



Caravel Frame and SoC

Version 6.1.0

Caravel Frame and SoC documentation

Table of Contents

Caravel Overview	5
• How do I use this guide?	7
Features of Caravel	8
• Caravel Padframe General Features	9
• Caravel Management SoC Features	12
Caravel Pinouts	15
• Caravel pins and functions	16
• Caravel QFN-64 pinout	21
• Caravel bare die pinout	22
• Caravel WLCSP pinout	23
Getting Started	
• Essentials	25
• Block diagram	26
• caravel_user_project template	27
• Power-up/boot process	27
Clocking and DLL/DCO	29
• DLL (Delay-Locked Loop)	30
• DCO (Digitally-Controlled Oscillator)	31
• wb_clk_i	32
• user_clock2	33
• Clock monitoring	35
Firmware/Programming	
• Firmware Flash SPI interface	36
• Writing and compiling basic firmware for the Caravel RISC-V CPU	38
• Writing and simulating testbenches with Verilog or Cocotb	39
• Other important programming information	40
Caravel Full-chip Simulation	41

General Purpose Input/Output	42
• Management GPIO pin	42
• GPIO configuration by firmware or HKSPI	44
• GPIO power-on configuration by user_defines	45
• Standard GPIO configuration mode constants	46
Logic Analyzer	47
UART (RS232 Serial Interface)	48
Wishbone Interface	49
SPI Controller	14
Counter/Timer	51
Housekeeping and HKSPI	52
• Housekeeping SPI pins	53
• Housekeeping SPI protocol definition	54
• Housekeeping SPI modes	55
• Housekeeping SPI Pass-through mode	57
• Housekeeping SPI addresses	58
Interrupts (IRQs)	62
Registers and Memory Map	63
Analog Connections	64
• NOTES:	65
Advanced Guides	66
• Executing code from RAM	67
• Custom ISRs	68
• Power-on behavior	69
• Management Core wrapper	70
• Building Caravel using Litex	71
• Misc	72
Bringup (Caravel Eval)	

Caravan Specifics	74
• Common features	74
• Caravan differences	76
Caravel Mini Specifics	77
• Common features	74
• Caravel Mini differences	79
Supplementary Figures	80
Specifications and Ratings	81
• Absolute maximum ratings	82
Schematics and PCB Design	83
• KiCad schematic and footprint	84
• PCB design guidelines	85
• Caravel Eval Board and M.2 card	86
Glossary	87


Caravel Overview

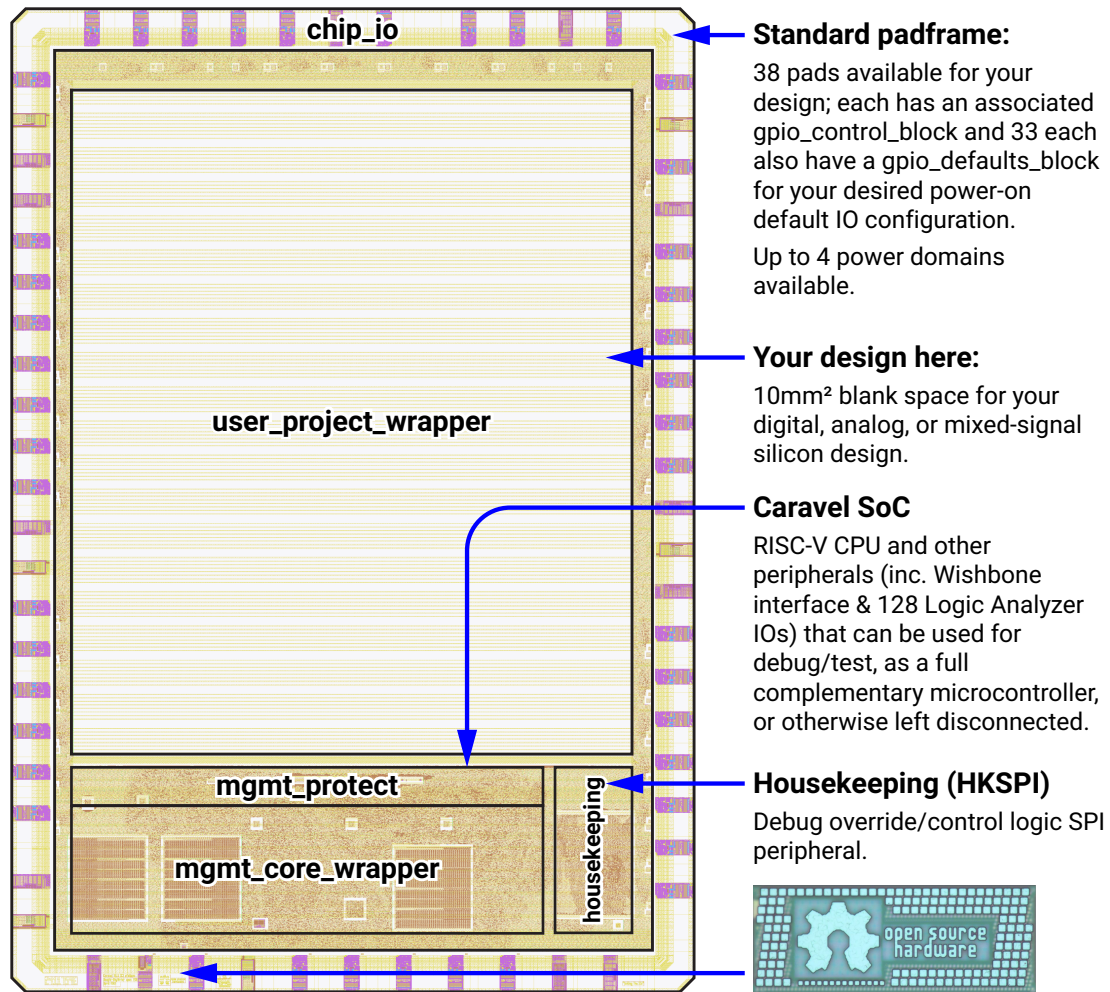
Designing and fabricating an ASIC with [Efabless chipIgnite](#)  requires that you use an existing supported template (or “frame”), and our free and open-source **Caravel** is recommended as the most popular, feature-rich, and production-ready template. The chipIgnite submission process automatically integrates your user project design area into the frame, so you don’t have to worry about it.

