|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| BE/CO/HT  2009-06-26 | | | | | |
| Technical Note | | | | | |
| WORLDFIP INSOURCING PROJECT  nanoFIP WP3  nanoFIP  Functional Specification | | | | | |
| Abstract  The nanoFIP is an FPGA component implementing the WorldFIP protocol that can be used in field devices able to communicate at the three standard speeds: 31.25 kbit/s, 1 Mbit/s and 2.5 Mbit/s. It is designed to be radiation tolerant by using different single event upset mitigation techniques. To improve the radiation tolerance, the nanoFIP implements a minimal subset of the WorldFIP services that it provides without the need for any microprocessor. WorldFIP variables are transferred either directly on 16‑bit I/O ports or via a memory included in the nanoFIP component. | | | | | |
| Authors:  E. van der Bij  E. Gousiou | | | Checked by:  R. Brun  F. Formenti  Q. King  J. Palluel G. Peñacoba J. Serrano B. Todd | |  |
|  | | | | | |
| ***History of Changes*** | | | | | |
| Rev. No. | Date | Pages | | Description of Changes | |
| 1.0  1.1  1.2  1.3 | 09-06-2009  22-06-2009  26-06-2009  25-10-2010 | All  All  All  All | | First draft  Title: functional specification. Name changed to nanoFIP. Added definitions of PDU and MPS. Variable E0xy needs station address as data for selective resets. Comments about presence variable contents. VARx\_RDY/VARx\_ACC signals introduced. U\_ACER status added. NOSTAT signal introduced. Received PDU\_TYPE and length of variable stored in memory. Modified text for external logic for reset by long message. Added VAR1B0 reset assist output. Multiplexed identification inputs.  8-bit WISHBONE bus. Significance/refreshment style implementation. No VAR2B0. Reset variable adressed by broadcast. Examples of consumed and produced data. RSTON not activated on RSTIN.  WISHBONE interface with 8 bits DAT\_I/DAT\_O[7:0]; DAT\_I/DAT\_O[15:8] defined as extra bits for stand-alone operation.  Signal CYC\_I added to WISHBONE to comply with WISHBONE rule 3.40.  Memory addresses MSB reserved (Page 13, Table 9 modified).  Signals FX\_TXD/RXA/RXD renamed to FD\_ TXD/RXA/RXD.  FD\_RXA renamed to FD\_RXCDN, to avoid confusions with RXA, analogue output of FIELDRIVE.  Page 5 bullet 12 and Page 6, 2.2 and Page 9, 3.4: PDU\_TYPE and Length bytes are checked on reception.  Page 6 bullets 3,5,6 and Page 12, 4.1 and Page 14, 6.2: PDU\_TYPE and Length added to the number of consumed/ produced bytes.  Page 10, 3.5: broadcast vars are accepted in stand-alone mode.  Page 11, 19: Four output pins added to provide user access to the nanoFIP status bits 2 to 5.  In stand-alone mode, if while consuming more than 2 bytes of data arrive, the frame is still considered valid and the first 2 bytes arrive to DAT\_O.  PoR input added. | |
| 1.4 | 15-03-2011 | All | | EDMS doc no. added  Fixed References broken links.  no broadcast variables in stand-alone mode (would be a problem with FIPdiag).  broadcast var 91h (04h is a produced var in microfip).  Turnaround & Silence times changed according to measurements & datasheets of microFIP.  Changes on nanoFIP status byte, bits 4 & 5.  WISHBONE clock renamed to wclk; 40MHz clock to uclk.  Figures redone. nanoFIP instead of NanoFIP.  Write Only (instead of Write/ Read) access rights to the Produced memory. | |
| 1.5 | 31-10-2011 | All | | Added JTAG feature description  Status byte, bit 5, Manchester encoding check removed | |

Table of Contents (not compulsory, can be removed)

1 INTRODUCTION 5

2 Functionality 6

2.1 Variable Structure and Addressing 6

2.2 Functionality of variables 6

2.3 FRAME DELIMITERS AND FRAME CHECKSUM 7

2.4 Turnaround and Silence time 7

3 Variable handling 8

3.1 MPS Status definition 8

3.2 presence variable (14xyh) 8

3.3 identification variable (10xyh) 9

3.4 Consumed variable VAR1 (05xyh) 9

3.5 consumed broadcast variable VAR2 (91..h) 10

3.6 produced variable VAR3 (06xyh) 10

4 resets 12

4.1 Consumed variable (E0..h) 12

4.2 power on reset 12

4.3 External reset logic 12

4.4 resets overview 12

4.5 Memory map 13

5 user interface 14

5.1 startup 14

5.2 Stand-alone mode 14

5.2.1 consumed variable 14

5.2.2 Produced variable 15

5.3 Memory mode 15

5.3.1 Consumed variables 15

5.3.2 produced variable 16

5.4 User access verification 17

6 interface signal definition 18

6.1 Pin assignment 18

6.2 Wishbone datasheet 21

7 References 21

annex 22

1 JTAG feature 22

1.1 introduction 22

1.2 JTAG CONSUMED variable VAR4 (AAxyh) 22

1.3 JTAG PRODUCED variable VAR5 (Abxyh) 24

<Page Intentionally Left Blank>

# INTRODUCTION

The nanoFIP is an FPGA component implementing the WorldFIP protocol [1] that can be used in field devices able to communicate at the three standard speeds. The nanoFIP, that is developed as part of the WorldFIP insourcing project [2], is designed to be radiation tolerant by using different single event upset mitigation techniques such as triple module redundancy.

The device is used in conjunction with a FIELDRIVE driver chip and FIELDTR insulating transformer, both available from the company ALSTOM [3]. For implementing a robust reset, external logic is needed.

To improve the radiation tolerance, the nanoFIP implements a minimal subset of the WorldFIP services. The main specifications are:

* No need for a processor to set up data transfers.
* Consumption of one addressed variable.
* Consumption of one broadcast variable.
* Production of one addressed variable.
* Consumed variable size between 1 and 124 bytes.
* Produced variable size between 2 and 124 bytes settable by hardwired pins.
* Fixed variable addresses.
* WorldFIP speed settable to 31.25 kbit/s, 1 Mbit/s or 2.5 Mbit/s.
* Frame delimiters and Frame Checksum (FCS) using the WorldFIP/IEC standard.
* Produced variable uses PDU\_TYPE 40h (compact value Protocol Data Unit).
* nanoFIP status (optional) and MPS (Manufacturing Periodical/Aperiodical Services) status sent together with the produced variable.
* Consumed variables are only updated when correct PDU\_TYPE, Length and FCS bytes have been received.
* Consumed freshness and significance data available in memory but not treated.
* Produced freshness and significance show data update status.
* Simple interface to variable data
  + Variable data transfer over an integrated memory accessible with an 8‑bit WISHBONE System-on-Chip interconnection.
  + Possibility of stand-alone mode with 16 input and 16 output lines without the need to transfer data to or from memory.
  + Separate data valid outputs for each variable (consumed and produced).
* Several reset possibilities
  + External reset pin.
  + Addressed reset by broadcast consumed variable validated by station address as data.
  + Reset output available to external logic.
* Capability to program/ reprogram a companion FPGA over JTAG.

# Functionality

## Variable Structure and Addressing

The station address xyh is set by the input pins SUBS[7:0].

