White paper

for the design of a

Modular Mikrocontroller Development System. (mMDS)

Version 0.1
29. December 2008

Author: Sebastian Dietz
sebastian_dietz@gmx.net

Content

Purpose

The designed system should allow a student to easily develop and implement a microcontroller based application while minimizing the hardware development effort for a prototype. It should offer a hardware and software framework for mMDS based applications.

This white paper collects all design considerations for the initial prototype.

Copyright

History

Version	Release Date	Reason	Superseded Version
0.1	29. December 2008	Initial Document	None
0.2	30. December 2008	Changed BUS layout	0.1
0.3	11. January 2009	Added PLD Design	0.2

Hardware design

The mMDS will be designed using a euro bus with 96 pin female connectors. The initial design uses a COTS board from the vendor Rademacher (www.rademacher.de) , order number: Art-Nr.: 940, Conrad electronic (www.conrad.de) order number: 521238. The bus connectors on the board can be obtained from Conrad with the order number: 742301.

The Euro-Bus system employed on this board uses 64 of the available 96 pins of the connectors. Row B and C of the connectors are shortened. (B2 = C2, B4 = C4, etc)

The Bus system

The bus system is common to all boards connected. The system should offer a high flexibility for future add-ons.

The following components are proposed:

Power

(1 a,b,c)GND (32 a,b,c) VCC

This is given by the board design since 1 a,b,c and 32 a,b,c are each shortened.

(31 a,c)	Special voltage (I.e. +15V)
(30 a,c)	Special voltage (I.e. -15V)

Communication

(2 a c)	RS485
(3 a c) - (6 a c)	High speed channels
(29 a c)	TWI (I²C)

Parallel IO

(7 a) – (25 a) Data, address and control lines

(25 c) /IRQ1	Interrupt 1
(26 a) /IRQ2	Interrupt 2
(26 c) /Reset	Global reset line

Expansion

(27 a c) – (28 a c)	reserved for future add-ons.
(2 b) – (31 b)	The b row should not be used by any board since this might be used for an enhanced version of the mMDS.

A detailed pin out of the configuration used for the prototype can be found in a later section.

Design considerations

Each board designed for the bus should take the proposed configuration into account. It will be the users responsibility to ensure that only compatible boards are added to the bus, however the bus master should be able to detect and disable incompatible boards. Care needs to be taken if the special voltages are used.

The reference design will use an Atmel AVR RISC MCU for the Master Card.

OS Design considerations

The initial mMDS OS will only master the bus and offer a basic functionality to read and write memory / IO addresses in the system. A future version of the mMDS OS should offer a high level language interface, such as a BASIC interpreter, that allows easy and quick development of automated solutions.

It is envisioned that a RAM card present in the system can be loaded at runtime from a mass media like SDCARD and execute user code. Since the AVR CPU is not able to address the available bus address space, or execute code from the RAM, a simple interpreter should be provided with the mMDS OS.

The AVR BASIC Interpreter developed by Jörg Wolfram (www.jcwolfram.de) will serve as a basis for the development.

Bus Cards

A bus card is a circuit board that utilizes the bus infrastructure as either master or slave. Only one master is allowed on the bus.

Bus master

The bus master controls utilization on the bus.

There will be two types of slave cards on the bus. Active Cards and Passive Cards. Active Cards have their own microcontroller that control their behaviour. Passive cards are cards that are fully dependant on the microcontroller of the bus master or an active slave card.

During initialization, the bus master will query all other cards via the TWI bus in order to recognize available active add-ons.

Active cards will have the possibility to request Direct Parallel IO in order to perform high speed parallel IO between bus cards. During this Direct Parallel IO (DPIO), the bus master will assume the role of a bus client for the parallel IO part of the bus.

Active Cards

Active cards need to be able to listen on the TWI bus. Each card has a unique card id that consists of an available 7bit TWI address. A proposed list of card IDs can be found at the end of this document.

Active cards communicate their capabilities as part of the initialization to the bus master. This capability list includes: DPIO capable, High speed IO, required memory... tbd

Passive Cards

Passive cards do not need to be able to listen on the TWI interface. These cards are designed to serve as a client to master cards. This could be a special IO card, memory, etc. These cards are addressed using the parallel IO bus.

