Bridge of Life Education

# Caravel SOC Introduction

Housekeeping GPIO SPI MMIO

Willy Chiang

# Topics

- System Block Diagram ( modules )
- Processor VexRISCV functions & its interface
- Reset / POR
- Management Protect Are – power control
- Clocking / DLL, configuration SPI register
- Housekeeping SPI registers
- GPIO
- SPI
- IRQ
- Memory-mapped IO address
- SRAM – Management area/Storage area
- Bus - Wishbone
- Peripherals – Counter/Timer/UART
- User project design (counter) Interface
- Testbench
- Firmware code

# Topics

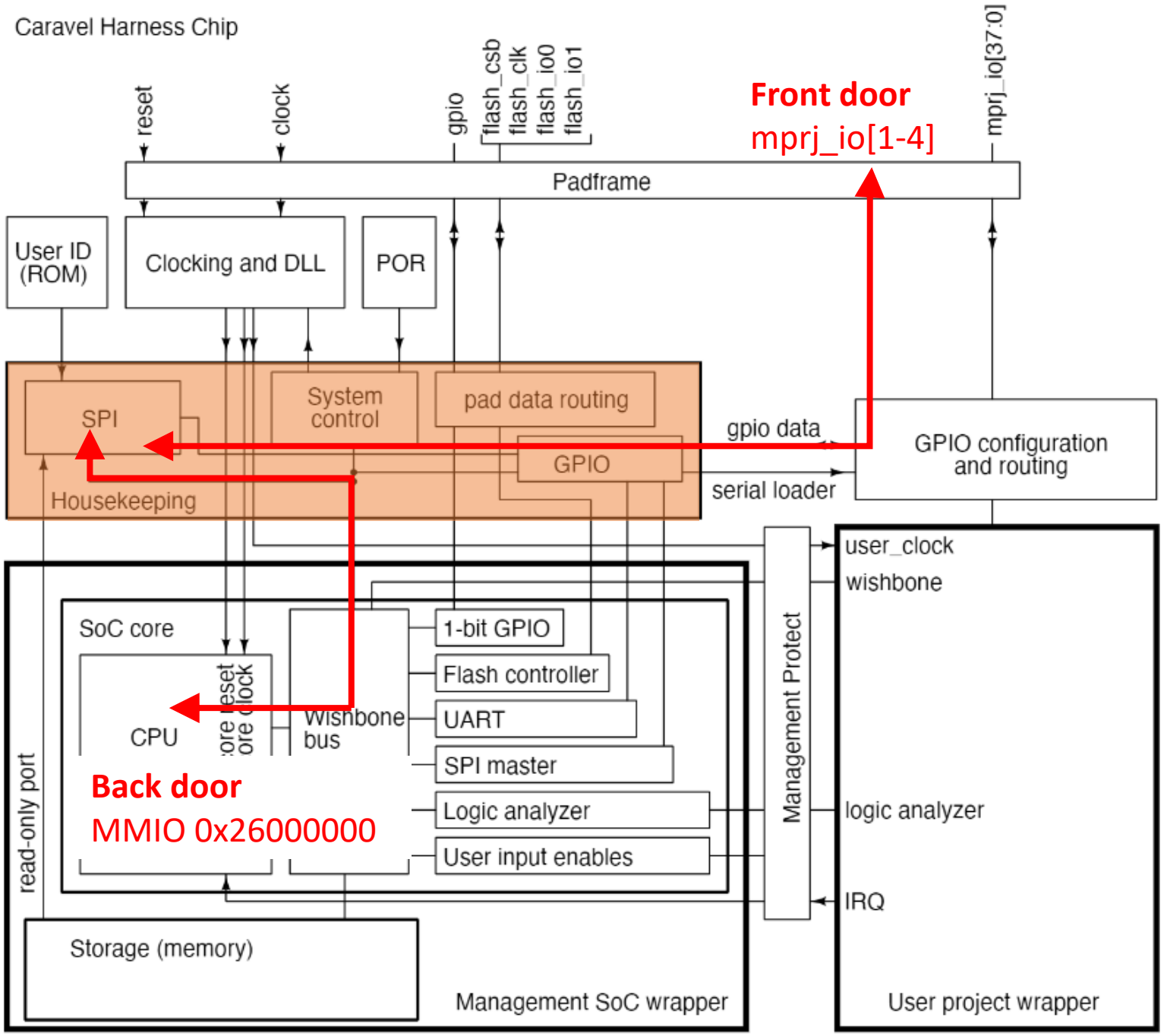
- System Block Diagram ( modules )
- Processor VexRISCV functions & its interface
- Reset / POR
- Management Protect Are – power control
- Clocking / DLL, configuration SPI register
- Housekeeping SPI registers
- GPIO
- SPI
- IRQ
- Memory-mapped IO address
- SRAM – Management area/Storage area
- Bus - Wishbone
- Peripherals – Counter/Timer/UART
- User project design (counter) Interface
- Testbench
- Firmware code

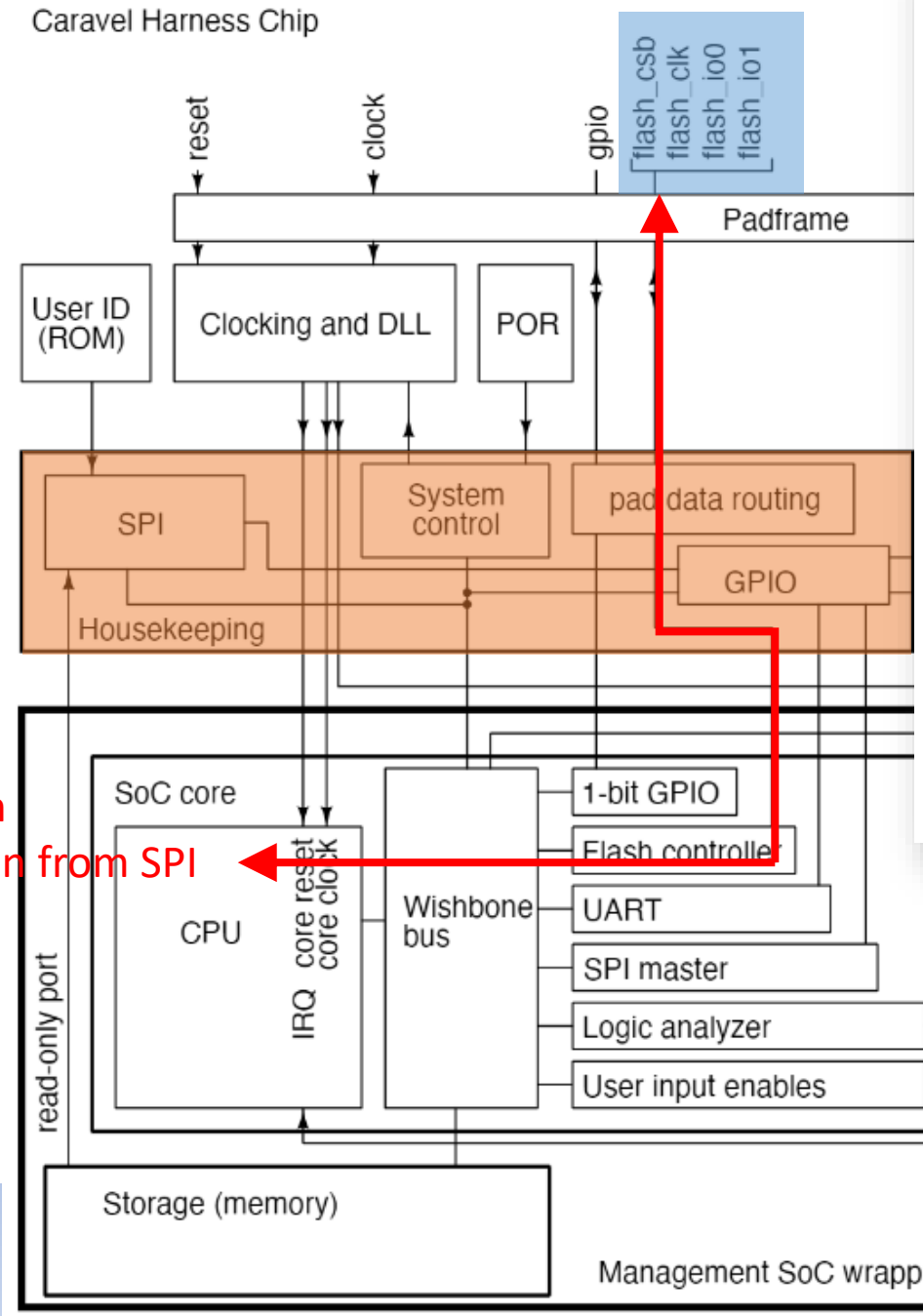
## Front door (SPI):