After power-on the nanoFIP works as a station handling the following variables addressed by:

* ID\_DAT = 14xyh.  
  For the presence variable with a content as described in Section .
* ID\_DAT = 10xyh.  
  For the identification variable with a content as described in Section .
* ID\_DAT = 05xyh.  
  The station consumes a variable of any length up to 124 bytes preceded by the PDU\_TYPE and Length bytes and followed by the MPS status byte.
* ID\_DAT = 91..h.  
  The station consumes a variable of any length up to 124 bytes preceded by the PDU\_TYPE and Length bytes and followed by the MPS status byte. As the station address is not checked, this variable can be used for the consumption of variables that need to be broadcast to all stations on the segment.
* ID\_DAT = 06xyh.  
  The station produces a variable with a user-settable length content (2, 8,   
  16, 32, 64 or 124 user-data bytes), an optional 1-byte nanoFIP status plus the MPS status, the PDU\_TYPE and the Length bytes (in total maximum 128 bytes).
* ID\_DAT = E0..h  
  The station consumes this 2-byte variable (plus PDU\_TYPE, Length and MPS status bytes). As the station address is not checked, this variable can be broadcast to all stations on the segment. If the first data byte contains the station address it will reset the nanoFIP and the FIELDRIVE component. If the second data byte contains the station address it will assert the reset output pin RSTON. If none of the bytes contains the station address the variable is ignored.

## Functionality of variables

The produced and consumed variables are handled as follows:

* The consumed variables are only updated (assertion of VARx\_RDY) when the PDU\_TYPE, Length and Frame Checksum (FCS) bytes are correct.
* The PDU\_TYPE and Length fields of consumed variables are stored in the communication memory before the actual data.
* The refreshment and significance status of consumed variables is available in the communication memory as the last byte after the data. This byte is the MPS (Manufacturing Periodical/Aperiodical Services) status.
* The produced variable contains optionally (configurable by the pin NOSTAT) as the one but last byte the nanoFIP status. As last byte it contains the MPS status with the refreshment and the significance status as described in section .
* The produced data variable uses PDU\_TYPE 40h (compact value Protocol Data Unit).

## FRAME DELIMITERS AND FRAME CHECKSUM

The nanoFIP uses the frame delimiters and Frame Checksum (FCS) of the WorldFIP/IEC standard. The FIP/NFC coding is not supported.

## Turnaround and Silence time

The WorldFIP Turnaround and Silence times are fixed, but dependent on the chosen bit rate as is shown in Table 1. The Turnaround time refers to a produced variable and defines the minimum time that the nanoFIP waits to deliver its RP\_DAT reply, after the reception of an ID\_DAT. The Silence time refers to a consumed variable and defines the maximum time that the nanoFIP waits for an RP\_DAT, after the reception of an ID\_DAT; after this time, the nanoFIP returns to the idle state. The values of the Turnaround times are taken after measurements on microFIP chips configured with the fastest settings. The values of the Silence times are taken from the recommendations of document [4].

|  |  |  |
| --- | --- | --- |
| Speed | Turnaround time | Silence time |
| 31.25 kbit/s | 480 μs | 4096 μs |
| 1 Mbit/s | 14 μs | 150 μs |
| 2.5 Mbit/s | 6 μs | 96 μs |

Table : **Turnaround and Silence times**

# Variable handling

## MPS Status definition

When transmitting a Manufacturing Periodical/Aperiodical Services (MPS) variable the status byte (last transmitted) is formatted as shown in Table 2:

|  |  |  |
| --- | --- | --- |
| **MPS Status bit** | **Name** | **Contents** |
| 0 (last transmitted) | refreshment | 1/0 |
| 1 |  | 0 |
| 2 | significance | 1/0 |
| 3 |  | 0 |
| 4-7 |  | 0000 |

Table : **MPS Status**

In stand-alone mode (SLONE=Vcc) the MPS Status byte will have the refreshment and significance bits always set to 1. In memory mode (SLONE=Gnd) the refreshment and significance bits will be set to 1 if the user has updated the produced variable (i.e. any address of the produced variable has been written to) since the last transmission of the variable (assertion of VAR3\_RDY); otherwise the refreshment and significance bits will be set to 0. This emulates a refreshment time equal to the WorldFIP cycle time.

## presence variable (14xyh)

After the power-on reset, the nanoFIP is able to produce the SMMPS presence variable. The frame contents dedicated to the nanoFIP embedded presence variable are shown in Table 3. No MPS status is associated with this variable [5].

|  |  |  |
| --- | --- | --- |
| **Byte** | **Field name** | **Contents** |
| 0 | PDU\_TYPE | 50h |
| 1 | Length | 05h |
| 2 | 2 | 80h |
| 3 | 3 | 03h |
| 4 | LG\_VAR\_I | 00h |
| 5 | Reserved | F0h |
| 6 | BA status | priority | 00h |

Table : **Presence variable fields**

Implementation detail: for compatibility the nanoFIP sends the same presence variable as the MICROFIP. However, according to [5] bytes 2 and 3 (without a name or definition for these) contain the shown fixed values, byte 4 should in fact contain the length of the Identification variable, byte 5 is reserved and should contain 00h while byte 6 contains the BA status and priority (00h: BA not supported, BA priority le+).

## identification variable (10xyh)

After the power-on reset, the nanoFIP is able to produce the SMMPS identification variable. The contents of the Constructor and Model fields are partially configurable by connecting C\_ID[3:0] and M\_ID[3:0] to Ground, Vcc, S\_ID[1] or S\_ID[0] (see section ). The frame contents dedicated to the nanoFIP identification variable is shown in Table 4. No MPS status is associated with this variable [5].

|  |  |  |
| --- | --- | --- |
| **Byte** | **Field name** | **Contents** |
| 0 | PDU\_TYPE | 52h |
| 1 | Length | 08h |
| 2 | WorldFIP profile | 01h |
| 3 | Class | 00h |
| 4 | Constructor (first byte) | 00h |
| 5 | Constructor (second byte) | Const[7:0]h |
| 6 | Model (first byte) | Model[7:0]h |
| 7 | Model (second byte) | 00h |
| 8 | Version | 00h |
| 9 | User | 00h |

Table : **Identification variable fields**

## Consumed variable VAR1 (05xyh)

Variables of any size up to 124 user-data bytes can be received with the variable address 05xyh. The PDU\_TYPE, Length and Frame Checksum (FCS) are verified and should be valid in order to declare the variable valid. If the received variable is not valid, VAR1\_RDY will not be asserted and the communication memory (see section ) may contain invalid data.

The PDU\_TYPE and Length fields of consumed variables are available in the communication memory as the first bytes before the actual data (see Table 5). The MPS status byte (see section ) received from the network is also stored in the communication memory, directly after the data. This MPS status byte, containing the refreshment and significance bits, is not interpreted by the nanoFIP.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Memory offset** | **Contents** |  | **Memory offset** | **Contents** |
| 0 | PDU\_TYPE |  | 0 | 40h |
| 1 | Length (n-1) |  | 1 | 09h |
| 2 | Data byte 0 |  | 2 | 01h |
| 3 | Data byte 1 |  | 3 | 02h |
| … |  |  | … | … |
| n-1 | Data byte n-2 |  | 9 | 08h |
| n | MPS Status |  | 10 | 05­­­­­­h |

Table : **Consumed variable in communication memory with example**

## consumed broadcast variable VAR2 (91..h)

Any variable with the address 91..h (i.e. independent of the station address) is received in the same way as a normal consumed variable. This broadcast variable uses the signal VAR2\_RDY. In memory mode the broadcast variable has its own area in the communication memory (see section ) and the WISHBONE interface for accessing it. Stand-alone mode does not support consumed broadcast variables.

