Squid Design Specifications

Revision 1.1

Assaf Gilboa and Yisrael Shulman

Table of Contents

[1. Glossary 3](#_Toc373223009)

[2. Abstract 3](#_Toc373223010)

[3. Feature list 4](#_Toc373223011)

[4. IO's and PIN assignment 6](#_Toc373223012)

[5. Squid blocks 9](#_Toc373223013)

[a. SPI Slave Specification 9](#_Toc373223014)

[I. SPI protocol 9](#_Toc373223015)

[II. SPI Low level Operation 10](#_Toc373223016)

[III. SPI High level protocol 10](#_Toc373223017)

[b. Address decoder 11](#_Toc373223018)

[c. Serial Queue 12](#_Toc373223019)

[I. Serial packet general bits usage 12](#_Toc373223020)

[d. Packet Formatter 13](#_Toc373223021)

[e. Loop Management 13](#_Toc373223022)

[I. Bypass 13](#_Toc373223023)

[II. Error Detection 13](#_Toc373223024)

[f. Service routine - TBD 14](#_Toc373223025)

[g. Interrupt service 14](#_Toc373223026)

[h. Register set 15](#_Toc373223027)

[I. REVISION 15](#_Toc373223028)

[II. STATUS 16](#_Toc373223029)

[III. ISR 16](#_Toc373223030)

[IV. IMR 16](#_Toc373223031)

[V. ICR 17](#_Toc373223032)

[VI. ITR 17](#_Toc373223033)

[VII. PONG 17](#_Toc373223034)

[VIII. LOOP RESET 18](#_Toc373223035)

[IX. LOOP BYPASS 18](#_Toc373223036)

[X. LOOP\_TIMEOUT 18](#_Toc373223037)

[XI. SCRATCHPAD 18](#_Toc373223038)

[XII. COMMAND 19](#_Toc373223039)

[XIII. QUEUE STATUS 19](#_Toc373223040)

[XIV. SERVICE\_STATUS 19](#_Toc373223041)

[XV. SR\_CONTROL 20](#_Toc373223042)

[XVI. SR\_MEMORY 20](#_Toc373223043)

[XVII. SERIAL\_WRITE 20](#_Toc373223044)

[XVIII. SERIAL\_READ 21](#_Toc373223045)

[XIX. SERVICE\_READ 21](#_Toc373223046)

[Figure 1 - Squid system view 4](#_Toc373223047)

[Figure 2 - Squid block diagram 9](#_Toc373223048)

[Figure 3 - SPI protocol 10](#_Toc373223049)

[Figure 4 - Squid transaction 11](#_Toc373223050)

[Figure 5 - Serial Packet general bit 12](#_Toc373223051)

[Figure 6 - Loop logic 14](#_Toc373223052)

# Glossary

***Serial Loop*** – The serial loop that connects 8 Hammers in serial protocol and the Squid. The serial loop has serial\_in, serial\_out, serial\_rstn, serial\_clk\_0 and serial\_clk\_1

***Serial Chain*** – The serial chain is the serial structure which is built by concatenation of all active serial loop and at its maximum is 24 ***Serial Loop.*** The serial chain starts at dispatch point for serial packet and end at Squid as final stop

***SPI*** – Serial Peripheral Interface standard the-facto by Motorola for synchronous serial data link

# Abstract

The Squid is bridging FPGA designed to connect a single CPU with multiple ASICs.

The Squid target FPGA is Altera Cyclone 4: 15K logic elements, 343 user-IOs part# EP4CE15F23C8N package F484 (next pin compatible Altera is 30K).

Utilization target: release with less than 70%



Figure 1 - Squid system view

# Feature list

* SW interface
  + SPI Slave IF - CPOL = 0 CPHA = 0
  + “Squid” serial protocol
    - Squid header
      * Read/ write
      * Squid address
      * Length in 32b
    - Payload
      * Squid register value 32bit
      * Serial transaction packet/s per length
  + Squid register set access
  + Interrupt mechanism
  + 4 GPIO In/Out – Default as Input
  + Serial SW Queue
    - TX Queue for serial packet send
    - RX Queue for serial packet returning to SW (ex. ASIC reads)
    - Can support conditional queuing
* Service routines to reduce SW polling overhead
  + HW periodic polling on 16 preconfigured serial packets
  + Each service routine can generate Interrupt/ status according to response pre
  + Configurable period between HW polling according to desired time (HW timer)
* Chain management – 24 serial loops
  + ASIC serial protocol compliancy
  + Single chain ( 24 serial loops of 8 ASICs -> 192 ASICs)
  + Active loop configuration
    - Any serial chain can be bypassed to handle faulty chain or for address allocation purposes
    - Chain can be reset independently
  + 10K Job per second maximal rate
  + Timeout to detect failing transactions
* Debug
  + HW debug I/F 10 bit bus
  + I2C Slave IF – TBD where to hook it to?

