Project „BlinkenBone“

BlinkenBus Specifications

Rev 0.4, 13.1.2013

Send all comments and errors to the author:

j_hoppe@t-online.de (Jörg Hoppe)

Change history:

Rev	Date	Changes	By
0.1	28.02.2012	initial version	JH
0.2	29.02.2012	changed register layout in address map	JH
0.3	13.04.2012	name changed to „BlinkenBus“	JH
0.4	13.1.2013	Added BBUCI interface	JH

Table of Contents

1 BlinkenBus concepts

BlinkenBus is the connection between the BeagleBone (which contains the BlinkenLight logic) and the I/O boards (called „BlinkenBoards“), which drive lamps and sense switches on the BlinkenLight consoles.

BlinkenBus is a parallel bus with 9 bit addresses and 8 bit data width, routed over flexible flat cable.

Connected is a BeagleBone as bus master and many slave I/O boards, each containing 16 registers, 8 bit width.

Slave I/O boards are called „BlinkenBoards“. They are equipped with

- individual addressable parallel I/O register chips (like 74x373, or 74x244),
- registers implemented by an onboard microController. („BBUCI“ - „BlinkenBoard MicroController Interface“)

Each boards has a 5 bit address selector switch, addressable are 32 boards. Every board implements also a special „board control register“.

With 9 bit address lines, total register address space is 512 registers.

Boards with different I/O characteristics can be mixed on the BlinkenBus. The current board type has 5 input registers and 11 output registers. So the maximum of outputs on a BlinkenBus is 11 * 8 * 32 = 2816 pins, max inputs are 5 * 8 * 32 = 1280 pins.

This is enough to drive the „mother of all blinkenlights“: the IBM 360/91 console.

2 BlinkenBus address map

(Valid for current BlinkenBoard, other board types may have other register maps)

Address (hex)	use	I/O/CTL	board space
1ff	Board control register	CONTROL	I/O board 31 register addr = register no. + 16 * board no.
1fe	BBUCI registers IN14 and OUT14	BBUCI Inputs, BBUCI Outputs	
...	...		
1fb	BBUCI registers IN11 and OUT11		
1fa	I/O register OUT10 and BBUCI IN10	BBUCI Inputs, I/O Outputs	
...	...		
1f5	I/O register OUT5 and BBUCI IN5		
1f4	I/O registers IN4 and OUT4	I/O Inputs (read only), I/O Outputs (write only)	
...	...		
1f0	I/O registers IN0 and OUT0		
1ef ... 010	board 30 ... board 1		
00f	Board control register	CONTROL	I/O board 0 register addr = register no. + 16 * board no.
00e	BBUCI registers IN14 and OUT14	BBUCI Inputs, BBUCI Outputs	
...	...		
00b	BBUCI registers IN11 and OUT11		
00a	I/O register OUT10 and BBUCI IN10	BBUCI Inputs, I/O Outputs	
...	...		
005	I/O register OUT5 and BBUCI IN5		
004	I/O registers IN4 and OUT4	I/O Inputs (read only), I/O Outputs (write only)	
...	...		
000	I/O registers IN0 and OUT0		

Input and output registers share the same register address in the range 0...14.

The board control registers uses address 15 on each board.

3 BlinkenBus Board Control Register

Every board has one 8 bit control register. It uses address 15 on each board. Bit assignment is:

bit	info	r/w
7:6	State of DIP switches #2 and #3, unused.	r
5	State of BBUCI timing DIP switch. This is DIP switch #1 on a standard BlinkenBoard.	r
4:1	board type ("0" for first "IN5/OUT11" model)	r
0	1 = "Disable OUTPUTS“, all outputs are tristate (default after reset = 1)	r/w

If a board is not connected: board control register must be read as 0xFF (all DATA lines high).

So special setting "switches = 111“ and "board type = „1111“ gives "disable" bit pattern.

4 BlinkenBus signals on 40 pin flat cable

I/O boards are connected over a flexible bus, so boards can be mounted where necessary.