With **Caravel**, you have a ready-to-use chip harness for creating your own ASIC design and getting it fabricated for prototype or production purposes. It includes a **standardized padding**, blank silicon **design area of 10mm²** and optional **on-chip SoC** (microcontroller/management/test framework). You can use it whether you are creating a proprietary/private chip, one for commercial purposes, or an open-source design.

Hint

While this documentation describes **Caravel**, it also covers **Caravan** (our frame with some dedicated bare analog pads, suitable for prototyping) and **Caravel Mini**. All 3 are very similar, with some specific distinctions marked where appropriate.

For more information about the different frames and options, see: [Which chipIgnite template should I use?](#) 



Caravel die floorplan

How do I use this guide?

It is intended that you can work through each of the following sections in order to learn the basics of Caravel and how to use it, before reaching more advanced topics.

This guide covers general use of Caravel as a chip padding/harness for essential support of your design, as well as programming the Caravel SoC as a microcontroller and interfacing it with your design, and more-advanced concepts like analog connections.

Features of Caravel

As a padframe, Caravel offers easy-to-connect standard features such as GPIO pins, power delivery (optionally with multiple power domains), clocking, and reset logic (both a dedicated reset pin, and a **POR** circuit). These are well-characterized and qualified to reduce the overhead of the designer.

As an **SoC**, Caravel provides a wide range of optional functionality on-chip that can either be largely ignored or can be used for any of **bringing up** your design, debugging and providing a diagnostic/maintenance interface, controlling different peripherals that your design can leverage, or even as a full microcontroller (including CPU with external firmware interface, basic UART, and SPI controller).

Caravel Padframe General Features

These features are universally available to any chip based on the Caravel frame, even without specific use of the **Management SoC**.

User project wrapper

Caravel includes a 10mm² “user project wrapper” design area. The `user_project_wrapper` is what a user submits as a fixed-area **GDS** macro. It is then automatically integrated (by the Efabless chipIgnite submission process) into the rest of the Caravel chip harness to produce the final GDS that is submitted for fabrication. This design area, in the 130nm node, is enough for on the order of **6 million transistors or 1 million logic gate primitives**.

The designer can choose whether the design they include in the user project wrapper will interface with the Caravel SoC, the padframe GPIOs, the built-in clock sources, the **PDN** (Power Delivery Network) or any combination of these. For advanced users ordering bare dice and implementing a highly-specialized design, the user design may even use none of these features (though this is not recommended).

The user project wrapper provides **support for digital, analog, and mixed-signal projects**.

Housekeeping SPI

The chip’s power-on state defaults to assigning 4 GPIOs as a **Housekeeping SPI (HKSPI)** interface for an external SPI controller to assert debugging control over certain base configuration, debugging, and clocking of the chip.

This also includes **SPI pass-through**, able to be driven via HKSPI to take control (for reading/writing) of a firmware SPI ROM connected to the Management SoC.

See: [Housekeeping and HKSPI](#)

38 GPIOs

Caravel provides **38 GPIO pins** that can be used interchangeably by the user's own digital logic, the Caravel CPU, or in some cases as analog connections:

- These 38 can be configured (and reconfigured via CPU firmware or HKSPI) to function as outputs, bidirectional, or inputs (including optional pull-up or pull-down).
- They also have built-in ESD protection, level shifting, and buffering, thus simplifying the job of the designer.
- 33 support power-on default configuration specified in silicon by the designer, while the remaining 5 have Caravel-dedicated power-on functions (that can be overridden by CPU firmware or HKSPI).
- 29 optionally support direct pad connections for analog signals [1].
- For Caravan : 11 are “bare analog” pads without GPIO circuitry and without ESD protection.
- Some offer additional Caravel SoC “management” functions (such as UART and additional debug functions).

See: [General Purpose Input/Output](#)

Footnotes

[1]

Caravel direct analog pad connections include ESD protection which typically limits full swing signals to about 50MHz.

Dedicated clock input and DLL/DCO configurable clocking

Caravel and/or the user design can optionally receive (and modify) a clock signal via a dedicated clock input pin that includes circuitry for multiplying/dividing the input clock frequency.

Note

The DLL/DCO is inactive by default, passing the optional dedicated clock input directly through to the user project wrapper. If the DLL/DCO is to be used, it must be explicitly enabled via

HKSPI or firmware running on the **Management SoC** . For more information, see the section on **Clocking, DLL and DCO** .

See: [Clocking and DLL/DCO](#)

POR (Power-On Reset) module

See: [Power-up/boot process](#)

Four power domains

Caravel provides **4 power domains** : two intended as nominal 1.8V digital supplies, two intended as analog supplies in the range 1.8V to 5.5V. It also includes **vddio** for setting the desired external digital logic level (as a reference voltage in the range 1.8V-5.5V) for compatibility with a wide range of device.

Caravel Management SoC Features

The Management SoC's RISC-V CPU (RV32I) is built into the die area, adjacent the user project wrapper, and can be interfaced with your design, as well as externally to the chip.

The SoC is generated using [Litex](#) and includes the following peripherals and capabilities that can be optionally enabled/disabled on subsets of the GPIO pins, to make the SoC useful both as a general-purpose microcontroller and a specialized test/debug interface for your design...

