IAR Application Note 6811-002

Using DALLAS 1-Wireä interface with the Motorola 68HC11

by Stefan Nyman
[mailto:Stefan.Nyman@micronym.se]

SUMMARY

This application note describes how to interface Dallas MicroLAN devices to a Motorola 68HC11.

KEYWORDS

Dallas MicroLAN, interface

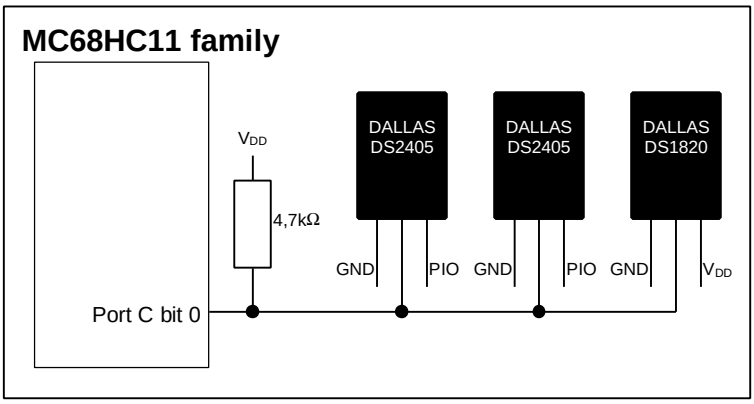
PROBLEM

The 1-Wire™ interface, also called MicroLAN™, is a method for communicating with several components of different types via a single data line. Every component has its own identity (a 64-bit number) and is powered via the single line.

The application note describes how to interface several 1-Wire devices to a Motorola 68HC11. It shows an implementation of the search command that can be used with other microcontrollers.

The program library for the DS2405 includes Read ROM, Match ROM, Search ROM and Active-Only Search ROM.

SOLUTION



The hardware is very simple (as shown above). For further details how to use the PIO pin on the DS2405 and the V_{DD} pin on the DS1820, see the Data Sheets.

The software is divided into **3 levels**.

The **Hardware level** routines are hardware specific. They are:
- Reset Pulse
- Write 0
- Write 1
- Read Data

In addition there are two delay routines to ensure correct timing.

The **Medium level** routines are:
- 3 routines used by the search command

The **Application level** routines consists of the high-end commands:
- Read Byte
- Write Byte — used for sending a command.
- Read ROM — gets the identity of one component.
- Match ROM — to address one specific switch.
- Search ROM — made to perform the Active-only search command as well.

Hardware Level

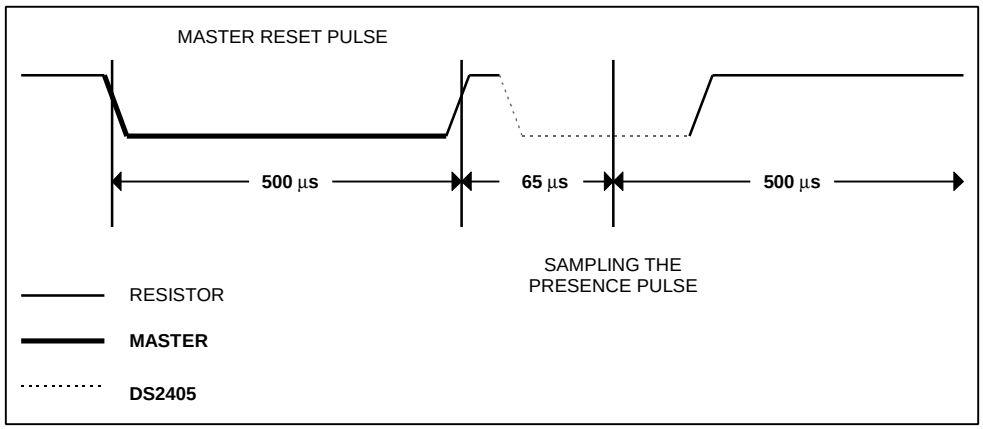
This level contains the shell of routines which are hardware dependant. You have to change these if you do not have an 8 MHz crystal or if the wire is connected to a different pin. In these routines, the wire is connected to bit 0 of Port C.