## produced variable VAR3 (06xyh)

In memory mode the size of the produced variable is configured by the external pins P3\_LGTH[2:0] to 2, 8, 16, 32, 64 or 124 user-data bytes (see pin description, section ). In stand-alone mode, the size is predefined to 2 user-data bytes.

Table 6 shows how the data is sent from the communication memory to the WorldFIP bus. If the pin NOSTAT is grounded, the one but last byte sent contains the status of the nanoFIP interface as described in Table 8. If the pin NOSTAT is connected to Vcc, this status byte is not sent, as is shown in Table 7. As the last byte the MPS status (see section ) is automatically added at the end of the variable and sent to the network.

The Length byte represents the sum of user-data bytes (P3\_LGTH) plus the nanoFIP status (if applicable) and the MPS status bytes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Memory offset** | **Contents** |  | **Memory offset** | **Contents** |
| not from memory | PDU\_TYPE |  | not from memory | 40h |
| not from memory | Length (n-1) |  | not from memory | 0Ah |
| 2 | Data byte 0 |  | 2 | 01h |
| 3 | Data byte 1 |  | 3 | 02h |
| … |  |  | … | … |
| n-2 | Data byte n-3 |  | 9 | 08h |
| n-1 | nanoFIP status |  | 10 | 00h |
| n | MPS Status |  | 11 | 05h |

Table : **Produced variable as sent to WorldFIP (NOSTAT=Gnd) with example**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Memory offset** | **Contents** |  | **Memory offset** | **Contents** |
| not from memory | PDU\_TYPE |  | not from memory | 40h |
| not from memory | Length (n-1) |  | not from memory | 09h |
| 2 | Data byte 0 |  | 2 | 01h |
| 3 | Data byte 1 |  | 3 | 02h |
| … |  |  | … | … |
| n-1 | Data byte n-2 |  | 9 | 08h |
| n | MPS Status |  | 10 | 05h |

Table : **Produced variable as sent to WorldFIP (NOSTAT=Vcc) with example**

| **nanoFIP Status bit** | **Name** | **Contents** |
| --- | --- | --- |
| 0 | r1 | reserved, ignore on read |
| 1 | r2 | reserved, ignore on read |
| 2 | u\_cacer | User consumed variable access error |
| 3 | u\_pacer | User produced variable access error |
| 4 | r\_tler | Received Control or PDU\_TYPE error or incoherent length |
| 5 | r\_fcser | Received FCS or bit number error |
| 6 | t\_txer | Transmit error (FIELDRIVE) |
| 7 | t\_wder | Watchdog error (FIELDRIVE) |

Table : **nanoFIP Status**

Bits 0 and 1 Reserved bits, ignore the value.

Bit 2=1 The user logic accessed a consumed variable (VAR1\_ACC or VAR2\_ACC high) when this variable was not ready (VAR1\_RDY respectively VAR2\_RDY was low). This may mean that the consumed variable was read while it was being updated from the WorldFIP bus.

Bit 3=1 The user logic accessed the produced variable (VAR3\_ACC high) when this variable was not ready (VAR3\_RDY low). This may mean that the produced variable was modified by the user while it was being sent to the WorldFIP bus.

Bit 4=1 A consumed variable has arrived for this station with a wrong Control or PDU\_TYPE byte or with an excessive length or with a Length byte different than the actual length received.

Bit 5=1 A consumed variable has arrived for this station with a wrong Frame Check Sum or with a number of bits not corresponding to a multiple of 8.

Bit 6=1 The FIELDRIVE has signalled a transmission error.

Bit 7=1 The FIELDRIVE has signalled a watchdog timer problem.

Bits 2 to 5 contain the status since the previously produced variable was sent to the WorldFIP bus. They are reset after the transmission of the produced variable.

Bits 6 and 7 are directly coming from the FIELDRIVE signals and are only reset when the nanoFIP is reset (nanoFIP internal reset, Figure 1).

Bits 2 to 5 are also available to the user on the output pins U\_CACER, U\_PACER, R\_TLER, R\_FCSER (see section ).

# resets

## Consumed variable (E0..h)

Either in memory or in stand-alone mode the station consumes the 2-byte variable along with the PDU\_TYPE, the Length and the MPS status bytes with the address E0..h (i.e. independent of the station address). Whenever it receives this variable it will verify the data, without writing them to the memory or to the DATO bus. If the first data byte contains the station address it will perform a reset of the nanoFIP and the FIELDRIVE. If the second data byte contains the station address it will assert the reset output pin RSTON for 8 clock cycles. If none of the two data bytes contains the station address the variable is ignored.

Note that if the nanoFIP or FIELDRIVE chip has serious problems (e.g. a state-machine blocked or the analogue receiver not working), it will not be able to correctly receive this variable. For critical applications it is therefore suggested to implement external logic to reset the nanoFIP.

## power on reset

The RSTPON pin should be held low for at least 2 ms to bring the nanoFIP to an initial defined state.

## External reset logic

External logic acting on the input RSTIN may implement a very robust reset mechanism based on the timing of the Carrier Detect signal from the FIELDRIVE. For example, if the external logic detects bus activity with a data transfer of messages with a length of over 200 bytes it could activate the reset input of the nanoFIP.

Other user logic may implement a reset based on the data contents or on detecting no activity on the VARx\_RDY signals.

## resets overview

Figure 1 summarises all the reset inputs and outputs of the nanoFIP.

## Picture6.png

Figure 1: **Overview of the reset logic**

## Memory map

When the pin SLONE is grounded, the exchange of the data of the three variables goes over three memory blocks organised as shown in Table 9. Access is done via an 8-bit wide WISHBONE interface [6].

|  |  |  |  |
| --- | --- | --- | --- |
| **Address ADR\_I[9:0]** | **Contents** | **Size** | **Access** |
| **0\_00**\_0000000b | PDU\_TYPE Variable 1 | 1 byte | Read only |
| **0\_00**\_0000001b | Length Variable 1 | 1 byte | Read only |
| **0\_00**\_0000010b | **0\_00**\_1111110b | **Variable 1 05xyh**  **(Consumed)** | max. 124 bytes followed by MPS status | Read Only |
| **0\_00**\_1111111b | reserved | 1 byte | Do not access |
| **0\_01**\_0000000b | PDU\_TYPE Variable 2 | 1 byte | Read only |
| **0\_01**\_0000001b | Length Variable 2 | 1 byte | Read only |
| **0\_01**\_0000010b | **0\_01**\_1111110b | **Variable 2 91..h (Consumed, broadcast)** | max. 124 bytes followed by MPS status | Read Only |
| **0\_01**\_1111111b | reserved | 1 byte | Do not access |
| **0\_10**\_0000000b | reserved | 1 byte | Do not access |
| **0\_10**\_0000001b | reserved | 1 byte | Do not access |
| **0\_10**\_0000010b | **0\_10**\_1111101b | **Variable 3 06xyh (Produced)** | max. 124 bytes | Write Only |
| **0\_10**\_1111110b | reserved | 1 byte | Do not access |
| **0\_10**\_1111111b | reserved | 1 byte | Do not access |
| **0\_11**\_0000000b | **0\_11**\_1111111b | reserved | 128 bytes | Do not access |
| **1\_xx\_**xxxxxxxb | reserved | 512 bytes | Do not access |

Table : **nanoFIP memory map**

# user interface

## startup

After power-on, the nanoFIP automatically starts up using the settings as defined by the input pins RATE[1:0], SUBS[7:0], M\_ID[3:0], C\_ID[3:0] and P3\_LGTH[2:0].

