PROJECT NOTE

SKARAB ADC4x3G-14
Getting Started Guide

PRODUCT/PROCEDURE NAME:	SKARAB ADC4x3G-14
PRODUCT NUMBER:	N/A
DOCUMENT NAME:	PN-SKARAB ADC4x3G_14 GSG.doc
DOCUMENT VERSION:	1

VERSION HISTORY

Description of Change	Document Version	Edited by	Date
Initial Release	1	GT	08-August-18
Added Bypass-Mode Yellow Block Info	2	DFT	30-August-19

Contents

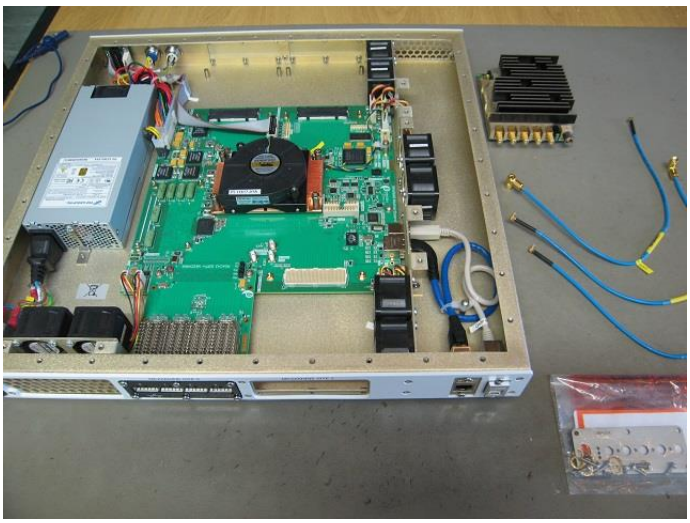
1 INSTALLATION PROCEDURE

The following section details the procedure to follow to install a SKARAB ADC4x3G-14 mezzanine module in a SKARAB LRU.

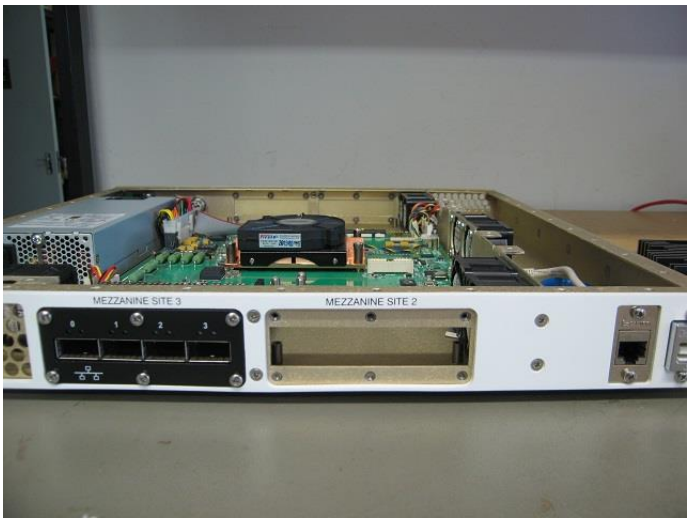
Before starting the installation, check that the following is available:

- ESD safe environment
- SKARAB LRU
- SKARAB ADC4x3G-14 mezzanine
- Pozidrive screwdriver suitable for M2 and M3 screws (the same screwdriver is used for all screws during the installation of the SKARAB ADC4x3G-14 mezzanine)
- Three ADC SYNC cables (CAB-INT-SIG-10281.01B)
- Packet of mechanicals for the SKARAB ADC4x3G-14 mezzanine
- Torque spanner / SMA spanner

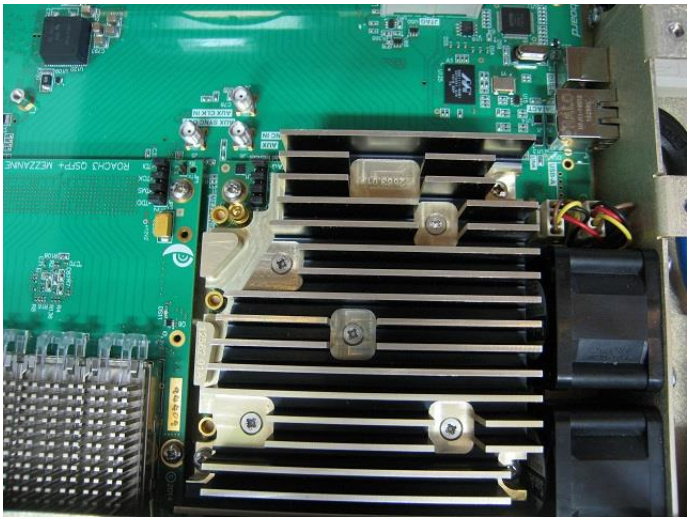
Remove the lid of the SKARAB LRU.