The following definitions are used to make the routines more readable:

```
ML_port equ $03
ML_ddr  equ $07
ML_pin  equ %00000001
IOBASE  equ $1000
```

Master Reset Time Slot

The bus master transmits a reset pulse (minimum 480 µs). The bus master then releases the line and goes to receive mode. The 1-Wire bus is pulled to a high state via the 4,7 kΩ pull-up resistor. After detecting the rising edge on the data pin, the device waits 15 - 60 µs and then transmits the presence pulse (60 - 240 µs). A reset pulse shall always be sent before any ROM command.



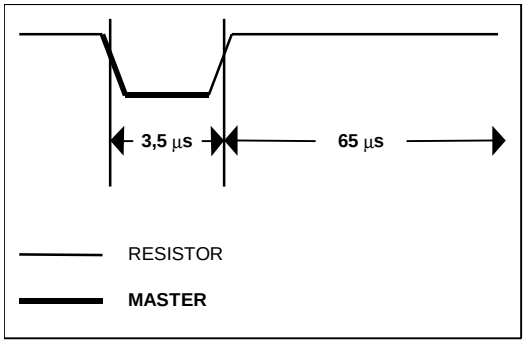

```
************************************
* Subroutine ResetML
* Sends reset pulse to all devices.
* Returns C=1 if anyone present.
```

```
*_____________________________________
ResetML  psha
         pshx
         ldx   #IOBASE
    sei
         bclr  ML_port,X,ML_pin    ML_pin=0
         bset  ML_ddr,X,ML_pin     Make it output
         jsr   Del_500             Wait for 500 µs
         bclr  ML_ddr,X,ML_pin     Release the low level
         jsr   Del_65              Wait for presence pulse
         ldaa  ML_port,X
    cli
         anda  #ML_pin
         cmpa  #1                  Makes C=1 if presence
         jsr   Del_500             Wait for 500 µs
         pulx
         pula
         rts
```

Write Data

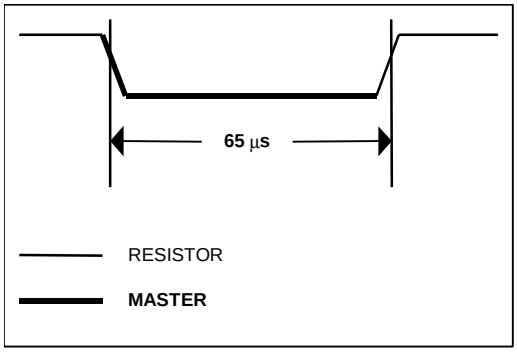
The falling edge of the data line synchronizes the 1-Wire device to the master by triggering an internal delay circuit. The delay determines when the device will sample the data line. This time is 15 - 60 µs.

Write-One Time Slot




```
*********************************
* Wr_One writes a one to the 1-wire bus
*_____________________________________
Wr_One   pshx
         ldx   #IOBASE
    sei
         bclr  ML_port,X,ML_pin
         bset  ML_ddr,X,ML_pin      3,5 µs
         bclr  ML_ddr,X,ML_pin
    cli
         jsr   Del_65
         pulx
         rts
```

Write-Zero Time Slot

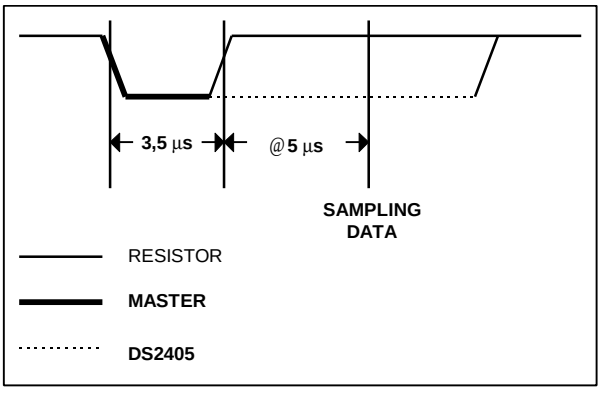



```
*********************************
* Wr_Zero writes a zero to the 1-wire bus
*_______________________________
Wr_Zero  pshx
         ldx   #IOBASE
    sei
         bclr  ML_port,X,ML_pin
         bset  ML_ddr,X,ML_pin
         jsr   Del_65            65 µs
         bclr  ML_ddr,X,ML_pin
    cli
         pulx
         rts
```