VexRiscv RISC-V CPU core

The CPU core is a [VexRiscv](#) minimal+debug configuration. Intended for use either as a microcontroller, general-purpose CPU, control or debug interface to the user design. It can be considered as a lightweight single-core bare-metal microcontroller, programmable in C or RISC-V assembly (RV32I, 32-bit instructions specifically), and it comprises:

- **Dedicated firmware ROM SPI master** for [XIP](#) loading of firmware code from an external SPI memory into a local 16-word (64-byte) instruction cache.
- **1.5kByte** local SRAM for stack, scratch space/variables, or small high-speed in-memory executable subroutines.
- **Interrupt and exception handling**.
- **Dedicated power domain**.
- **GPIO control** : Ability to reconfigure the 38 GPIOs (**27 for Caravan**), including taking over GPIOs as “management mode”, either to activate SoC peripherals that have external interfaces, or for firmware to directly use some GPIOs.
- **Single management GPIO pin** . See: [Management GPIO pin](#)

See: [Firmware, Flash SPI, and Programming Guide](#)

The CPU core also has **access to a range of other SoC peripherals** as described below.

Note

If you don't intend to make use of the Management SoC at all, you can simply choose to not connect to its ports in the users project wrapper, and you can optionally tie its `RESETb` signal low externally to hold it in reset.

Logic Analyzer interface

The **Logic Analyzer** comprises 128 internal IO pins that can optionally be connected with your design in the user project wrapper.

See: [Logic Analyzer](#)

Wishbone master

The user project wrapper has an incoming **Wishbone master** interface from the CPU. This includes the `wb_clk_i` signal. It is implemented as a 32-bit Classic-Wishbone-based memory map expansion of the CPU for addresses in the range `0x30000000` – `0x300FFFFF`.

See: [Wishbone Interface](#)

UART

The SoC includes a UART that is accessible only to the CPU. It can be enabled to take over two specific GPIO pins for transmit and/or receive, and it has the following features:

- Fixed baud rate proportional to 9,600 baud at a 10MHz core clock (i.e. 19,200 baud if the Caravel core clock is set to 20MHz).
- Fixed 8N1 configuration.
- 16-byte FIFO for each of transmit and receive.
- TX/RX runs independently of the CPU.
- CPU can poll the FIFO state or enable an IRQ for the UART.

See: [UART \(RS232 Serial Interface\)](#)

SPI Controller

SPI master for direct control by user firmware.

See: [SPI Controller](#)

6 user IRQs

- 3 internally-driven by the user project.
- 2 externally-driven (optional) by GPIO pins.
- 1 internally-driven by SoC peripherals.

See: [Interrupts \(IRQs\)](#)

Counter-timers

See [Counter/Timer](#)

Caravel Pinouts

All chips fabricated using the Caravel harness have a standard pinout with some pins dedicated to the SoC (i.e. the CPU), some for power, most for user-configurable GPIOs, and a few with shared functions.

Caravel chips can be ordered in a 64-pin **QFN** package, or as bare dice (unpackaged, bare silicon chips).

Older generations of chipIgnite and the Open MPW shuttles also supplied **WLCSP** packaged parts.

Caravel pins and functions

Pin description

Name	Type	Description
<code>mprj_io[37:0]</code>	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.
<code>flash_clk</code>	Digital out	Flash SPI clock
<code>flash_csb</code>	Digital out	Flash SPI chip select
<code>flash_io[1:0]</code>	Digital I/O	Flash SPI data input/output
<code>clock</code>	Digital in	External CMOS 3.3V clock source
<code>resetb</code>	Digital in	SoC system reset (sense inverted)
<code>SD0</code>		Housekeeping serial interface data output

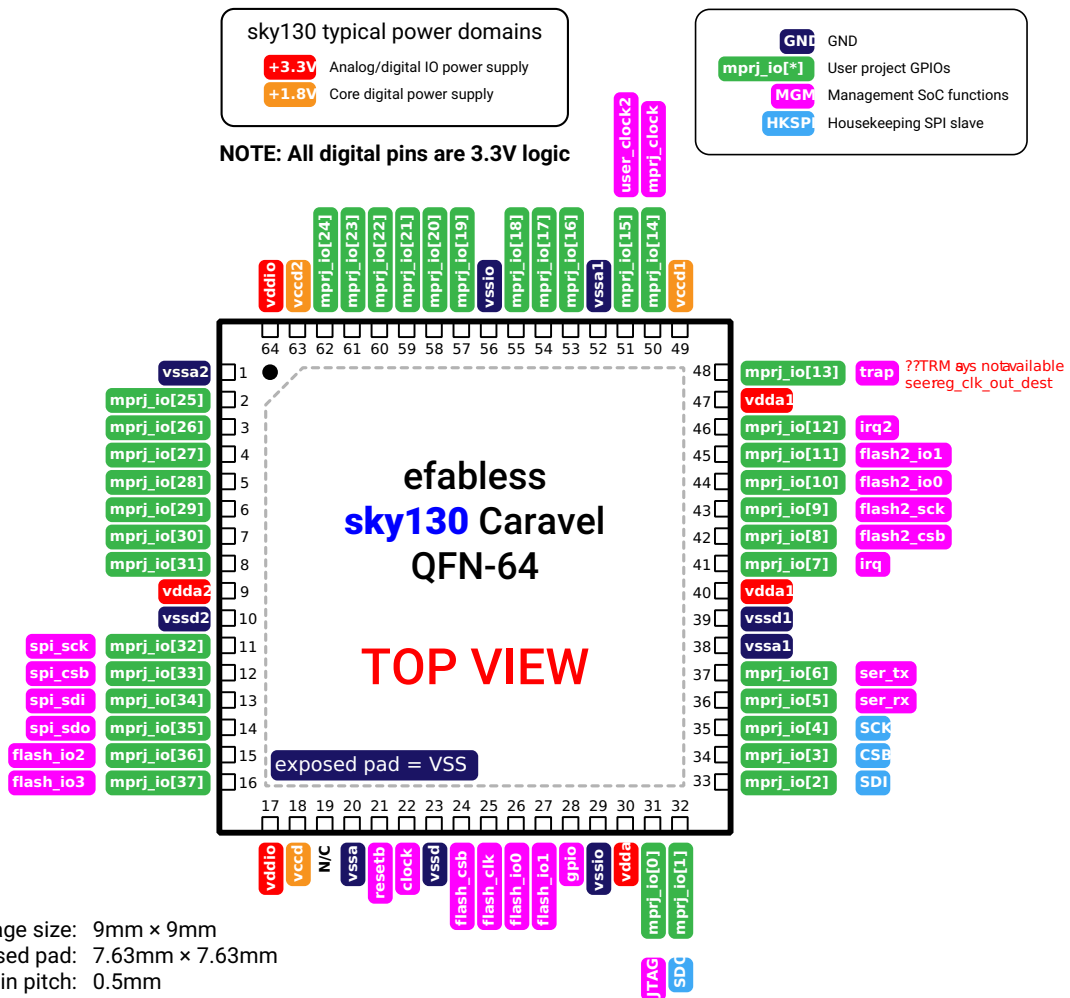
Name	Type	Description
	Digital out	
SDI	Digital in	Housekeeping serial interface data input
CSB	Digital in	Housekeeping serial interface chip select
SCK	Digital in	Housekeeping serial interface clock
ser_tx	Digital out	UART transmit channel
ser_rx	Digital in	UART receive channel
irq	Digital in	External interrupt
gpio	Digital I/O	Management GPIO/user power enable
JTAG	Digital I/O	JTAG system access
flash2_csb	Digital out	User area QSPI flash enable (sense inverted)