Master Card

Only one master card can be installed per bus.

It is intended to keep the management functions of the card as limited as possible in order to allow the developer to host additional functionality on the master card. The management functions will be bundled in a mMDS OS. The initial mMDS OS requires at least two 8 bit IO ports, the UART and the TWI. When the BASIC interpreter has been integrated into the mMDS OS, the master card will not be able to hold any user specific code. All user code needs to be stored in a mass storage device on the bus.

Initialization of the bus

After the initial POST and start of the operating system on the master card, the card will initialize the parallel IO control lines and start querying the slave cards via the Two Wire

Interface.

TWI Communication

The master will query all available card addresses on the TWI bus and request a presence answer from the slave cards. If at least one slave card has answered the call, the master will query the card capabilities.

Address Space for slave cards: 0x50 to 0x57 using the 24c02 EEprom for passive slaves and 0x70 to 0x77 for active slave cards.

These capabilities include: DPIO Capable, IO usage, RAM usage, high speed channel requests, etc.

Sequence of events:

- Power on
- Master does presence call
- Master queries capabilities of the active slaves
- Master queries capabilities of passive slaves.
- Master finishes OS start

The Master will also query common TWI addresses for passive slave cards that might offer special capabilities like RAM, IO, etc. These cards will only provide a small TWI EEprom that contains the offered capability list and a null terminated human readable string describing the card.

During the initialization of the cards, the master will assign a DPIO id to each cards offering the DPIO capability. This id is used to de-conflict multiple DPIO capable cards on the bus. A maximum of 8 DPIO capable cards are possible in the same system unless more slave addresses are assigned to the active slaves on the TWI bus.

Direct Parallel IO – DPIO

DPIO can be requested by an active slave. The slave will check the DPIOA line if no one else is performing a DPIO action. If the DPIOA line is already 0, the slave will wait until the line is reset. It then set the DPIOR line to 0 and wait for the master to acknowledge the request by setting the DPIOA line to 0.

After the master sets the DPIOA line to 0, the master performs a read operation. The client has to present its DPIO address at the data bits. The master will then perform a write operation and publishes the just read address. The address presented at the data bits is read by the slave card requesting the DPIO. The master will clear the bus and the DPIO starts.

NOTE: The Address during the read and write operation of the master are to be ignored.

The master will wait until the slave performing the DPIO resets the DPIOR line to 1. The Master will then set the DPIOA to 1 and resume mastering the parallel IO bus.

Multiple DPIO capable cards

In order to enable a de-conflicted access to DPIO, the active slave is only allowed to access the parallel bus if the master has acknowledged the DPIO request with the correct slave

address. If the address presented at the data bits is not the address of the card requesting the DPIO, the card has to release the DPIOR line immediately since a conflict might have occurred.

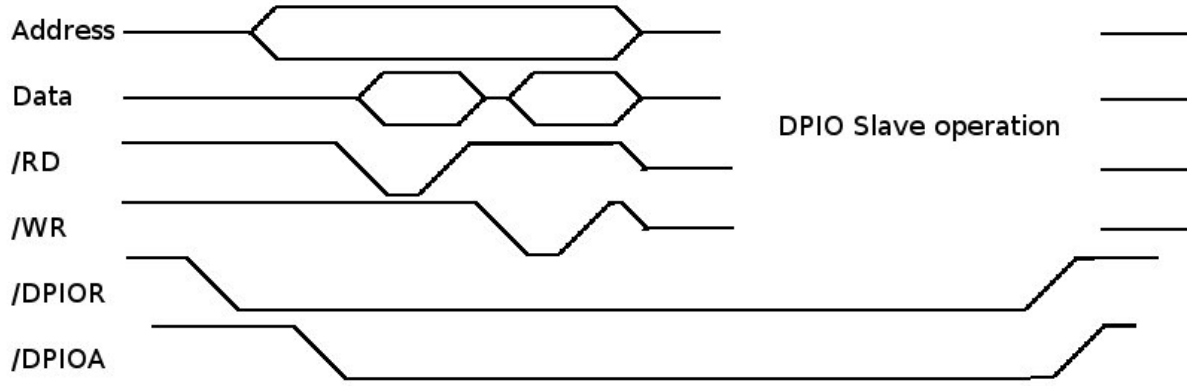
The card with DPIO address 0 would always be able to request the bus since a conflict on the data bus will result in address 0 to be displayed. Due to this possible privilege and PLD design constrains, DPIOID 0 will not be used. If two cards with higher addresses collide on the bus, the resulting address represented on the bus will not match any card and no card will start the DPIO. Both cards will release the DPIOR and the master will resume the bus mastering. Each card detecting a conflict should wait a random time frame before attempting to master the bus again. With a limited number of DPIO capable cards, the slaves should wait 10ms multiplied by their DPIO address.