Read Data

The falling edge of the data line synchronizes the device to the master by triggering an internal delay circuit. If a '0' is to be transmitted, the delay circuit determines how long the device will hold the data line low, overriding the '1' generated by the pullup resistor. If a '1' is to be transmitted, the device will leave the read data time slot unchanged.

Read Data Time Slot



```
************************************
* Subroutine Rd_data
* Returns the data in Carry
*_____________________________________
Rd_Data  pshx
         ldx   #IOBASE
    sei
         bclr  ML_port,X,ML_pin
         bset  ML_ddr,X,ML_pin        0 for 3,5 µs
         bclr  ML_ddr,X,ML_pin
         bclr  ML_ddr,X,ML_pin        Wait some µs
         brclr ML_port,X,ML_pin,low   Poll wire
         sec                          and update carry
         bra   *+3
low      clc
    cli
         jsr   Del_65
         pulx
         rts

* Delay for 500µs (8 MHz crystal)
Del_500  psha                 3 c
         ldaa  #196           2 c
         deca                 2 c   2,5 µs x 196 = 490 µs
         bne   *-1            3 c
         pula                 4 c
         rts                  5 c

* Delay 65µs (8 MHz crystal)
Del_65   psha                 3 c
         ldaa  #22            2 c
         deca                 2 c   2,5 µs x 22 = 55 µs
         bne   *-1            3 c
         pula                 4 c
         rts                  5 c
```

Medium Level

This level contains routines used by the high-end commands. The application engineer does not have to know about these.

```
*******************************
* Set_Bit
* Sets a bit in a 64-bit array
* pointed out by X
* Bit number 1-64 in acc B
*_____________________________
Set_Bit  psha
         pshb
         pshx
         decb
         tba
         lsrb
         lsrb
         lsrb
         comb
         andb  #7
         abx          X points to actual byte
         ldab  #1
         anda  #7
settst   beq   setrdy
```

```
         lslb
         deca
         bra   settst
setrdy   orab  0,X
         stab  0,X
         pulx
         pulb
         pula
         rts

*******************************
* Clr_Bit
* Clears a bit in a 64-bit array
* pointed out by X
* Bit number 1-64 in acc B
*_____________________________
Clr_Bit  psha
         pshb
         pshx
         decb
         tba
         lsrb
         lsrb
         lsrb
         comb
         andb  #7
         abx          X points to actual byte
         ldab  #1
         anda  #7
clrtst   beq   clrrdy
         lslb
         deca
         bra   clrtst
clrrdy   comb
         andb  0,X
         stab  0,X
         pulx
         pulb
         pula
         rts

*******************************
* Tst_Bit
* Tests a bit in the 64-bit array
* pointed out by X
* Bit number 1-64 in acc B
* Returns zero or nonzero
*_____________________________
Tst_Bit  psha
         pshb
         pshx
         decb
         tba
         lsrb
         lsrb
         lsrb
         comb
         andb  #7
         abx          X points to actual byte
         ldab  #1
         anda  #7
```

```
tsttst   beq   tstrdy
         lslb
         deca
         bra   tsttst
tstrdy   andb  0,X
         pulx
         pulb
         pula
         rts
```

Application Level

This level consists of the highest layer of routines concerning the 1-wire protocol. The application engineer has to know how to use them.

