# Spinnaker to Neuelab [Spin2Neu]

Author: Francesco Diotalevi

Company: Istituto Italiano di Tecnologia - EDL

|  |  |  |  |
| --- | --- | --- | --- |
| History: | | | |
| **Revision** | **Date** | **Author** | **Description** |
| 1.0 | May 16th, 2018 | Francesco Diotalevi | First Release (DRAFT) after G. De Robertis design in 2015 |
|  |  |  |  |

Contents

[Spinnaker to Neuelab [Spin2Neu] 1](#_Toc514253862)

[Introduction 3](#_Toc514253863)

[1.1 Input SpiNNaker Link 3](#_Toc514253864)

[1.2 Output SpiNNaker Link 3](#_Toc514253865)

[2 Spin2Neu Diagram 5](#_Toc514253866)

[3 Main components of the Spin2Neu architecture 7](#_Toc514253867)

[3.1 Spinn\_neu I/f 7](#_Toc514253868)

[3.2 CoreMonSeq 7](#_Toc514253869)

[3.2.1 Monitor 7](#_Toc514253870)

[3.2.2 Timestamper 7](#_Toc514253871)

[3.2.3 AEXSequencer 7](#_Toc514253872)

[3.2.4 InFIfo 7](#_Toc514253873)

[3.2.5 OutFIfo 7](#_Toc514253874)

[3.2.6 TimeStampWrap detector 7](#_Toc514253875)

[3.3 AXIStream I/f 7](#_Toc514253876)

[3.4 AXI4Lite I/f 7](#_Toc514253877)

[4 Spin2Neu internal registers 8](#_Toc514253878)

[4.1 Control register (CTRL\_REG) 9](#_Toc514253879)

[4.2 RX DATA buffer register (RXDATA\_REG) 10](#_Toc514253880)

[4.3 RX Time buffer register (RXTIME\_REG) 11](#_Toc514253881)

[4.4 TX Data buffer register (TXDATA\_REG) 12](#_Toc514253882)

[4.5 DMA register (DMA\_REG) 13](#_Toc514253883)

[4.6 RAW Status Register (STAT\_RAW\_REG) 14](#_Toc514253884)

[4.7 IRQ Register (IRQ\_REG) 15](#_Toc514253885)

[4.8 Mask Register (MSK\_REG) 16](#_Toc514253886)

[4.9 Wrapping TimeStamp Register (WRAPTimeStamp\_REG) 17](#_Toc514253887)

[4.10 Identification register (ID\_REG) 18](#_Toc514253888)

[5 References 19](#_Toc514253889)

[6 Appendixes 20](#_Toc514253890)

# Introduction

The Spin2Neu design aims to deliver and receive data to a spinnaker through 2 different unidirectional links: the input SpiNNaker Link and the output SpiNNaker link.

The user sends data to spinnaker by using the *Axi4 Stream Output Bus lines*  and receives data from spinnaker by using the Axi4 Stream Input Bus lines.

The user can control the whole data processing thanks to the AXI 4 Lite Bus interface that is able to communicate with the Spin2Neu internal registers that will be detailed in Section 4.

## Input SpiNNaker Link

The input SpiNNaker Link consists in 7 data wires named *data\_2of7\_from\_spinnaker* and an acknowledge wire *ack\_to\_spinnaker*.

## Output SpiNNaker Link

The output SpiNNaker Link consists in 7 data wires named *data\_2of7\_to\_spinnaker* and an acknowledge wire *ack\_from\_spinnaker*.

In the Table 1 the input and output ports of the Spin2Neu module are listed.