# IO's and PIN assignment

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Pin Name | Direction | Description | FPGA PIN | Standart | Notes |
| GPIO[0] | Inout |  | PIN\_J1 | 3.3-V LVCMOS |  |
| GPIO[1] | Inout |  | PIN\_H6 | 3.3-V LVCMOS |  |
| GPIO[3] | Inout |  | PIN\_H2 | 3.3-V LVCMOS |  |
| GPIO[2] | Inout |  | PIN\_H5 | 3.3-V LVCMOS |  |
| dbg\_bus[0] | Out |  | PIN\_M1 | 3.3-V LVCMOS |  |
| dbg\_bus[9] | Out |  | PIN\_R1 | 3.3-V LVCMOS |  |
| dbg\_bus[1] | Out |  | PIN\_V1 | 3.3-V LVCMOS |  |
| dbg\_bus[2] | Out |  | PIN\_N2 | 3.3-V LVCMOS |  |
| dbg\_bus[3] | Out |  | PIN\_V2 | 3.3-V LVCMOS |  |
| dbg\_bus[4] | Out |  | PIN\_P2 | 3.3-V LVCMOS |  |
| dbg\_bus[5] | Out |  | PIN\_U1 | 3.3-V LVCMOS |  |
| dbg\_bus[6] | Out |  | PIN\_P1 | 3.3-V LVCMOS |  |
| dbg\_bus[7] | Out |  | PIN\_U2 | 3.3-V LVCMOS |  |
| dbg\_bus[8] | Out |  | PIN\_R2 | 3.3-V LVCMOS |  |
| i2c\_scl | Inout |  | PIN\_G5 | 3.3-V LVCMOS |  |
| i2c\_sda | Inout |  | PIN\_G4 | 3.3-V LVCMOS |  |
| int\_squid | Out |  | PIN\_B1 | 3.3-V LVCMOS |  |
| IN\_REFCLK | In |  | PIN\_G1 | 3.3-V LVCMOS |  |
| ser\_clk0[0] | Out |  | PIN\_AA21 | 1.8 V |  |
| ser\_clk0[9] | Out |  | PIN\_AA7 | 1.8 V |  |
| ser\_clk0[10] | Out |  | PIN\_AB5 | 1.8 V |  |
| ser\_clk0[11] | Out |  | PIN\_V11 | 1.8 V |  |
| ser\_clk0[12] | Out |  | PIN\_W10 | 1.8 V |  |
| ser\_clk0[13] | Out |  | PIN\_Y3 | 1.8 V |  |
| ser\_clk0[14] | Out |  | PIN\_AB19 | 1.8 V |  |
| ser\_clk0[15] | Out |  | PIN\_V12 | 1.8 V |  |
| ser\_clk0[16] | Out |  | PIN\_AA16 | 1.8 V |  |
| ser\_clk0[17] | Out |  | PIN\_V14 | 1.8 V |  |
| ser\_clk0[18] | Out |  | PIN\_W17 | 1.8 V |  |
| ser\_clk0[1] | Out |  | PIN\_N19 | 1.8 V |  |
| ser\_clk0[19] | Out |  | PIN\_B19 | 1.8 V |  |
| ser\_clk0[20] | Out |  | PIN\_B16 | 1.8 V |  |
| ser\_clk0[21] | Out |  | PIN\_D19 | 1.8 V |  |
| ser\_clk0[22] | Out |  | PIN\_F12 | 1.8 V |  |
| ser\_clk0[23] | Out |  | PIN\_A4 | 1.8 V |  |
| ser\_clk0[2] | Out |  | PIN\_R20 | 1.8 V |  |
| ser\_clk0[3] | Out |  | PIN\_U21 | 1.8 V |  |
| ser\_clk0[4] | Out |  | PIN\_W20 | 1.8 V |  |
| ser\_clk0[5] | Out |  | PIN\_C22 | 1.8 V |  |
| ser\_clk0[6] | Out |  | PIN\_F19 | 1.8 V |  |
| ser\_clk0[7] | Out |  | PIN\_H21 | 1.8 V |  |
| ser\_clk0[8] | Out |  | PIN\_AA3 | 1.8 V |  |
| ser\_clk1[0] | Out |  | PIN\_M19 | 1.8 V |  |
| ser\_clk1[9] | Out |  | PIN\_AA8 | 1.8 V |  |
| ser\_clk1[10] | Out |  | PIN\_AB7 | 1.8 V |  |
| ser\_clk1[11] | Out |  | PIN\_V5 | 1.8 V |  |
| ser\_clk1[12] | Out |  | PIN\_W6 | 1.8 V |  |
| ser\_clk1[13] | Out |  | PIN\_Y6 | 1.8 V |  |
| ser\_clk1[14] | Out |  | PIN\_AA13 | 1.8 V |  |
| ser\_clk1[15] | Out |  | PIN\_W13 | 1.8 V |  |
| ser\_clk1[16] | Out |  | PIN\_AB16 | 1.8 V |  |
| ser\_clk1[17] | Out |  | PIN\_V15 | 1.8 V |  |
| ser\_clk1[18] | Out |  | PIN\_Y17 | 1.8 V |  |
| ser\_clk1[1] | Out |  | PIN\_N20 | 1.8 V |  |
| ser\_clk1[19] | Out |  | PIN\_A19 | 1.8 V |  |
| ser\_clk1[20] | Out |  | PIN\_B20 | 1.8 V |  |
| ser\_clk1[21] | Out |  | PIN\_E12 | 1.8 V |  |
| ser\_clk1[22] | Out |  | PIN\_F14 | 1.8 V |  |
| ser\_clk1[23] | Out |  | PIN\_B5 | 1.8 V |  |
| ser\_clk1[2] | Out |  | PIN\_R21 | 1.8 V |  |
| ser\_clk1[3] | Out |  | PIN\_U22 | 1.8 V |  |
| ser\_clk1[4] | Out |  | PIN\_W21 | 1.8 V |  |
| ser\_clk1[5] | Out |  | PIN\_D20 | 1.8 V |  |
| ser\_clk1[6] | Out |  | PIN\_F21 | 1.8 V |  |
| ser\_clk1[7] | Out |  | PIN\_H22 | 1.8 V |  |
| ser\_clk1[8] | Out |  | PIN\_AB3 | 1.8 V |  |
| ser\_out[0] | In |  | PIN\_M21 | 1.8 V |  |
| ser\_out[9] | In |  | PIN\_AB10 | 1.8 V |  |
| ser\_out[10] | In |  | PIN\_AB9 | 1.8 V |  |