Read Byte

Calls the Rd_data 8 times and shifts the data into the accumulator A. Used by the Read ROM command and when reading the scratchpad in the DS1820.

```
***********************************
* Subroutine Rd_Byte
* Returns one byte from the 1-Wire bus
* in acc A
*_________________________________________
Rd_Byte  pshb
         ldab  #8
rd_bnext jsr   Rd_Data
         rora
         decb
         bne   rd_bnext
         pulb
         rts
```

Write Byte

Calls the Wr_One or Wr_Zero according to the content of accumulator A. Can be used to send commands or write to memory in the DS1820.

```
*********************************
* Wr_Byte sends a byte (accA)
*_______________________________________
Wr_Byte  psha
         pshb
         pshx
         ldx   #IOBASE
         ldab  #8
sh_b     lsra
         bcc   wr_0
         jsr   Wr_One
         bra   next_b
wr_0     jsr   Wr_Zero
next_b   decb
         bne   sh_b
         pulx
         pulb
         pula
         rts
```

Read ROM

Read ROM allows the bus master to read one device's 8-bit family code, 48-bit serial number and 8-bit CRC. If more than one device is present, they will produce a wired AND result.

```
************************************
* Subroutine Read_ROM
* Executes the Read ROM command
* Gets the identity of one device.
* ID is stored in 8-byte area [CRC,Serial[6],Family]
*_____________________________________
Read_ROM psha
         pshx
         ldx   #Family
         ldaa  #R_ROM
         jsr   Wr_Byte
rd_ROM   jsr   Rd_Byte
         staa  0,X
         dex
         cpx   #Family-8
         bne   rd_ROM
         pulx
         pula
         rts
```

Match ROM

Match ROM followed by a 64-bit ROM sequence, allows the bus master to address a specific device. Only the one that exactly matches the 64-bit sequence will respond to the following memory function command. All others will wait for a reset pulse.

```
***********************************
* Match
* Addresses one specific device
* Toggles the PIO (DS2405).
* At entry: X points to the
* data area for this device
*_____________________________________
Match    psha
         pshb
         pshx
         ldaa  #M_ROM
         jsr   Wr_Byte
         clrb
         incb
match    jsr   Tst_Bit
         beq   *+7
         jsr   Wr_One
         bra   *+5
         jsr   Wr_Zero
         incb
         cmpb  #64
         bls   match
         pulx
         pulb
         pula
         rts
```

Search ROM

When a system is initially brought up, the HC11 must know the 64-bit codes of all devices. This is done by a 3-step routine on each bit of the ID-code.