Name	Type	Description
flash2_sck	Digital out	User area QSPI flash clock
flash2_io[1:0]	Digital I/O	User area QSPI flash data
spi_sdo	Digital out	Serial interface controller data output
spi_sck	Digital out	Serial interface controller clock
spi_csb	Digital out	Serial interface controller chip select
spi_sdi	Digital in	Serial interface controller data input
vddio	3.3V Power	ESD and padframe power supply
vdda	3.3V Power	Management area power supply

Name	Type	Description
<code>vccd</code>	1.8V Power	Management area digital power supply
<code>vssio</code> / <code>vssa</code> / <code>vssd</code>	Ground	ESD, padframe, and management area ground
<code>vdda1</code>	3.3V Power	User area 1 power supply
<code>vccd1</code>	1.8V Power	User area 1 digital power supply
<code>vssa1</code>	Ground	User area 1 ground
<code>vssd1</code>	Ground	User area 1 digital ground
<code>vdda2</code>	3.3V Power	User area 2 power supply
<code>vccd2</code>	1.8V Power	User area 2 digital power supply
<code>vssa2</code>	Ground	User area 2 ground

Name	Type	Description
vssd2	Ground	User area 2 digital ground

Caravel QFN-64 pinout



Caravel QFN64 pinout

Caravel bare die pinout

Caravel bare dice have bond pads in a standard padding and are numbered starting at 1 on the top of the left-hand edge, incrementing counter-clockwise up to pad 63.

Caravel WLCSP pinout

Wafer-level chip-scale packaging is no longer offered by Efabless for standard chipignite orders, but may be specially-ordered and customized for large-volume production orders.

Older generations of Caravel chip already fabricated as WLCSP instead of QFN or bare dice had the following pinout:

Getting Started with Caravel

Essentials

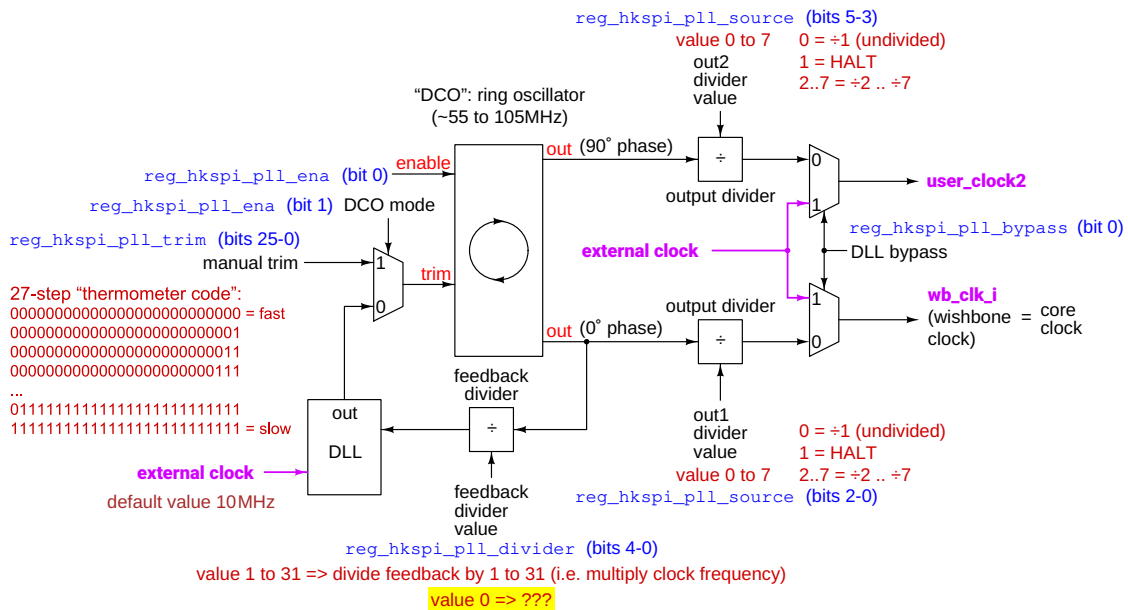
Block diagram

caravel_user_project template

Power-up/boot process

Clocking and DLL/DCO

Caravel has a dedicated **clock** input pin. While it is not mandatory to make use of this for any design submitted to chipignite, an external clock source is (in most cases) required in order for most parts of the Caravel SoC to be used.