If a system requires more than 5 DPIO cards, it is recommended to utilize a dedicated high speed channel that can be requested by the card during initialization instead of the DPIO feature.

DPIO Handling

This project will provide a PLD performing the required DPIO verification. It will accept the DPIO address and offer an MCU interface that contains a data latch and control lines to detect a DPIO conflict.

DPIO Timing



Slave Cards

Slave cards are add-ons to the mMDS system. They provide features like RAM, IO, Network, Video, etc.

It is the main purpose of this project to design a system that has a great flexibility for user cards that can be easily attached to the mMDS.

Active Cards

Active cards are cards that contain their own MCU. Possible card types are Video, Network, active bus interfaces, etc.

The main feature of active cards is the possibility to master the bus, or to use the high speed channels on the bus.

Active cards need to communicate with the bus master via the TWI interface. They will act as slaves to the bus and are addressed in the range described later in this document. It is the users responsibility to configure the slave address unique on the bus.

Initialization

During initialization the active card has to report it's capabilities and requirements to the mMDS OS. The bus master will send command 0x00 via the TWI to the card address. The card needs to answer this command with 0xff to notify the mMDS OS of it's presence.

Requesting Resources

After the presence call, the mMDS OS will query capabilities of the card. The mMDS OS will send the 0x01 command (query capabilities) and expects a list of capabilities as result from the card.

Passive Cards

Passive cards do not need to communicate with the master via the TWI. However, in order to keep the design flexible, it is recommended to implement a small TWI EEprom 24c02 on each passive card that contains the type of card and a list of capabilities.

The EEprom content should be:

Address:	Content:
0x00	Type of card: 0x01: IO 0x02: RAM
0x01-0x03	Start address of card
0x04-0x06	End address of card
0x07-	Null terminated string describing the card. (clear text)

The EEproms on the passive cards can have a device select code from 0x50 to 0x57 allowing up to 8 cards with this type of EEprom on the TWI bus.

The mMDS OS will provide means to initialize a TWI EEprom on a newly created card.

Bus Access

All cards, except the card having Direct Parallel IO – DPIO access, have to listen on the bus until they are addressed by the bus master. During the start of the system, the master card is the only card allowed to control the bus.

BUS description

The system should offer a bus as described in the hardware design. The maximum amount of cards are defined by the TWI address space limitations. The maximum amount of TWI addresses for the 24c02 EEprom of the passive cards limit the amount of passive cards to 8

unless a different EEprom is added to the additional cards with a different device select code. The mMDS OS should be designed to check for at least 8 passive cards.