Pin	Signal	BeagleBone		Pin	Signal	BeagleBone
1	ADDR7	P8.40 = GPIO2_13		23	DATA5	P8.22 = GPIO1_5
3	ADDR6	P8.39 = GPIO2_12		25	DATA4	P8.23 = GPIO1_4
5	ADDR5	P8.42 = GPIO2_11		27	DATA3	P8.6 = GPIO1_3
7	ADDR4	P8.41 = GPIO2_10		29	DATA2	P8.5 = GPIO1_2
9	ADDR3	P8.44 = GPIO2_9		31	DATA1	P8.24 = GPIO1_1
11	ADDR2	P8.43 = GPIO2_8		33	DATA0	P8.25 = GPIO1_0
13	ADDR1	P8.46 = GPIO2_7		35	STROBE*	P8.21 = GPIO1_30
15	ADDR0	P8.45 = GPIO2_6		37	ADDR8	P8.20 = GPIO1_31
17	R/W*	P8.26 = GPIO1_29		39	n.c.	
19	DATA7	P8.4 = GPIO1_7		4, 6, …, 36, 38	GND	
21	DATA6	P8.3 = GPIO1_6		2, 40	+3.3V	

19 wires are used for data . Signal voltage is 3.3V. Max load without repeater is 10??? boards. Max length is ??? m.

A bidirektional repeater could be inserted berween cable segments. This would be a uni/bidirektional driver, powered by +3.3V. Direction of driver is controlled by STROBE* and R/W*.

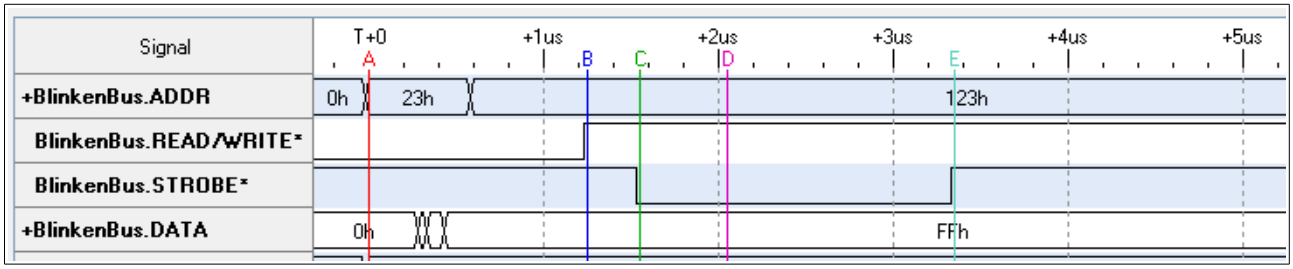
5 BlinkenBus Timing

All timing is generated by the BeagleBone. Signals are generated by software in the kernel mode driver „kblinkenbus“. Timing is not high speed optimized … slow speed makes the bus signals more reliable.

With approx. 4 µs per cycle, the whole 512 byte address space can be accessed in 2ms … this is fast enough.

5.1 Read cycle

Example: read address 0x123 (register IN3 on board 18), result data is 0xFF.



A: BeagleBone outputs address onto A0...A8

B: BeagleBone sets R/W* to READ.

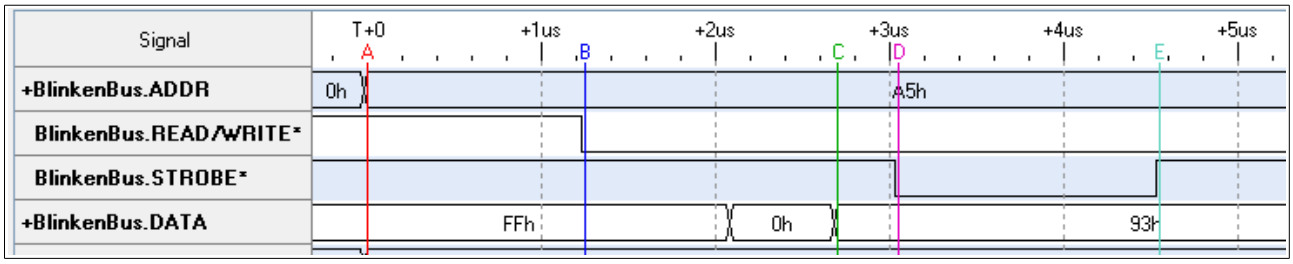
C: BeagleBone activates STROBE*. CPLD switches data pins of input register chips onto BlinkenBus DATA lines. (no change in signals visible here).