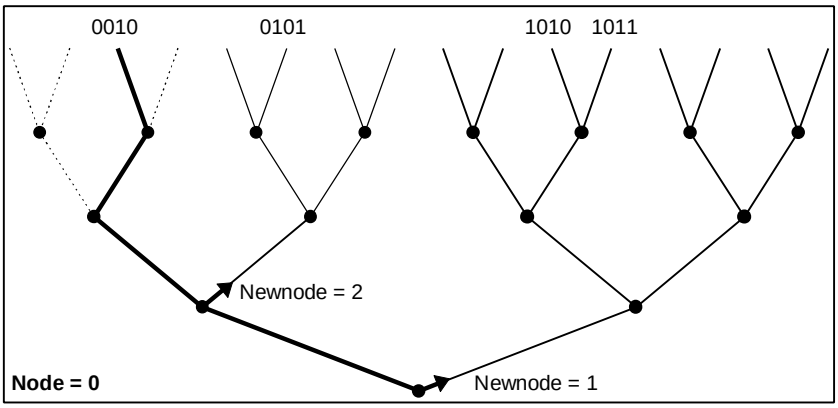
```
* Search ROM variables:
* Last branch level where the zero-branch was chosen
Node     rmb 1
Nextnode  rmb 1

**************************************
* Search ROM
* Search for identities and status
* for all devices
* At entry: A holds the command Search ROM ($F0)
* or command Active-Only Search ROM ($EC)
* X points to an area
* where identities can be stored:
* Dev #1: CRC, Serial, Family, Status (9 bytes)
* Dev #2: CRC, Serial, Family, Status   etc.
* B holds the Tree position
* Node is the last branch level where
* the zero-branch was chosen
* Returns Carry set if any device is found
*_____________________________________
Search   pshb
         pshx
         psha

         clr   Nextnode   Initial value = 0

N_branch ldaa  Nextnode
         staa  Node       Update Node value
         clr   Nextnode
         jsr   Reset
         bcc   none
         pula
         psha             Update command
         jsr   Wr_Byte
         clrb
         incb             Reset bit pointer

RDnext   cmpb  Node       If Node points higher than
         bhs   Normal     current position

L_trace  jsr   Rd_Data    then follow last trace
         bcs   noturn
         jsr   Rd_Data
         bcs   tstbit
         jsr   Tst_Bit
         bne   tstbit
         stab  Nextnode   Here is a new branch - remember it
         bra   tstbit
noturn   jsr   Rd_Data
tstbit   jsr   Tst_Bit    Choose the correct way
         beq   *+7
         jsr   Wr_One
         bra   *+5
         jsr   Wr_Zero
         bra   NextPos    --------->
```

```
Normal   jsr   Rd_Data    else do normal search
         bcc   two?
* Only 1-branch?
         jsr   Rd_Data
         bcc   one
* No active switch:
         clc
none     bra   None


*_____________________________
* [10] Only 1-branch
one      jsr   Set_Bit
         jsr   Wr_One
         bra   NextPos    --------->

* Perhaps two branches:
two?     jsr   Rd_Data
         bcc   two

*_____________________________
* [01] Only 0-branch:
         bra   Zerobr
*_____________________________
* [00] Two branches:
two      cmpb  Node       Been here before?
         beq   Onebr      Yes, choose a 1

         stab  Nextnode   No, save this index

Zerobr   jsr   Clr_Bit    and select 0-branch
         jsr   Wr_Zero    Write a zero to choose this branch
         bra   NextPos    --------->

* Select 1-branch:
Onebr    jsr   Set_Bit    Set bit in ID
         jsr   Wr_One     Write a one to choose this branch
NextPos  incb
         cmpb  #64        All the way done?
         bls   RDnext     No
         jsr   Rd_Data    Yes, check state (DS2405)
         clra
         rola
         staa  8,X

         tst   Nextnode
         sec
         beq   None       No more branches detected
         ldab  #9         Yes, there are more
         ldaa  0,X
         staa  9,X        Copy last ID
         inx
         decb
         bne   *-6
         jmp   N_branch   Do the next branch

None     pula
         pulx
         pulb
         rts
```

How the search is done

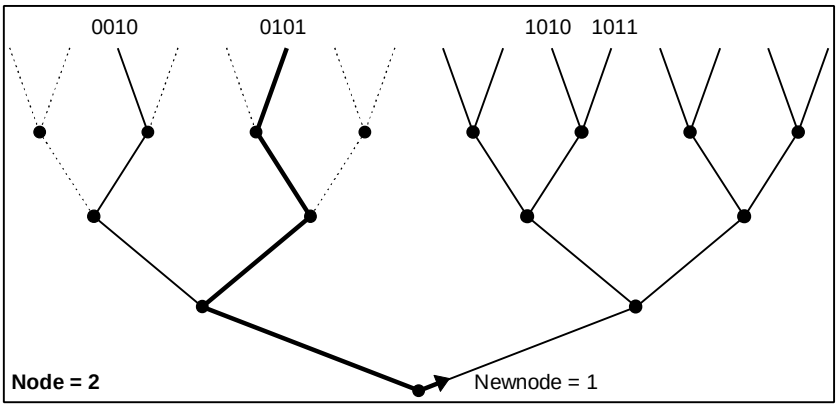
When the search starts, the routine selects the leftmost branch, i.e. when a choice can be done, the zero-branch is selected. It remembers the branch level where to turn right at the next time.

To illustrate the mechanism, this example shows how to extract the 4-bit code of 4 devices.