Signal description

PIN	Function	Description	PIN	Function	Description
1 a	GND	Supply Ground	1 c	GND	Supply Ground
2 a	RS485+	RS 485 Bus	2 c	RS485-	RS 485 Bus
3 a	HSC1	High Speed Channel 1	3 c	HSC1	High Speed Channel 1
4 a	HSC2	High Speed Channel 2	4 c	HSC2	High Speed Channel 2
5 a	HSC3	High Speed Channel 3	5 c	HSC3	High Speed Channel 3
6 a	HSC4	High Speed Channel 4	6 c	HSC4	High Speed Channel 4
7 a	A0	Address bit 0	7 c	A1	Address bit 1
8 a	A2	Address bit 2	8 c	A3	Address bit 3
9 a	A4	Address bit 4	9 c	A5	Address bit 5
10 a	A6	Address bit 6	10 c	A7	Address bit 7
11 a	A8	Address bit 8	11 c	A9	Address bit 9
12 a	A10	Address bit 10	12 c	A11	Address bit 11
13 a	A12	Address bit 12	13 c	A13	Address bit 13
14 a	A14	Address bit 14	14 c	A15	Address bit 15
15 a	A 16	Address bit 16	15 c	A 17	Address bit 17
16 a	A 18	Address bit 18	16 c	A 19	Address bit 19
17 a	A 20	Address bit 20	17 c	A 21	Address bit 21
18 a	A 22	Address bit 22	18 c	A 23	Address bit 23
19 a	D0	Data bit 0	19 c	D4	Data bit 1
20 a	D1	Data bit 2	20 c	D5	Data bit 3
21 a	D2	Data bit 4	21 c	D6	Data bit 5
22 a	D3	Data bit 6	22 c	D7	Data bit 7
23 a	/RD	Read Data	23 c	/WD	Write Data
24 a	MEM /IO	Memory or IO operation	24 c	/DPIOR	DPIO Request
25 a	/DPIOA	DPIO Acknowledge	25 c	IRQ1	Interrupt 1
26 a	IRQ2	Interrupt 2	26 c	/RESET	System Reset

PIN	Function	Description	PIN	Function	Description
27 a	reserved		27 c	reserved	
28 a	reserved		28 c	reserved	
29 a	SDA	TWI Interface	29 c	SCL	TWI Interface
30 a	VDD	Special Voltage	30 c	VDD	Special Voltage
31 a	VDD	Special Voltage	31 c	VDD	Special Voltage
32 a	VCC	Supply Voltage + 5V	32 c	VCC	Supply Voltage + 5V

NOTE: Row b of the connectors should not be wired at any board in order to be compatible with future enhanced implementations based on this proposed bus.

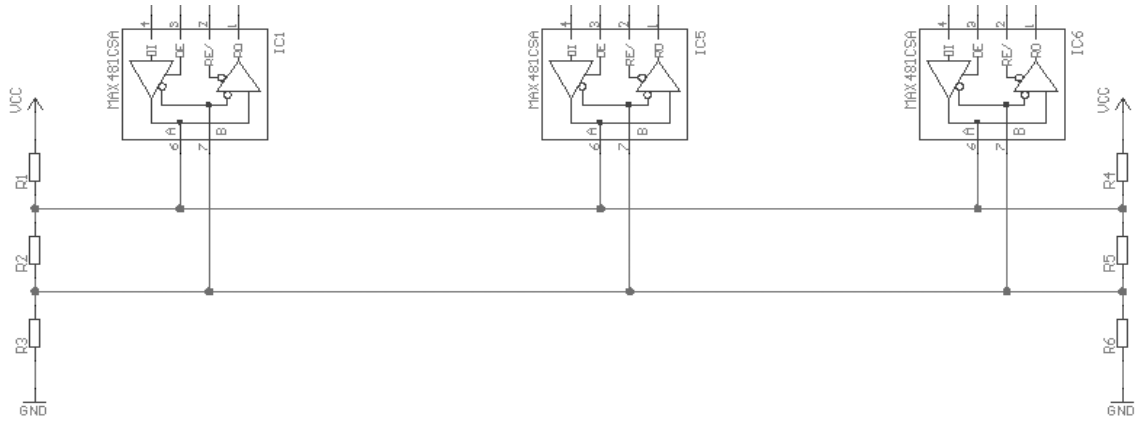
Parallel IO

Parallel IO is controlled by the master card. After the system start, an active slave might request Direct Parallel IO access as described in the master card section above.

RS485 connection

The RS485 connection on the bus is not used by the master card in the initial design. The master card should offer an external access to the bus via a standardized 9 pin connector on the front.

RS485 access for client cards:



R1, R3, R4, R6: 100kOhm

R2, R5: 120 Ohm

Resistors R1, R3, R4, and R6 can be omitted for short distances.