D: I/O register on BlinkenBoard drives valid data.

E: BeagleBone deactivate STROBE* after reading DATA. CPLD stops driving DATA lines.

5.2 Write cycle

Example: data 0x93 is written to address 0xA5 (register OUT5 on board 10).



A: BeagleBone outputs address onto A0...A8

B: BeagleBone sets R/W* to WRITE

C: BeagleBone drives DATA

D: BeagleBone activates STROBE*. CPLD switches output registers transparent, DATA appear on driver outputs.

E: BeagleBone deactivates STROBE*. Output registers latch and hold data.

- BeagleBone should stop driving DATA lines

6 Timing modifications for BBUCI register accesses

Several addresses (OUT10..14, IN6..14) are not implemented as physical I/O register chips, but by means of an µController (example: AVR Atmega). Since bus access is implemented with „Pin Change Interrupts“ and ISRs, the bus timing has to be slower than.

Slower BlinkenBus timing is to be used, if

1. A BlinkenBoard is setup to implement BBUCI, by setting the „BBUCI timing DIP switch“. Software reads Bit 5 of Board Control Register 15 as „1“ then.

AND

2. One of the BBUCI registers is accessed (Access to parallel I/O registers is always fast).

6.1 Local BBUCI bus signals

The CPLD on the BlinkenBoard generates these signals for the local BBUCI interface:

BBUCI_DATA: just a copy of BLINKENBUS.DATA

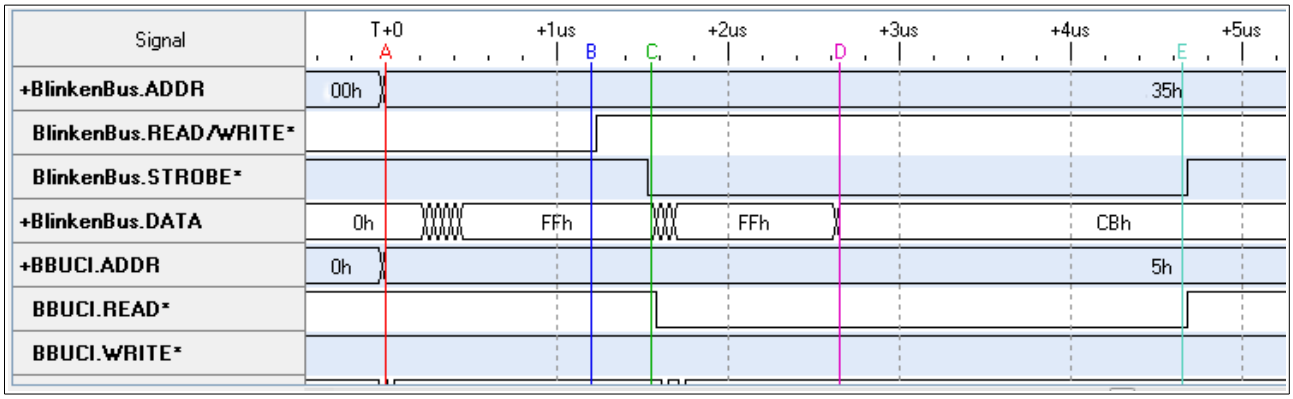
BBUCI_ADDR0..BBUCI_ADDR3: just a copy of BLINKENBUS.ADDR0..ADDR3

BBUCI_READ*: strobes a read access (when data is to be read from the µC). Goes LOW like STROBE*, if one of the BBUCI IN registers is accessed (IN6..IN14).