Table 1 Spin2Neu interface signals description

| **Comment** | **Port name** | **Width** | **Dir** | **Description** |
| --- | --- | --- | --- | --- |
| AXI4 Lite Bus lines | S\_AXI\_ACLK | 1 | I | AXI Clock, System clock line |
| S\_AXI\_ARESETN | 1 | I | AXI Reset active low line |
| S\_AXI\_AWADDR | 32 | I | AXI Write address |
| S\_AXI\_AWVALID | 1 | I | Write address valid |
| S\_AXI\_WDATA | 32 | I | Write data |
| S\_AXI\_WSTRB | 4 | I | Write strobes |
| S\_AXI\_WVALID | 1 | I | Write valid |
| S\_AXI\_BREADY | 1 | I | Response ready |
| S\_AXI\_ARADDR | 32 | I | Read address |
| S\_AXI\_ARVALID | 1 | I | Read address valid |
| S\_AXI\_RREADY | 1 | I | Read ready |
| S\_AXI\_ARREADY | 1 | O | Read address ready |
| S\_AXI\_RDATA | 32 | O | Read data |
| S\_AXI\_RRESP | 2 | O | Read response |
| S\_AXI\_RVALID | 1 | O | Read valid |
| S\_AXI\_WREADY | 1 | O | Write ready |
| S\_AXI\_BRESP | 2 | O | Write response |
| S\_AXI\_BVALID | 1 | O | Write response valid |
| S\_AXI\_AWREADY | 1 | O | Write address ready |
| Axi4 Stream Input Bus lines | S\_AXIS\_TREADY | 1 | O | Ready to receive data |
| S\_AXIS\_TVALID | 1 | I | Read valid |
| S\_AXIS\_TDATA | 32 | I | Read data |
| S\_AXIS\_TLAST | 1 | O | Signal indicating the last data of the burst |
| Axi4 Stream Output Bus lines | M\_AXIS\_TREADY | 1 | I | Ready to receive data |
| M\_AXIS\_TVALID | 1 | O | Write valid |
| M\_AXIS\_TDATA | 32 | O | Write data |
| M\_AXIS\_TLAST | 1 | O | Signal indicating the last data of the burst |
| Input Spinnaker Link | Data\_2of7\_from\_spinnaker | 7 | I | Self-timed 2-of-7 protocol data input |
| Ack\_to\_spinnaker | 1 | O | Acknowledge signal to spinnaker chip |
| Input Spinnaker Link | Data\_2of7\_to\_spinnaker | 7 | O | Self-timed 2-of-7 protocol data output |
| Ack\_from\_spinnaker | 1 | I | Acknowledge signal from spinnaker chip |
|  | nRst | 1 | I | Active Low reset signal |
| Clk\_Spinn | 1 | I | Spinnaker clock |
| Clk\_Core | 1 | I | Core clock |
| Interrupt\_o | 1 | O | Interrupt active high signal for micro |
| LpbkDefault | 3 | I | Loopback configure [1] |

# Spin2Neu Diagram

The Figure 1 shows all the modules used in the Spin2Neu design. The Spin2Neu design integrates the following modules that will be explained into the respective subsections into the next chapter:

* Spinn\_Neu I/f,
* CoreMonSeq,
  + Monitor,
  + TimeStamper,
  + AEXSequencer,
  + InFIfo,
  + OutFifo
  + Timestamp wrap detector,
* AXI Stream I/f,
* AXI Lite REGs I/f.

CoreMonSeq

AXI 4 Lite I/f

AXI Lite REGs I/f

InFifo

OutFifo

AEXSequencer

Monitor

TimeStamper

TimeStamp Wrap  
detector

Spinn\_neu If

AXI Stream I/f

AXI Stream Ouptut bus I/f

AXI Stream Input bus I/f

Spinnaker Input Link

Spinnaker Output Link

7

7

Output AER I/f

Input AER I/f

Figure 1 Spin2Neu architecture

# Main components of the Spin2Neu architecture

In this Section we will describe the main components of the Spin2Neu architecture.

## Spinn\_neu I/f

This module (coming from the University of Manchester) is used to interface the *Spinnaker input link* and *Spinnaker Output link* with, respectively, the *output AER I/f* and the *input AER I/f*.

## CoreMonSeq

The *CoreMonSeq* module by using one Input Fifo and one Output Fifo can manage the two events data flows that are the two spinnaker links. Using an internal 32 bit counter, it can identify any data with the specific time stamp.