The nanoFIP will directly be available to respond to the different WorldFIP requests such as Presence and Identification and it will consume and produce the variables.

## Stand-alone mode

When the pin SLONE is connected to Vcc, the nanoFIP will operate in stand-alone mode. In this mode the nanoFIP will consume a variable containing 2 bytes of data along with PDU\_TYPE, Length and MPS status bytes (5 bytes total). Any extra bytes will be ignored. nanoFIP will produce 6-byte variables (2 bytes of user-data + PDU\_TYPE+ Length + nanoFIP status + MPS status). If the signal NOSTAT is connected to Vcc, the nanoFIP status will not be sent (i.e. it will produce only 5 bytes).

### consumed variable

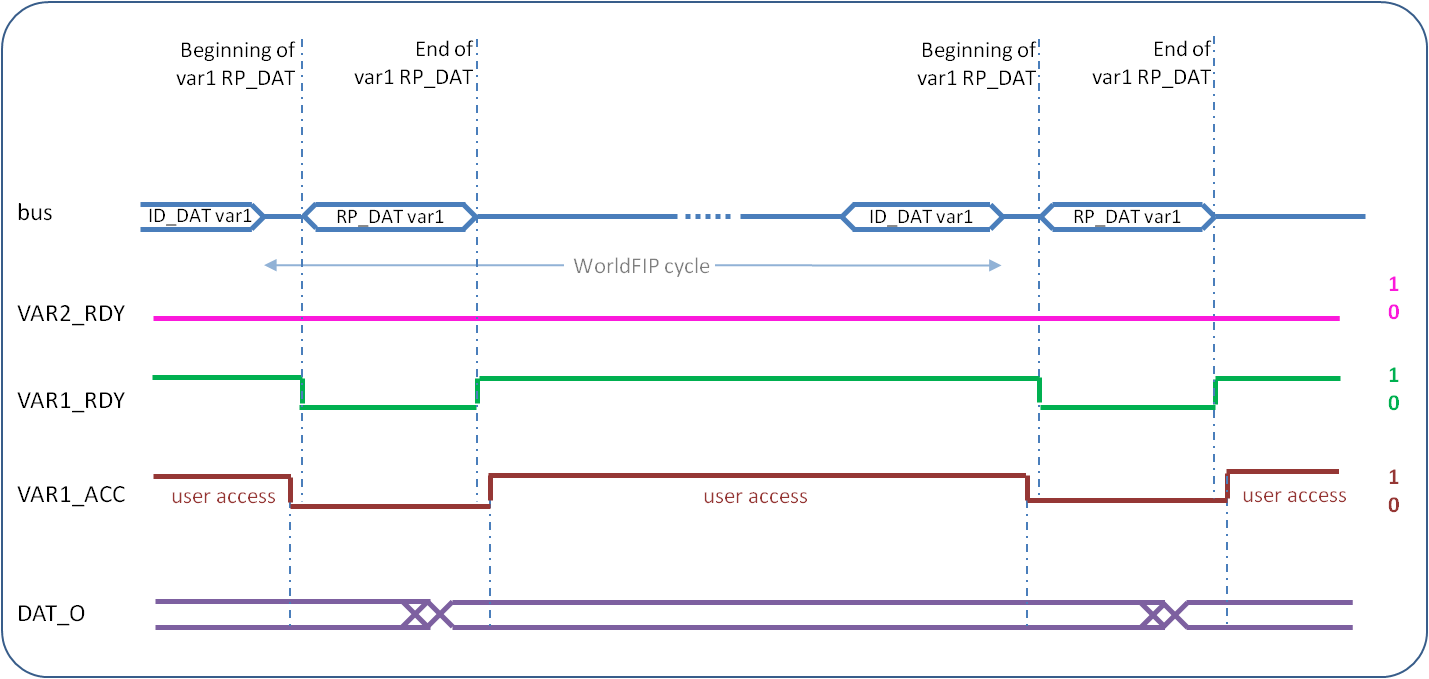
In stand-alone mode the consumed variable (05xyh) will be available on the pins DAT\_O[15:0]. The first data byte will be on DAT\_O[7:0] and the second on DAT\_O[15:8]. The reception of new variable data will be signalled by the VAR1\_RDY signal (see ). The assertion of VAR1\_RDY indicates that a valid update (with correct PDU\_TYPE, Length and FCS bytes) has been received for this variable. When VAR1\_RDY is asserted, DAT\_O[15:0] will be stable and can safely be read. When an update of the variable is about to arrive over WorldFIP (end of ID\_DAT), VAR1\_RDY will be de-asserted and the output may change. It is the user’s responsibility to copy the data if needed before (e.g. on the rising edge of VAR1\_RDY).

Figure 2: **Consumed variable timing in stand-alone mode**

### Produced variable

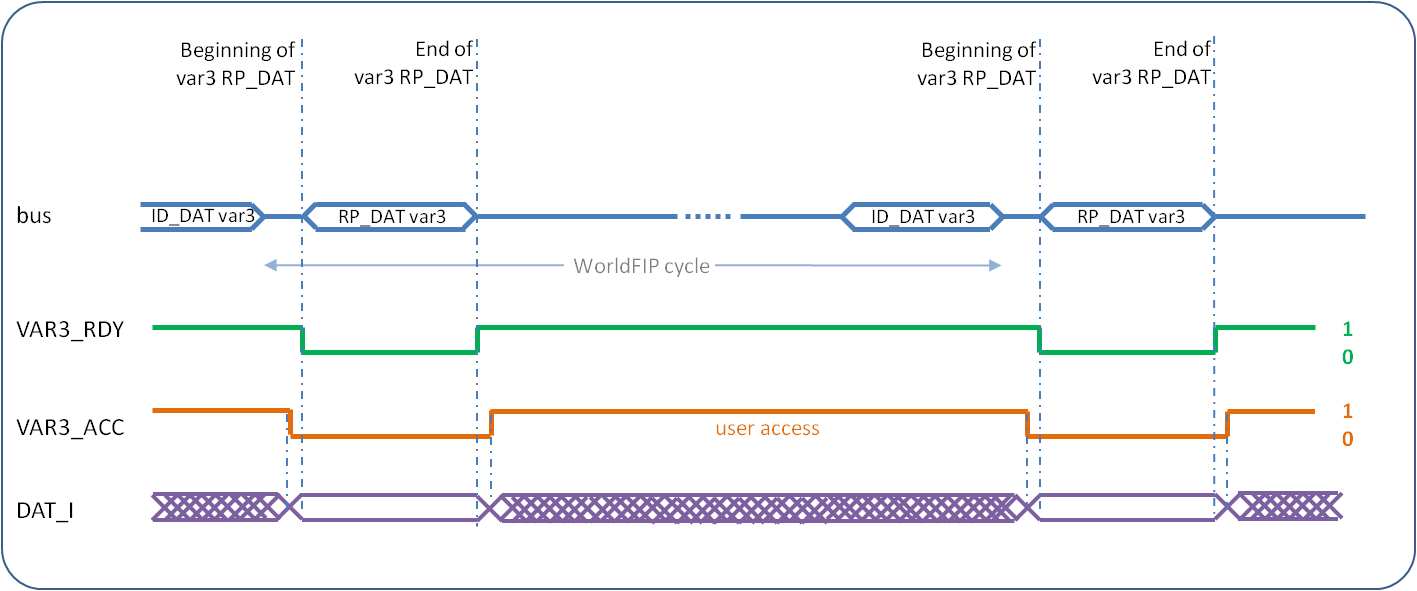
In stand-alone mode the produced variable (06xyh) should be presented at the pins DAT\_I[15:0]. When VAR3\_RDY is asserted, the user may change the input and the nanoFIP will sample the data on the first clock cycle after it has de-asserted VAR3\_RDY (see ). It is the user’s responsibility to make sure the data is stable at the moment of sampling.