Caravel clocking diagram

The Caravel SoC's RISC-V CPU derives its core clock from the **clock** pin, as do the following peripherals that can be used by the CPU:

- UART
- SPI Controller
- Counter/Timer

Note that **HKSPI** can be used without using the Caravel **clock** pin, as it has its own dedicated **SCK** clock input.

At power-on (and during any reset state) the clock signal present on the **clock** pin is passed, unmodified, directly to the CPU and other dependent peripherals, and is also available as an incoming signal in the user project area via both **wb_clk_i** and **user_clock2**.

The CPU or HKSPI may modify this clock path, e.g. to enable the **DLL** (and hence multiply and divide it to derive a different clock frequency), or to switch to using Caravel's internal ring oscillator (**DCO**) as the clock source.

While the Caravel SoC has limited clock source options, if *your own user project* requires a clock source then you might choose to use any of:

- `wb_clk_i` – recommended for synchronous interfacing with the Caravel CPU (e.g. via [Wishbone Interface](#) or [Logic Analyzer](#)).
- `user_clock2`
- Any GPIO input pin
- A Caravel-controlled [Logic Analyzer](#) pin
- Your own internal oscillator, e.g. a ring oscillator

Of the options given, `wb_clk_i` and `user_clock2` are preferred because they have [known timing characteristics](#) and can either be derived from Caravel's internal ring oscillator ([DCO](#)) or from Caravel's external dedicated [clock](#) input pin – in both cases optionally also being modified by the DLL/divider circuits as described. In turn this means that they can be controlled, to a degree, by firmware running on the CPU.

DLL (Delay-Locked Loop)

The Caravel DLL is like a PLL as found in an FPGA. The Caravel frame has a dedicated “clock” input pin.

DCO (Digitally-Controlled Oscillator)

This is an internal ring oscillator with a fixed base frequency and which can be “trimmed” by up to 26 steps to control its actual output frequency. It is used by the DLL to generate a **reasonably stable** multiple of the input clock source, but can also be used simply as a direct clock source instead of the **clock** input pin. In this mode it can optionally be divided by two independent integer dividers (to produce each of **wb_clk_i** and **user_clock2**).

Note that the actual internal DCO frequency is **PVT** -dependent.

wb_clk_i

`wb_clk_i` is the core clock used by the Caravel CPU and related peripherals. It is also available inside the [User Project Wrapper](#) .

user_clock2

When the DLL is enabled, the clock source feeding `user_clock2` 's divider is 90 degrees out of phase. That is, `user_clock2` lags `wb_clk_i` by a quarter-cycle.

Clock monitoring

It's possible to monitor `wb_clk_i` and/or `user_clock2` via GPIO pins, which can help with debugging your clock source and DLL/DCO behavior.

Caravel can use GPIOs 15 and 14 for this purpose, while Caravan (for chips fabricated as of June 2023) can use GPIOs 31 and 30 instead.

Enabling this feature requires setting the respective GPIO(s) to `GPIO_MODE_MGMT_STD_OUTPUT` mode, and then setting one or two bits of the clock monitor register: HKSPI register 0x1B (in firmware, this is `reg_clk_out_dest` or address `0x26200004`).

- For Caravel: Monitoring of `wb_clk_i` and `user_clock2` is via GPIOs 15 and 14 respectively. These are respectively enabled by writing `1` to bits 2 and/or 1 of the clock monitor register.
- For Caravan: Monitoring of `wb_clk_i` and `user_clock2` is via GPIOs 31 and 30 respectively. These are respectively enabled by writing `1` to bits 4 and/or 3 of the clock monitor register.

Firmware, Flash SPI, and Programming Guide

Firmware Flash SPI interface

Writing and compiling basic firmware for the Caravel RISC-V CPU

Writing and simulating testbenches with Verilog or Cocotb

Other important programming information

Caravel Full-chip Simulation

General Purpose Input/Output

Management GPIO pin

GPIO configuration by firmware or HKSPI

GPIO power-on configuration by user_defines

Standard GPIO configuration mode constants

GPIO_MODE_MGMT_STD_OUTPUT

TBC

Logic Analyzer

UART (RS232 Serial Interface)

Wishbone Interface

SPI Controller

Tip

Not to be confused with the **firmware SPI bus** or **HKSPI**.

Counter/Timer

Housekeeping and HKSPI

Housekeeping (HK) describes a subset of SoC control registers which – besides being addressable by the Caravel CPU – have been made externally-accessible through a “Housekeeping SPI” (HKSPI) interface. This interface coexists on four of the Caravel GPIO pins (`mprj_io[4:1]`) and is always enabled at power-on (but can be deactivated at run-time). Importantly, this means any simple external SPI controller can always take control over certain blocks of the chip frame/SoC. This feature is typically used for **bring-up** debugging purposes, and could be used for diagnostic/maintenance purposes in a field application.

With Housekeeping, you can externally access certain SoC registers (some read-only, some read/write) to inspect some aspects of the SoC state or otherwise control its behaviour, including to:

- Verify the chipIgnite product ID and read your chip’s unique **Project ID** .
- Alter the chip’s core clock paths/speed via **DLL** and **DCO** .
- Redirect the internal clock signals out via GPIO pins.
- Reset the Caravel RISC-V CPU.
- Reconfigure GPIO pin functions.
- Take over and optionally reprogram a firmware SPI Flash ROM chip connected to the Caravel CPU.