### Monitor

This module extract the data from the output AER I/f and feeds the InFifo with the data and relative time stamp.

### Timestamper

This module is a 32 bit counter used to identify the events coming from/to the spinnaker links.

### AEXSequencer

This module takes the data coming from the OutFifo and generates (depending on the timing information inside the couple Data/Timestamp of the OutFifo) the AER Input I/f of the *Spinn\_neu I/f* module.

### InFIfo

This Fifo is 2048 data of 64bit length. Both Fifo ports are 64 bits sized.

### OutFIfo

This Fifo is 2048 data of 64bit length. The input port is 32 bits sized and the output port is 64 bits sized.

### TimeStampWrap detector

This module is used to detect the wrapping of the *Timestamper*.

## AXIStream I/f

The *AXIStream I/f* interfaces the AXI stream output and input I/f with the FIFOs interfaces.

## AXI4Lite I/f

The AXI4Lite I/f module is used to interface the CPU with the internal Spin2Neu registers.

# Spin2Neu internal registers

In this Section a detailed view of the registers internal to the Spin2Neu module.

|  | **Offset** | **Mnemonic** | **Description** | **Type** | **Reset Value** |
| --- | --- | --- | --- | --- | --- |
|  | 0x00 | Control register (CTRL\_REG) | Control register | rw | 0x0?000000 |
|  | 0x08 | RX DATA buffer register (RXDATA\_REG) | RX Data Buffer | ro | 0x00000000 |
|  | 0x0C | RX Time buffer register (RXTIME\_REG) | RX Time Buffer | ro | 0x00000000 |
|  | 0x10 | TX Data buffer register (TXDATA\_REG) | TX Data Buffer | rw | 0x00000000 |
|  | 0x14 | DMA register (DMA\_REG) | DMA Burst Register | rw | 0x00000100 |
|  | 0x18 | RAW Status Register (STAT\_RAW\_REG) | Status RAW register | ro | 0x00000000 |
|  | 0x1C | IRQ Register (IRQ\_REG) | IRQ register | rc | 0x00000000 |
|  | 0x20 | Mask Register (MSK\_REG) | Mask register for the IRQ\_REG register | rw | 0x00000000 |
|  | 0x28 | Wrapping TimeStamp Register (WRAPTimeStamp\_REG) | Wrapping TimeStamp Register | rc | 0x00000000 |
|  | 0x5C | Identification register (ID\_REG) | ID Register | ro | 0x53324E10 |

## Control register (CTRL\_REG)

This register is used to control the Spin2Neu IP.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CTRL\_REG (Spin2Neu Base + 0x00)** | | | | | | | | Reset Value: **0x0?000000** | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | LocalFar LB | Local LB | Remote LB | Reserved | | | | | | | |
|  |  |  |  |  | ro | ro | ro |  |  |  |  |  |  |  |  |
|  | | | | | | | |  | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | Rearm DMA | Reserved | | | | | | | FlushFifos | Reserved | IE enable | EN DMA | Reserved |
|  |  |  | wc |  |  |  |  |  |  |  | wc |  | r/w | r/w |  |

* En DMA
  + If ‘1’ the DMA I/f is enabled
* IE Enable
  + If ‘1’ the Spin2Neu generates Interrupt depending on the unmasked bit of
* FlushFifo
  + If ‘1’ the internal Spin2Neu Fifos are flushed. It’s automatically cleared by the hardware.
* RearmDMA
  + If ‘1’ the DMA is re-armed. It’s automatically cleared by the hardware.
* Remote Loopback,
* Local Loopback and
* LocalFar Loopback
  + See **Error! Reference source not found.** for details.

## RX DATA buffer register (RXDATA\_REG)

This register contains the data (read from the InFifo) coming from the output AER I/f of the Spinn\_neu I/f module.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **RXDATA\_REG (Spin2Neu Base + 0x08)** | | | | | | | | Reset Value: **0x00000000** | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Data | | | | | | | | | | | | | | | |
| ro | | | | | | | | | | | | | | | |
|  | | | | | | | |  | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Data | | | | | | | | | | | | | | | |
| ro | | | | | | | | | | | | | | | |

* Data
  + 32 bit length data

## RX Time buffer register (RXTIME\_REG)

This register contains the timestamp associated to the received data (see RX DATA buffer register (RXDATA\_REG)) from the output AER I/f of the Spinn\_neu I/f module.