R2 and R5 are required.

Since only one endpoint of the RS485 bus is on the mMDS, it is the users responsibility to ensure a proper termination of the bus. The last card in the mMDS using the RS485 bus needs to implement the termination as well as any external connection made via the port at the bus master card.

TWI connection

The TWI connection is only used internally during the initialization of the mMDS. Slave cards can attach additional components to the TWI bus like voltage sensors, IO devices, etc. that do not collide with the already used addresses for the mMDS management. These devices can be addressed via the mMDS OS basic interpreter once employed.

High speed channels

There are four high speed channels on the bus. The initial design does not define the detailed implementation of these channels. However it is recommended to use a differential signal with speeds 10mbit or faster for each channel. RS422 might be a possible candidate, but this is left to the designers of the slave cards using this interface.

During the initialization, the master card will query the active slaves in order to recognize their requirements. During this initialization, the slave card should report which high speed channel it intents to use. If the slave card for whatever reason decides to not report it's high speed channel usage, the master will not be able to detect and report any conflicts on the high speed channels.

Card Id's

TWI Address space:

0x50 – 0x57 passive slave cards (using the 24c02 EEprom)

0x70 – 0x77 active slave cards.

All other addresses might be freely used to connect additional components to the TWI bus. Care must be taken to not exceed the maximum allowed

Proposed TWI addresses

Passive cards		**Active cards**	
0x50	RAM	0x70	Network
0x51	Digital IO	0x71	Video
0x52	Digital IO	0x72	E-Bus
0x53	Analogue IO	0x73	CAN
0x54	Analogue IO	0x74	Mass Storage
0x55	free	0x75	free
0x56	free	0x76	free
0x57	free	0x77	free

Card Communication

mMDS OS commands during active card initialization:

mMDS OS command:	**Card answer:**
0x00 presence call.	0xff
0x01 capability call.	capability list.
0x02 description call.	Null terminated string with card description. (clear text)
0x03 + DPIO ID	DPIO ID
0xff Initialization complete	

Capability list

Once the mMDS OS has requested the capabilities list from the active card, the card has to send a list of capabilities in accordance to the list below.

0x00	DPIO capable
0x01 + 6 byte	provides IO range (6 bytes are the start and end IO address)
0x02 + 6 byte	provides RAM range (6 bytes are the start and end RAM

address)