**SPI interface** connected to the **padframe** through GPIO pins

## Back door (MMIO):

**wishbone** interface connected to the **management SoC**



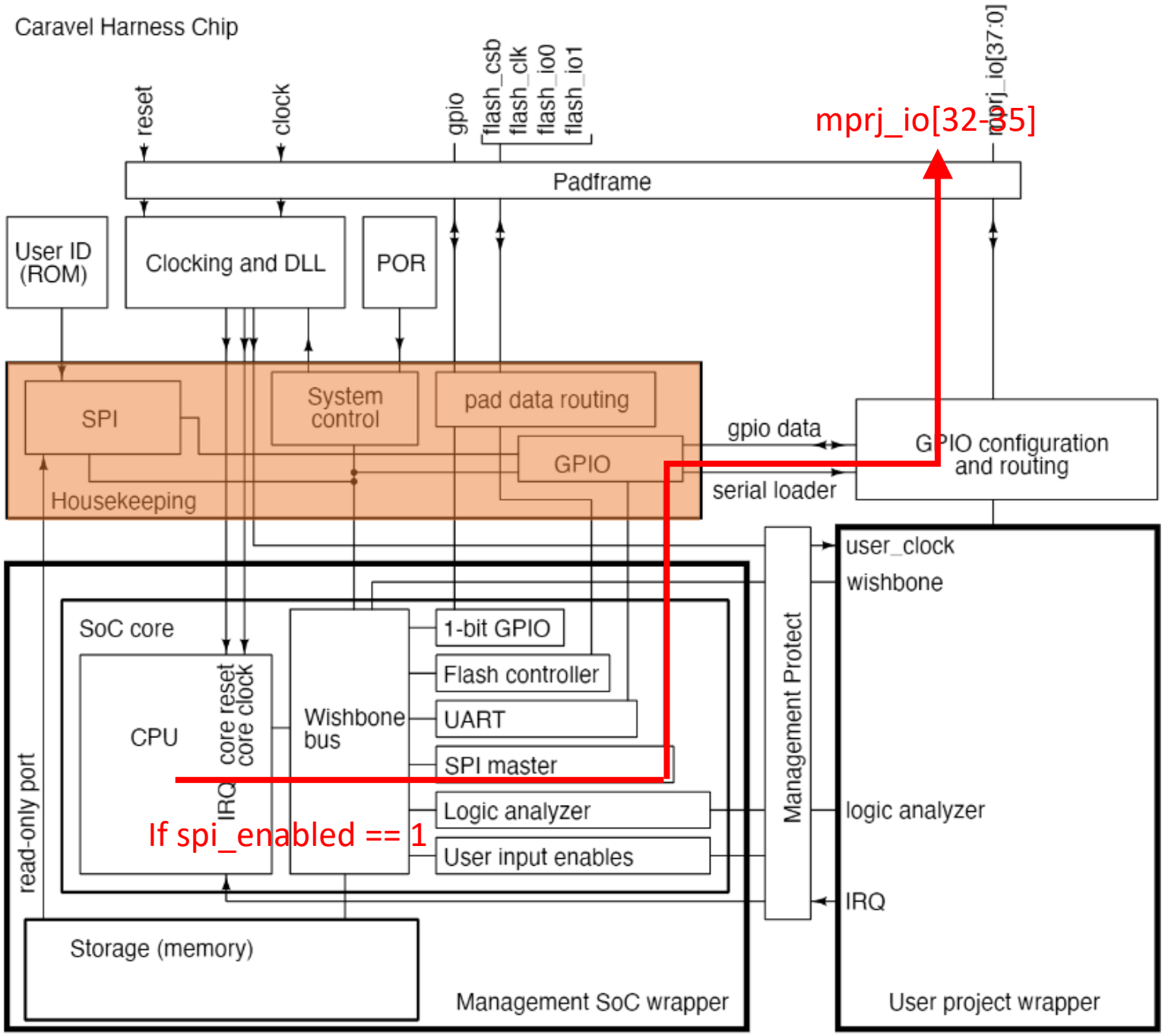


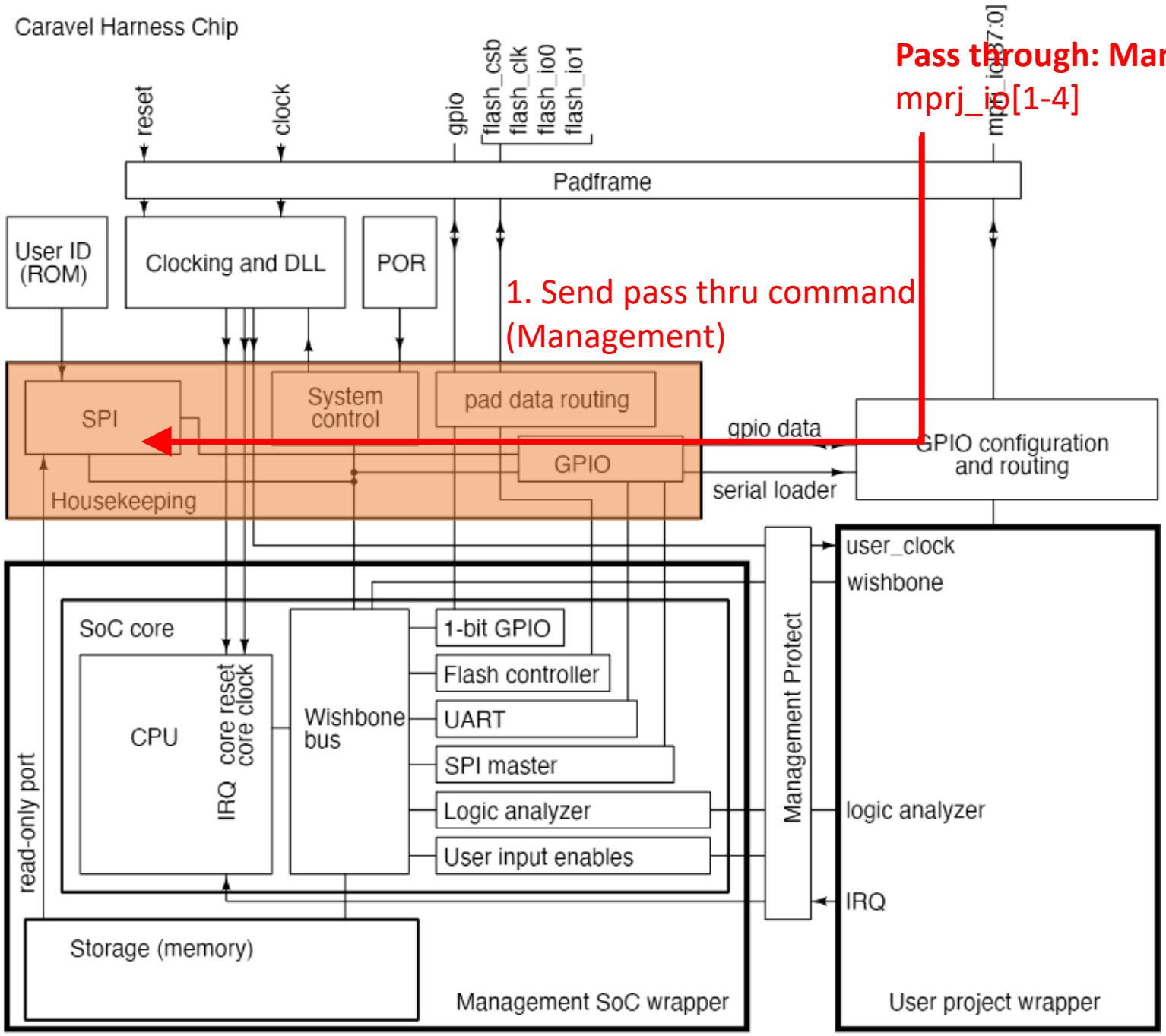

CPU fetch instruction from SPI flash

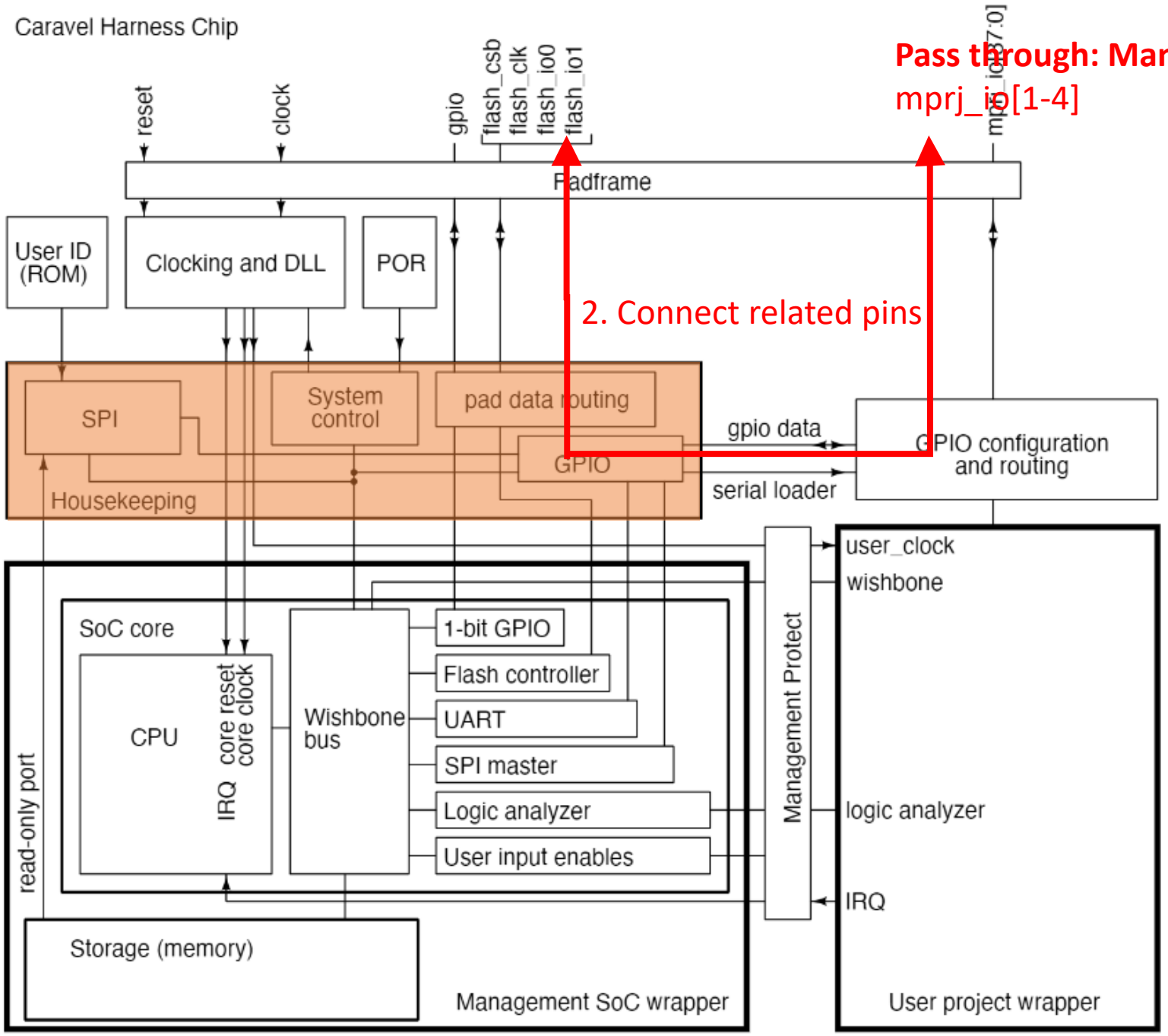
counter_la_tb.v:

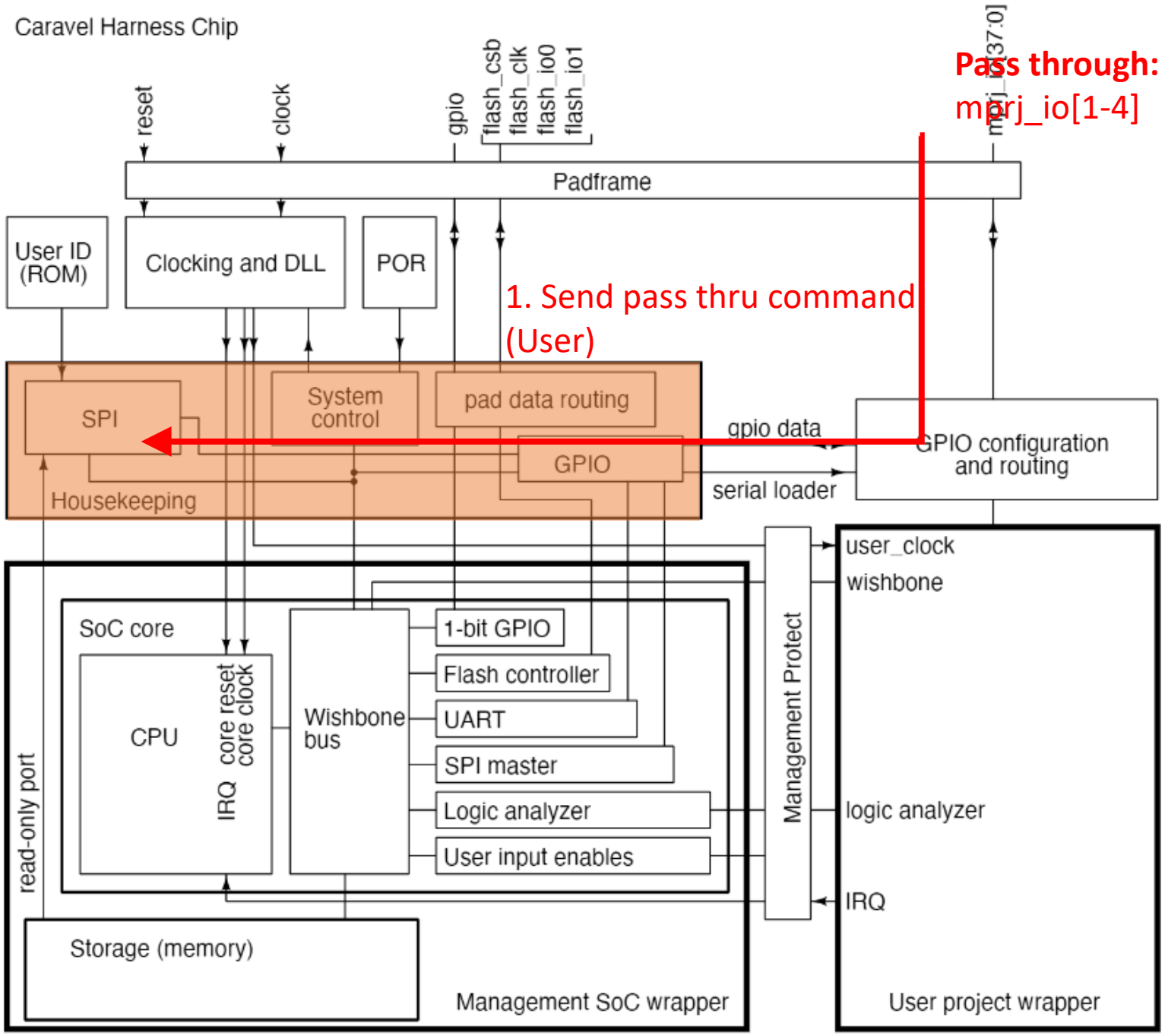
```
caravel uut (
    .clock      (clock),
    .gpio       (gpio),
    .mprj_io    (mprj_io),
    .flash_csb(flash_csb),
    .flash_clk(flash_clk),
    .flash_io0(flash_io0),
    .flash_io1(flash_io1),
    .resetb     (RSTB)
);

spiflash #(
    .FILENAME("counter_la.hex")
) spiflash (
    .csb(flash_csb),
    .clk(flash_clk),
    .io0(flash_io0),
    .io1(flash_io1),
    .io2(),             // not used
    .io3()              // not used
);
```

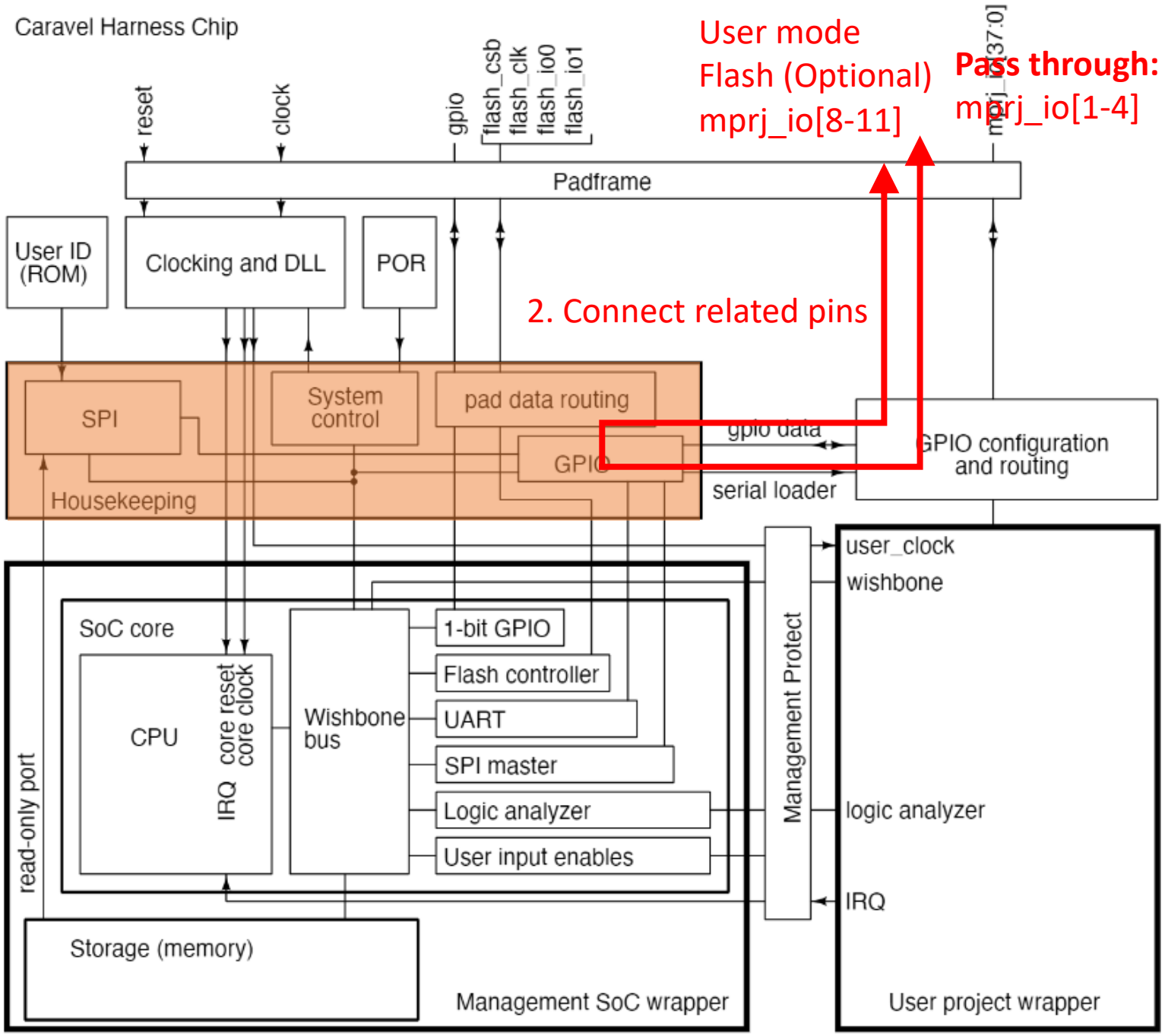
Note: **counter_la**.hex is building from **counter_la.c**


Caravel Harness Chip
reset
clock
gpio
flash_csb
flash_clk
flash_io0
flash_io1
mprj_io[37:0]
mprj_io[32-35]
Padframe
User ID (ROM)
Clocking and DLL
POR
SPI
System control
pad data routing
GPIO
Housekeeping
gpio data
serial loader
GPIO configuration and routing
user_clock
wishbone
SoC core
CPU
core reset
core clock
IRQ
Wishbone bus
1-bit GPIO
Flash controller
UART
SPI master
Logic analyzer
User input enables
If spi_enabled == 1
read-only port
Management Protect
logic analyzer
IRQ
Storage (memory)
Management SoC wrapper
User project wrapper

