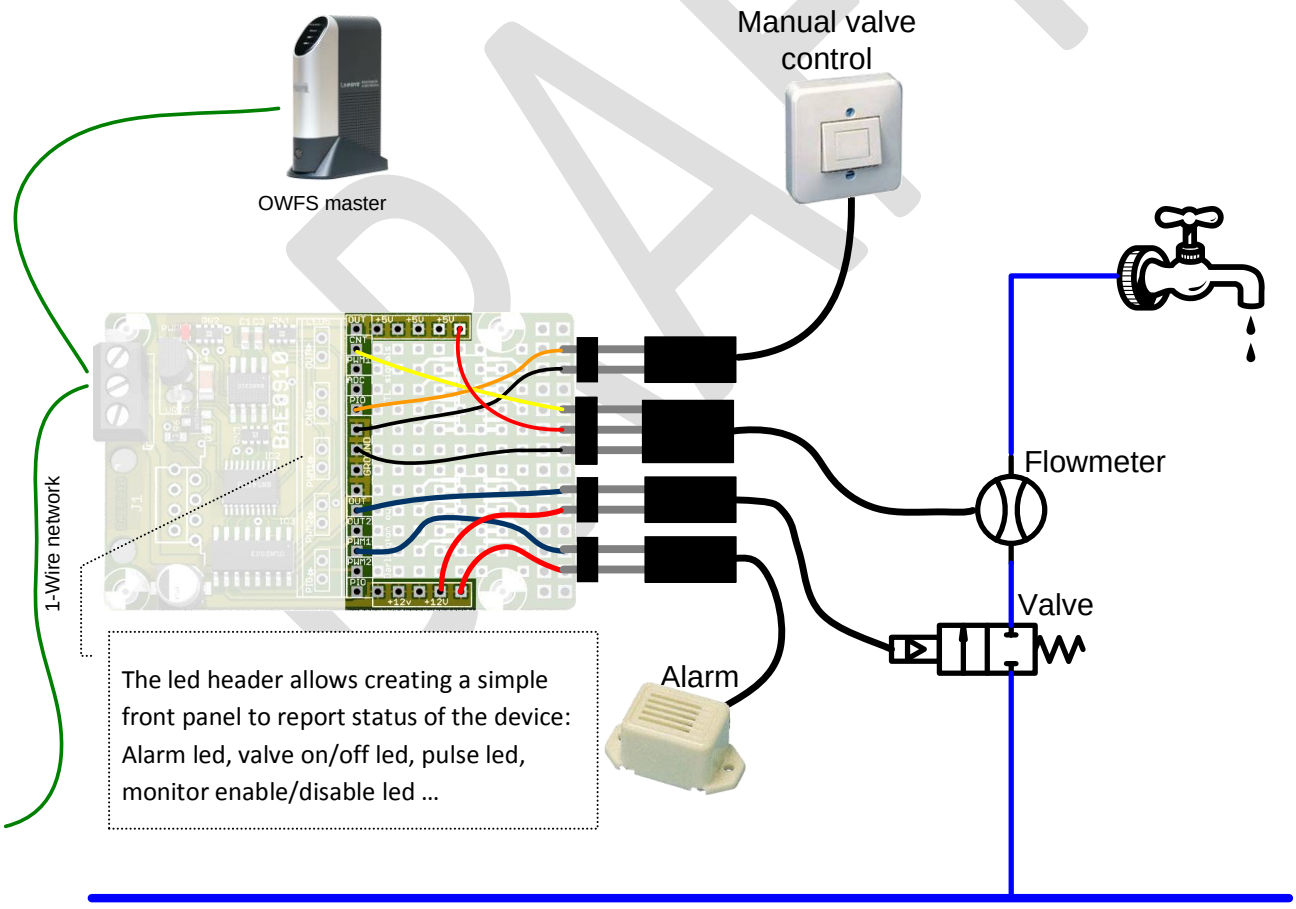
ADVANCED USE

A COUNT input could be used for many applications beyond the simple data logger.
With the Automation Engine of the BAE0910, it is possible to embed reactions based on the inputs avoiding latency due to master polling. This allows designing real-time and master independent logic to conditionally control valves, alarms, status led, etc.
Examples:

Consumption / quota management by controlling a valve based on volume total or volume delta,
Volume aware automated lawn watering system,
Leakage detection system with autonomous logic to control a shutoff valve,
Closed loop flow regulation with proportional valve,
Monitoring for beverage volume
Controlling regeneration cycle for a water softener unit based on actual volume consumption.