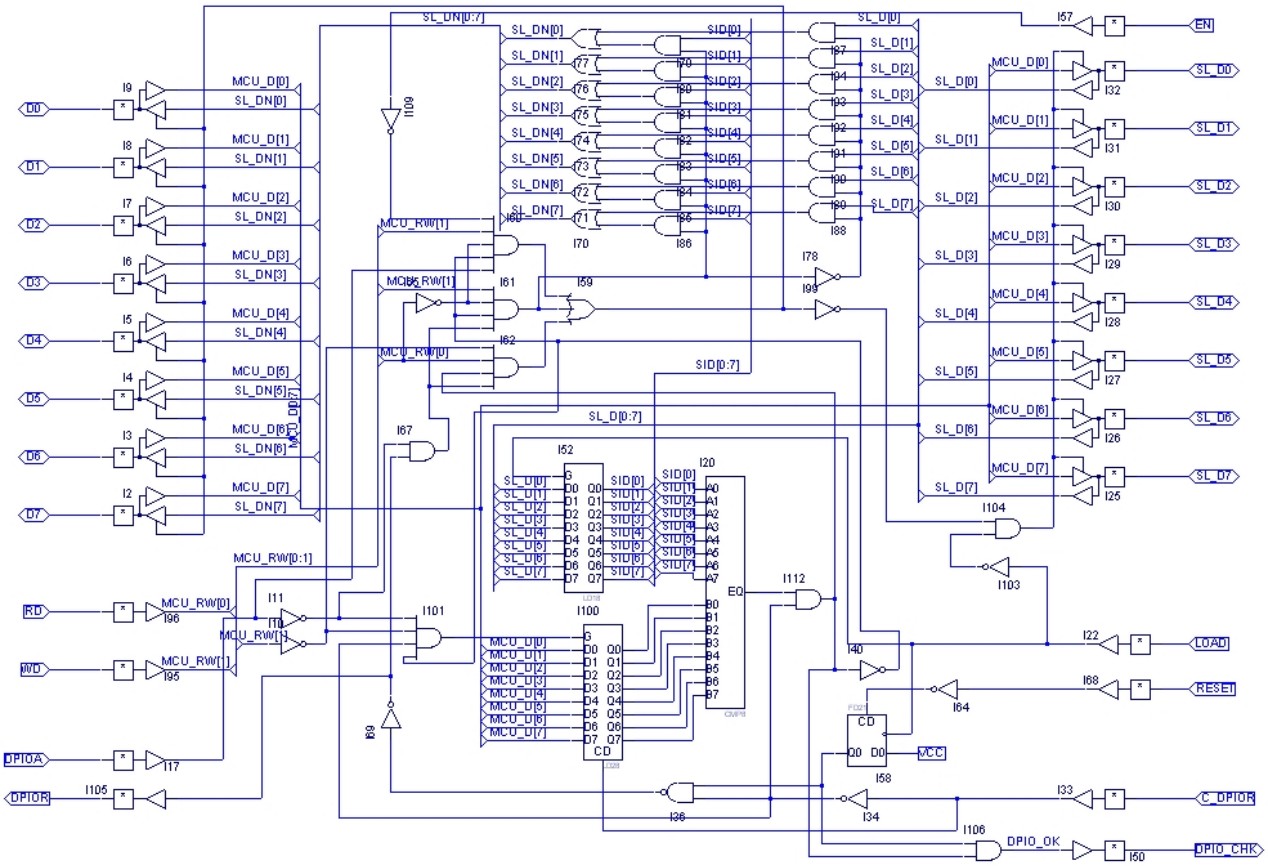
0x03 + 1 byte	High speed channel user. (followed by the channel number)

NOTE: capability 0x03 can be send multiple times in order to use more than one high speed channel.

Further capabilities are to be defined.

DPIO PLD

PLD Schematic



Output DPIOR of the PLD needs to drive the base of a transistor that pulls down the DPIOR line on the bus. This was done in order to fit the design into a small PLD like the ispLSI 1016 from Lattice. Using a tri state output for that port would have exceeded the available output enable pins in this chip.

PLD Pin out

To be defined.