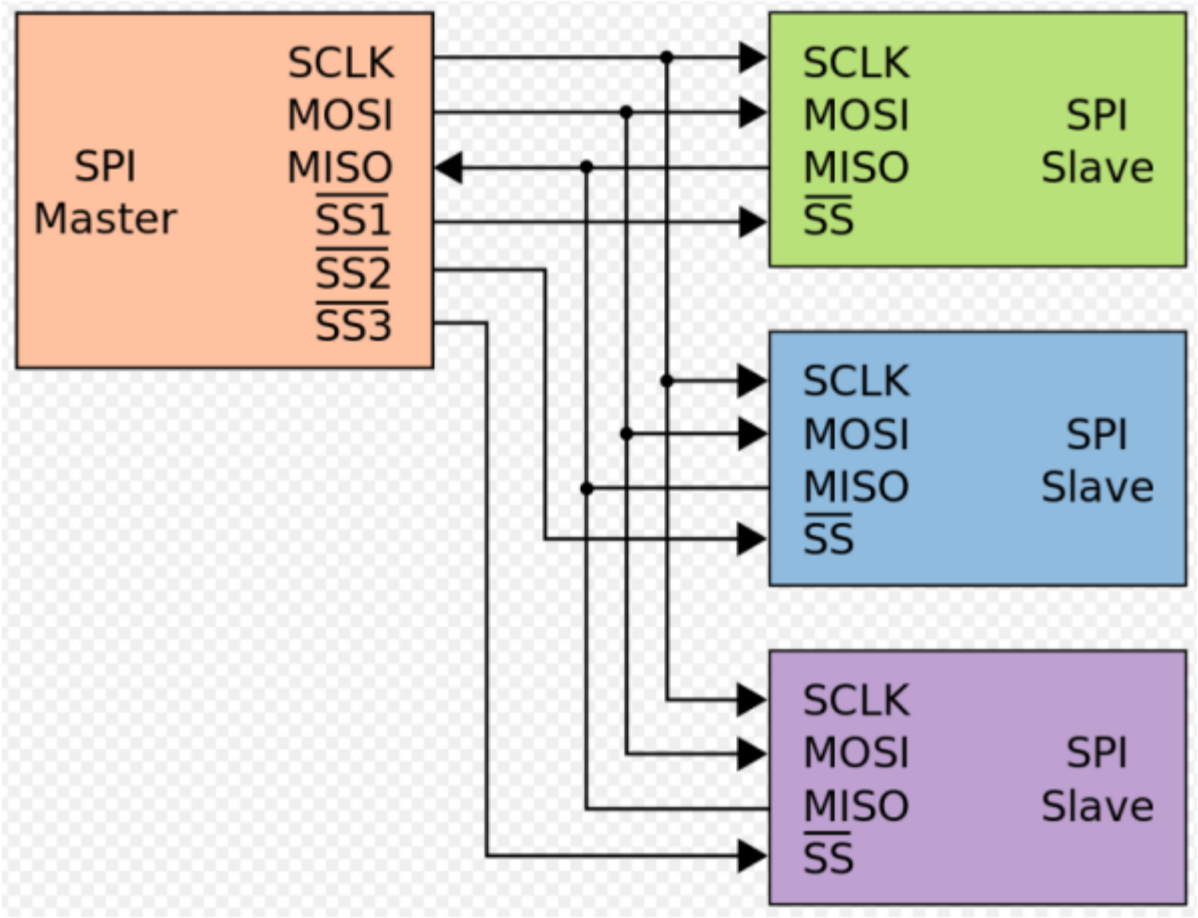

Caravel Harness Chip
Pass through: Management mode
mprj_io[1-4]
reset
clock
gpio
flash_csb
flash_clk
flash_io0
flash_io1
mprj_io[37:0]
Padframe
User ID (ROM)
Clocking and DLL
POR
1. Send pass thru command (Management)
SPI
System control
pad data routing
GPIO
Housekeeping
gpio data
serial loader
GPIO configuration and routing
user_clock
wishbone
SoC core
CPU
IRQ
core reset
core clock
Wishbone bus
1-bit GPIO
Flash controller
UART
SPI master
Logic analyzer
User input enables
Management Protect
logic analyzer
IRQ
read-only port
Storage (memory)
Management SoC wrapper
User project wrapper


Caravel Harness Chip
reset
clock
gpio
flash_csb
flash_clk
flash_io0
flash_io1
mprj_io[37:0]
Pass through: Management mode
mprj_io[1-4]
Padframe
User ID (ROM)
Clocking and DLL
POR
2. Connect related pins
SPI
System control
pad data routing
GPIO
Housekeeping
gpio data
serial loader
GPIO configuration and routing
user_clock
wishbone
SoC core
CPU
IRQ
core reset
core clock
Wishbone bus
1-bit GPIO
Flash controller
UART
SPI master
Logic analyzer
User input enables
Management Protect
logic analyzer
IRQ
read-only port
Storage (memory)
Management SoC wrapper
User project wrapper

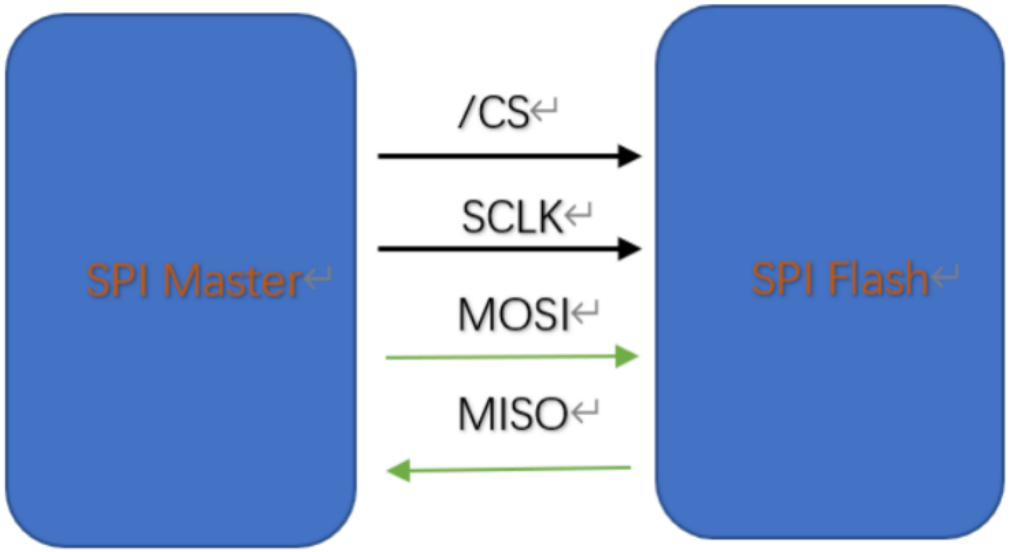

Caravel Harness Chip
reset
clock
gpio
flash_csb
flash_clk
flash_io0
flash_io1
mprj_io[37:0]
Pass through: User mode
mprj_io[1-4]
Padframe
User ID (ROM)
Clocking and DLL
POR
1. Send pass thru command (User)
SPI
System control
pad data routing
GPIO
Housekeeping
gpio data
serial loader
GPIO configuration and routing
user_clock
wishbone
SoC core
CPU
IRQ
core reset
core clock
Wishbone bus
1-bit GPIO
Flash controller
UART
SPI master
Logic analyzer
User input enables
Management Protect
logic analyzer
IRQ
read-only port
Storage (memory)
Management SoC wrapper
User project wrapper


Caravel Harness Chip
reset
clock
gpio
flash_csb
flash_clk
flash_io0
flash_io1
mprj_io[37:0]
User mode
Flash (Optional)
mprj_io[8-11]
Pass through: User mode
mprj_io[1-4]
Padframe
User ID (ROM)
Clocking and DLL
POR
2. Connect related pins
SPI
System control
pad data routing
GPIO
Housekeeping
gpio data
serial loader
GPIO configuration and routing
user_clock
wishbone
SoC core
CPU
IRQ
core reset
core clock
Wishbone bus
1-bit GPIO
Flash controller
UART
SPI master
Logic analyzer
User input enables
read-only port
Management Protect
logic analyzer
IRQ
Storage (memory)
Management SoC wrapper
User project wrapper

# SPI bus topology




# Standard SPI Mode

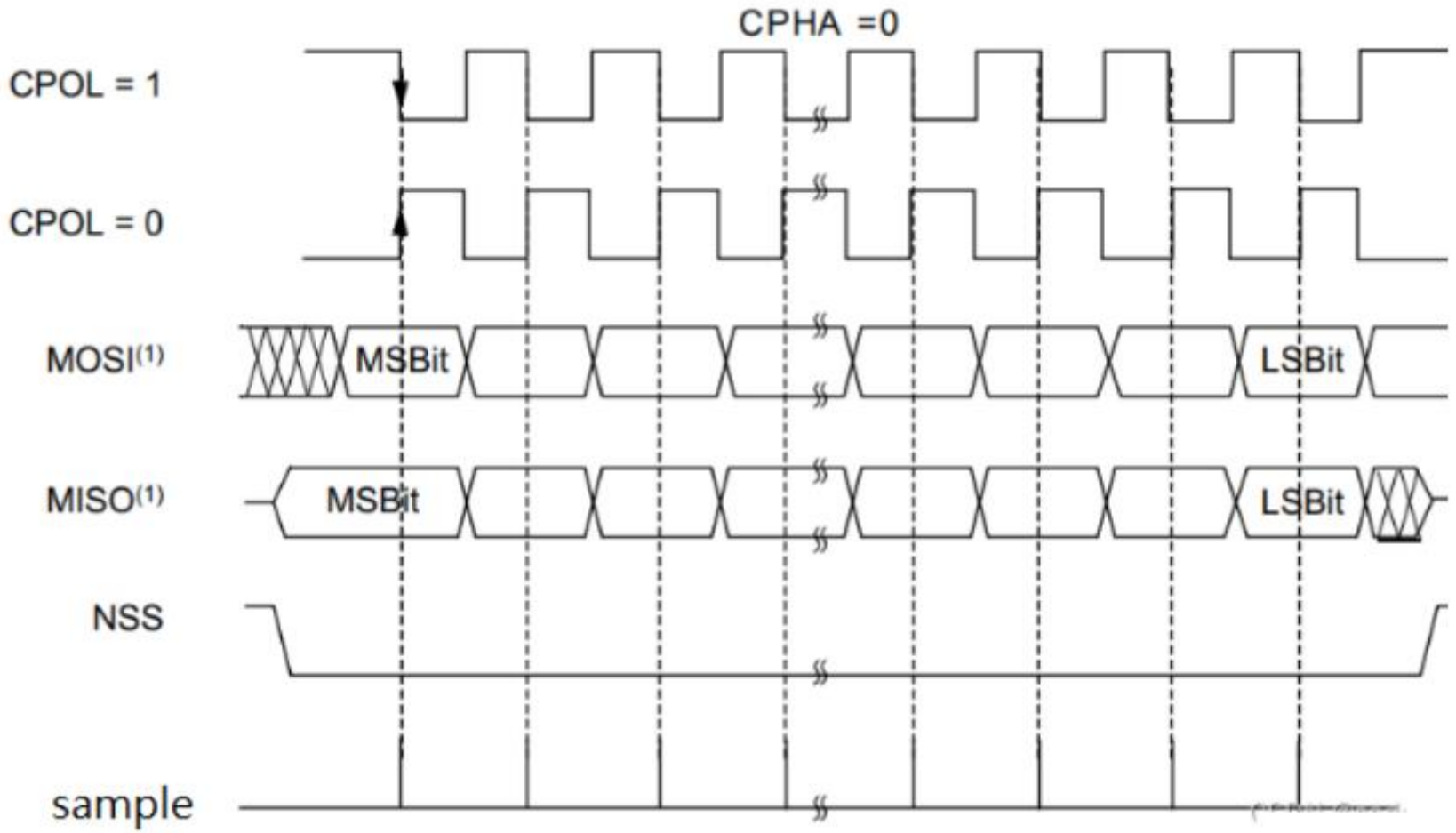


# SPI mode

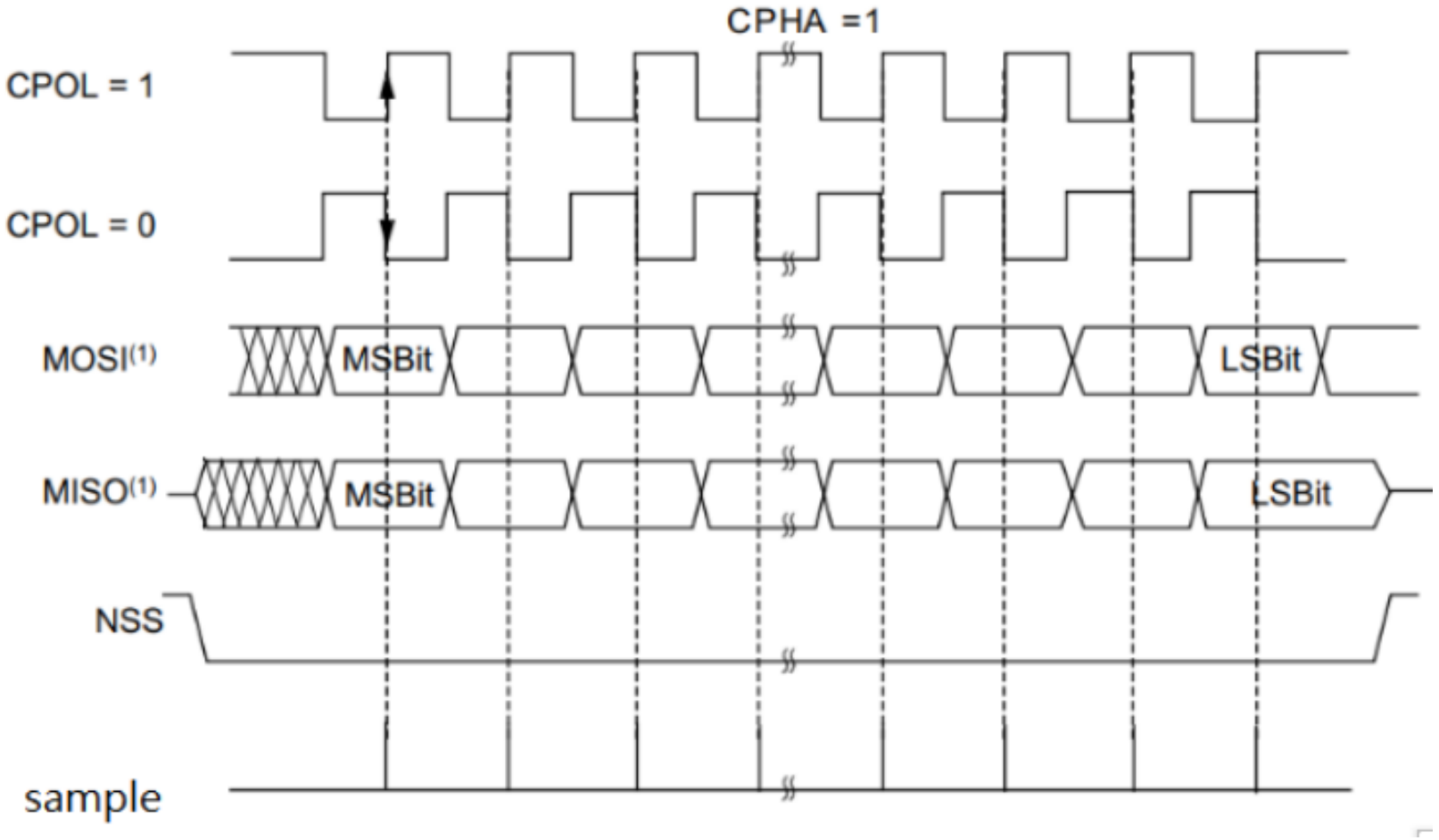
| mode | CPOL | CPHA |
|---|---|---|
| mode 0 | 0 | 0 |
| mode 1 | 0 | 1 |
| mode 2 | 1 | 0 |
| mode 3 | 1 | 1 |