EXAMPLE: LEAKAGE DETECT, ALERT AND SHUTOFF SYSTEM

This example shows an autonomous leakage detection system that includes flow meter, valve, alarm and pushbutton.
The pushbutton allows manual control on the valve and alarm condition clearing.
The 1-Wire connection to an OWFS system allows centralized and remote control of the system.



leakage detection example with BAE0910 – www.brain4home.eu

When grouping detection, actuation and logic in the same board, it becomes possible to setup a leak detector with automatic shutoff valve. Instead of implementing the logic at the master side, the logic is can be embedded within the eeprom of the BAE chip. This allows independent monitoring while still allowing master supervision and data logging.

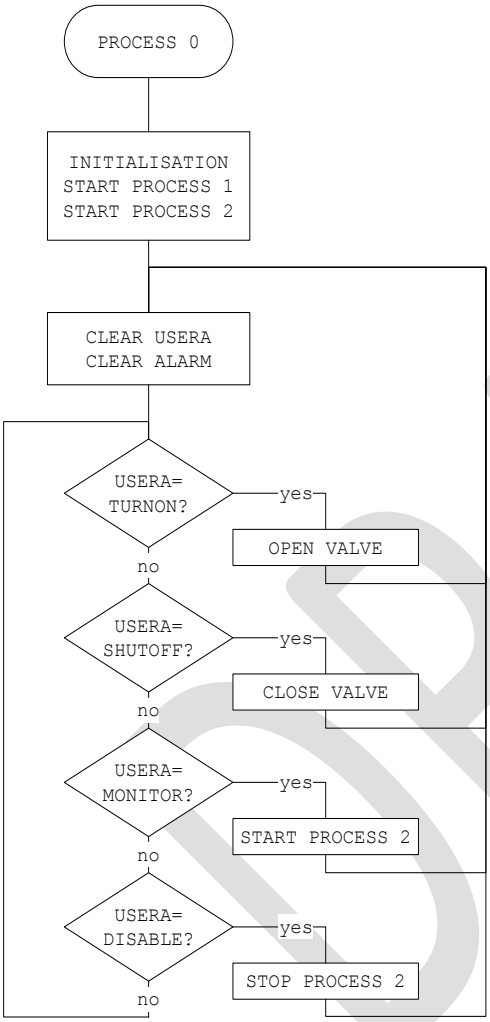
AE CODE LEAKAGE DETECTION EXAMPLE

This AE code monitors water flow to detect leak condition. The leak condition is based on a configurable maximum duration of uninterrupted consumption. The maximum acceptable duration is specified in minutes in the range of 1 to 255 minutes. A minimum flow threshold is also configurable.

This example uses three distinct processes: the worker (process 0), the user interface (process 1), the monitor (process 2).

PROCESS 0: THE WORKER

It initializes I/O and loop on USERA register to check if a command is to execute. It then executes the commands requested by owfs or other processes.