“HKSPI” is an **SPI** responder that can be accessed from an external controller (e.g. the **Caravel Eval Board**) through a standard 4-pin SPI serial interface. The SPI implementation is **mode 0** [↗](#) , with new data on **SDI** captured on the **SCK** rising edge, and output data presented on the falling edge of **SCK** (to be sampled on the next **SCK** rising edge). The SPI pins are shared with user area GPIO.

Housekeeping SPI pins

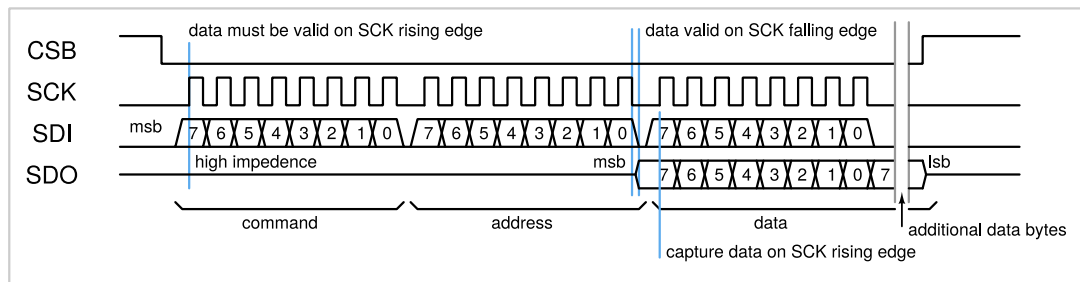
Housekeeping SPI external interface pins

GPIO pin	HKSPI pin	Dir	Function
<code>mprj_io[1]</code>	SDO	Output	Serial data out, clocked out on falling edge of <code>SCK</code>
<code>mprj_io[2]</code>	SDI	Input	Serial data in, clocked in on rising edge of <code>SCK</code>
<code>mprj_io[3]</code>	CSB	Input	“Chip Select bar” (falling edge starts an HKSPI transaction)
<code>mprj_io[4]</code>	SCK	Input	Serial clock.

Housekeeping SPI protocol definition

All input is in groups of 8 bits. Each byte is input MSB (most-significant-bit) first.

Every command sequence requires one command word (8 bits), followed by one address word (8 bits), followed by one or more data words (8 bits each), according to the data transfer modes described in [Housekeeping SPI modes](#).



Housekeeping SPI signalling

Addresses are read in sequence from lower values to higher values.

Therefore groups of bits larger than 8 should be grouped such that the lowest bits are at the highest address. Any bits additional to an 8-bit boundary should be at the lowest address.

Data is captured from the register map in bytes on the falling edge of the last SCK before a data byte transfer. Multi-byte transfers should ensure that data do not change between byte reads.

CSB pin must be low to enable an SPI transmission. Data are clocked by pin **SCK**, with data valid on the rising edge of **SCK**. Output data is received on the **SDO** line. **SDO** is held high-impedance when **CSB** is high and at all times other than the transfer of data bits on a read command. **SDO** outputs become active on the falling edge of **SCK**, such that data are written and read on the same **SCK** rising edge.

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

The first transferred byte is the command word, interpreted according to the [Housekeeping SPI command word definition](#).

Housekeeping SPI command word definition

Word	Meaning
00000000	No operation

Word	Meaning
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)

Note

All other words are reserved and act as no-operation if not defined by the SPI responder module.

Housekeeping SPI modes

The two basic modes of operation are **streaming mode** and **n-byte mode** .

In **streaming mode** operation, the data is sent or received continuously, one byte at a time, with the internal address incrementing for each byte. Streaming mode operation continues until **CSB** is raised to end the transfer.

In **n-byte mode** operation, 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 (note that a value of zero implies streaming mode). After **n** bytes have been read and/or written, the SPI returns to waiting for the next command. No toggling of CSB is required to end the command or to initiate the following command.

Housekeeping SPI Pass-through mode

The pass-through mode puts the CPU into immediate reset, then sets `FLASH_CSB` low to initiate a data transfer to the CPU's attached SPI flash. After the pass-through command byte has been issued, all subsequent SPI signaling on `SDI` and `SCK` are applied directly to the SPI flash (pins `FLASH_I00` and `FLASH_CLK`, respectively), and the SPI flash data output (pin `FLASH_I01`) is applied directly to `SDO`, until the `CSB` pin is raised. When `CSB` is raised, the `FLASH_CSB` is also raised, terminating the data transfer to the SPI flash. The CPU is brought out of reset, and starts executing instructions at the program start address.

This mode allows the SPI flash to be programmed from the same SPI communication channel as the housekeeping SPI, without the need for additional wiring to the SPI flash chip.

There are two pass-through modes. The first one corresponds to the primary SPI flash used by the management SoC. The second one corresponds to a secondary optional SPI flash that can be defined in the user project.

The pass-through mode allows a communications chip external to the Caravel chip program either SPI flash chip from a host computer without requiring separate external access to the SPI flash. Both pass-through modes only connect to I/O pins 0 and 1 of the SPI flash chips, and so must operate only in the 4-pin SPI mode. The user project may elect to operate the SPI flash in quad mode using a 6-pin interface.

Housekeeping SPI addresses

The purpose of the housekeeping SPI is to give access to certain system values and controls independently of the CPU. The housekeeping SPI can be accessed even when the CPU is in full reset. Some control registers in the housekeeping SPI affect the behaviour of the CPU in a way that can be potentially detrimental to the CPU operation, such as adjusting the trim value of the digital frequency-locked loop generating the CPU core clock.

While both the CPU and HKSPI can access the same registers that control/inspect certain SoC functions, the addresses are different between the two interfaces. Namely, accessing these registers via HKSPI uses an 8-bit address only, while accessing them via the CPU uses 32-bit addresses scattered through the range `0x26000000` – `0x262FFFFF` with no correlation between the addresses of the two interfaces.