- **CPOL（Clock Polarity）**
  - CPOL=1 CLK=high @ bus idle
  - CPOL=0 CLK=low @ bus idle
- **CPHA（Clock Phase)**
  - CPHA=0 sample data @ 1st edge > Rising Edge
  - CPHA=1 sample data @ 2nd edge > Falling Edge

# SPI mode 0 and mode 2



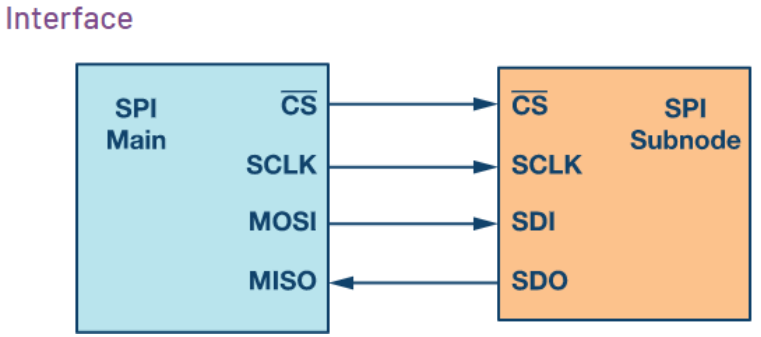
# SPI mode 1 and mode 3



# Housekeeping SPI

SDI (pin F9), CSB (pin E8), SCK (pin F8), and SDO (pin E9)

- An SPI slave that can be accessed from remote host
- The SPI implementation is **mode 0**
  - New data on **SDI** captured on the **SCK** rising edge
  - Output data on the falling edge of **SCK**
- SPI pin are shared with user area GPIO.



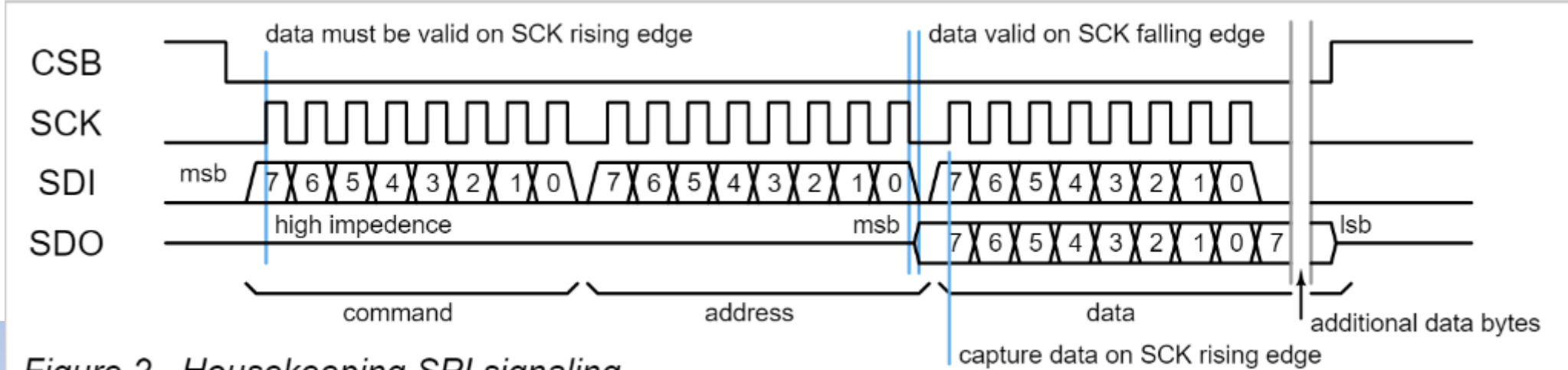




*Figure 2. Housekeeping SPI signaling*

# SPI protocol definition

- After **CSB** is set low, the SPI is always in the “command” state, awaiting a new command.

*Table 8* Housekeeping SPI command word definition

| | |
|---|---|
| 00000000 | No operation |
| 10000000 | Write in streaming mode |
| 01000000 | Read in streaming mode |
| 11000000 | Simultaneous Read/Write in streaming mode |
| 11000100 | Pass-through (management) Read/Write in streaming mode |
| 11000110 | Pass-through (user) Read/Write in streaming mode |
| 10nnn000 | Write in n-byte mode (up to 7 bytes). |
| 01nnn000 | Read in n-byte mode (up to 7 bytes). |
| 11nnn000 | Simultaneous Read/Write in n-byte mode (up to 7 bytes). |

housekeeping_spi.v

```
// Command format:
// 00000000  No operation
// 10000000  Write until CSB raised
// 01000000  Read  until CSB raised
// 11000000  Simultaneous read/write until CSB raised
// 11000100  Pass-through read/write to management area flash SPI until CSB raised
// 11000010  Pass-through read/write to user area flash SPI until CSB raised
// wrnnn000  Read/write as above, for nnn = 1 to 7 bytes, then terminate
```

Document typo

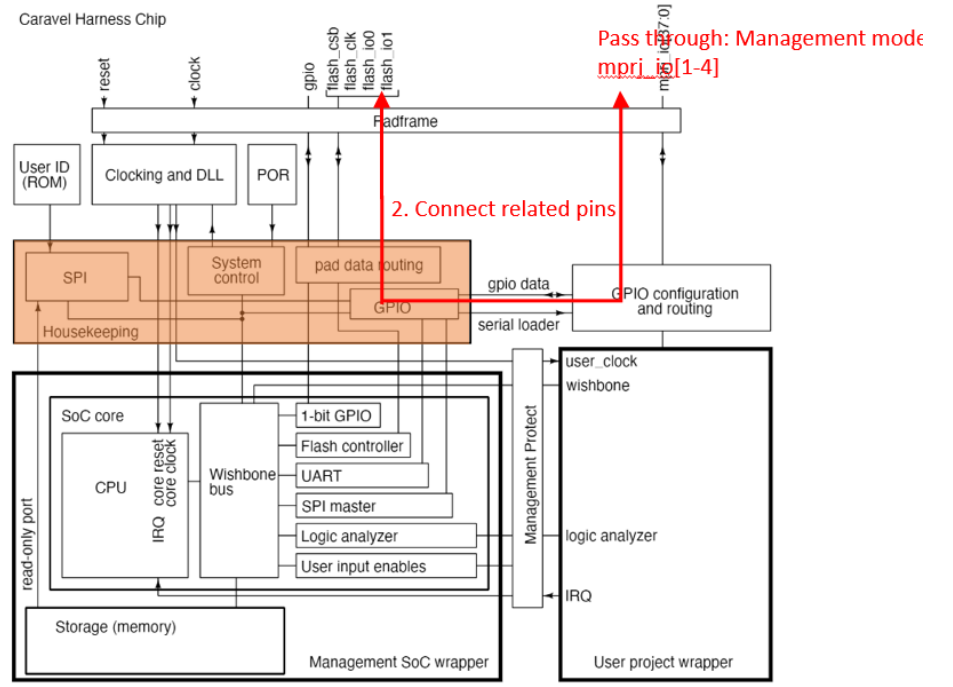
# Housekeeping SPI modes

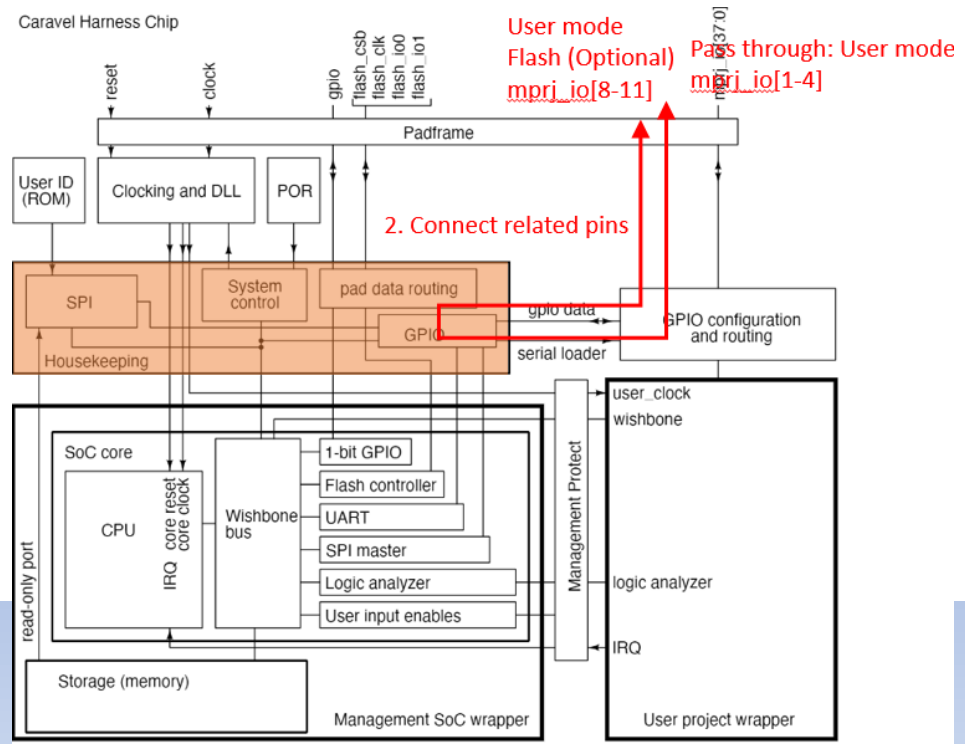
| | |
|---|---|
| 00000000 | No operation |
| 10000000 | Write in streaming mode |
| 01000000 | Read in streaming mode |
| 11000000 | Simultaneous Read/Write in streaming mode |
| 11000100 | Pass-through (management) Read/Write in streaming mode |
| 11000110 | Pass-through (user) Read/Write in streaming mode |
| 10nnn000 | Write in n-byte mode (up to 7 bytes). |
| 01nnn000 | Read in n-byte mode (up to 7 bytes). |
| 11nnn000 | Simultaneous Read/Write in n-byte mode (up to 7 bytes). |

- Streaming Mode
  - The data is sent or received continuously, one byte at a time, with the internal address incrementing for each byte.
  - Operation continues until **CSB** is raised to end the transfer.
- N-byte mode
  - The number of bytes to be read and/or written is encoded in the command word, and may have a value from **1 to 7**
  - After n bytes have been read and/or written, the SPI returns to waiting for the next command.

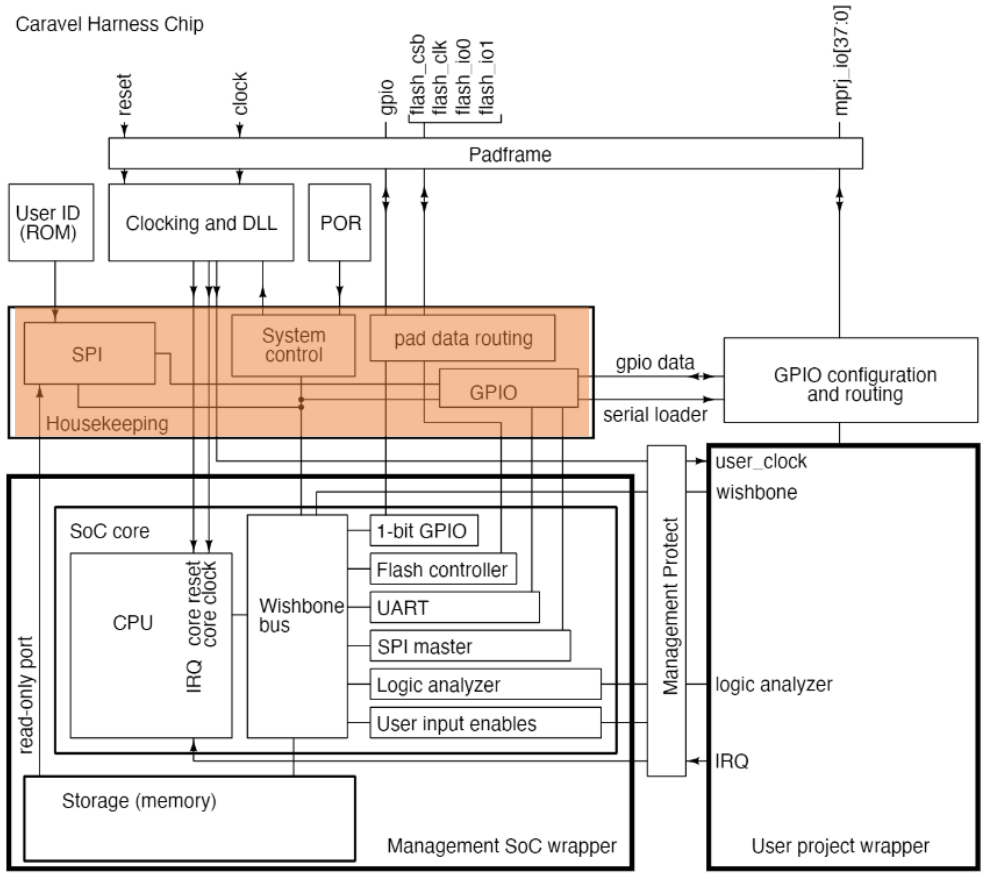
# Pass-thru Mode

- The pass-thru mode puts the CPU into immediate reset, then sets **FLASH_CSB** low to initiate a data transfer to the QSPI flash
- After the pass-thru command byte has been issued
  - SDI → FLASH_IO0
  - SCK → FLASH_CLK
  - FLASH_IO1 → SDO
- When **CSB** is raised, the **FLASH_CSB** is also raised, terminating the data transfer to the QSPI flash. The CPU is brought out of reset, and starts executing instructions at the program start address.