Figure 3: **Produced variable timing in stand-alone mode**

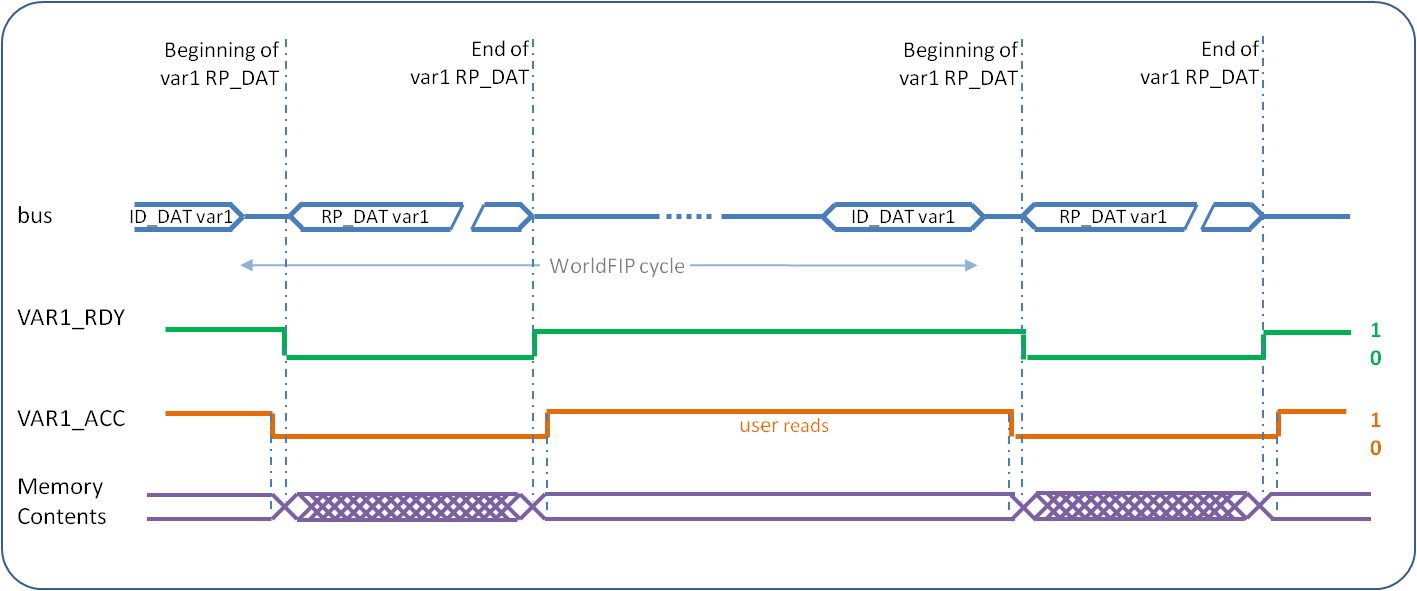
## Memory mode

When the pin SLONE is grounded, the nanoFIP will operate in memory mode. In this mode the nanoFIP will have its full functionality and variables of any size up to 124 bytes can be consumed and those set by the P3\_LGTH[2:0] pins can be produced.

### Consumed variables

Whenever a consumed variable is received the VAR1\_RDY (05xyh) or VAR2\_RDY (91..h) signal will be asserted (see ). During the time the VAR1\_RDY or VAR2\_RDY is asserted, the data in the memory will be stable and the user may safely read the contents of the memory. The data will be stable in the memory until a new consumed variable is received from the WorldFIP bus, which will be signalled by the de-assertion of VAR1\_RDY or VAR2\_RDY. It is the user’s responsibility to copy the data if needed before, e.g. by guaranteeing to read the variable within the variable’s WorldFIP scanning period after assertion of VAR1\_RDY or VAR2\_RDY.

Note: The three VARx\_RDY signals are independent and only related to their corresponding memory block; the user for example could read a variable 2, when a variable 1 is being written.

Figure 4: **Consumed variable timing in memory mode**

### produced variable

When VAR3\_RDY is asserted, the user may write new variable data into the memory (see ). When the variable is requested and about to be sent to the WorldFIP bus (end of the next ID\_DAT) the nanoFIP will de-assert VAR3\_RDY. It is the user’s responsibility to guarantee that the data in the memory is stable during the time VAR3\_RDY is de-asserted, e.g. by guaranteeing to write the variable within the variable’s WorldFIP scanning period after assertion of VAR3\_RDY.

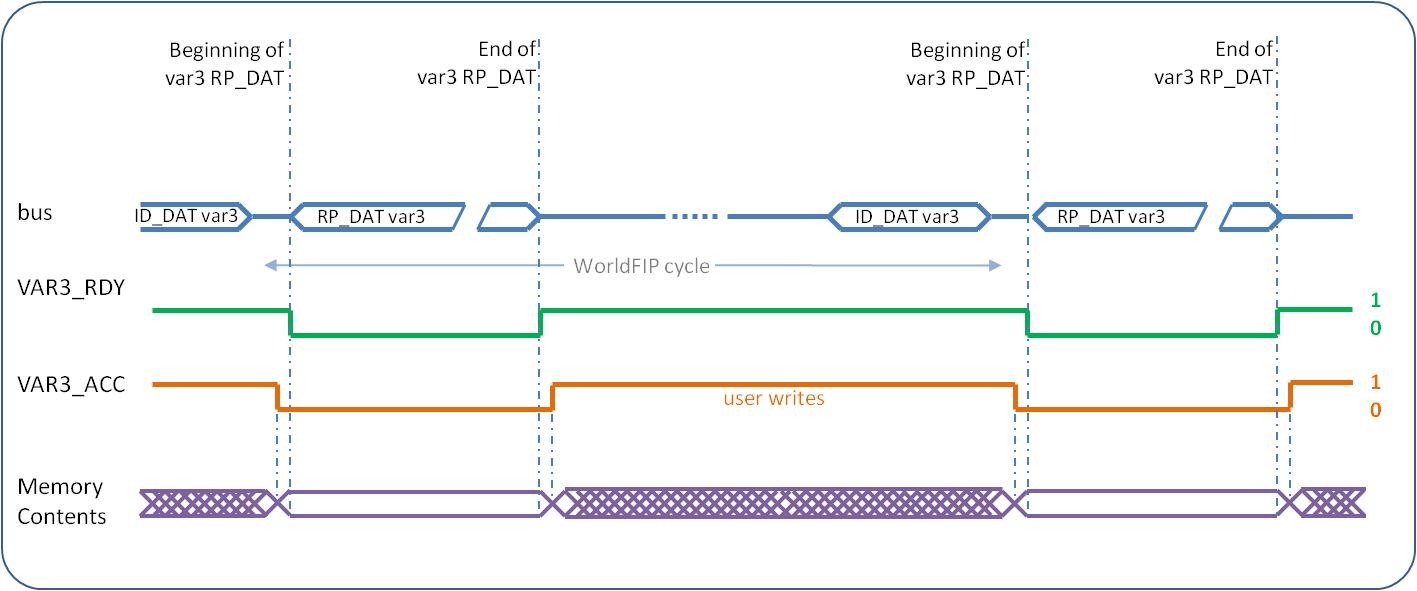
Implementation detail: after a reset, the nanoFIP will respond with the non-initialised contents of the memory with the MPS status byte set to 0. The MPS status byte will also be 0 if none of the produced variable addresses has been accessed since the last assertion of VAR3\_RDY (see section ).