BBUCI_WRITE*: strobes a write access (when data is to written into the µC). Goes LOW like STROBE*, if one of the BBUCI OUT registers is accessed (OUT11..OUT14).

6.2 BBUCI Read cycle:

Example: address 0x35 is read. The µController ouptputs a data value of 0xC8.



A: BeagleBone outputs address onto A0...A8. On BBUCI_ADDR, the lower 3 bits ADDR2..ADDR0 appear.

B: BeagleBone sets R/W* to READ.

C: BeagleBone activates STROBE*. CPLD switches data pins of input register chips onto BlinkenBus DATA lines.

BBUCI_READ* signal is activated and interrupts the µController.

The BlinkenBus master must wait 2µs after STROBE* 1->0, before he reads the DATA back (transition „E“).

The µController must interrupt on BBUCI_READ* 1->0, and deliver data with 2 µs after begin of the Interrupot Service Routine (ISR).

D: the µController on BlinkenBoard drives valid data.

E: BeagleBone deactivate STROBE* after reading DATA.

READ* goes inactive at same time.

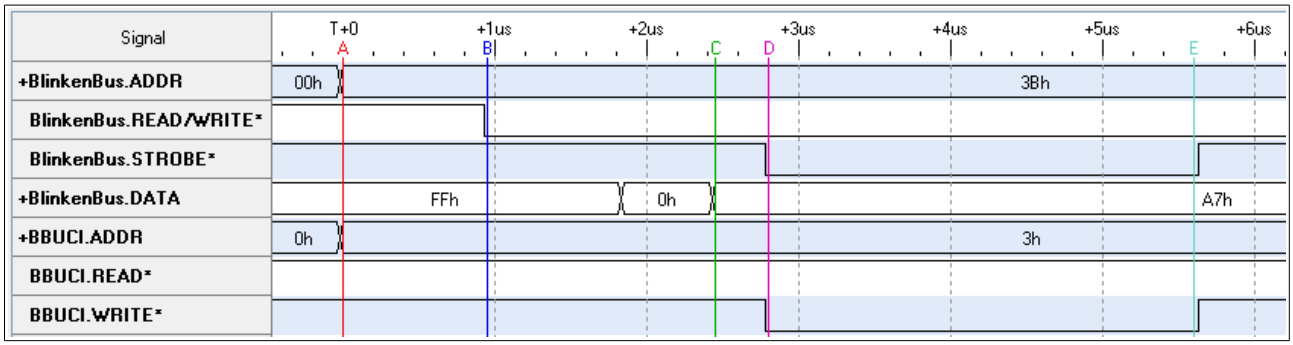
µController stops driving data at end of ISRDATA must go tristate again with 500ns after STROBE 0->1.CPLD

stops driving DATA lines.

The next BBUCI cycle (read or write) must not start before 500ns have passed after BBUCI_READ* goes high. The µController needs this time to terminate the ISR.

6.3 BBUCI Write cyle

Example: the Busmaster writes a data value of 0xA7 into address 0x3B.



A: BeagleBone outputs address onto A0...A8. On BBUCI_ADDR, the lower 3 bits ADDR2..ADDR0 appear.

B: BeagleBone sets R/W* to WRITE

C: BeagleBone drives DATA (0xA7)

D: BeagleBone activates STROBE*.

BBUCI_WRITE* goes active and interrupts the µController.

The µController samples DATA in the ISR, and waits for BBUCI_WRITE* go inactive again.

The BlinkenBus master must wait 2µs after STROBE* 1->0, before STROBE* goes high again. (transition „E“).

E: BeagleBone deactivates STROBE*. The µController stops sampling DATA.

The next BBUCI cycle (read or write) must not start before 500ns have passed after BBUCI_WRITE* goes high. The µController needs this time to terminate the ISR.