# Housekeeping SPI registers




| Register Address | msb 7 | 6 | 5 | 4 | 3 | 2 | 1 | lsb 0 | comments |
|---|---|---|---|---|---|---|---|---|---|
| 0x00 | SPI status and control | | | | | | | | unused/ undefined |
| 0x01 | unused | | | | manufacturer_ID[11:8] (= 0x4) | | | | read-only |
| 0x02 | manufacturer_ID[7:0] (= 0x56) | | | | | | | | read-only |
| 0x03 | product_ID (= 0x10) | | | | | | | | read-only |
| 0x04–0x07 | user_project_ID (unique value per project) | | | | | | | | read-only |
| 0x08 | unused | | | | | | PLL DCO enable | PLL enable | default 0x02 |
| 0x09 | unused | | | | | | | PLL bypass | default 0x01 |
| 0x0A | unused | | | | | | | CPU IRQ | default 0x00 |
| 0x0B | unused | | | | | | | CPU reset | default 0x00 |
| 0x0C | unused | | | | | | | CPU trap | read-only |
| 0x0D–0x10 | DCO trim (26 bits) (= 0x3ffefff) | | | | | | | | default 0x3ffefff |
| 0x11 | unused | | PLL output divider 2 | | | PLL output divider | | | default 0x12 |
| 0x12 | unused | | | | PLL feedback divider | | | | default 0x04 |

# docs/other/ memory_map.txt

```
SPI register   description     signal                  memory map address
-------------------------------------------------------------------------
 00 SPI status (reserved)  (undefined)                    2610_0000
 01 Manufacturer ID (high) mfgr_id[11:8]                  2610_0006
..
 03 Product ID         prod_id[7:0]                    2610_0004
 04 User project ID        mask_rev[31:24]                 2610_000b
..

 08 PLL enables        pll_dco_ena, pll_ena      2610_000c
 09 PLL bypass         pll_bypass            2610_0010
 0a IRQ            irq                2610_0014
 0b     Reset              reset                   2610_0018
 0c CPU trap state         trap                         2610_0028
 0d PLL trim           pll_trim[31:24]           2610_001f
..
 11 PLL source         pll90_sel[2:0], pll_sel[2:0]  2610_0020
 12 PLL divider        pll_div[4:0]          2610_0024
 13 GPIO control           serial_resetn/clock/data      2600_0000

 14 SRAM read-only control hkspi_sram_clk/csb        2610_0034
 15 SRAM read-only address hkspi_sram_addr           2610_0030
 16 SRAM read-only data    hkspi_sram_data[31:24]        2610_002f
 17 SRAM read-only data    hkspi_sram_data[23:16]        2610_002e
 18 SRAM read-only data    hkspi_sram_data[15:8]         2610_002d
 19 SRAM read-only data    hkspi_sram_data[7:0]      2610_002c

 1a Power monitor          usr1/2_vcc/vdd_pwrgood        2620_0000
 1b Output redirect        clk1/clk2/trap_output_dest    2620_0004
 1c Input redirect         irq_8/7_inputsrc          2620_000c

 1d GPIO[0] configure      gpio_configure[0][12:8]       2600_0025
 ...
 68 GPIO[37] configure     gpio_configure[37][7:0]       2600_00b8
 69 GPIO data          mgmt_gpio_in[37:32]       2600_0010
..
 6e Power control          pwr_ctrl_out[3:0]         2600_0004
 6f HK SPI disable         hkspi_disable             2620_0010
```

# rtl/housekeeping.v

- Base Decode
  - GPIO_BASE_ADR = 32'h2600_0000,
  - SPI_BASE_ADR = 32'h2610_0000,
  - SYS_BASE_ADR = 32'h2620_0000,
- Module enable status from SoC
  - **input qspi_enabled** // Flash SPI is in quad mode
  - **input spi_enabled** // SPI master is enabled
- SPI master interface to/from SoC
  - output spi_sdi,
  - input spi_csb,
  - input spi_sck,
  - input spi_sdo,
  - input spi_sdoenb,
- PassThru Mode detect by housekeeping_spi
  - wire pass_thru_mgmt
  - wire pass_thru_user
- GPIO data management (to padframe)
  - input [`MPRJ_IO_PADS-1:0] mgmt_gpio_in,
  - output [`MPRJ_IO_PADS-1:0] mgmt_gpio_out,
  - output [`MPRJ_IO_PADS-1:0] mgmt_gpio_oeb,
- SPI flash management (management SoC side)
  - input spimemio_flash_xx
  - output spimemio_flash_xx
- SPI flash management (padframe side)
  - output pad_flash_xx
  - input pad_flash_xx

## Front door

| | |
|---|---|
| mprj_io[4] | mprj_io[3] |
| SCK | CSB |
| mprj_io[2] | mprj_io[1] |
| SDI | SDO |

| | | | |
|---|---|---|---|
| E9 | SDO | Digital out | Housekeeping serial interface data output |
| F9 | SDI | Digital in | Housekeeping serial interface data input |
| E8 | CSB | Digital in | Housekeeping serial interface chip select |
| F8 | SCK | Digital in | Housekeeping serial interface clock |

## Spi_master

```
* spi_sck    = mprj_io[32]        (output)
* spi_csb    = mprj_io[33]        (output)
* spi_sdi    = mprj_io[34]        (input)
* spi_sdo    = mprj_io[35]        (output)
```

## Document typo

| | | | |
|---|---|---|---|
| F9 | spi_sdo | Digital out | Serial interface controller data output |
| F8 | spi_sck | Digital out | Serial interface controller clock |
| E8 | spi_csb | Digital out | Serial interface controller chip select |
| E9 | spi_sdi | Digital in | Serial interface controller data input |

| | | | |
|---|---|---|---|
| D8 | flash_clk | Digital out | Flash SPI clock |
| C10 | flash_csb | Digital out | Flash SPI chip select |
| D9, D10 | flash_io[1:0] | Digital I/O | Flash SPI data input/output |

## Flash_Controller or

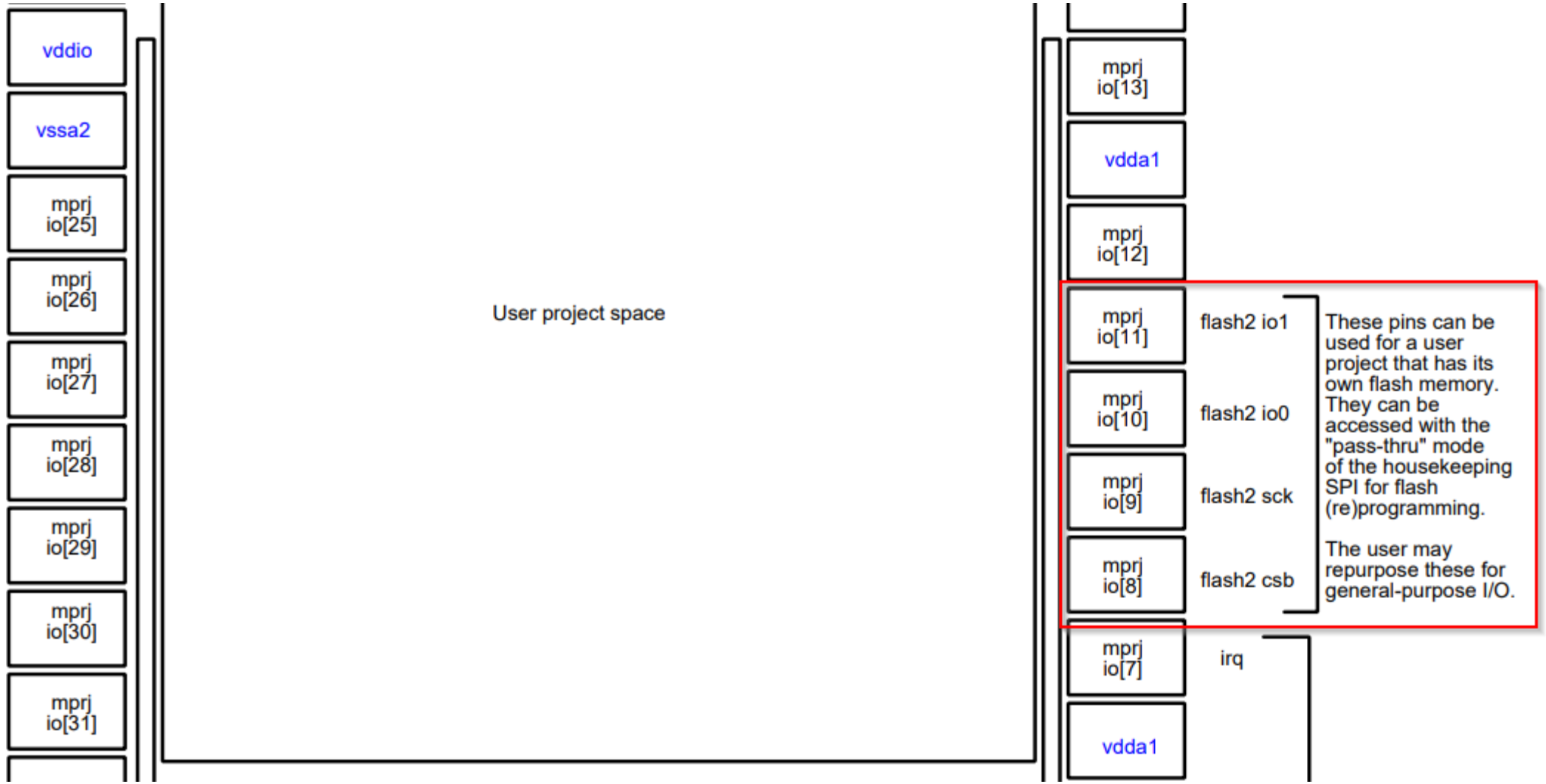
## Pass-thru Management mode

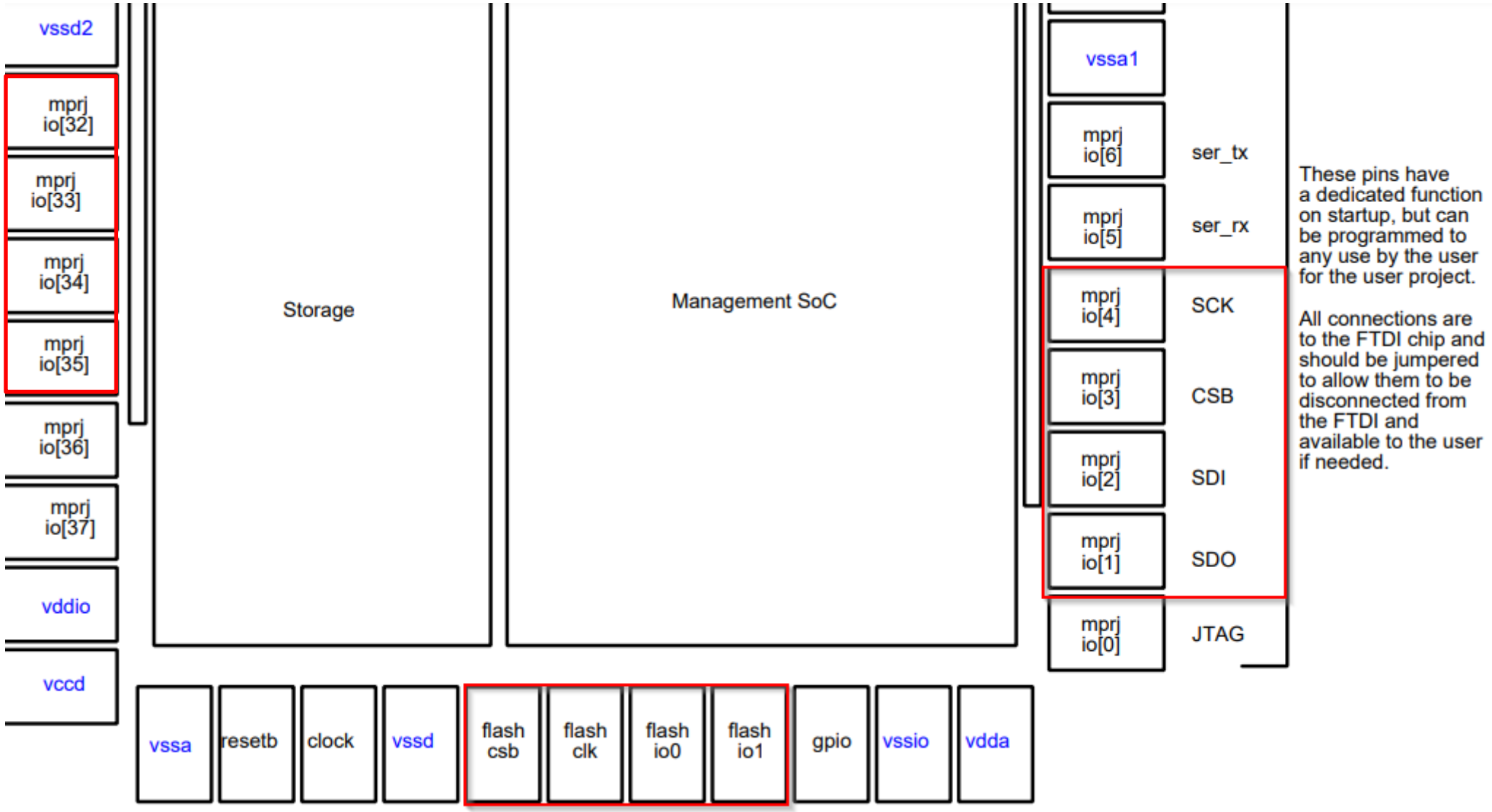
| | |
|---|---|
| flash_clk | mprj_io[33] |
| flash_io[1] | clock |
| flash_io[0] | flash_csb |

| | | | |
|---|---|---|---|
| F5 | flash2_csb | Digital out | User area QSPI flash enable (sense inverted) |
| E4 | flash2_sck | Digital out | User area QSPI flash clock |
| E3, F4 | flash2_io[1:0] | Digital I/O | User area QSPI flash data |

## Pass-thru User mode

| | |
|---|---|
| mprj_io[12] | mprj_io[11] |
| | flash2_io[1] |
| mprj_io[10] | mprj_io[9] |
| flash2_io[0] | flash2_sck |
| mprj_io[8] | mprj_io[7] |
| flash2_csb | irq |


vddio
vssa2
mprj io[25]
mprj io[26]
mprj io[27]
mprj io[28]
mprj io[29]
mprj io[30]
mprj io[31]
User project space
mprj io[13]
vdda1
mprj io[12]
mprj io[11]
flash2 io1
mprj io[10]
flash2 io0
mprj io[9]
flash2 sck
mprj io[8]
flash2 csb
These pins can be used for a user project that has its own flash memory. They can be accessed with the "pass-thru" mode of the housekeeping SPI for flash (re)programming.
The user may repurpose these for general-purpose I/O.
mprj io[7]
irq
vdda1




These pins have a dedicated function on startup, but can be programmed to any use by the user for the user project.