The SKARAB ADC4x3G-14 mezzanine module can be populated in any of the four mezzanine sites. If only a single SKARAB ADC4x3G-14 mezzanine module exists in a SKARAB LRU, then this is typically placed in mezzanine site 2. The QSFP+ mezzanine is populated in mezzanine site 3. Mezzanine site 2 has a blanking plate that is easily removed allowing the SKARAB ADC4x3G-14 mezzanine module to be populated on this mezzanine site. Mezzanine site 0 and 1 can be used but would require removing the SKARAB LRU front plate.



Feed the front panel SMA connectors of the SKARAB ADC4x3G-14 mezzanine through the opening of the chosen mezzanine site. Make sure that the mezzanine connector of the SKARAB ADC4x3G-14 mezzanine lines up with the motherboard mezzanine connector before pushing the SKARAB ADC4x3G-14 mezzanine connector into place.



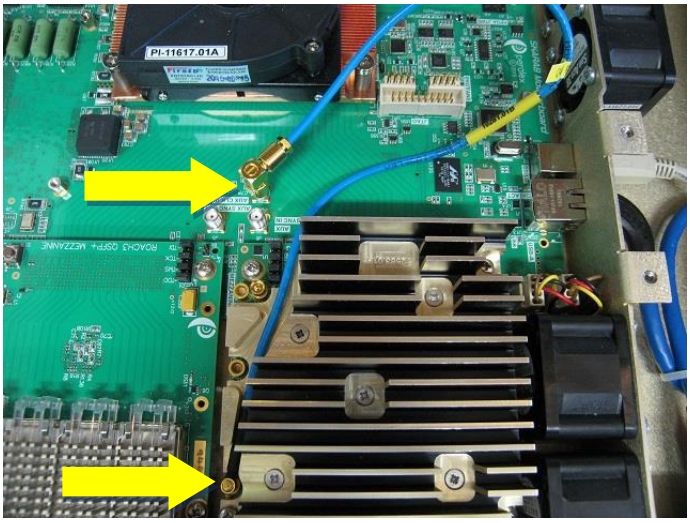
Once the SKARAB ADC4x3G-14 mezzanine is in place, use the following screws to secure the mezzanine module (orientation according to the above photo):

- Top left: M3 x 4 (with flat and wavy washer)
- Top right: M3 x 10 (with flat and wavy washer)
- Bottom left and bottom right: M3 x 12 (with flat and wavy washer)

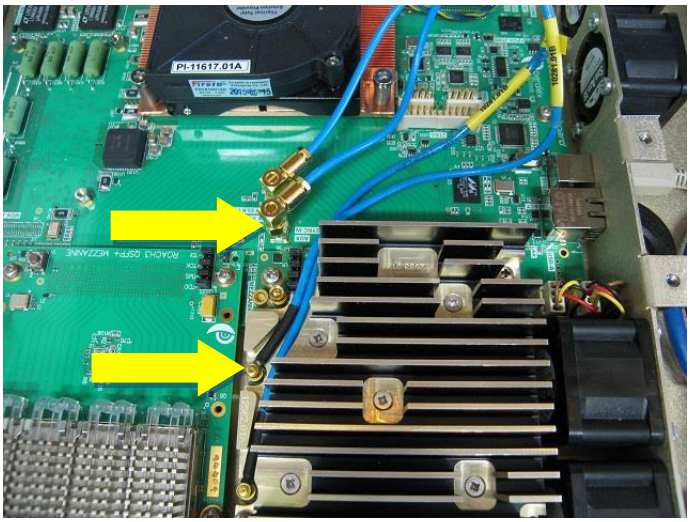
Secure the SKARAB ADC4x3G-14 mezzanine face plate with the following screws:

- Six M3 x 8
- Four M2 x 4
- SMA washers and nuts (use a torque spanner or SMA spanner to secure the SMA nuts)

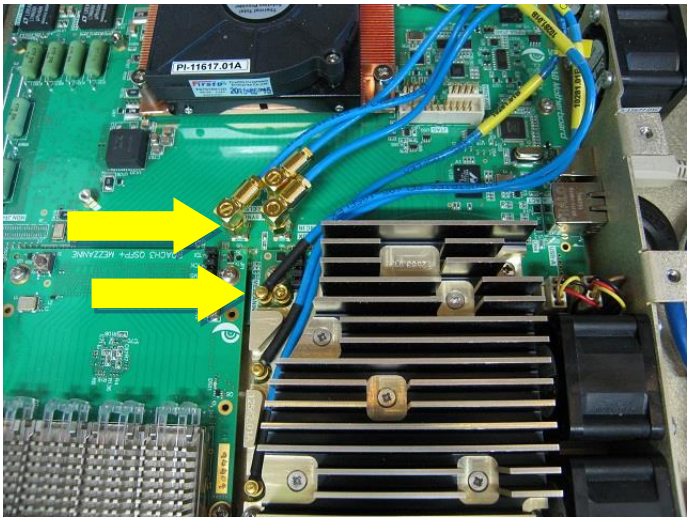
Connect the first ADC SYNC cable from AUX CLK IN to the SMP header closest to the face plate (see below).



Connect the second ADC SYNC cable from AUX SYNC IN to the middle SMP header (see below).



Connect the third ADC SYNC cable from AUX SYNC OUT to the top SMP header, closest to the edge of the PCB (see below).



Replace the SKARAB LRU lid using the Pozidrive screwdriver.

2 DETAILS OF THE SKARAB ADC4X3G-14 MEZZANINE

The hardware details of the SKARAB ADC4x3G-14 mezzanine are provided in the following hardware user manual: HUM-SKARAB ADC4X3G-01.doc

The following is a summary of the SKARAB ADC4x3G-14 mezzanine functionality to enable a discussion of how to interact with the SKARAB ADC4x3G-14 mezzanine.

The SKARAB ADC4x3G-14 mezzanine has two dual channel 14-bit 3Gsps ADCs. Depending on the build of the SKARAB ADC4x3G-14, one of two possible ADC devices can be populated on the board:

- TI ADC32RF80 ADC: This ADC supports a number of DDC options. The widest output bandwidth achievable is when this ADC is configured for decimate by 4. It is important to note that is this device does not support raw ADC output sample data. Only decimated output data is available. Depending on the configuration of the DDC this can be complex IQ samples or real samples centered on fs/4. See the ADC32RF80 datasheet for details of the output DDCs.
- TI ADC32RF45 ADC: This ADC supports a number of DDC options, as well as raw ADC output samples. It includes all of the functionality of the ADC32RF80 ADC. As mentioned, it also includes the option of raw 12-bit or 14-bit ADC sample data. Again it is important to note that there are limitations in terms of what the standard SKARAB LRU can support. The standard SKARAB LRU motherboard includes a XC7VX690T-2FFG1927C FPGA. The '-2' FPGA GTH transceivers support up to 11.3Gbps. There are fixed ratios for the ADC32RF45 ADC that define the JESD204B output data rate relative to the ADC sample rate. When configured for raw ADC samples, the ratios are as follows:
    - 14-bit raw ADC sample data: ratio = 5, therefore maximum ADC sample rate = 11.3 / 5 = 2.26Gsps.
    - 12-bit raw ADC sample data: ratio = 4, therefore maximum ADC sample rate = 11.3 / 4 = 2.825Gsps.

  Upon special request, it will be possible to populate the '-3' FPGA on the SKARAB LRU motherboard. The '-3' FPGA GTH transceivers support up to 13.1Gbps. With the '-3' FPGA populated on the SKARAB LRU motherboard:
    - 14-bit raw ADC sample data: ratio = 5, therefore maximum ADC sample rate = 13.1 / 5 = 2.62Gsps. However, it is limited by the ADC32RF45 to 2.5Gsps in this mode.
    - 12-bit raw ADC sample data: ratio = 4, therefore maximum ADC sample rate = 13.1 / 4 = 3.275Gsps. However, it is limited by the ADC32RF45 to 3Gsps in this mode.