The time stamp value read from this register is the time stamp that the Spinn2Neu module sticks to the received Data available into the RX DATA buffer register (RXDATA\_REG).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **RXTIME\_REG (Spin2Neu Base + 0x0C)** | | | | | | | | Reset Value: **0x00000000** | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TimeStamp | | | | | | | | | | | | | | | |
| ro | | | | | | | | | | | | | | | |
|  | | | | | | | |  | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TimeStamp | | | | | | | | | | | | | | | |
| ro | | | | | | | | | | | | | | | |

* TimeStamp
  + 32 bit length data

## TX Data buffer register (TXDATA\_REG)

This register is used to fill the OutFifo.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TXDATA\_REG (Spin2Neu Base + 0x10)** | | | | | | | | Reset Value: **0x00000000** | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TXData | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
|  | | | | | | | |  | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXData | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

When writing to this register, keep in mind that it is used by the internal hardware as follows: The register needs to be written twice to enable the correct behaviour. The first data written into the register represents the time, elapsed which, the second written data into this register is delivered to the Spinn\_neu I/f module.

## DMA register (DMA\_REG)

This register is used to set the behaviour of the AXIStream interface. It represents the number of data (32 bit size length) sent to the DMA interface.

NOTE: This register can be written only if CTRL\_REG.ENDMA=’0’.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **DMA\_REG (Spin2Neu Base + 0x14)** | | | | | | | | Reset Value: **0x000001n00** | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
|  | | | | | | | |  | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | DMA Lenght | | | | | | | | | | |
|  |  |  |  |  | rw | | | | | | | | | | |

For example, if it is set to 8, the burst from/to the AXI Stream I/f will be in terms of 8 data length.

T1

T2

T3

T4

D4

D3

D2

D1

M\_axis\_tdata

M\_axis\_tready

M\_axis\_tvalid

M\_axis\_tlast

clock

## RAW Status Register (STAT\_RAW\_REG)

When read, this register gives a snapshot of the status of warning or errors signals. It is a Read Only register.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **STAT\_RAW\_REG (Spinn2Neu Base + 0x18)** | | | | | | | | Reset Value: **0x00000000** | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | | | | | | | | | |
|  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | |  | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | RxModemErr | RXParityErr | TxDumpMode | Reserved | | RX FIFONot Empty | RxBufferReady | TimeStamp Wrapped | Reserved | TX Data Full | TX Data Almost Full | TX Data Empty | RX Data Full | RX Data Almost Empty | RX Data Empty |
|  | ro | ro | ro |  |  | ro | ro | ro |  | ro | ro | ro | ro | ro | ro |

* RxDataEmpty
  + When ‘0’, the INFIFO is not empty
  + When ‘1’ the INFIFO is empty
* RxDataAlmostEmpty
  + When ‘1’ the INFIFO has 1 or 0 data to be read.
  + When ‘0’ the INFIFO has more or equal two data to be read.
* RxDataFull
  + When ‘1’ the INFIFO is full.
  + When ‘0’ the INFIFO is not full.
* TxDataEmpty
  + When ‘0’, the OUTFIFO is not empty
  + When ‘1’ the OUTFIFO is empty
* TxDataAlmostFull
  + When ‘1’ the OUTFIFO has 2047 or 2048 data within himself.
  + When ‘0’ the OUTFIFO has less than 2047 data within himself.
* TxDataFull
  + When ‘1’ the OUTFIFO is full.
  + When ‘0’ the OUTFIFO is not full.
* Time stamp wrapped (this bit is high for one clock period only, when the counter wraps its value)
  + When ‘1’ the counter inside the TimeStamp module has wrapped its value.
  + When ‘0’ the counter inside the TimeStamp module has not yet wrapped its value
* RXBufferReady
  + When ‘1’ the Rx Fifo has at least DMA\_REG value of data available
  + When ‘0’ the Rx Fifo has less than DMA\_REG value of data available
* RXFifoNotEmpty
  + When ‘1’ the RX Fifo is not empty.
  + When ‘0’ the RX Fifo is empty
* TxDumpMode
  + When ‘1’ the spinn\_neu I/f module through its in\_mapper module has highlighted a DumpMode event.
* RXParityErr
  + When ‘1’ the spinn\_neu I/f module through its out\_ module has highlighted a Parity error event.
* RxModeErr
  + When ‘1’ the spinn\_neu I/f module through its spinn\_receiver module has highlighted a Rx Mode error event.