| ser\_out[11] | In |  | PIN\_V7 | 1.8 V |  |
| ser\_out[12] | In |  | PIN\_W8 | 1.8 V |  |
| ser\_out[13] | In |  | PIN\_Y8 | 1.8 V |  |
| ser\_out[14] | In |  | PIN\_AA14 | 1.8 V |  |
| ser\_out[15] | In |  | PIN\_AA15 | 1.8 V |  |
| ser\_out[16] | In |  | PIN\_AB17 | 1.8 V |  |
| ser\_out[17] | In |  | PIN\_AB18 | 1.8 V |  |
| ser\_out[18] | In |  | PIN\_AB20 | 1.8 V |  |
| ser\_out[1] | In |  | PIN\_P22 | 1.8 V |  |
| ser\_out[19] | In |  | PIN\_A18 | 1.8 V |  |
| ser\_out[20] | In |  | PIN\_C19 | 1.8 V |  |
| ser\_out[21] | In |  | PIN\_E15 | 1.8 V |  |
| ser\_out[22] | In |  | PIN\_F16 | 1.8 V |  |
| ser\_out[23] | In |  | PIN\_C3 | 1.8 V |  |
| ser\_out[2] | In |  | PIN\_U19 | 1.8 V |  |
| ser\_out[3] | In |  | PIN\_V22 | 1.8 V |  |
| ser\_out[4] | In |  | PIN\_Y21 | 1.8 V |  |
| ser\_out[5] | In |  | PIN\_D22 | 1.8 V |  |
| ser\_out[6] | In |  | PIN\_H19 | 1.8 V |  |
| ser\_out[7] | In |  | PIN\_J22 | 1.8 V |  |
| ser\_out[8] | In |  | PIN\_AA4 | 1.8 V |  |
| ser\_in[0] | Out |  | PIN\_M20 | 1.8 V |  |
| ser\_in[1] | Out |  | PIN\_P21 | 1.8 V |  |
| ser\_in[9] | Out |  | PIN\_AA9 | 1.8 V |  |
| ser\_in[10] | Out |  | PIN\_AB8 | 1.8 V |  |
| ser\_in[11] | Out |  | PIN\_V6 | 1.8 V |  |
| ser\_in[12] | Out |  | PIN\_W7 | 1.8 V |  |
| ser\_in[13] | Out |  | PIN\_Y7 | 1.8 V |  |
| ser\_in[14] | Out |  | PIN\_AB13 | 1.8 V |  |
| ser\_in[15] | Out |  | PIN\_Y13 | 1.8 V |  |
| ser\_in[16] | Out |  | PIN\_AA17 | 1.8 V |  |
| ser\_in[17] | Out |  | PIN\_W15 | 1.8 V |  |
| ser\_in[18] | Out |  | PIN\_AA20 | 1.8 V |  |
| ser\_in[19] | Out |  | PIN\_A16 | 1.8 V |  |
| ser\_in[20] | Out |  | PIN\_C17 | 1.8 V |  |
| ser\_in[21] | Out |  | PIN\_E13 | 1.8 V |  |
| ser\_in[22] | Out |  | PIN\_F15 | 1.8 V |  |
| ser\_in[23] | Out |  | PIN\_C10 | 1.8 V |  |
| ser\_in[2] | Out |  | PIN\_R22 | 1.8 V |  |
| ser\_in[3] | Out |  | PIN\_V21 | 1.8 V |  |
| ser\_in[4] | Out |  | PIN\_W22 | 1.8 V |  |
| ser\_in[5] | Out |  | PIN\_D21 | 1.8 V |  |
| ser\_in[6] | Out |  | PIN\_F22 | 1.8 V |  |
| ser\_in[7] | Out |  | PIN\_J21 | 1.8 V |  |
| ser\_in[8] | Out |  | PIN\_AA10 | 1.8 V |  |
| ser\_rstn[0] | Out |  | PIN\_M22 | 1.8 V |  |
| ser\_rstn[9] | Out |  | PIN\_AB4 | 1.8 V |  |
| ser\_rstn[10] | Out |  | PIN\_V10 | 1.8 V |  |
| ser\_rstn[11] | Out |  | PIN\_V8 | 1.8 V |  |
| ser\_rstn[12] | Out |  | PIN\_Y10 | 1.8 V |  |
| ser\_rstn[13] | Out |  | PIN\_AA19 | 1.8 V |  |
| ser\_rstn[14] | Out |  | PIN\_AB14 | 1.8 V |  |
| ser\_rstn[15] | Out |  | PIN\_AB15 | 1.8 V |  |
| ser\_rstn[16] | Out |  | PIN\_V13 | 1.8 V |  |
| ser\_rstn[17] | Out |  | PIN\_AA18 | 1.8 V |  |
| ser\_rstn[18] | Out |  | PIN\_U15 | 1.8 V |  |
| ser\_rstn[1] | Out |  | PIN\_R19 | 1.8 V |  |
| ser\_rstn[19] | Out |  | PIN\_A20 | 1.8 V |  |
| ser\_rstn[20] | Out |  | PIN\_D15 | 1.8 V |  |
| ser\_rstn[21] | Out |  | PIN\_E16 | 1.8 V |  |
| ser\_rstn[22] | Out |  | PIN\_A10 | 1.8 V |  |
| ser\_rstn[23] | Out |  | PIN\_D10 | 1.8 V |  |
| ser\_rstn[2] | Out |  | PIN\_U20 | 1.8 V |  |
| ser\_rstn[3] | Out |  | PIN\_W19 | 1.8 V |  |
| ser\_rstn[4] | Out |  | PIN\_Y22 | 1.8 V |  |
| ser\_rstn[5] | Out |  | PIN\_F17 | 1.8 V |  |
| ser\_rstn[6] | Out |  | PIN\_H20 | 1.8 V |  |
| ser\_rstn[7] | Out |  | PIN\_K15 | 1.8 V |  |
| ser\_rstn[8] | Out |  | PIN\_AA5 | 1.8 V |  |
| spi\_clk | In |  | PIN\_E1 | 3.3-V LVCMOS |  |
| spi\_cs | In |  | PIN\_F2 | 3.3-V LVCMOS |  |
| spi\_mosi | In |  | PIN\_E3 | 3.3-V LVCMOS |  |
| spi\_miso | Out |  | PIN\_F1 | 3.3-V LVCMOS |  |
| uart\_rxd | In |  | PIN\_C2 | 3.3-V LVCMOS | Removed |
| uart\_txd | Out |  | PIN\_C1 | 3.3-V LVCMOS | Removed |