PLD Test Script

```
MODULE dpio01

" TOOL:      ispLEVER
" DATE:      Sun Jan 11 14:14:57 2009
" TITLE:     dpio01.bls
" MODULE:    dpio01
" DESIGN:    dpio01
" FILENAME:  dpio01.abt
" PROJECT:   dpio
" VERSION:   1.2.00.11.33.08_Classic_M4KZE_update
```

```
" Inputs
    WD pin;
    RESET pin;
    RD pin;
    LOAD pin;
    EN pin;
    DPIOA pin;
    C_DPIOR pin;


" Outputs
    DPIO_CHK pin;
    DPIOR pin;


" Bidirs
    SL_D7 pin;
    SL_D6 pin;
    SL_D5 pin;
    SL_D4 pin;
    SL_D3 pin;
    SL_D2 pin;
    SL_D1 pin;
    SL_D0 pin;
    D7 pin;
    D6 pin;
    D5 pin;
    D4 pin;
    D3 pin;
    D2 pin;
    D1 pin;
    D0 pin;


      x=.x.;
      z = .z.;

Test_vectors


([EN,RD,WD,RESET,LOAD,DPIOA,C_DPIOR,SL_D7,SL_D6,SL_D5,SL_D4,SL_D3,SL_D2,SL
_D1,SL_D0,D7,D6,D5,D4,D3,D2,D1,D0] ->
[DPIO_CHK,DPIOR,SL_D7,SL_D6,SL_D5,SL_D4,SL_D3,SL_D2,SL_D1,SL_D0,D7,D6,D5,D
4,D3,D2,D1,D0])

"Initial reset
[0,0,0,0,0,0,0, z,z,z,z,z,z,z,z, z,z,z,z,z,z,z,z] -> [0,0,
x,x,x,x,x,x,x,x, x,x,x,x,x,x,x,x];
"Master Read
[0,0,1,1,0,1,1, 1,1,1,0,0,0,1,1, z,z,z,z,z,z,z,z] -> [0,0,
x,x,x,x,x,x,x,x, 1,1,1,0,0,0,1,1];
"Master Write
[0,1,0,1,0,1,1, z,z,z,z,z,z,z,z, 0,0,0,1,1,1,0,0] -> [0,0,
0,0,0,1,1,1,0,0, x,x,x,x,x,x,x,x];
"Master Read wo enable
[1,0,1,1,0,1,1, 1,1,1,0,0,0,1,1, z,z,z,z,z,z,z,z] -> [0,0,
x,x,x,x,x,x,x,x, x,x,x,x,x,x,x,x];
"Master Write wo enable
[1,1,0,1,0,1,1, z,z,z,z,z,z,z,z, 0,0,0,1,1,1,0,0] -> [0,0,
x,x,x,x,x,x,x,x, x,x,x,x,x,x,x,x];
```

```
"DPIO request WO load
[0,1,1,1,0,1,0, z,z,z,z,z,z,z,z,z, z,z,z,z,z,z,z,z,z] -> [0,0,
x,x,x,x,x,x,x,x, x,x,x,x,x,x,x,x,x];
"DPIO Load Address
[0,1,1,1,1,1,1,1, 1,0,1,0,1,1,1,1, z,z,z,z,z,z,z,z,z] -> [0,0,
x,x,x,x,x,x,x,x, x,x,x,x,x,x,x,x,x];
"DPIO Request w load
[0,1,1,1,0,1,0, z,z,z,z,z,z,z,z,z, z,z,z,z,z,z,z,z,z] -> [0,1,
x,x,x,x,x,x,x,x, x,x,x,x,x,x,x,x,x];

"DPIO Acknowledge with conflict
[0,1,1,1,0,1,0, z,z,z,z,z,z,z,z,z, z,z,z,z,z,z,z,z,z] -> [0,1,
x,x,x,x,x,x,x,x, x,x,x,x,x,x,x,x,x];
[0,0,1,1,0,0,0, z,z,z,z,z,z,z,z,z, z,z,z,z,z,z,z,z,z] -> [0,1,
x,x,x,x,x,x,x,x, 1,0,1,0,1,1,1,1];
[0,1,0,1,0,0,0, z,z,z,z,z,z,z,z,z, 1,0,1,0,0,1,1,1] -> [0,1,
x,x,x,x,x,x,x,x, x,x,x,x,x,x,x,x,x];
[0,1,1,1,0,0,1, z,z,z,z,z,z,z,z,z, z,z,z,z,z,z,z,z,z] -> [0,0,
x,x,x,x,x,x,x,x, x,x,x,x,x,x,x,x,x];

"DPIO Acknowledge without conflict