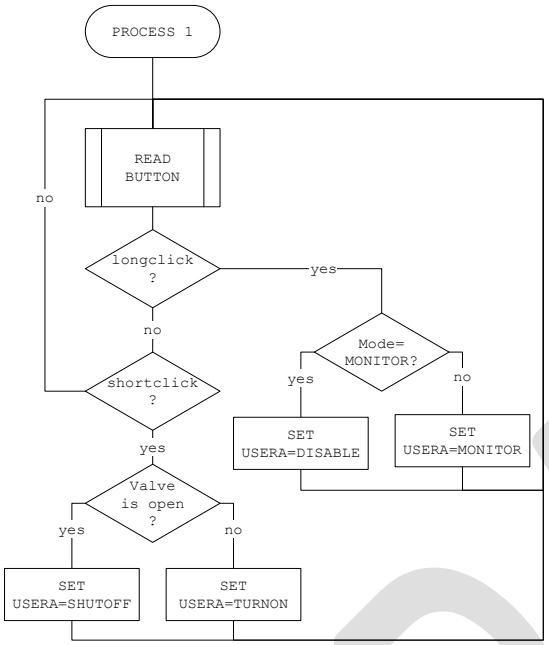



```
// AutomationEngine assembly source for BAE0910. May 2010
// This code monitors water flow to detect and react on leak condition.
// VARIABLES:
// B_USERA command to execute: TURNON,SHUTOFF,MONITOR,DISABLE
// B_USERB retain current state, DISABLE/MONITOR
//          read userb from owfs to know current mode
// B_USERC countdown, remaining minutes before alarm
// B_ALARM alarm flag, a value of 128 means leak condition
// this automatically showup the device in alarmed owfs folder

// PERIPHERALS:
// flow sensor connected on CNT
// shutoff valve connected OUT
// alarm connected on PWM1
// pushbutton connected on PIO

#define ALLOWED_DURATION 30 //after 30 minutes, trigger an alarm
                            //and shutoff
#define MIN_THRESHOLD    9  //minimum volume per minute to
                            //consider real consumption
                            //using a flowmeter with 855 pulses/liter
                            //9 units= 10ml

#define TURNON   1
#define SHUTOFF  2
#define MONITOR  3
#define DISABLE  4

#include "bae0910.inc"
#eeprom          0,0      // start_page, end_page
#org             $00      // Code generated at address specified.

begin:
        NOP      // to allows auto start of process #0 on poweron,
                 // first byte of page0 should be a NOP
init:
        SET.B    B_CNTC, 0          // enable CNT pin as counter
                                    // on falling edge
        SET.B    B_OUTC, 16         // OUT pin is enabled
        SET.B    B_PIOC,18          // enable pio as input with pullup
        SET.B    B_TPM1C,4          // Pre Scaler (1us resolution)
        WAIT     1
        SET.W    W_PERIOD1,500      //2000 Hertz
        WAIT     1
        CLR.L    L_COUNT            // set counter to zero
        SET.B    B_OUT,1            //turn on valve
        SET.B    B_ALARMC,$80       // only user2 flag is meaningfull
        START    1,check_button
        START    2,leak_detect

main:  //the main loop wait for a command in usera
        CLR.B    B_USERA //clear last command
        CLR.W    W_DUTY1  //clear alarm buzzer
        CLR.B    B_ALARM  //clear alarmed flag to not reply to conditional
                          //search from owfs
wait_command:
        WAIT     1
        CMP.B    B_USERA,TURNON
        BEQ      turnon
        CMP.B    B_USERA,SHUTOFF
        BEQ      shutoff
        CMP.B    B_USERA,MONITOR
        BEQ      monitor
        CMP.B    B_USERA,DISABLE
        BEQ      disable
        BRA      wait_command
turnon:
        SET.B    B_OUT,1  //turn on
        BRA      main
shutoff:
        CLR.B    B_OUT   //shutoff the valve
        BRA      main
monitor:
        START    2,leak_detect    //start leak detect process
        BRA      main
disable:
        SUSPEND  2        //leak detect process is disabled
        SET.B    B_USERB,DISABLE //update state
        BRA      main
```

PROCESS 1: THE USER INTERFACE

This process handles the button behavior; It toggles valve state and clear alarm on key press.
It also toggle monitor enable/disable mode on long keypress.

The real processing of the action is not handled by this process, it only 'post' a command to the worker process via the usera register.