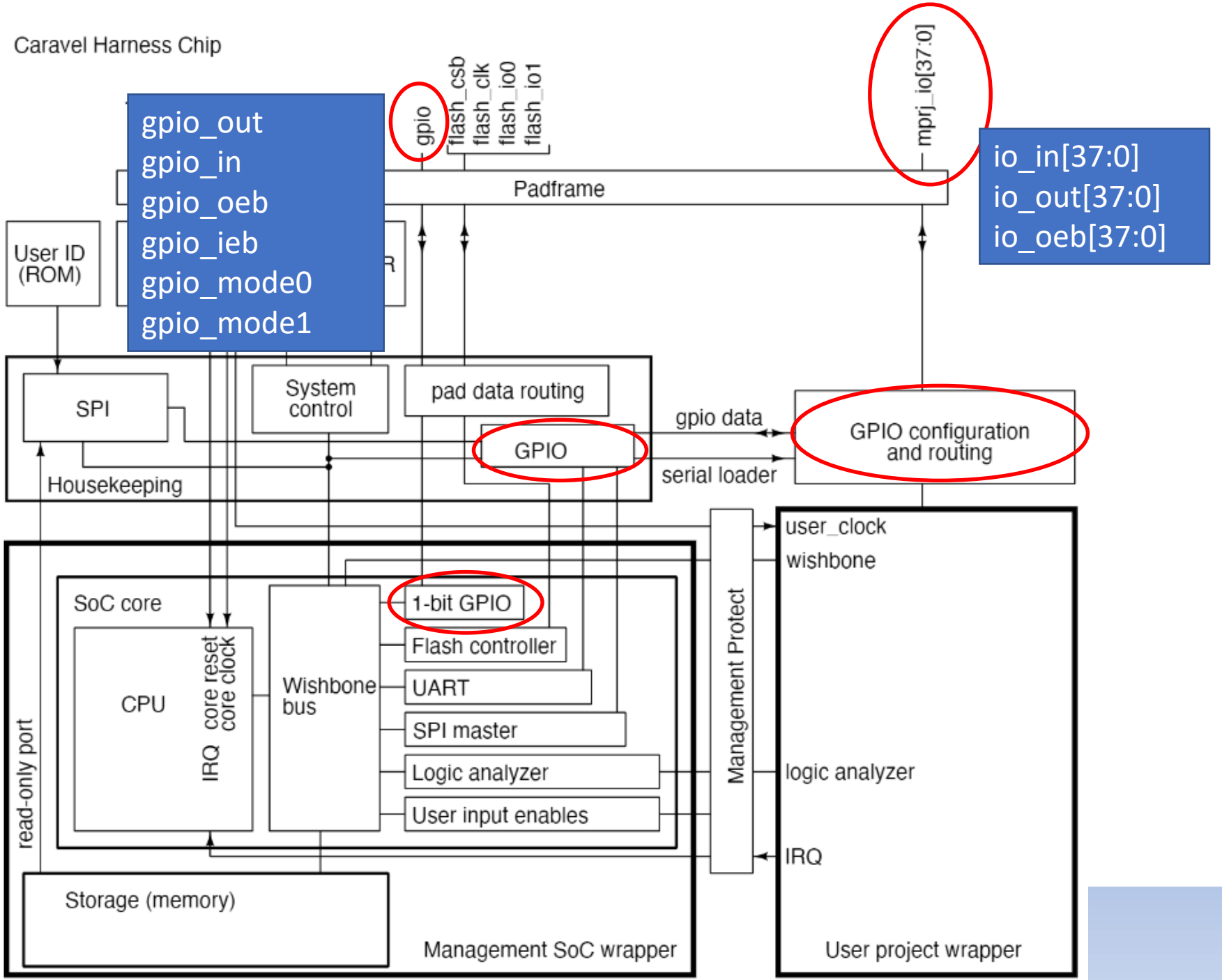
All connections are to the FTDI chip and should be jumpered to allow them to be disconnected from the FTDI and available to the user if needed.

# Topics

- System Block Diagram ( modules )
- Processor VexRISCV functions & its interface
- Reset / POR
- Management Protect Are – power control
- Clocking / DLL, configuration SPI register
- Housekeeping SPI registers
- GPIO
- SPI
- IRQ
- Memory-mapped IO address
- SRAM – Management area/Storage area
- Bus - Wishbone
- Peripherals – Counter/Timer/UART
- User project design (counter) Interface
- Testbench
- Firmware code

# Agenda

- Architecture Introduction
- One pin GPIO
- MPRJ_IO

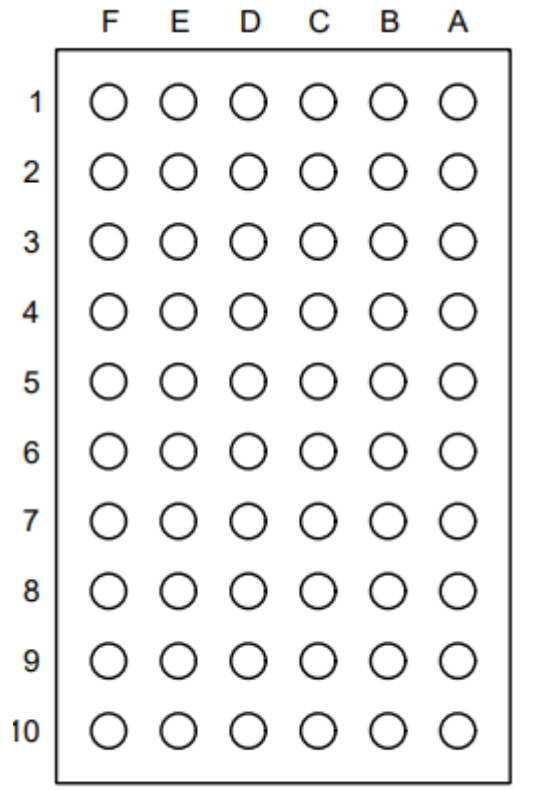

Caravel Harness Chip
gpio_out
gpio_in
gpio_oeb
gpio_ieb
gpio_mode0
gpio_mode1
gpio
flash_csb
flash_clk
flash_io0
flash_io1
mprj_io[37:0]
io_in[37:0]
io_out[37:0]
io_oeb[37:0]
Padframe
User ID (ROM)
SPI
System control
pad data routing
GPIO
gpio data
serial loader
GPIO configuration and routing
Housekeeping
user_clock
wishbone
SoC core
1-bit GPIO
Flash controller
UART
SPI master
Logic analyzer
User input enables
CPU
IRQ
core reset
core clock
Wishbone bus
read-only port
Management Protect
logic analyzer
IRQ
Storage (memory)
Management SoC wrapper
User project wrapper

# Descriptions

- In Harness Definition
  - GPIO handling moved out of management SoC and into SPI.
  - On power-up, the SPI automatically configures the GPIO. Manual load is possible from both the SPI and from the wishbone bus.
- In Management Area
  - Configure User Project I/O pads (**mprj_io**)
  - Control the User Project power supply **(one bit gpio)** [Not implement]
- In User Project Area
  - This is the user space. It has a limited silicon area 2.92mm x 3.52mm as well as a fixed number of **I/O pads** 38 and **power pads** 4.

# Pin description (6x10 WLCSP)

| Pin # | Name | Type | Summary description |
|---|---|---|---|
| A9, B9, A8, B8, C8, A7, A6, B6, A5, B5, A4, B4, A3, C3, A1, B2, B1, C2, C1, D1, D2, E1, F1, E2, D3, F3, E3, F4, E4, F5, E5, F7, E7, F8, E8, F9, E9, D7 | `mprj_io[37:0]` | Digital I/O | General purpose configurable digital I/O with pullup/pulldown, input or output, enable/disable, analog output, high voltage output, slew rate control. Shared between the user project area and the management SoC. |
| E10 | `gpio` | Digital I/O | Management GPIO/user power enable |




*Package as viewed from the bottom.*

# Agenda

- Architecture Introduction
- One pin GPIO
- MPRJ_IO

# General Purpose Input/Output (pin E10)

| | | | |
|---|---|---|---|
| E10 | gpio | Digital I/O | Management GPIO/user power enable |

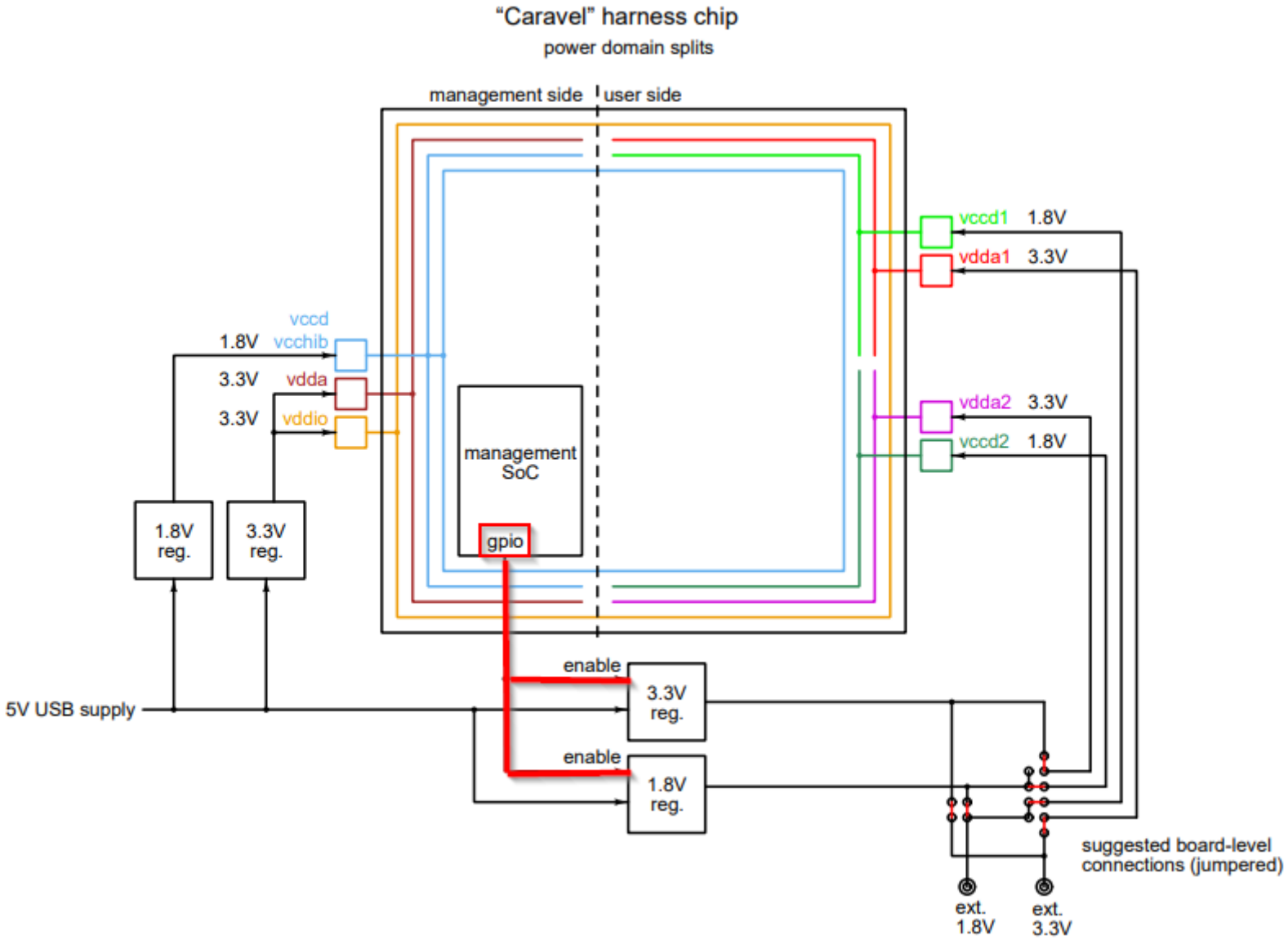
- A single GPIO port is provided from the Management SoC as general indicator and diagnostic for programming or as a means to control functionality off chip.
- One example user case is to set an enable for an off-chip LDO enabling a controlled power-up sequence for the user project.