The SKARAB ADC4x3G-14 mezzanine includes an ARM microcontroller to handle the configuration of the ADC sample clock generator and the ADCs themselves. The ARM microcontroller is accessed through the mezzanine I2C interface as an I2C slave device. A basic I2C memory map is implemented in the ARM microcontroller. This allows the SKARAB ADC43G-14 mezzanine to be managed from a remote host via the standard SKARAB LRU Ethernet command / response packets. Specifically, only accesses to the mezzanine I2C interface are required to completely control and monitor the status of the SKARAB ADC4x3G-14 mezzanine.

Most high level accesses are performed simply through this I2C register map. The ADC sample clock generator and the ADCs themselves are connected to the ARM microcontroller with an SPI bus. The ARM microcontroller receives the high level command via the I2C interface and automatically performs the required sequence of SPI writes and reads to execute the high level command. If, for any reason, it is necessary to perform low level SPI reads and writes that are not catered for as high level commands, then the I2C register map caters for this as well. It is possible to access the SKARAB ADC4x3G-14 mezzanine board level SPI bus via a sequence of I2C writes and reads.

The user accessible portion of the ARM microcontroller I2C memory map is detailed below:

FIRMWARE_VERSION_MAJOR_REG: Address 0x7E
FIRMWARE_VERSION_MINOR REF: Address 0x7F

Contains the ARM microcontroller embedded software version.

MEZ_CONTROL_REG: Address 0x01

BITS	RW	DESCRIPTION
0	W	Enable PLL SYNC: '1' = connect input SYNC signal to PLL SYNC input.
1	W	Enable ADC SYNC: '1' = connect input SYNC signal to ADC SYNC input.

2	RW	Update voltage readings: Write '1' = update I2C register map with latest on-board power supply voltage readings. Cleared to '0' by ARM microcontroller when voltage update complete.
3	RW	Update timestamp registers: Write '1' = update I2C register map with latest timestamp received from external GPS module. Cleared to '0' by ARM microcontroller when timestamp update complete.

1V9_AVDD_REG: Address 0x06 (MSB 15..8) 0x07 (LSB 7..0)

Contains 12 bit ADC value for +1V9_AVDD on-board power supply. It can be converted to a voltage using the following formula:
+1V9_AVDD voltage = 1V9_AVDD_REG x 3.3 / 4096

1V15_AVDD_REG: Address 0x08 (MSB 15..8) 0x09 (LSB 7..0)

Contains 12 bit ADC value for +1V15_AVDD on-board power supply. It can be converted to a voltage using the following formula:
+1V15_AVDD voltage = 1V15_AVDD_REG x 3.3 / 4096

1V15_DVDD_REG: Address 0x0A (MSB 15..8) 0x0B (LSB 7..0)

Contains 12 bit ADC value for +1V15_DVDD on-board power supply. It can be converted to a voltage using the following formula:
+1V15_DVDD voltage = 1V15_DVDD_REG x 3.3 / 4096

3V3_CLK_REG: Address 0x0C (MSB 15..8) 0x0D (LSB 7..0)

Contains 12 bit ADC value for +3V3_CLK on-board power supply. It can be converted to a voltage using the following formula:
+3V3_CLK voltage = 3V3_CLK_REG x 2 x 3.3 / 4096

TIMESTAMP_SECONDS_REG: Address 0x10 (MSB 31..24) 0x11 (23..16) 0x12 (15..8) 0x13 (LSB 7..0)
TIMESTAMP_MICROSECONDS_REG: Address 0x14 (MSB 31..24) 0x15 (23..16) 0x16 (15..8) 0x17 (LSB 7..0)

Contains the latest time from the external GPS module.

GAIN_CONTROL_REG: Address 0x18

This depends on the build of the SKARAB ADC4x3G-14 mezzanine. The default build of the SKARAB ADC4x3G-14 mezzanine module has the ADC input gain stages bypassed for maximum analogue input bandwidth.