# Squid blocks



Figure 2 - Squid block diagram

## SPI Slave Specification

### SPI protocol

The SPI mode that in use in the Squid is CPOL = 0 CPHA = 0

The SPI works is also supporting continuous mode of operation where the transaction is longer than a single byte.

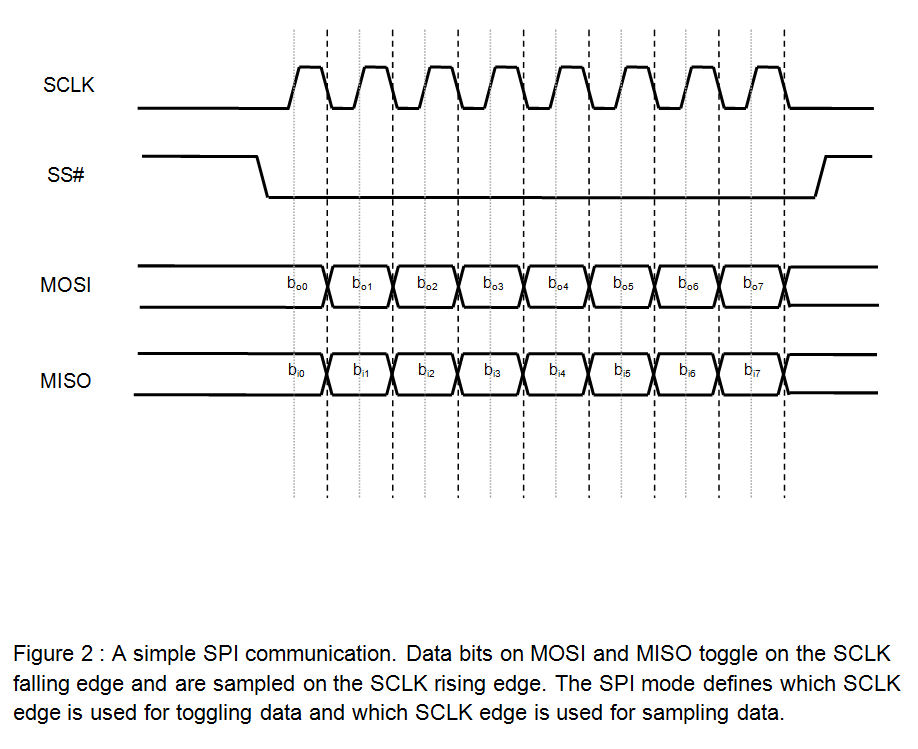


Figure 3 - SPI protocol

### SPI Low level Operation

The SPI I/F is implementing the SPI shift registers. Each sample bit is shifted till 8 shifts are done and entire byte is latched into the next block. Each 8 bits the SPI I/F is latching a ready byte into transmission.