Figure 5: **Produced variable timing in memory mode**

## User access verification

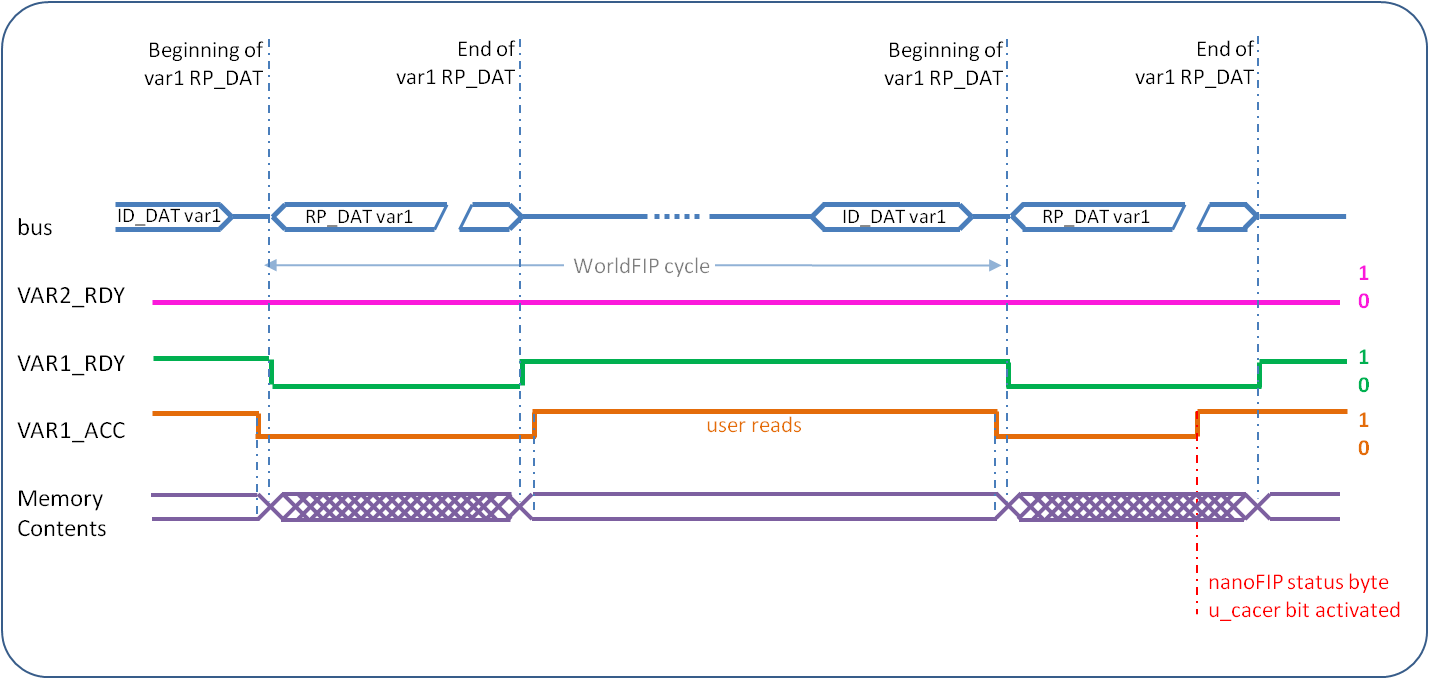
The user may signal to the nanoFIP that it is accessing a variable by asserting the signal VAR1\_ACC, VAR2\_ACC or VAR3\_ACC. When the VARx\_ACC signal is asserted the nanoFIP will verify that the corresponding VARx\_RDY signal is asserted (showing that the variable safely can be accessed). If the nanoFIP detects that the variable was not ready to be accessed (i.e. a consumed variable was being received from WorldFIP or the produced variable was being sent to WorldFIP), the nanoFIP status bit u\_cacer or u\_pacer respectively will be set (see Table 8), showing that certain data may be corrupted (see ). When this happens the nanoFIP will continue to produce and consume variables. For applications not requiring this functionality the VARx\_ACC signals may be grounded.

Figure 6: **User access error on a consumed variable**

Note: The WISHBONE signals (STB\_I and ADR\_I) give to the nanoFIP a clear indication of the accesses in memory attempted by the user. The extra access verification with the VARx\_ACC signals allows the detection of the user’s access to the WISHBONE if a state machine gets blocked (with STB\_I to ‘0’) without having read the complete memory.