BITS	RW	DESCRIPTION
1..0	W	Gain stage select: 0 = ADC input 0 1 = ADC input 1 2 = ADC input 2 3 = ADC input 3
6..2	W	Gain control value: 1dB steps -6dB = 21 0dB = 15 +15dB = 0
7	RW	Update gain: Write '1' = Update the ADC input gain stage. Cleared to '0' by ARM microcontroller when gain update complete.

DECIMATION_RATE_REG: Address 0x19

Specify the desired decimation rate of the DDCs inside the ADCs. Only applied when 'Apply DDC change' is set to '1'.

DDC0_NCO_SETTING_REG: Address 0x1A (MSB 15..8) 0x1B (LSB 7..0)

Specify the NCO register setting for DDC0 inside the ADCs. Only applied when 'Apply DDC change' is set to '1'. Calculate the NCO register setting as follows:
NCO register setting = 65536 x DDC center frequency / ADC sample rate

DDC1_NCO_SETTING_REG: Address 0x1C (MSB 15..8) 0x1D (LSB 7..0)

Specify the NCO register setting for DDC1 inside the ADCs. Only used when running in dual band mode. Only applied when 'Apply DDC change' is set to '1'. Calculate the NCO register setting as follows:
NCO register setting = 65536 x DDC center frequency / ADC sample rate

DDC_CONTROL_REG: Address 0x1E

BITS	RW	DESCRIPTION
1..0	W	ADC and DDC channel select: 2 = ADC input 0 0 = ADC input 1 3 = ADC input 2 1 = ADC input 3
2	W	Dual band enable: '1' = enable both DDC0 and DDC1
3	W	Nyquist zone: '0' = first Nyquist zone 1 = second Nyquist zone
6	W	Enable real DDC output: '1' = enable real DDC output samples (see the ADC datasheet for details)
7	RW	Apply DDC change: Write '1' = Update the DDC settings. Cleared to '0' by ARM microcontroller when DDC update complete.

DIRECT_SPI_ADDRESS_REG: Address 0x20 (MSB 15..8) 0x21 (LSB 7..0)

Specify address for a SPI transaction on the SKARAB ADC4x3G-14 mezzanine module.

DIRECT_SPI_DATA_REG: Address 0x22 (MSB 15..8) 0x23 (LSB 7..0)

Specify the write data for an SPI write transaction on the SKARAB ADC4x3G-14 mezzanine module. Contains the read data from an SPI read transaction on the SKARAB ADC4x3G-14 mezzanine module.

DIRECT_SPI_CONTROL_REG: Address 0x24

BITS	RW	DESCRIPTION
3..0	W	SPI destination: 0 = PLL 2 = ADC0 3 = ADC1 4 = GAIN0 5 = GAIN1 6 = GAIN2 7 = GAIN3 8 = ADC0 and ADC1
4	W	SPI read / not write: '0' = SPI write '1' = SPI read
7	RW	Start SPI transaction: Write '1' = Start SPI transaction. Cleared to '0' by ARM microcontroller when SPI transaction complete.

The ARM microcontroller on the SKARAB ADC4x3G-14 mezzanine starts up in bootloader mode by default. This allows for remote upgrades of the ARM microcontroller embedded software. In this bootloader mode, all three front panel LEDs are solid on. During startup of the SKARAB LRU, the Microblaze in the SKARAB LRU motherboard FPGA issues a sequence of I2C writes and reads to the ARM microcontroller to read the embedded software version of the bootloader before forcing the ARM microcontroller to exit bootloader mode and start executing the normal runtime embedded software. When in this mode, front panel LED 1 should constantly flash indicating that the ARM microcontroller is running. If the ARM microcontroller is staying in bootloader mode (front panel LEDs are solid on), confirm that the FPGA firmware generated for the SKARAB LRU motherboard includes the latest updates. The ARM microcontroller also has an RS232 interface via the front panel LEMO connector X0. Connecting this to a

terminal program on a host can also provide guidance as to the state of the ARM microcontroller. As detailed in the hardware user manual, configured the terminal for 115200 8N1.