**Register Listing for GPIO**

| Register | Address |
|---|---|
| GPIO_MODE1 | 0xf0002800 |
| GPIO_MODE0 | 0xf0002804 |
| GPIO_IEN | 0xf0002808 |
| GPIO_OE | 0xf000280c |
| GPIO_IN | 0xf0002810 |
| GPIO_OUT | 0xf0002814 |

**defs.h**

```
#define reg_gpio_mode1  (*(volatile uint32_t*) CSR_GPIO_MODE1_ADDR)
#define reg_gpio_mode0  (*(volatile uint32_t*) CSR_GPIO_MODE0_ADDR)
#define reg_gpio_ien    (*(volatile uint32_t*) CSR_GPIO_IEN_ADDR)
#define reg_gpio_oe     (*(volatile uint32_t*) CSR_GPIO_OE_ADDR)
#define reg_gpio_in     (*(volatile uint32_t*) CSR_GPIO_IN_ADDR)
#define reg_gpio_out    (*(volatile uint32_t*) CSR_GPIO_OUT_ADDR)
```



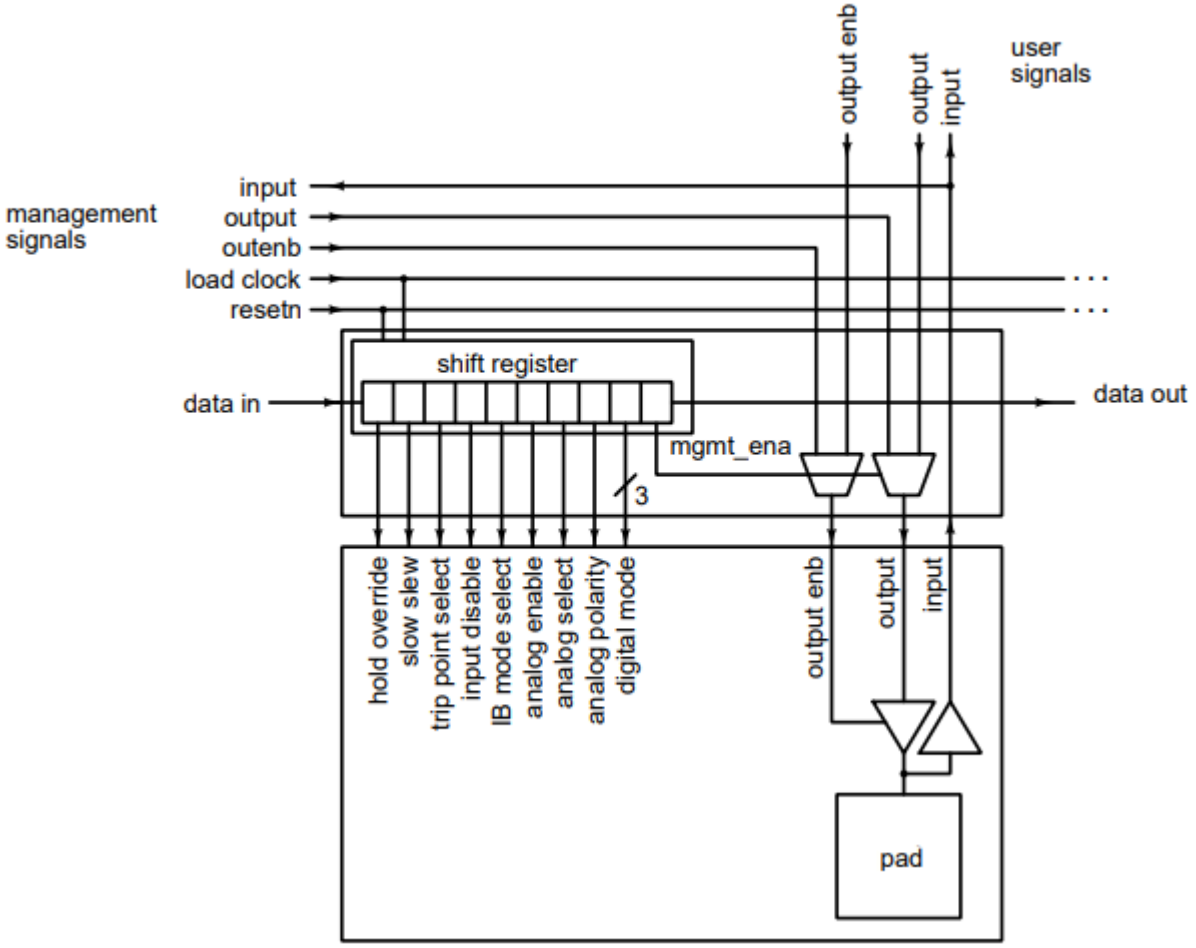

"Caravel" harness chip

# Agenda

- Architecture Introduction
- One pin GPIO
- MPRJ_IO

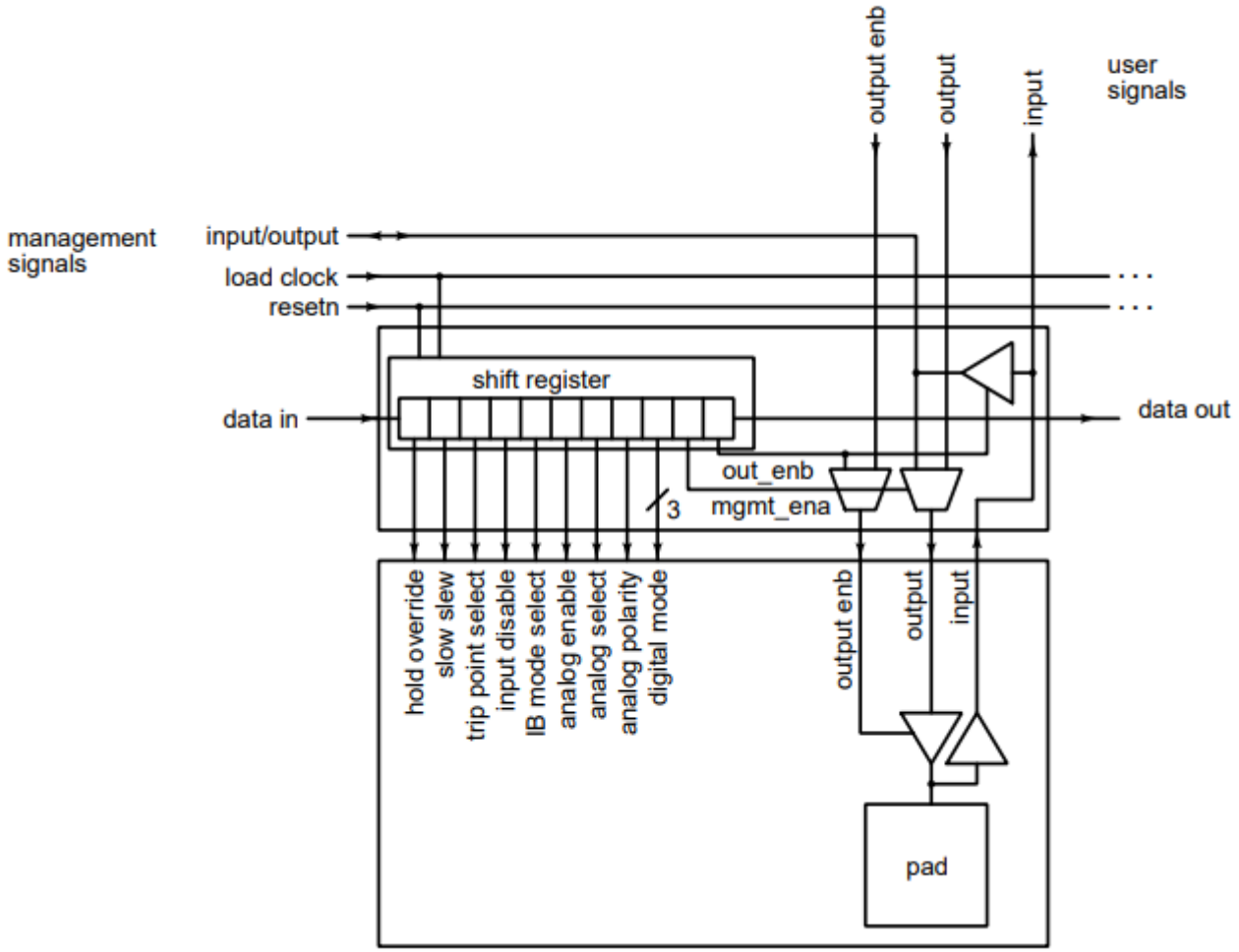
# GPIO Pad Structure

Used for pad 0 (JTAG) and pad 1 (SDO)

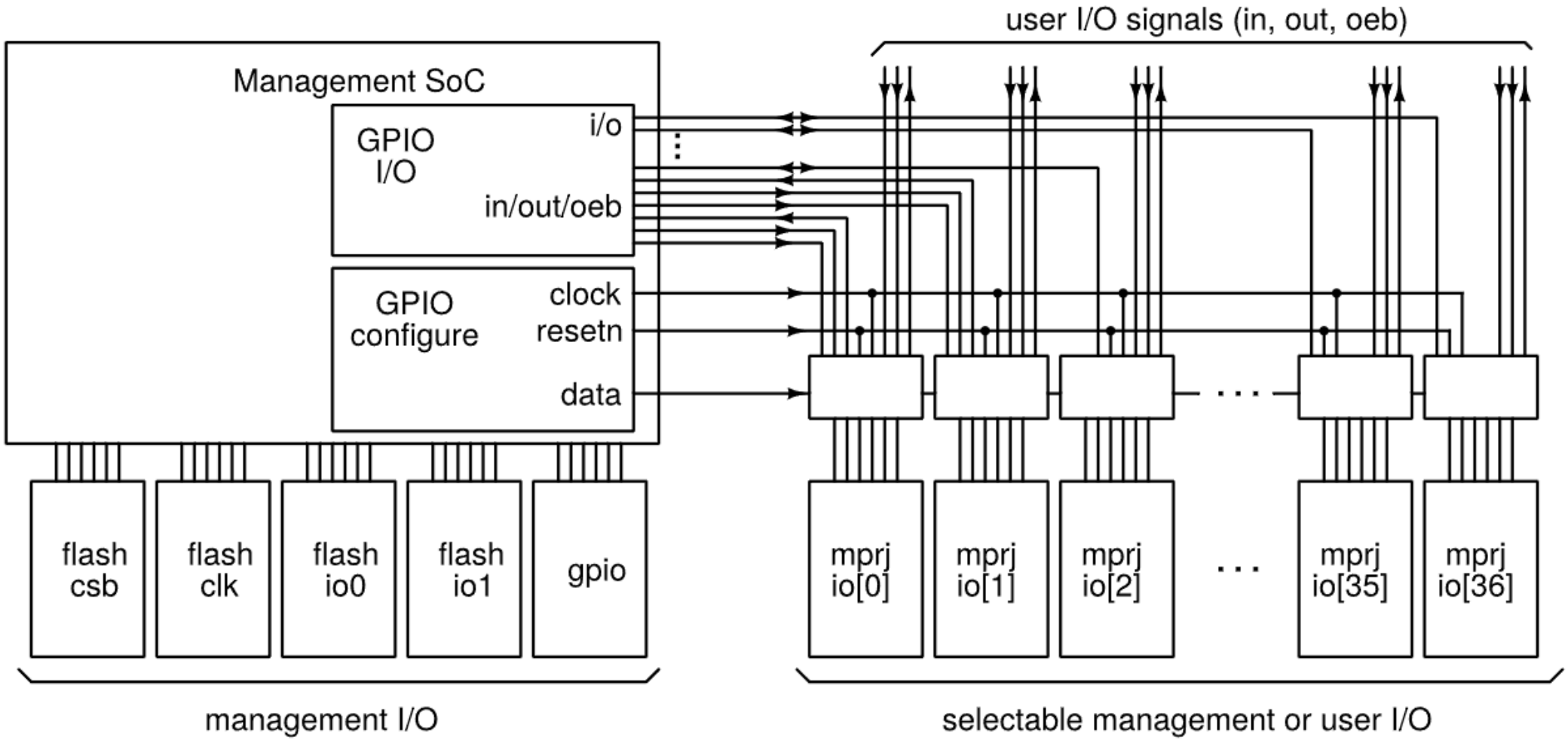



Single GPIO pad structure

Used for all pads except 0 and 1



# GPIO Pad Structure



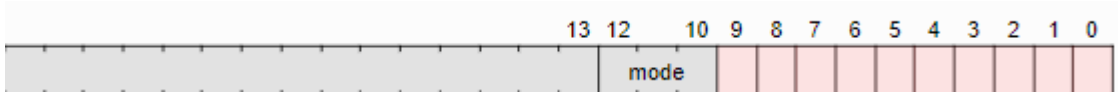
# GPIO Configuration

- Configuration settings define whether the GPIO is configured to connect to the user project area or the management SoC
- GPIOs are configured by assigning **predefined values** for each IO in the file rtl/header/**user_defines.v**
- Assigned configuration values for GPIO[5] thru GPIO[37], but GPIO[0] thru GPIO[4] are preset and cannot be changed.
- The following values are redefined for assigning to GPIOs:

GPIO_MODE_MGMT_STD_INPUT_NOPULL
GPIO_MODE_MGMT_STD_INPUT_PULLDOWN
GPIO_MODE_MGMT_STD_INPUT_PULLUP
GPIO_MODE_MGMT_STD_OUTPUT
GPIO_MODE_MGMT_STD_BIDIRECTIONAL
GPIO_MODE_MGMT_STD_ANALOG

GPIO_MODE_USER_STD_INPUT_NOPULL
GPIO_MODE_USER_STD_INPUT_PULLDOWN
GPIO_MODE_USER_STD_INPUT_PULLUP
GPIO_MODE_USER_STD_OUTPUT
GPIO_MODE_USER_STD_BIDIRECTIONAL
GPIO_MODE_USER_STD_OUT_MONITORED
GPIO_MODE_USER_STD_ANALOG

# mprj_io[i] Control Register Descriptions




| Mask bit | Default | Description |
|---|---|---|
| 10-12 | 001 | Digital mode |
| 9 | TODO | input voltage trip point select |
| 8 | 0 | slow slew (0 - fast slew, 1 - slow slew) |
| 7 | TODO | analog bus polarity |
| 6 | TODO | analog bus select |
| 5 | TODO | analog bus enable (0 - disabled, 1 - enabled) |
| 4 | TODO | IB mode select |
| 3 | 0 | input disable (0 - input enabled, 1 - input disabled) |
| 2 | 0 | hold override value (value is the value during hold mode) |
| 1 | 1 | output disable (0 - output enabled, 1 - output disabled) |
| 0 | 1 | management control enable (0 - user control, 1 - management control) |

| Digital mode bits (bit 12 11 10) | Digital mode description |
|---|---|
| 0 0 0 | analog function (all digital buffers disabled) |
| 0 0 1 | input (output disabled), no pullup or pulldown |
| 0 1 0 | input (output disabled), with pullup |
| 0 1 1 | input (output disabled), with pulldown |
| 1 0 0 | open drain (to power) output |
| 1 0 1 | open drain (to ground) output |
| 1 1 0 | output enabled |
| 1 1 1 | weak output (through 5k resistor) |