# interface signal definition

## Pin assignment

| Pin | Type | Description | MicroFIP equivalent |
| --- | --- | --- | --- |
| POWER SUPPLY | | | |
| P3V3 | Supply | 3.3 Volt power supply |  |
| P1V5 | Supply | 1.5 Volt power supply |  |
| Gnd | Supply | Ground |  |
| WORLDFIP SETTINGS | | | |
| RATE[1:0] | Input | Bit rate  00: 31.25 kbit/s 01: 1 Mbit/s 10: 2.5 Mbit/s 11: reserved, do not use | A[1:0] |
| SUBS[7:0] | Input | Subscriber number coding. Station address. |  |
| S\_ID[1:0] | Output | Identification selection (see M\_ID, C\_ID) |  |
| M\_ID[3:0] | Input | Identification variable settings (see section ). Connect the ID inputs either to Gnd, Vcc, S\_ID[0] or S\_ID[1] to obtain different values for the Model data (i=0,1,2,3).  M\_ID[i] connected to: Gnd S\_ID0 S\_ID1 Vcc Model [2\*i] 0 1 0 1 Model [2\*i+1] 0 0 1 1 |  |
| C\_ID[3:0] | Input | Identification variable settings (see section ). Connect the ID inputs either to Gnd, Vcc, S\_ID[0] or S\_ID[1] to obtain different values for the Constructor data (i=0,1,2,3).  C\_ID[i] connected to: Gnd S\_ID0 S\_ID1 Vcc Constructor [2\*i] 0 1 0 1 Constructor [2\*i+1] 0 0 1 1 |  |
| P3\_LGTH[2:0] | Input | Produced variable data length  000: 2 Bytes 001: 8 Bytes 010: 16 Bytes 011: 32 Bytes 100: 64 Bytes 101: 124 Bytes 110: reserved, do not use 111: reserved, do not use  Actual size: +1 PDU\_TYPE byte +1 Length byte +1 nanoFIP Status byte +1 MPS Status byte (last transmitted) |  |
| FIELDRIVE | | | |
| FD\_RSTN | Output | Initialisation control, active low |  |
| FD\_WDGN | Input | Watchdog on transmitter | WTC1X |
| FD\_TXER | Input | Transmitter error | TER1X |
| FD\_TXENA | Output | Transmitter enable | TXE |
| FD\_TXCK | Output | Line driver half bit clock | TXCK |
| FD\_TXD | Output | Transmitter data | TXD |
| FD\_RXCDN | Input | Reception activity detection, active low | RXA1X |
| FD\_RXD | Input | Receiver data |  |
| USER INTERFACE, GENERAL SIGNALS | | | |
| UCLK | Input | 40 MHz clock | CLK |
| SLONE | Input | Stand-alone mode | SLONE |
| NOSTAT | Input | If connected to Vcc, disables sending of nanoFIP status together with the produced data. |  |
| RSTIN | Input | Initialisation control, active low.  Resets the nanoFIP and the FIELDRIVE (FD\_RSTN). It should remain active for at least 8 CLK cycles. |  |
| RSTON | Output | Reset output, active low. Active when the reset variable is received and the second byte contains the station address. It is active for 8 CLK cycles. A glitch may appear at Power On. |  |
| RSTPON | Input | Power On Reset, active low. It has to remain active for at least 2 ms. |  |
| USER INTERFACE, NON WISHBONE | | | |
| VAR1\_RDY | Output | Signals new data is received and can safely be read (Consumed variable 05xyh). In stand-alone mode one may sample the data on the first clock edge VAR1\_RDY is high. |  |
| VAR1\_ACC | Input | Signals that the user logic is accessing variable 1. Only used to generate a status that verifies that VAR1\_RDY was high when accessing. May be grounded. |  |
| VAR2\_RDY | Output | Signals new data is received and can safely be read (Consumed broadcast variable 91xyh). |  |
| VAR2\_ACC | Input | Signals that the user logic is accessing variable 2. Only used to generate a status that verifies that VAR2\_RDY was high when accessing. May be grounded. |  |
| VAR3\_RDY | Output | Signals that the variable can safely be written (Produced variable 06xyh). In stand-alone mode, data is sampled on the first clock after VAR\_RDY is de-asserted. |  |
| VAR3\_ACC | Input | Signals that the user logic is accessing variable 3. Only used to generate a status that verifies that VAR3\_RDY was high when accessing. May be grounded. |  |
| U\_CACER | Output | nanoFIP status byte - bit 2 |  |
| U\_PACER | Output | nanoFIP status byte - bit 3 |  |
| R\_TLER | Output | nanoFIP status byte - bit 4 |  |
| R\_FCSER | Output | nanoFIP status byte - bit 5 |  |
| DAT\_I[15:8] | Input | Data In of stand-alone mode (used in addition to DAT\_I[7:0] of the WISHBONE interface. |  |
| DAT\_O[15:8] | Output | Data Out of stand-alone mode (used in addition to DAT\_O[7:0] of the WISHBONE interface. |  |
| USER INTERFACE, WISHBONE SLAVE | | | |
| WCLK\_I | Input | WISHBONE clock. May be independent of CLK. |  |
| DAT\_I[7:0] | Input | WISHBONE Data In |  |
| DAT\_O[7:0] | Output | WISHBONE Data Out |  |
| ADR\_I[9:0] | Input | Address |  |
| RST\_I | Input | WISHBONE reset. Does not reset other internal logic. |  |
| STB\_I | Input | Strobe |  |
| ACK\_O | Output | Acknowledge |  |
| WE\_I | Input | Write enable |  |
| CYC\_I | Input | Cycle (not used, but essential for the compliance to the WISHBONE interface rules) |  |
| USER INTERFACE, JTAG CONTROLLER (refer to Annex) | | | |
| JC\_TCK | Output | 5 MHz clock to target FPGA |  |
| JC\_TMS | Output | Test Mode Select to target FPGA |  |
| JC\_TDI | Output | Test Data In to target FPGA |  |
| JC\_TDO | Input | Test Data Out from target FPGA |  |

## Wishbone datasheet

|  |  |
| --- | --- |
| Description | Specification |
| General description | 3 times 128 x 8-bit memory |
| Supported cycles | SLAVE READ/WRITE |
| Data port, size Data port, granularity Data port, maximum operand size Data transfer ordering Data transfer sequencing | 8-bit 8-bit 8-bit Big endian and/or little endian Undefined |
| Clock frequency constraints | 40 MHz max |
| Supported signal list and cross reference to equivalent WISHBONE signals | Signal name WISHBONE Equiv. ACK\_O ACK\_O ADR\_I[9:0] ADR\_I[] CLK\_I CLK\_I DAT\_I[7:0] DAT\_I[] DAT\_O[7:0] DAT\_O[] STB\_I STB\_I WE\_I WE\_I  RST\_I RST\_I  CYC\_I CYC\_I |
| Special requirements | None |

# References

[1] IEC fieldbus standards. [IEC-61784-1 Industrial communication networks: Profiles : Part 1: Fieldbus profiles](http://cdsweb.cern.ch/record/1169412)

[2] CernFIP pages. <http://www.ohwr.org/projects/cern-fip>

[3] ALSTOM FIELDRIVE and FIELDTR documentation. <http://www.fipware.fr/>

[4] Adaptation des Temps de Retournement et Silence des Agents (MICROFIP) sur ceux du MANAGER (FULLFIP), CERN AB-Note-2005-023(CO), R. Brun, 2005.  
<http://cdsweb.cern.ch/record/834924>

[5] FIP Device Manager Software Version 4 User Reference Manual, ALS50278h-en. <http://cern-worldfip.web.cern.ch/cern-worldfip/Docs/FIPWARE/User%20documentation/Software/FDM_FULLFIP2.pdf>

[6] Specification for the: WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores, Revision: B.3, Released: September 7, 2002.  
<http://www.opencores.org/downloads/wbspec_b3.pdf>

# annex

# JTAG feature

## introduction

This feature gives the possibility to reprogram a companion FPGA though the IEEE 1149.1 JTAG interface. WorldFIP frames arriving to nanoFIP are translated to the equivalent TMS, TDI and TCK signals that drive the Test Access Port (TAP) of the target FPGA. At the same time the TDO input is monitored. The TRST signal is not used.

Two variables have been added for this feature:

* ID\_DAT= AAxyh.

The station consumes this variable of any size up to 124 bytes preceded by the PDU\_TYPE and Length bytes and followed by the MPS status byte.

* ID\_DAT = ABxyh.

The station produces this variable with a predefined length of 1 data-byte, followed by the nanoFIP status and the MPS bytes and preceded by the PDU\_TYPE and the Length.