Register Address	msb				lsb				
	7	6	5	4	3	2	1	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) (= 0x3ffeff)								default 0x3ffeff
0x11	unused		PLL output divider 2			PLL output divider			default 0x12
0x12	unused			PLL feedback divider					default 0x04

Housekeeping SPI register map

Housekeeping SPI registers

Name	Register address	Description
manufacturer_ID	0x01 (low 4 bits) and 0x02	The 12-bit manufacturer ID for efabless is 0x456
product_ID	0x03	The product ID for the Caravel harness chip is 0x10
user_project_ID	0x04 to 0x07	The 4-byte (32bit) user project ID is metal-mask programmed on each project before tapeout, with a unique number given to each user project.
PLL enable	0x08 bit 0	This bit enables the digital frequency-locked-loop clock multiplier. The enable should be applied prior to turning off the PLL bypass to allow the PLL time to stabilize before using it to drive the CPU clock.
PLL DCO enable	0x08 bit 1	The PLL can be run in DCO mode, in which the feedback loop to the driving clock is removed, and the system operates in free-running mode, driven by the ring oscillator which can be tuned between approximately 90 to 200MHz by setting the trim bits (check PLL trim) (NEED TO UPDATE THIS TO MATCH LEO'S RECENT CHARACTERIZATION and do some more char)
PLL bypass	0x09 bit 0	When enabled, the PLL bypass switches the clock source of the CPU from the PLL output to the external CMOS clock (pin C9). The default value is 0x1 (CPU clock source is the external CMOS clock).
CPU IRQ	0x0A bit 0	This is a dedicated manual interrupt driving the CPU IRQ channel 6. The bit is not self-resetting, so while the rising edge will trigger an interrupt, the signal must be manually set to zero before it can trigger another interrupt.

CPU reset	<code>0x0B</code> 0	bit	The CPU reset bit puts the entire CPU into a reset state. This bit is not self-resetting and must be set back to zero manually to clear the reset state
CPU trap	<code>0x0C</code> 0	bit	If the CPU has stopped after encountering an error, it will raise the trap signal. The trap signal can be configured to be read from a GPIO pin, but as the GPIO state is potentially unknowable, the housekeeping SPI can be used to determine the true trap state.
PLL trim	<code>0x0D</code> <code>0x10</code> (all bits) to (lower two bits)	(all bits) to (lower two bits)	The 26-bit trim value can adjust the DCO frequency over a factor of about two from the slowest (trim value <code>0x3ffffff</code>) to the fastest (trim value <code>0x0</code>). Default value is <code>0x3ffefff</code> (1 step higher than the slowest trim). Note that this is a thermometer-code trim, where each bit provides an additional (approximately) 250ps delay (on top of a fixed delay of 4.67ns). The fastest output frequency is approximately 215MHz while the slowest output frequency is approximately 90MHz (check PLL trim) (NEED TO UPDATE THIS TO MATCH LEO'S RECENT CHARACTERIZATION and do some more char)
PLL output divider	<code>0x11</code> 2-0	bits	The PLL output can be divided down by an integer divider to provide the core clock frequency. This 3-bit divider can generate a clock divided by 2 to 7. Values 0 and 1 both pass the undivided PLL clock directly to the core (and should not be used, as the processor does not operate at these frequencies).
PLL output divider (2)	<code>0x11</code> 5-3	bits	The PLL 90-degree phase output is passed through an independent 3-bit integer clock divider and provided to the user project space as a secondary clock. Values 0 and 1 both pass the undivided PLL clock, while values 2 to 7 pass the clock divided by 2 to 7, respectively.
PLL feedback divider	<code>0x12</code> 4-0	bits	The PLL operates by comparing the input clock (pin <code>C9</code>) rate to the rate of the PLL clock divided by the feedback divider value (when running in PLL mode, not DCO mode). The feedback divider must be set such that the external clock rate multiplied by the feedback divider value falls between 90 and 214 MHz (preferably centered on this range, or approximately 150 MHz) (check PLL trim) (NEED TO UPDATE THIS, and the calculation below, TO MATCH LEO'S

RECENT CHARACTERIZATION and do some more char) . For example, when using an 8 MHz external clock, the divider should be set to 19 ($19 * 8 = 152$). The DCO range and the number of bits of the feedback divider implies that the external clock should be no slower than around 4 to 5 MHz.

Interrupts (IRQs)

Registers and Memory Map

Analog Connections

NOTES:

- Allowing a pin to switch between digital and analog modes should be possible, but probably difficult. Might really only work well for OpenFrame, given how long it takes the CPU to signal everything in the GPIOs/chip.
- Comment on io_oeb and io_out when pads are configured for analog

Advanced Guides

Executing code from RAM

Custom ISRs

Power-on behavior

Management Core wrapper

The Caravel “Management Core Wrapper” is designed in a way to allow the implementation of different management cores. Early versions of the Google-sponsored “Open MPW” Caravel SoCs used a PicoRV32 core, while the current version officially offered by Efabless is “Caravel V6.0” (in use since Open MPW-6) and is generated using Litex with a VexRiscv CPU core.

The common parts of the management core wrapper include:

- Housekeeping and HKSPI.
- GPIO configuration blocks (power-on defaults and runtime-reprogrammable).
- User Project Wrapper interface pins, namely: Logic Analyzer; IRQs; Wishbone (inc. `wb_rst_i` , and `wb_clk_i` and `user_clock2` clock sources); and power rings (for a PDN of up to 4 power domains).
- Management protection.
- Clocking module (DLL/DCO).
- POR (Power-On Reset) module.