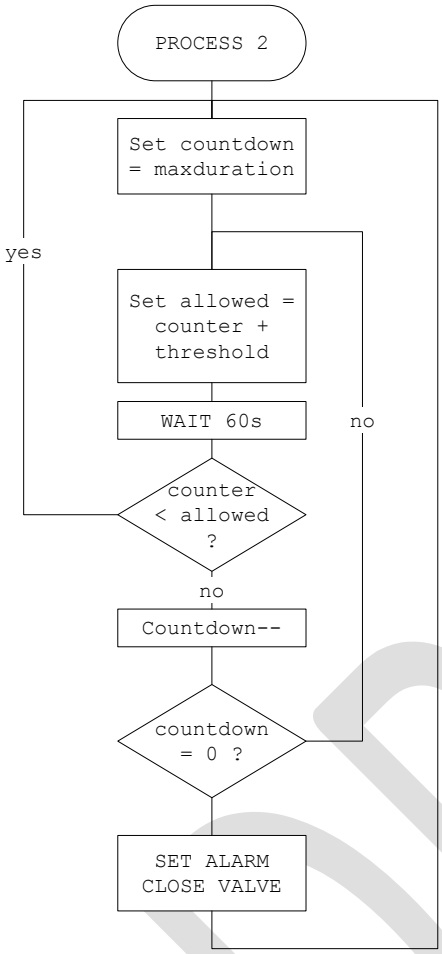



```
// PROCESS 1 for handling button behaviour
check_button: // process that check the local button,
              //only one button to simply invert state on each click.
        AIS     1 // reserve one byte for readbutton result
loop_button:
        SET.B   @-1,32   //exit read button after
                         //two seconds if no keypress
        CALL    readbutton
        CMP.B   @-1,16   // was pressed more than one second ?
        BGR     longclick
        CMP.B   @-1,0    // was pressed for more than zero second?
        BGR     shortclick
        BRA     loop_button

longclick:  //toggle mode monitor/disable
        CALL    beep1  // play two beep on long button press
        CALL    beep1
        CMP.B   B_USERB,DISABLE
        BEQ     ask_monitor
        CMP.B   B_USERB,MONITOR
        BEQ     ask_disable
        BRA     loop_button
shortclick: //toggle valve
        CALL    beep1 // play one beep on short button press
        CMP.B   B_OUT,1
        BEQ     ask_close
        CMP.B   B_USERB,MONITOR
        BEQ     ask_open
        BRA     loop_button

ask_monitor:
        SET.B   B_USERA,MONITOR
        BRA     loop_button
ask_disable:
        SET.B   B_USERA,DISABLE
        BRA     loop_button
ask_close:
        SET.B   B_USERA,SHUTOFF
        BRA     loop_button
ask_open:
        SET.B   B_USERA,TURNON
        BRA     loop_button
```

PROCESS 2: THE MONITOR

This process monitors the counter for leakage detection.

When a leakage condition is met this process signals the alarm state, turn on the buzzer and ask to shutoff the valve.




```
// PROCESS 2 for leak detect
leak_detect:
        SET.B   B_USERB,MONITOR  //update runningstate
reset_countdown:
        SET.B   B_USERC,ALLOWED_DURATION //count down=allowed duration
loop_detect:
        PUSH.L  L_COUNT  // store current counter
        PUSH.L  MIN_THRESHOLD
        ADD.L   // stack contains value to check against counter
        WAIT    240 //wait 15 seconds (240/16)
        WAIT    240
        WAIT    240
        WAIT    240 // four time 15s to wait one minute
        PUSH.L  L_COUNT
        SUB.L   // calculate allowed - counter
        AIS     -4  // remove result from stack
        BGR     reset_countdown  //if not exceeded threshold,
                                 //then no significant flow
flowing:                 // flow is present
        DEC.B   B_USERC  // decrement countdown
        BEQ     reached_maxduration //countdown reached zero
        BRA     loop_detect
reached_maxduration:
        SET.B   B_USERA,SHUTOFF  // shutoff the valve
        SET.B   W_DUTY1,250      // turn on buzzer
        SET.B   B_ALARM,$80      // set alarmed flag to allows
                                 // owfs showing alarmed device
        BRA     leak_detect
```

The compiled program is only 246 bytes long on a total of 1024 bytes available.

While fully functional, this example leave room for improvements likes handling a warning level before shutoff, restricting volume consumption per time window (quota), adding automatic alarm reset after a defined period of time, or many other customizations.