## JTAG CONSUMED variable VAR4 (AAxyh)

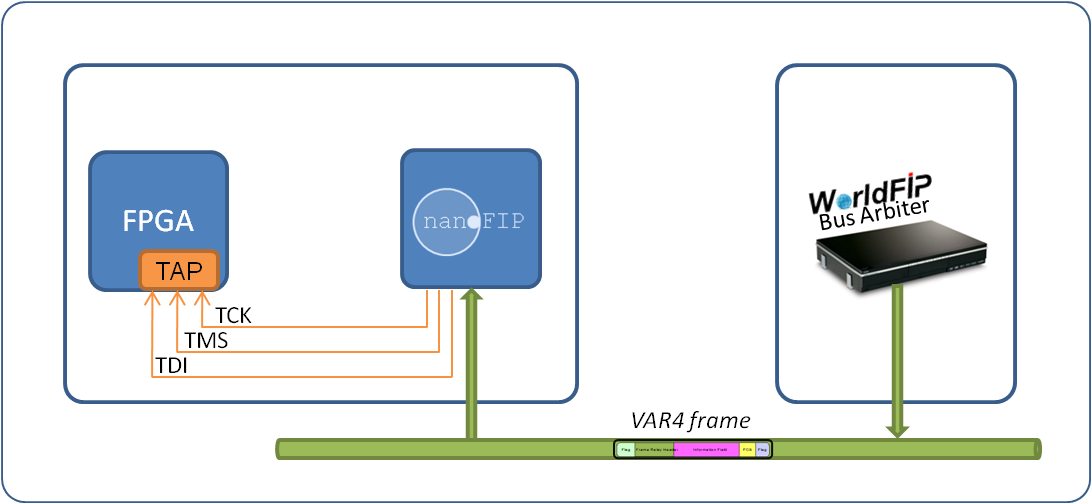
The JTAG consumed variable VAR4 is received by nanoFIP exactly as a consumed VAR1 and it is stored in a dedicated memory (Table 1). After its validation nanoFIP translates the received data to the equivalent TMS, TDI and TCK bits that drive the Test Access Port (TAP) of the companion FPGA.

Figure 1: **JTAG Controller Consumption**

|  |  |  |  |
| --- | --- | --- | --- |
| **Address** | **Contents** | **Size** | **Access** |
| 0000000b | PDU\_TYPE Variable 4 | 1 byte | Not accessible by external logic |
| 0000001b | Length Variable 4 | 1 byte |
| 0000010b | 1111110b | **Variable 4**  **AAxyh**  **(JTAG Controller, Consumed)** | max. 124 bytes followed by MPS status |
| 1111111b | reserved | 1 byte |

Table 1:**JTAG Controller Consumed memory**

As Figure 2 indicates, the consumed VAR4 bytes 4 to n-1 contain pairs of TMS and TDI bits. Since the number of those bits varies and may not be a multiple of 8, bytes 2 and 3 concatenated in big endian encoding are used to indicate the exact number of TMS and TDI bits that have to be treated.

In the example of Figure 2: n = 7Fh = 127d and the concatenation of bytes 2 and 3 = 03CAh = 970d. Therefore 121 complete bytes (4 to 125) and two bits of byte 126 contain TMS and TDI information.

|  |  |  |
| --- | --- | --- |
| 7(MSB) | TMS (3) | 1 |
| 6 | TDI (3) | 0 |
| 5 | TMS (2) | 1 |
| 4 | TDI (2) | 0 |
| 3 | TMS (1) | 1 |
| 2 | TDI (1) | 1 |
| 1 | TMS (0) | 0 |
| 0(LSB) | TDI (0) | 0 |

|  |  |  |
| --- | --- | --- |
| **Memory offset** | **Description** | **Content** |
| 0 | PDU\_TYPE | 40h |
| 1 | Length | 7Eh [n-1] |
| 2 | Number of TMS, TDI bits to be considered  (big endian encoding) | 03h |
| 3 | CAh |
| 4 | TMS, TDI bits | ACh |
| 5 | TMS, TDI bits | F0h |
| ... | ... | ... |
| ... | ... | ... |
| 126 [n-1] | Some TMS, TDI bits | 40h |
| 127 [n] | MPS status | 05h |

|  |  |  |
| --- | --- | --- |
| 7(MSB) | TMS (3) | 0 |
| 6 | TDI (3) | 1 |
| 5 | Not considered | 0 |
| 4 | Not considered | 0 |
| 3 | Not considered | 0 |
| 2 | Not considered | 0 |
| 1 | Not considered | 0 |
| 0(LSB) | Not considered | 0 |

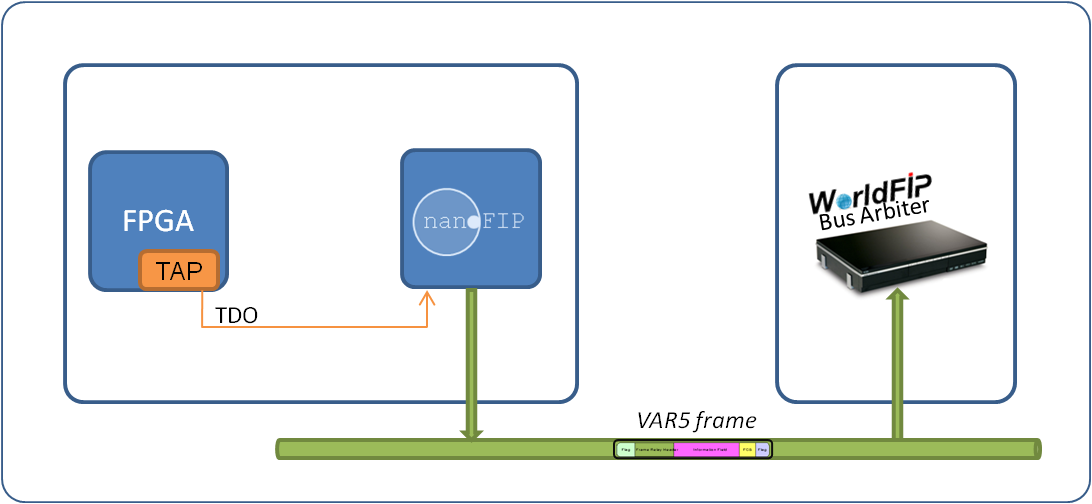
Figure 2: **JTAG Controller Consumed variable in communication memory with example**

After the reception of the complete frame and its validation in terms of PDU\_TYPE, Length and Frame Checksum (FCS), nanoFIP is responsible for the generation of a 5 MHz clock TCK. The clock is generated using the system clock, uclk, of 40 MHz.

A new TCK cycle is created for every TMS-TDI pair; the clock stays inactive if a new TMS-TDI is not available.

nanoFIP will output the corresponding values for TMS and TDI on the falling edge of the TCK, starting from byte 4, TMS-TDI (0).

## JTAG PRODUCED variable VAR5 (Abxyh)

While driving the TAP, nanoFIP is also sampling the TAP input signal TDO; it stores it and sends it back to the Bus Arbiter (BA) at the next JTAG produced variable VAR5.

The production of a VAR5 follows the same principles as a produced VAR3. However, regardless of the P3\_LGTH input, one user-data byte is sent. Moreover, the nanoFIP status byte is always sent, regardless of the NOSTAT input, so as to inform the BA of the status of a previous consumption.

Figure 3: **JTAG Controller Production**

Since only one byte is sent, there is no need to use the memory for JTAG produced variables.

The TDO input is sampled at the rising edge of the TCK. In reality, only the last sampled TDO bit is significant and used by the software on the BA side; this bit is added at the end of an array of seven zeroes to form the one user-data byte that is delivered with VAR5.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Byte number** | | **Description** | | **Content** |
| 0 | PDU\_TYPE | | 40h | |
| 1 | Length | | 03h | |
| 2 | user-data byte | | 01h | |
| 3 | nanoFIP status byte | | 00h | |
| 4 | MPS status byte | | 05h | |

Figure 4: **JTAG Controller Produced variable with example**

|  |  |  |
| --- | --- | --- |
| 7(MSB) | No information | 0 |
| 6 | No information | 0 |
| 5 | No information | 0 |
| 4 | No information | 0 |
| 3 | No information | 0 |
| 2 | No information | 0 |
| 1 | No information | 0 |
| 0(LSB) | TDO | 1 |