Building Caravel using Litex

Misc

- CPU as a testbench, e.g. using the CPU to drive **inputs** into UPW on GPIO pins.
- Reset behaviour, sources, options, e.g. is it a good idea to rely solely on *wb_rst_i* ?

Caravel Chip Bring-up and the Caravel Evaluation Board

Caravan Specifics

This is a summary of all things that are common and different between Caravan and the normal Caravel.

Common features

Caravan differences

Caravel Mini Specifics

This is a summary of all things that are common and different between Caravel Mini and the normal Caravel.

Common features

Caravel Mini differences

Supplementary Figures

Specifications and Ratings

Absolute maximum ratings

Type	minimum	typical	maximum	units
Supply voltage (VDDIO)	1.8	3.3	5.0	V
Core digital supply voltage (VCCD)	1.62	1.8	1.98	V
Junction temperature	-40	27	100	$^{\circ}\text{C}$
V_{OH}	$(0.8 \cdot \text{VDDIO})$		V	
V_{OL}			0.4	V
Management area power		TBD		mW
Storage area power		TBD		mW
GPIO voltage	0		VDDIO	V
GPIO frequency	0		50	MHz
Management clock	0		40	MHz

Schematics and PCB Design

KiCad schematic and footprint

PCB design guidelines

Caravel Eval Board and M.2 card

Glossary

bring-up

The process of getting a chip to operate for the first time after receiving it as a fabricated device. During this time, many measurements and experiments may be required, as well as debugging. This includes to verify that the core features of the frame/SoC are functioning as expected, and that your design is able to respond in a set of your own expectations.

Caravel Eval Board

The bring-up/development/evaluation board for chipIgnite/Caravel chips. Typically one board is supplied with every chipIgnite order that includes QFN-packaged Caravel chips. For more information, see https://github.com/efabless/caravel_board [↗](#) and note that you can [purchase a demo board from the Efabless Store](#) [↗](#) – the demo board includes 1 Caravel demo chip.

Caravel SoC

[Define me](#)

crt0

Initial “C Runtime” bootstrapping routines. Code built into the assembly process of a C program that is executed before the `main()` function is called. Responsible for loading the initial system/memory state, including initializing any global/static variables and optionally loading read-only data.

DLL

Delay-Locked Loop. Very similar to a PLL. In Caravel, [there is a DLL](#) which is an all-digital SoC peripheral that can be used to generate new clock frequencies from an internal or external clock source.

GDS

[Define me](#)

Hardening

The process of generating a final silicon layout (and hence [GDS](#) file) from potentially multiple parts, including synthesis of higher-level descriptions of digital logic.

Litex

[Define me](#)

Management Area

[Define me](#)

Management Core

Define me

Management Core Wrapper

Define me

Management SoC

Define me

PDN

Power Delivery Network.


PLL

Phase-Locked Loop. A device commonly used in FPGAs and in Caravel to derive a new clock frequency/phase from a supplied clock source. Typically allows for a clock source to be multiplied in frequency by an integer value, and then divided by a second integer value to produce a new clock frequency. Sometimes may offer multiple multipliers/dividers in order to produce multiple clocks. Compare: [DLL](#)

POR

Power-On Reset. A circuit that ensures a stable reset sequence during chip power-on, thus ensuring a stable system state if a dedicated external reset is not otherwise implemented.

Project ID

Every unique silicon layout (e.g. customer project) fabricated with Efabless chipIgnite has a unique 32-bit “Project ID” assigned by Efabless and included in the silicon layout. The Project ID is accessible by the Caravel SoC (and via [HKSPI](#)) as a read-only 32-bit value, but is also present as “GDS art” text in the padding, rendered as 8 hex digits. Most Project IDs are of the pattern `YYMMhhhh` where `hhhh` is a random value assigned by Efabless at the initialization of the project, and `YYMM` is the shuttle number (e.g. `2409`) and itself is formed of the last two digits of the shuttle year and the month number. An example Project ID (as a hex string) is `240476A0` which is [Tiny Tapeout 6](#) , on the April 2024 shuttle.

Note that when using the SoC or HKSPI to read the 32-bit value of the Project ID, some shuttles had the project ID bits in reverse order, e.g. `240476A0` (which in binary is `0010_0100_0000_0100_0111_0110_1010_0000`) would be read as `056E2024` (which is the binary string in reverse: `0000_0101_0110_1110_0010_0000_0010_0100`).

PVT

Short for “Process, Voltage, Temperature” and typically used in the context “PVT-dependent”, meaning that the exact behaviour/characteristics of something is affected (or otherwise likely to deviate from typical stated figures) by virtue of: variations that naturally occur in the fabrication process; variations in precise voltages in the circuit; and variations in ambient temperature.


QFN

Quad Flat No-leads IC package  . A plastic-encapsulated chip package with pin pads around all 4 sides.

SoC

System on a Chip. A combination of chip modules that provide a system of functionality, often including a CPU and other useful peripheral devices implemented in silicon.

SPI

Serial Peripheral Interface  . A common 4-wire interface for simple serial communication with a peripheral device, driven by a controller. Often used between chips, and capable of multi-megabit-per-second transfers.

UPW

Short for **User Project Wrapper** .

User Project Wrapper


The design area reserved for a user project. It has a fixed location and dimensions within the overall Caravel chip die area, and fixed pin placements around all 4 edges that a user design must connect to in order to interface with the Caravel SoC and/or GPIOs.

Typically the User Project Wrapper also has a **PDN** that is generated by the **hardening** flow.

VexRiscv

[Define me](#)

WLCSP

Wafer-Level redistribution Chip-Scale Package  . A minimal chip package usually with a “redistribution” layer that attaches bare bond pads of a silicon die to ball grid array (BGA) solder balls via tiny wires.

XIP

Execute In Place: Code is directly loaded and executed from an external memory as needed, without the need for user-driven caching control, buffering, translation, logic, etc.