# Mprjp_io MMIO Registers

| Region | Address | Size |
|---|---|---|
| dff | 0x00000000 | 0x00000400 |
| sram | 0x01000000 | 0x00000800 |
| flash | 0x10000000 | 0x01000000 |
| hk | 0x26000000 | 0x00100000 |
| user project | 0x30000000 | 0x10000000 |
| csr | 0xf0000000 | 0x00010000 |
| vexriscv_debug | 0xf00f0000 | 0x00000100 |

| | | |
|---|---|---|
| **0x26 00 00 00** | User project area GPIO data transfer (bit 0, auto-zeroing) | Mirrors housekeeping register 0x13 |
| **0x26 00 00 04** | User project area GPIO power[0] configure | (These are currently undefined/unused.) Housekeeping register 0x6e |
| **0x26 00 00 0c** | User project area GPIO data (L) (GPIO 31 to 0) | Housekeeping registers 0x6a—0x6d |
| **0x26 00 00 10** | User project area GPIO data (H) (GPIO 37 to 32, upper bits unused) | Housekeeping register 0x69 |
| **0x26 00 00 24** | User project area GPIO mprj_io[0] configure | Housekeeping registers 0x1d, 0x1e— |
| ⋮ | ⋮ | ⋮ |
| **0x26 00 00 b8** | User project area GPIO mprj_io[37] configure | Housekeeping registers —0x67, 0x68 |

# rtl/header/user_defines.v

```
// Useful GPIO mode values.  These match the names used in defs.h.
`define GPIO_MODE_MGMT_STD_INPUT_NOPULL    13'h0403
`define GPIO_MODE_MGMT_STD_INPUT_PULLDOWN  13'h0c01
`define GPIO_MODE_MGMT_STD_INPUT_PULLUP    13'h0801
`define GPIO_MODE_MGMT_STD_OUTPUT          13'h1809
`define GPIO_MODE_MGMT_STD_BIDIRECTIONAL   13'h1801
`define GPIO_MODE_MGMT_STD_ANALOG          13'h000b

`define GPIO_MODE_USER_STD_INPUT_NOPULL    13'h0402
`define GPIO_MODE_USER_STD_INPUT_PULLDOWN  13'h0c00
`define GPIO_MODE_USER_STD_INPUT_PULLUP    13'h0800
`define GPIO_MODE_USER_STD_OUTPUT          13'h1808
`define GPIO_MODE_USER_STD_BIDIRECTIONAL   13'h1800
`define GPIO_MODE_USER_STD_OUT_MONITORED   13'h1802
`define GPIO_MODE_USER_STD_ANALOG          13'h000a
```

| Mask bit | Default | Description |
|---|---|---|
| 10-12 | 001 | Digital mode |
| 9 | TODO | input voltage trip point select |
| 8 | 0 | slow slew (0 - fast slew, 1 - slow slew) |
| 7 | TODO | analog bus polarity |
| 6 | TODO | analog bus select |
| 5 | TODO | analog bus enable (0 - disabled, 1 - enabled) |
| 4 | TODO | IB mode select |
| 3 | 0 | input disable (0 - input enabled, 1 - input disabled) |
| 2 | 0 | hold override value (value is the value during hold mode) |
| 1 | 1 | output disable (0 - output enabled, 1 - output disabled) |
| 0 | 1 | management control enable (0 - user control, 1 - management control) |

# rtl/header/user_defines.v

The power-on configuration for GPIO 0 to 4 is fixed and cannot be modified
(allowing the SPI and debug to always be accessible unless overridden by a flash program)

```
`define USER_CONFIG_GPIO_5_INIT  `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_6_INIT  `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_7_INIT  `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_8_INIT  `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_9_INIT  `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_10_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_11_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_12_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_13_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
```

```
// Configurations of GPIO 14 to 24 are used on caravel but not caravan.
`define USER_CONFIG_GPIO_14_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_15_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_16_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_17_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_18_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_19_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_20_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_21_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_22_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_23_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_24_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL

`define USER_CONFIG_GPIO_25_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_26_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_27_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_28_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_29_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_30_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_31_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_32_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_33_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_34_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_35_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_36_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
`define USER_CONFIG_GPIO_37_INIT `GPIO_MODE_MGMT_STD_INPUT_NOPULL
```

# firmware/caravel.h

```
#define reg_mprj_xfer (*(volatile uint32_t*)0x26000000)
#define reg_mprj_pwr  (*(volatile uint32_t*)0x26000004)
#define reg_mprj_irq  (*(volatile uint32_t*)0x26100014)
#define reg_mprj_datal (*(volatile uint32_t*)0x2600000c)
#define reg_mprj_datah (*(volatile uint32_t*)0x26000010)

#define reg_mprj_io_0 (*(volatile uint32_t*)0x26000024)
#define reg_mprj_io_1 (*(volatile uint32_t*)0x26000028)
#define reg_mprj_io_2 (*(volatile uint32_t*)0x2600002c)
#define reg_mprj_io_3 (*(volatile uint32_t*)0x26000030)
#define reg_mprj_io_4 (*(volatile uint32_t*)0x26000034)
#define reg_mprj_io_5 (*(volatile uint32_t*)0x26000038)
#define reg_mprj_io_6 (*(volatile uint32_t*)0x2600003c)

#define reg_mprj_io_7 (*(volatile uint32_t*)0x26000040)
#define reg_mprj_io_8 (*(volatile uint32_t*)0x26000044)
#define reg_mprj_io_9 (*(volatile uint32_t*)0x26000048)
#define reg_mprj_io_10 (*(volatile uint32_t*)0x2600004c)

#define reg_mprj_io_11 (*(volatile uint32_t*)0x26000050)
#define reg_mprj_io_12 (*(volatile uint32_t*)0x26000054)
#define reg_mprj_io_13 (*(volatile uint32_t*)0x26000058)
#define reg_mprj_io_14 (*(volatile uint32_t*)0x2600005c)

#define reg_mprj_io_15 (*(volatile uint32_t*)0x26000060)
#define reg_mprj_io_16 (*(volatile uint32_t*)0x26000064)
#define reg_mprj_io_17 (*(volatile uint32_t*)0x26000068)
#define reg_mprj_io_18 (*(volatile uint32_t*)0x2600006c)

#define reg_mprj_io_19 (*(volatile uint32_t*)0x26000070)
#define reg_mprj_io_20 (*(volatile uint32_t*)0x26000074)
#define reg_mprj_io_21 (*(volatile uint32_t*)0x26000078)
#define reg_mprj_io_22 (*(volatile uint32_t*)0x2600007c)
```

```
#define reg_mprj_io_23 (*(volatile uint32_t*)0x26000080)
#define reg_mprj_io_24 (*(volatile uint32_t*)0x26000084)
#define reg_mprj_io_25 (*(volatile uint32_t*)0x26000088)
#define reg_mprj_io_26 (*(volatile uint32_t*)0x2600008c)

#define reg_mprj_io_27 (*(volatile uint32_t*)0x26000090)
#define reg_mprj_io_28 (*(volatile uint32_t*)0x26000094)
#define reg_mprj_io_29 (*(volatile uint32_t*)0x26000098)
#define reg_mprj_io_30 (*(volatile uint32_t*)0x2600009c)
#define reg_mprj_io_31 (*(volatile uint32_t*)0x260000a0)

#define reg_mprj_io_32 (*(volatile uint32_t*)0x260000a4)
#define reg_mprj_io_33 (*(volatile uint32_t*)0x260000a8)
#define reg_mprj_io_34 (*(volatile uint32_t*)0x260000ac)
#define reg_mprj_io_35 (*(volatile uint32_t*)0x260000b0)
#define reg_mprj_io_36 (*(volatile uint32_t*)0x260000b4)
#define reg_mprj_io_37 (*(volatile uint32_t*)0x260000b8)
```

# testbench/counter_la/counter_la.c

```
void main()
{
...
        reg_mprj_io_31 = GPIO_MODE_MGMT_STD_OUTPUT;
...
        reg_mprj_io_16 = GPIO_MODE_MGMT_STD_OUTPUT;

        reg_mprj_io_15 = GPIO_MODE_USER_STD_OUTPUT;
..
        reg_mprj_io_0  = GPIO_MODE_USER_STD_OUTPUT;

        reg_mprj_io_6  = GPIO_MODE_MGMT_STD_OUTPUT;
...
    // Now, apply the configuration
    reg_mprj_xfer = 1;
    while (reg_mprj_xfer == 1);
...

    // Flag start of the test
    reg_mprj_datal = 0xAB400000;
...
    reg_mprj_datal = 0xAB510000;
}
```

# Topics

- System Block Diagram ( modules )
- Processor VexRISCV functions & its interface
- Reset / POR
- Management Protect Are – power control
- Clocking / DLL, configuration SPI register
- Housekeeping SPI registers
- GPIO
- SPI
- IRQ
- Memory-mapped IO address
- SRAM – Management area/Storage area
- Bus - Wishbone
- Peripherals – Counter/Timer/UART
- User project design (counter) Interface
- Testbench
- Firmware code

# Memory Regions

| Region | Address | Size |
|---|---|---|
| dff | 0x00000000 | 0x00000400 |
| sram | 0x01000000 | 0x00000800 |
| flash | 0x10000000 | 0x01000000 |
| hk | 0x26000000 | 0x00100000 |
| user project | 0x30000000 | 0x10000000 |
| csr | 0xf0000000 | 0x00010000 |
| vexriscv_debug | 0xf00f0000 | 0x00000100 |

| Region | Address | Registers |
|---|---|---|
| CTRL | 0xf0000000 | CTRL_RESET 0xf0000000<br>CTRL_SCRATCH 0xf0000004<br>CTRL_BUS_ERRORS 0xf0000008 |
| DEBUG_MODE | 0xf0000800 | DEBUG_MODE_OUT 0xf0000800 |
| DEBUG_OEB | 0xf0001000 | DEBUG_OEB_OUT 0xf0001000 |
| FLASH_CORE | 0xf0001800 | FLASH_CORE_MMAP_DUMMY_BITS 0xf0001800<br>FLASH_CORE_MASTER_CS 0xf0001804<br>FLASH_CORE_MASTER_PHYCONFIG 0xf0001808<br>FLASH_CORE_MASTER_RXTX 0xf000180c<br>FLASH_CORE_MASTER_STATUS 0xf0001810 |
| FLASH_PHY | 0xf0002000 | FLASH_PHY_CLK_DIVISOR 0xf0002000 |
| **GPIO** | 0xf0002800 | GPIO_MODE1 0xf0002800<br>GPIO_MODE0 0xf0002804<br>GPIO_IEN 0xf0002808<br>GPIO_OE 0xf000280c<br>GPIO_IN 0xf0002810<br>GPIO_OUT 0xf0002814 |

# Memory Regions

| Region | Address | Size |
|---|---|---|
| dff | 0x00000000 | 0x00000400 |
| sram | 0x01000000 | 0x00000800 |
| flash | 0x10000000 | 0x01000000 |
| hk | 0x26000000 | 0x00100000 |
| user project | 0x30000000 | 0x10000000 |
| csr | 0xf0000000 | 0x00010000 |
| vexriscv_debug | 0xf00f0000 | 0x00000100 |

| Region | Address | Registers |
|---|---|---|
| LA | 0xf0003000 | LA_IEN3-0 0xf0003000-0C<br>LA_OE3-0 0xf0003010-1C<br>LA_IN3-0 0xf0003020-2C<br>LA_OUT3-0 0xf0003030-3C |
| MPRJ_WB_IENA | 0xf0003800 | MPRJ_WB_IENA_OUT 0xf0003800 |
| SPI_ENABLED | 0xf0004000 | SPI_ENABLED_OUT 0xf0004000 |
| SPI Controller | 0xf0004800 | SPI_MASTER_CONTROL 0xf0004800<br>SPI_MASTER_STATUS 0xf0004804<br>SPI_MASTER_MOSI 0xf0004808<br>SPI_MASTER_MISO 0xf000480c<br>SPI_MASTER_CS 0xf0004810<br>SPI_MASTER_LOOPBACK 0xf0004814<br>SPI_MASTER_CLK_DIVIDER 0xf0004818 |

# Memory Regions

| Region | Address | Size |
|---|---|---|
| dff | 0x00000000 | 0x00000400 |
| sram | 0x01000000 | 0x00000800 |
| flash | 0x10000000 | 0x01000000 |
| hk | 0x26000000 | 0x00100000 |
| user project | 0x30000000 | 0x10000000 |
| csr | 0xf0000000 | 0x00010000 |
| vexriscv_debug | 0xf00f0000 | 0x00000100 |

| Region | Address | Registers |
|---|---|---|
| TIMER0 | 0xf0005000 | TIMER0_LOAD 0xf0005000<br>TIMER0_RELOAD 0xf0005004<br>TIMER0_EN 0xf0005008<br>TIMER0_UPDATE_VALUE 0xf000500c<br>TIMER0_VALUE 0xf0005010<br>TIMER0_EV_STATUS 0xf0005014<br>TIMER0_EV_PENDING 0xf0005018<br>TIMER0_EV_ENABLE 0xf000501c |
| **UART** | 0xf0005800 | UART_RXTX 0xf0005800<br>UART_TXFULL 0xf0005804<br>UART_RXEMPTY 0xf0005808<br>UART_EV_STATUS 0xf000580c<br>UART_EV_PENDING 0xf0005810<br>UART_EV_ENABLE 0xf0005814<br>UART_TXEMPTY 0xf0005818<br>UART_RXFULL 0xf000581c |
| **UART_ENABLED** | 0xf0006000 | UART_ENABLED_OUT 0xf0006000 |

# Memory Regions

| Region | Address | Size |
|---|---|---|
| dff | 0x00000000 | 0x00000400 |
| sram | 0x01000000 | 0x00000800 |
| flash | 0x10000000 | 0x01000000 |
| hk | 0x26000000 | 0x00100000 |
| user project | 0x30000000 | 0x10000000 |
| csr | 0xf0000000 | 0x00010000 |
| vexriscv_debug | 0xf00f0000 | 0x00000100 |

| Region | Address | Registers |
|---|---|---|
| USER_IRQ_0 | 0xf0006800 | USER_IRQ_0_IN 0xf0006800<br>USER_IRQ_0_MODE 0xf0006804<br>USER_IRQ_0_EDGE 0xf0006808<br>USER_IRQ_0_EV_STATUS 0xf000680c<br>USER_IRQ_0_EV_PENDING 0xf0006810<br>USER_IRQ_0_EV_ENABLE 0xf0006814 |
| USER_IRQ_1 | 0xf0007000 | Same as IRQ0 |
| USER_IRQ_2 | 0xf0007800 | Same as IRQ0 |
| USER_IRQ_3 | 0xf0008000 | Same as IRQ0 |
| USER_IRQ_4 | 0xf0008800 | Same as IRQ0 |
| USER_IRQ_5 | 0xf0009000 | Same as IRQ0 |
| USER_IRQ_ENA | 0xf0009800 | USER_IRQ_ENA_OUT 0xf0009800 |

# Question?