The SPI signals are synchronized and sampled in rate which is 8x the SPI data rate

### SPI High level protocol

Squid incoming transaction structure (multiple SPI byte transactions)

|  |  |  |  |
| --- | --- | --- | --- |
| Address[7:0] | CMD[1:0] | Length[5:0] | Data [length\*32-1:0] |
| Squid address | 1 – Write  0 – Read | Payload Length in Words (32b) | Payload |

Squid outgoing transaction structure (multiple SPI byte transactions)

|  |  |  |
| --- | --- | --- |
| Garbage[7:0] | Address[7:0] | Data [length\*32-1:0] |
| Don’t care | Incoming Squid address field | Payload:  Read – read register’s value  Write – incoming reflected write payload |

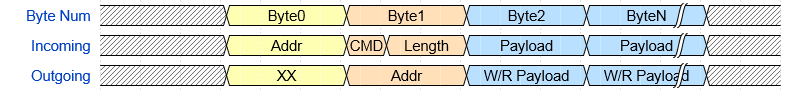


Figure 4 - Squid transaction

The Squid supports read and write transactions. The CPU is issuing a master “incoming Squid transaction” and **in parallel** the Squid is responding with outgoing transaction (see Figure #4).

When master is issuing WRITE, the Squid is reflecting at the response the written payload.

When master is issuing READ, the Squid is asserting the desired value in the “Payload” field while master “Payload” field is ignored (serves only as place holder)

The first outgoing byte is “Don’t care” since the Squid is decoding the Address at this stage.

## Address decoder

The Squid distinguish several types of addresses.

1. Register file - “Simple” configuration 32 bit wide registers
2. Service routine Memory which contains “Hammer Serial” service routine written by SW by address
3. Serial TX FIFO address which serves to store “Hammer’s serial” packets which destined to the Hammer chain. Each write is pushing the 32bit word into the Serial FIFO of transmit (no read access).
4. Serial RX FIFO address which serves to store “Hammer’s serial” packets which returned from the Hammer chain and are marked to be read by SW. Each SW read will pop a 32bit word from the FIFO to the SPI (no write access).
5. Service RX FIFO address which serves to store “Hammer’s serial” packets which returned from the Hammer chain, are marked to be read by SW and originated from the service routine. Each SW read will pop a 32bit word from the FIFO to the SPI (no write access).

|  |  |
| --- | --- |
| Address range | Destination |
| 0x00 – 0x3F | Register file |
| 0x40 – 0x7F | Service routine Memory |
| 0x80 | Serial TX FIFO |
| 0x81 | Serial RX FIFO |
| 0x82 | Service RX FIFO |

Table 1 - Address decoding

## Serial Queue

The serial Queue is allowing the SW to write continuously into the “Hammer serial” as long as there is an available space in the FIFO.

The TX and RX FIFOs will hold enough space to contain at least 2 full Jobs – 64x32bits words.

The Serial Queue will hold a SW readable status of the both TX Queue and RX Queue and will allow interrupts over full/empty state of the FIFO.

While the TX queue is filled by the SW, the returning RX Queue is filled by the HW. Each “Hammer Serial” transaction sent is returning to the Squid and the Squid parse the “Hammer serial” packet and acts according to the “general” bits at the tail of the packet.

The general bits provide the additional info from which the Squid is routing the “Hammer serial” returning packet to the right Queue or discard it all together

### Serial packet general bits usage

* Completion bit by SW (C) – 1 - Store the response if asserted (according to Condition if set). 0 – Discard the returning packet
* Conditional bit by SW (CC) – 1 - Test the “Hammer serial” value to be non-zero. If the value field 16lsb is non-zero and Completion bit asserted – store the packet. 0 – Don’t test the value, act by Completion bit only.
* RX Q destination (Q) – Set by HW (SW shouldn’t use this bit) to designate whether the packet arrived from service or serial Q and to store it at the right RX Q
* HW test (CNT) – bits set by HW to detect “Hammer serial” failures

|  |  |  |
| --- | --- | --- |
| **C bit** | **CC bit** | **Action** |
| 0 | 0 | No response for the serial packet |
| 0 | 1 | No response for the serial packet |
| 1 | 0 | Store serial packet in Q |
| 1 | 1 | Store serial packet in Q if “Value[15:0]” field is non-zero |

Table 2 – General bit Completion and Conditional Completion Action

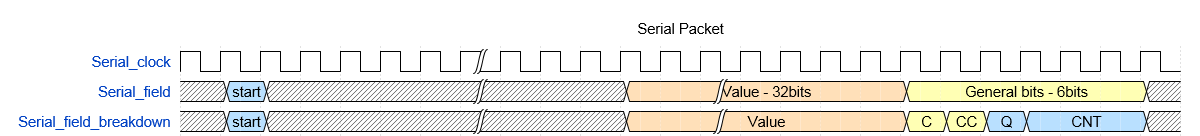


Figure 5 - Serial Packet general bit

## Packet Formatter

Packet Formatter will receive SPI bytes and format these bytes to 32bits transactions according to header. The packet formatter also toggle the bytes of the response to the direction of the SPI.