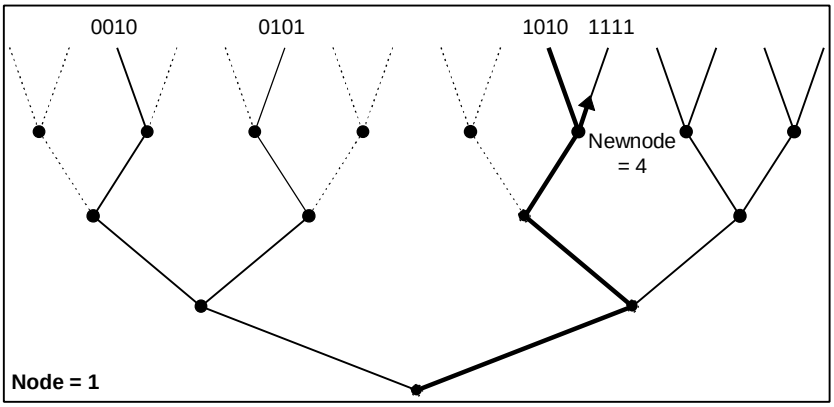


When the routine checks the first node, it notices two active branches. It selects the left (zero) and remembers Newnode=1. This indicates that it must later turn right at first level. At the second level there is another two-way node and the Newnode value is updated, Newnode=2. The old value is forgotten but is captured next time.

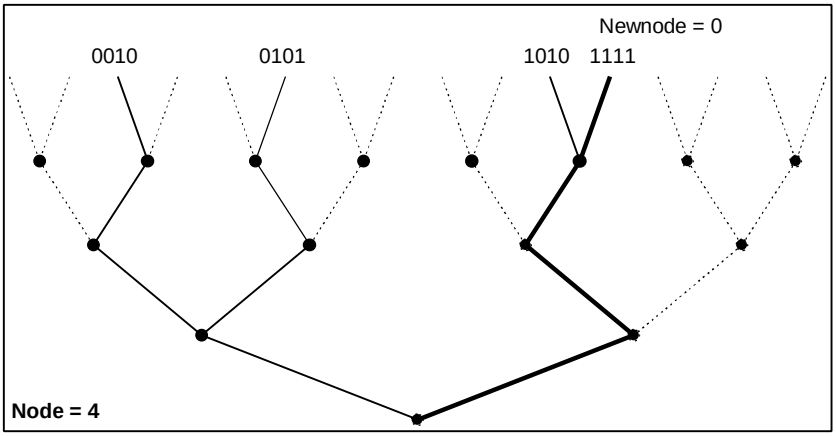
When the routine reaches the end, it has the code, 0010, and can start at the bottom again.



Node is now updated with the value of Newnode.
When the routine restarts, it notices that there are two active branches at the first level. Newnode is again set to 1. It follows the last trace until the current level equals the value of Node. Then it turns right and continues the check.



Node is again updated with the value of Newnode.
Since Node equals to 1, the routine immediately starts to the right and then holds to the left. At the fourth node it can see two active branches. Newnode is set to 4.



This last time it follows the latest track until the fourth level is reached and no more two-way branches are found. Newnode remains =0 and the search terminates. All device codes are now found.

CONTACT INFORMATION

USA
IAR Systems Inc.
One Maritime Plaza
San Francisco,
CA 94111
Tel: +1 415-765-5500
Fax: +1 415-765-5503
Email: info@iar.com

SWEDEN
IAR Systems AB
P.O. Box 23051
S-750 23 Uppsala
Tel: +46 18 16 78 00
Fax: +46 18 16 78 38
Email: info@iar.se

GERMANY
IAR Systems AG
Posthalterring 5
D-855 99 Parsdorf
Tel: +49 89 90 06 90 80
Fax: +49 89 90 06 90 81
Email: info@iar.de

UK
IAR Systems Ltd.
9 Spice Court,
Ivory Square
London SW11 3UE
Tel: +44 171 924 3334
Fax: +44 171 924 5341
Email: info@iarsys.co.uk

DENMARK
IAR Systems A/S
Elkjervej 30-32
DK-8230 Aabyhoj
Tel: +45 86 25 11 11
Fax: +45 86 25 11 91
Email: info@iar.dk

www.iar.com