[0,1,1,1,0,1,0, z,z,z,z,z,z,z,z,z, z,z,z,z,z,z,z,z,z] -> [0,1,
x,x,x,x,x,x,x,x, x,x,x,x,x,x,x,x,x];
[0,0,1,1,0,0,0, z,z,z,z,z,z,z,z,z, z,z,z,z,z,z,z,z,z] -> [0,1,
x,x,x,x,x,x,x,x, 1,0,1,0,1,1,1,1];
[0,1,0,1,0,0,0, z,z,z,z,z,z,z,z,z, 1,0,1,0,1,1,1,1] -> [1,1,
x,x,x,x,x,x,x,x, x,x,x,x,x,x,x,x,x];

"DPIO Terminate
[0,1,1,1,0,0,1, z,z,z,z,z,z,z,z,z, z,z,z,z,z,z,z,z,z] -> [0,0,
x,x,x,x,x,x,x,x, x,x,x,x,x,x,x,x,x];
[0,1,1,1,0,1,1, z,z,z,z,z,z,z,z,z, z,z,z,z,z,z,z,z,z] -> [0,0,
x,x,x,x,x,x,x,x, x,x,x,x,x,x,x,x,x];


"DPIO Request w load
[0,1,1,1,0,1,0, z,z,z,z,z,z,z,z,z, z,z,z,z,z,z,z,z,z] -> [0,1,
x,x,x,x,x,x,x,x, x,x,x,x,x,x,x,x,x];

"Master Read
[0,0,1,1,0,1,0, 1,1,1,0,0,0,1,1, z,z,z,z,z,z,z,z,z] -> [0,1,
x,x,x,x,x,x,x,x, 1,1,1,0,0,0,1,1];
"Master Write
[0,1,0,1,0,1,0, z,z,z,z,z,z,z,z,z, 0,0,0,1,1,1,0,0] -> [0,1,
0,0,0,1,1,1,0,0, x,x,x,x,x,x,x,x,x];

"DPIO Acknowledge with conflict
[1,1,1,1,0,1,0, z,z,z,z,z,z,z,z,z, z,z,z,z,z,z,z,z,z] -> [0,1,
x,x,x,x,x,x,x,x, x,x,x,x,x,x,x,x,x];
[1,0,1,1,0,0,0, z,z,z,z,z,z,z,z,z, z,z,z,z,z,z,z,z,z] -> [0,1,
x,x,x,x,x,x,x,x, 1,0,1,0,1,1,1,1];
[1,1,0,1,0,0,0, z,z,z,z,z,z,z,z,z, 1,0,1,0,0,1,1,1] -> [0,1,
x,x,x,x,x,x,x,x, x,x,x,x,x,x,x,x,x];
[1,1,1,1,0,0,1, z,z,z,z,z,z,z,z,z, z,z,z,z,z,z,z,z,z] -> [0,0,
x,x,x,x,x,x,x,x, x,x,x,x,x,x,x,x,x];

"DPIO Acknowledge without conflict
[1,1,1,1,0,1,0, z,z,z,z,z,z,z,z,z, z,z,z,z,z,z,z,z,z] -> [0,1,
x,x,x,x,x,x,x,x, x,x,x,x,x,x,x,x,x];
```

```
[1,0,1,1,0,0,0, z,z,z,z,z,z,z,z, z,z,z,z,z,z,z,z] -> [0,1,
x,x,x,x,x,x,x,x, 1,0,1,0,1,1,1,1];
[1,1,0,1,0,0,0, z,z,z,z,z,z,z,z, 1,0,1,0,1,1,1,1] -> [1,1,
x,x,x,x,x,x,x,x, x,x,x,x,x,x,x,x];

"DPIO read
[0,0,1,1,0,0,0, z,z,z,z,z,z,z,z, 1,1,1,1,0,0,0,0] -> [1,1,
1,1,1,1,0,0,0,0, x,x,x,x,x,x,x,x];
"DPIO write
[0,1,0,1,0,0,0, 0,0,0,0,1,1,1,1, z,z,z,z,z,z,z,z] -> [1,1,
x,x,x,x,x,x,x,x, 0,0,0,0,1,1,1,1];
"DPIO read/write wo enable
"DPIO read
[1,0,1,1,0,0,0, z,z,z,z,z,z,z,z, 1,1,1,1,0,0,0,0] -> [1,1,
1,1,1,1,0,0,0,0, x,x,x,x,x,x,x,x];
"DPIO write
[1,1,0,1,0,0,0, 0,0,0,0,1,1,1,1, z,z,z,z,z,z,z,z] -> [1,1,
x,x,x,x,x,x,x,x, 0,0,0,0,1,1,1,1];

"DPIO Terminate
[0,1,1,1,0,0,1, z,z,z,z,z,z,z,z, z,z,z,z,z,z,z,z] -> [0,0,
x,x,x,x,x,x,x,x, x,x,x,x,x,x,x,x];
[0,1,1,1,0,1,1, z,z,z,z,z,z,z,z, z,z,z,z,z,z,z,z] -> [0,0,
x,x,x,x,x,x,x,x, x,x,x,x,x,x,x,x];


END
```

PLD Test Waveforms