## Loop Management

Each Serial loop can route Serial\_in through the Hammer loop or simply short the loop and send Serial\_in to serial\_out directly. The loop manager is running 10MHz like the Hammer serial. The manager also provide serial clock to the Hammer serial

### Bypass

The Squid is enabling bypass per serial loop.

In order to bypass a loop:

1. Verify there is no serial packet in the chain:
   1. Stop the service routine from injecting packets
   2. Serial Q status should be empty
   3. Packets in the chain indication is zero
   4. Write the bypass register with the desired value
2. The loop which bypass is set will pass the serial packet and maintain the loop’s serial\_out in zero

### Error Detection

The serial chain logic is responsible to detect and report serial errors.

Since a packet is traversing a board and ~200 Hammer chips, there are a lot of failure points along the chain.

Types of errors:

1. Packet corruption – the packet is damaged by mal-operating component
2. Packet is lost – disconnect of serial somewhere

Detection:

1. Packet corruption
   1. Need to be end-to-end since Hammers can change data (parity, CRC will not work properly). The protection is per **Chain**
   2. Packet based since the packets are cut-thru
   3. Each serial packet entering will be marked with up-counting counter at its CNT general bits (see figure - Serial Packet general bit)
   4. Each serial packet exiting is expecting the “right” marking on the CNT bits
   5. Corrupted packets will have the marking wrong
   6. This marking will protect from serial packet shifts and systematic corruption.
   7. It will not protect from well-formed packets which interior data is bad
2. Packet Loss
   1. Packets which doesn’t return will not be detected by corruption since disappeared
   2. Every packet that goes into a loop MUST come out at the end of loop
   3. A loop is 8 Hammers long and a packet head is exiting a loop before tail has entered
   4. For each “start bit” on serial out trigger a timer, if start bit is not arriving on serial in after TBD configured cycles -> Error
   5. If arrived, reset trigger and wait for next “start bit”



Figure 6 - Loop logic

## Service routine - TBD

The Service routine is composed of 16 routines with 64 Words memory which hold Raw SPI protocol data with CMD, Address Etc. The service routine block will pump the data stored in his memory in predefined intervals. The certain service routine is enabled when the service routine are globally enabled as well as per routine enabled. The service routine are taking lower priority than the Serial Queue, and will be entering the serial only if the Serial Queue is idle. The service routine can be globally stopped to allow the SW to “drain” the Hammer serial in cases that reset or bypass are required.

## Interrupt service

The Squid has a dedicated line of interrupt towards the ARM. The Interrupt will be asserted when ISR register will hold a non-zero value.

The ISR is a register that holds vector of interrupts bitwise corresponding the STATUS register i.e. STATUS[0] will set ISR[0] if conditions are met.

ISR[i] asserted when:

STATUS[i] & IMR[i] asserted or ISR[i] & ITR[i] & ~ICR[i] & IMR[i]

The first (red) condition is referring to unmasked level events happening

The second (blue) condition is for temporary events that need capture, these events once captured are asserted till cleared.

Please refer to ITR register for sticky/ non-sticky type and ICR register for clear operation.

## Register set

|  |  |  |
| --- | --- | --- |
| **Register name** | **Address** | **Description** |
| [REVISION](#_REVISION) | 0x0 | Squid RTL revision |
| [STATUS](#_STATUS) | 0x1 | Squid status and error indications |
| [ISR](#_ISR) | 0x2 | Squid interrupt status register |
| [IMR](#_IMR) | 0x3 | Squid interrupt mask register |
| [ICR](#_ICR) | 0x4 | Squid interrupt clear register |
| [ITR](#_ITR) | 0x5 | Squid interrupt type register |
| [PONG](#_PONG) | 0xF | Squid fixed value register for “SPI health” read |
| [LOOP\_RESET](#_LOOP_RESET) | 0x10 | Squid loop reset vector |
| [LOOP\_BYPASS](#_LOOP_BYPASS) | 0x11 | Squid loop bypass vector |
| [LOOP\_TIMEOUT](#_LOOP_TIMEOUT) | 0x12 | Squid loop error of disconnection |
| [SCRATCHPAD](#_SCRATCHPAD) | 0x1F | Squid scratchpad register for write/read |
| [COMMAND](#_COMMAND) | 0x20 | Squid pulse command |
| [QUEUE\_STATUS](#_QUEUE_STATUS) | 0x21 | Squid Queue status |
| [SERVICE\_STATUS](#_SERVICE_STATUS) | 0x22 | Squid Service Routine status |
| [SR\_CONTROL](#_SR_CONTROL) | 0x30-0x3F | Squid Service routines control – 16 routines |
| [SR\_MEMORY](#_SR_MEMORY) | 0x40–0x5F | Squid “Serial packets” for service routine 2 per routine |
| [SERIAL\_WRITE](#_SERIAL_WRITE) | 0x80 | Squid serial queue FIFO write address |
| [SERIAL\_READ](#_SERIAL_READ) | 0x81 | Squid serial queue FIFO read address |
| [SERVICE\_READ](#_SERVICE_READ) | 0x82 | Squid service routine FIFO read address |