## IRQ Register (IRQ\_REG)

When read, this register gives the status of the collected warning or errors signals. It is a Read/Set register, i.e., to clear the warning/error bit the user has to write ‘1’ on the corresponding bit position.

Irq\_Set\_0

MASK\_REG

Interrupt active high

EN\_INT   
(from CTRL\_REG)

RAW Interrupt contributors

IRQ\_REG

Irq\_Set\_25

STAT\_RAW\_REG

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **IRQ\_REG (Spinn2Neu Base + 0x1C)** | | | | | | | | Reset Value: **0x00000000** | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | |  | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | RxModemErr | RXParityErr | TxDumpMode | Reserved | | RX FIFONot Empty | RxBufferReady | TimeStamp Wrapped | Reserved | TX Data Full | TX Data Almost Full | TX Data Empty | RX Data Full | RX Data Almost Empty | RX Data Empty |
|  | r/c | r/c | r/c |  |  | r/c | r/c | r/c |  | r/c | r/c | r/c | r/c | r/c | r/c |

The meaning of the masked contributors of this register is the same of the RAW Status Register (STAT\_RAW\_REG).

## Mask Register (MSK\_REG)

This is the Mask register used to mask the contributors for the interrupt signal.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **MSK\_REG (Spinn2Neu Base + 0x20)** | | | | | | | | Reset Value: **0x00000000** | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | | | | | | | |  | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | RxModemErr | RXParityErr | TxDumpMode | Reserved | | RX FIFONot Empty | RxBufferReady | TimeStamp Wrapped | Reserved | TX Data Full | TX Data Almost Full | TX Data Empty | RX Data Full | RX Data Almost Empty | RX Data Empty |
|  | r/w | r/w | r/w |  |  | r/w | r/w | r/w |  | r/w | r/w | r/w | r/w | r/w | r/w |

The meaning of the masked contributors of this register is the same of the RAW Status Register (STAT\_RAW\_REG).

## Wrapping TimeStamp Register (WRAPTimeStamp\_REG)

This register is used to read how many times the internal 32bit counter of the TimeStamp module has wrapped its value.

In case the user writes any value in this register, it will be cleared and also the internal 32bit counter of the TimeStamp module will be cleared.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **WRAPTIMESTAMP\_ REG (Spinn2Neu Base + 0x28)** | | | | | | | | Reset Value: **0x00000000** | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Wrapping times | | | | | | | | | | | | | | | |
| r/c | | | | | | | | | | | | | | | |
|  | | | | | | | |  | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Wrapping times | | | | | | | | | | | | | | | |
| r/c | | | | | | | | | | | | | | | |

## Identification register (ID\_REG)

This register contains the ID of the Spinn2Neu module.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID\_REG (Spin2Neu Base + 0x5C)** | | | | | | | | Reset Value: **0x53324E10** | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| S | | | | | | | | 2 | | | | | | | |
| r/o | | | | | | | | r/o | | | | | | | |
|  | | | | | | | |  | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| N | | | | | | | | Major | | | | Minor | | | |
| r/o | | | | | | | | r/o | | | | r/o | | | |

Minor = 1;

Major = 0;

# References

1. “Application Notes for Spin2Neu”, by F. Diotalevi, May 2016.

# Appendixes