The full source code and compiled code of this example is directly downloadable on www.brain4home.eu

INSTALLATION STEPS ON YOUR OWFS BASED SYSTEM

Download the compiled file "water-leak-detect.bin" from download area:
`wget http://www.brain4home.eu/attachments/water-leak-detect.bin`

erase the eeprom of your destination device:
`echo 1 >/path/to/your/owfs/FC.000000000nnn/eeprom/erase.0`

install AE program into the eeprom:
`cp water-leak-detect.bin /path/to/your/owfs/FC.000000000nnn/eeprom/page.0`

power cycle the bae board or start manually the AE code with:
`echo 1 >/path/to/your/owfs/FC.000000000nnn/910/pc0`

At this point, the bae device will toggle valve on every press on the pushbutton connected on PIO connector.
A long button press will toggle monitoring on/off.

UTILISATION FROM YOUR OWFS BASED SYSTEM

To switch on the valve from your Linux:
`echo 1 >/ path/to/your/owfs/FC.000000000nnn/910/usera`
To switch off the valve:
`echo 2 >/ path/to/your/owfs/FC.000000000nnn/910/usera`
To turn on the monitoring:
`echo 3 >/ path/to/your/owfs/FC.000000000nnn/910/usera`
To turn off the monitoring:
`echo 4 >/ path/to/your/owfs/FC.000000000nnn/910/usera`
To know the current mode:
`cat /path/to/your/owfs/FC.000000000nnn/910/userb`
3 : monitoring is enabled,
4 : monitoring is disabled

To know if the device is alarmed:
`cat /path/to/your/owfs/FC.000000000nnn/910/alarm`
0 : the device is not alarmed,
128 : the device is alarmed due to leak detection

To reset alarm condition, issue a command to usera. For example re-open the valve:
`echo 1 >/ path/to/your/owfs/FC.000000000nnn/910/usera`

To know the accumulated water volume:
`cat /path/to/your/owfs/FC.000000000nnn/910/count`
(to divide by the pulse/liter ratio that depends of flow sensor characteristics.)

To discover alarmed devices:
`ls /path/to/your/owfs/alarm`

CONCLUSION

The purpose of this application note and accompanying example is multiple:

1. Informative: I try to give basic information on how to use a particular feature.
2. Educational: the example show how to translate logic to AE code
3. Functional: by providing a concrete example, I also try to throw a light on the benefits to embed behavior directly within the slave. 1-Wire system becomes more than a datalogger without reliability issues.
4. Motivating: by giving the basics, you get the keys to incorporate your own imagination into a functional device.

I sincerely hope that I've met at least one of these purposes for anyone.

SUPPORT

Online support is available via the forum on www.brain4home.eu and via the discussion list.
To subscribe, list-subscribe@brain4home.eu

AVAILABILITY

Chips and boards can be ordered online on www.brain4home.eu

CONDITION OF USE

The BAE chips are intended for hobbyist usage and are not approved for use where it constitute or may constitute a danger to human life or health.

TERMS OF LICENSE

The software embedded in the chips is protected by copyright laws. Customer is not allowed to reverse engineer, decompile, or disassemble the embedded firmware.

ABOUT THE AUTHOR

I'm Pascal Baerten, an IT consultant with technical background in automation. I followed A2 technical studies until 1985 where I played with CNC machines and pneumatic automates. Graduated in Computer Sciences from the Robert Shuman High school in Belgium in 1989, My thesis was titled “A terminal emulator” where I mastered serial communication and networking programming.

My first computer was a Sinclair ZX81, where I learned the basics of exploiting very constrained computing resources in assembler. Later, a Commodore 64 opened the way to interfacing computers with electronic toys.

Since 1990 I developed network based resource sharing solutions in assembler and C.: Telex server, Fax server, Minitel server, mainframe front end, mail server, print server, text2speech telephone server, database gateway, IM server …

As skilled networking/server architect, I'm working as IT consultant for large financial companies since 1997.

In parallel, developments in home automation have contributed to accumulate some experience with microcontrollers and embedded computing.

REVISION HISTORY

Revision #	Date	Description
0.1	Feb 8, 2010	Initial draft
0.2	Mar 29, 2010	Update for prototyping board
0.3	May 15, 2010	Added leakage detect example