Table 3 – Squid Register Set

### REVISION

The register holds read only

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Default** | **Range** | **Description** |
| REVISION | RO |  | 15:0 | holds the RTL revision of the Squid |

### STATUS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Default** | **Range** | **Description** |
| **// Serial Queue indications** |  |  |  |  |
| Serial Q TX full | RO |  | 0 | The Serial TX FIFO is full |
| Serial Q TX not empty | RO |  | 1 | The Serial TX FIFO is not empty |
| Serial Q TX empty | RO |  | 2 | The Serial TX FIFO is empty |
| Serial Q RX full | RO |  | 3 | The Serial RX FIFO is full |
| Serial Q RX not empty | RO |  | 4 | The Serial RX FIFO is not empty |
| Serial Q RX empty | RO |  | 5 | The Serial RX FIFO is empty |
| **// Service Queue indications** |  |  |  |  |
| Service Q RX full | RO |  | 6 | The Service RX FIFO is full |
| Service Q RX not empty | RO |  | 7 | The Service RX FIFO is not empty |
| Service Q RX empty | RO |  | 8 | The Service RX FIFO is empty |
| **// Queue Errors** |  |  |  |  |
| FIFO Serial TX Error | RO |  | 9 | Serial TX FIFO underrun or overrun |
| FIFO Serial RX Error | RO |  | 10 | Serial RX FIFO underrun or overrun |
| FIFO Service RX Error | RO |  | 11 | Service RX FIFO underrun or overrun |
| **// Loop Indication** |  |  |  |  |
| Chain Empty | RO |  | 12 | The serial loops are empty – no packet is traversing the serial chain |
| Loop Timeout Error | RO |  | 13 | The loop has detected timeout on packet return see LOOP\_TIMEOUT |
| Loop corruption Error | RO |  | 14 | The serial loop has returned a malformed serial packet |
| Illegal access | RO |  | 15 | The SW has accessed a non-existing address |

### ISR

Interrupt Status Register holds the active interrupts of the Squid.

The register is Read Only, Width as STATUS reg.

The register when not zeroed is asserting the interrupt to CPU

Upon Interrupt the SW needs to read and this register.

The ISR bits’ offset is corresponding to the STATUS bit offset (interrupt is available per STATUS bit).

### IMR

Interrupt mask register. Each bit affects the bit order corresponding ISR bit.

The register is of Read-Write type , Width as STATUS reg.

1 - The interrupt will be mask

0 – The interrupt is enabled

Default value is 1 – No interrupts are enabled after reset

### ICR

Interrupt clear register. Each bit affects the bit order corresponding ISR bit.

The register is of Write-Only type (Read will return illegal value), Width as STATUS reg

The Clear act as a clearing pulse to sticky interrupts.

1 – Clear interrupt

0 – Don’t clear

Using the Clear on non-sticky interrupts will have no effect on interrupt

### ITR

Interrupt type register. Each bit affects the bit order corresponding ISR bit.

The register is of Write-Read, Width as STATUS reg.

The type can be Sticky or non-sticky.

Sticky interrupt is keeping interrupt asserted even if the event has passed ex. Pulse.

The ITR register will be set to the right default value and need not be set by SW (debug only).

By default:

Levels of FIFO (full/empty/non-empty) are non-sticky and shows current state.

Errors are sticky still collected by error pulses and are kept (if not masked) till cleared

### PONG

The register holds read only value.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Default** | **Range** | **Description** |
| PONG | RO | 0xDEADBEEF | 31:0 | holds the PONG value for SW polling |

The register is for SPI keep alive purposes.

Since the SPI is vital communication channel the SW polls periodically for “health check” of the SPI. The register serves as a fixed SW agreed value that once SW read is mismatched detect SPI hang/ bug.

SPI hang is fatal bug and require Squid reset

### LOOP RESET

The loop reset is a vector of reset to the Serial Loop. The reset is directly connected to the Squid SERIAL\_RSTN.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Default** | **Range** | **Description** |
| SERIAL\_RSTN | WR | 0x0 | 23:0 | Each bit is resetting a loop, 0 – reset is active; 1 – reset is inactive. |

### LOOP BYPASS

The loop bypass is a register that enable bypassing each loop (see “loop management”).

Note: changes are allowed only when no serial activity or pending packets exist

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Default** | **Range** | **Description** |
| LOOP BYPASS | WR | 0x0 | 23:0 | Each bit is bypassing a loop, 0 – don’t bypass; 1 – bypass loop insert zeroes to the loop. |

### LOOP\_TIMEOUT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Default** | **Range** | **Description** |
| LOOP TIMEOUT | WR | 0x0 | 23:0 | Bit per loop, 0 – no timeout occur; 1 – timeout occurred. |

The LOOP\_TIMEOUT holds per loop the event of packet loss in the loop. The loop is detecting “start\_bit” and expects to see a “start\_bit” returning. If the “start\_bit” is not returning throughout the packet (64 cycles), the loop will assert a “loop timeout” which designates a disconnection in the loop.

If “LOOP\_TIMEOUT” is non-zero the “STATUS. Loop Timeout Error” will be asserted.

Writing zero to this register will de-assert the “STATUS. Loop Timeout Error”. The register is written by HW when detecting a timeout and by SW clearing a timeout.

### SCRATCHPAD

A register for read write operations, has no design impact. For bring-up/ SW purposes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Default** | **Range** | **Description** |
| SCRATCHPAD | WR | 0x0 | 31:0 | Has no HW functionality |

### COMMAND

The Command register holds operation modes and reset ability to the Squid

Note: changes are allowed only when no serial activity or pending packets exist

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Default** | **Range** | **Description** |
| Serial Q TX FIFO reset | WR | 0x0 | 0 | Reset FIFO to empty state, data stored will be lost |
| Serial Q RX FIFO reset | WR | 0x0 | 1 | Reset FIFO to empty state, data stored will be lost |
| Service Q RX FIFO reset | WR | 0x0 | 2 | Reset FIFO to empty state, data stored will be lost |
| Loop logic reset | WR | 0x0 | 3 | Reset serial I/F logic (see detailed) |
| Service routine enable | WR | 0x0 | 4 | Enable the service routine feature |

**Loop logic reset** – **the reset should be done only in extreme cases** where the serial I/F is in error state (for debug purposes). The logic under this reset is the insertion of packets to the serial I/F, the bit aggregation of serial packets coming from the serial I/F and the service routine logic. Resetting the loop logic can cause Hammers to receive incomplete packets.

### QUEUE STATUS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Default** | **Range** | **Description** |
| Serial TX FIFO Used | RO |  | 7:0 | The number of occupied 32bits lines |
| Serial RX FIFO Used | RO |  | 15:8 | The number of occupied 32bits lines |
| Service RX FIFO Used | RO |  | 23:16 | The number of occupied 32bits lines |

### SERVICE\_STATUS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Default** | **Range** | **Description** |
| SERVICE\_STATUS | RO |  | 15:0 | Pending Service Routine |

The register states whether service routine is pending handling or not. The trivial case is that the SERVICE\_STATUS will be zeroes. The register is required to assure when “loop actions” such as bypass and reset are needed, the service routine is not pending completion. When a service routine is pending, need to wait to its completion.

1. Pending Request
2. No Pending request

### SR\_CONTROL

The register is containing the required timer that is used to create the period of the routine and enablement of the routine.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Default** | **Range** | **Description** |
| Service Routine Counter | WR | 0x7FFFFFFF | 30:0 | The timer of service routine |
| Service routine enable | WR | 0x0 | 31 | Enable bit of service routine |

The SR\_CONTROL registers are 16 registers for service routine 0 to 16.

Each SR\_CONTROL and service routine can be set and work independently.

The Service Routine Counter holds the service routine period:

Service Routine Period = Service Routine Counter holds x 100 nS;

Note: Changing timer should be done via enable = 0 only it is no allowed to change timer to a running routine.

The minimal value expected is 500 (50uS).

### SR\_MEMORY

The SR\_MEMORY is the service routine memory space. The Memory is can be read and written in random. No default value exists.

Each service routine takes up 2 consecutive 32bit address ex. Service routine #5 takes addresses (SR\_MEMORY\_BASE + 0xA) and (SR\_MEMORY\_BASE + 0xB).

The Service Routine Memory needs to written prior to the SR\_CONTROL which enables it.

### SERIAL\_WRITE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Default** | **Range** | **Description** |
| SERIAL\_WRITE | WO |  | 31:0 | Serial Q TX FIFO write address |

The register is access to the Serial TX Q FIFO. The write strobe is making a “push” to the FIFO. The register cannot be read. No reset value is relevant.

All accesses to serial I/F are done by writing this address.

Note: Prior to writing the address SW needs to assure the FIFO has sufficient place. Writing a full FIFO will be discarding the overflowing access and potentially make the Squid out of sync.

### SERIAL\_READ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Default** | **Range** | **Description** |
| SERIAL\_READ | RO |  | 31:0 | Serial Q RX FIFO read address |

The register is access to the Serial RX Q FIFO. The read strobe is making a “pop” to the FIFO. The register cannot be written. No reset value is relevant.

All accesses to returning serial packets are done by reading this address.

Note: Prior to reading the address SW needs to assure the FIFO is not empty. Reading an empty FIFO will be retrieving a non-relevant data and potentially make the Squid out of sync.

### SERVICE\_READ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Default** | **Range** | **Description** |
| SERVICE\_READ | RO |  | 31:0 | Service Q RX FIFO read address |

The register is access to the Service RX Q FIFO. The read strobe is making a “pop” to the FIFO. The register cannot be written. No reset value is relevant.

All accesses to service routine returning serial packets are done by reading this address.

Note: Prior to reading the address SW needs to assure the FIFO is not empty. Reading an empty FIFO will be retrieving a non-relevant data and potentially make the Squid out of sync.