The ARM microcontroller configured the ADC sample clock generator and the ADCs with the following default configuration when exiting bootloader model:

- ADC sample clock: 3GHz
- FPGA GTH reference clock: ADC sample clock divided by 16 = 187.5MHz
- ADC DDCs: decimate by 4, 1GHz center frequency, single band DDC, complex IQ output sample data, first Nyquist zone

There are two stages of synchronisation for the SKARAB ADC4x3G-14 mezzanine. The first stage is to synchronise the ADC sample clock generator PLL. This is referred to as PLL SYNC. This is required to synchronise the ADC sample clocks across both ADCs on the SKARAB ADC4x3G-14 mezzanine or across multiple SKARAB ADC4x3G-14 mezzanine modules. This ensures that phase accurate sampling can be achieved across ADCs. In terms of JESD204B, PLL SYNC ensures that the SYSREF clock within each ADC is aligned. PLL SYNC can only be performed if each SKARAB ADC4x3G-14 mezzanine module is provided with a 10MHz reference clock input. Note that it is possible to run the SKARAB ADC4x3G-14 mezzanine module without an external 10MHz reference (in other words, not perform PLL SYNC). The drawback of this is that the ADC sample clock will be off by the pull-range of the on-board VCXO.

PLL SYNC is typically performed before ADC SYNC. ADC SYNC is used to synchronise the JESD204B receiver firmware in the SKARAB LRU motherboard FPGA to the ADC output frames. ADC SYNC must be performed to read valid sample data out of the JESD204B receiver firmware embedded in the SKARAB ADC4x3G-14 Yellow Block.

Multiple steps are required to perform PLL SYNC and ADC SYNC. The detail of these steps is beyond the scope of this document. The steps are detailed in a Python script ‘adc_test.py’. This script performs the following:

- Read the ARM microcontroller embedded software version
- Configure the gain stage
- Configure the output DDC in the ADC
- Perform a basic test of the SKARAB ADC4x3G-14 SPI interface
- Optionally perform a PLL SYNC
- Perform an ADC SYNC
- Trigger ADC snapshots and write ADC DDC output samples to files

3 ADDING THE SKARAB ADC4X3G-14 MEZZANINE TO A DESIGN

The SKARAB ADC4x3G-14 mezzanine module has been integrated into the JASPER toolflow. Two Yellow Blocks have been developed. The first, 'skarab_adc4x3g14', should be used when using the DDCs of the module, while the second, skarab_adc4x3g14_byp, should be used when the DDCs are bypassed.

As detailed previously, the FPGA reference clock frequency is 187.5MHz. A cross clock FIFO is used inside both Yellow Blocks to move the ADC sample data to the DSP clock domain. When using the skarab_adc4x3g14 Yellow Block in a design, be sure to set the SKARAB DSP clock frequency higher than 187.5MHz to ensure that ADC sample data is not lost. When using the skarab_adc4x3g14_byp Yellow Block in a design, be sure to set the SKARAB DSP clock frequency higher than 280MHz to ensure that ADC sample data is not lost.

Note that the 'skarab_adc4x3g14' Yellow Block outputs four complex DDC samples (eight real DDC samples) in each DSP clock period. These are labelled as '…out0', '…out1', '…out2' and '…out3'. A sample from '…out0' represents the first in time of the four complex output samples and '…out3' represents the last in time of the four complex output samples. Note, however, that this ordering does change when configuring the DDC for real sample outputs or when running at DDC decimation rates other than four. Depending on the DDC decimation rate, some outputs will constantly be '0' because the corresponding JESD204B lane is unused. See the ADC datasheet for details of the JESD204B framing and how the DDC output samples are distributed across the JESD204B lanes. In summary, the 'skarab_adc4x3g14' Yellow Block assumes that data is distributed across all the JESD204B lanes. This is the case for decimate by four. Other decimation rates are supported but it will be up to the user of the 'skarab_adc4x3g14' Yellow Block to determine which of the outputs will contain valid DDC sample data and what the ordering of that data will be.

The ports of the 'skarab_adc4x3g14' Yellow Block are described below:

NAME	DIR	DESCRIPTION
adc_sync_start_in	IN	'0' to '1' transition starts ADC SYNC
adc_sync_complete_out	OUT	'1' when ADC SYNC complete, output data is now valid
pll_sync_start_in	IN	'0' to '1' transition starts PLL SYNC
pll_sync_complete_out	OUT	'1' when PLL SYNC complete
adc_trigger_out	OUT	'1' when trigger signal to mezzanine module is asserted, '0' otherwise
adc0_data_val_out	OUT	'1' when ADC0 output samples are valid
adc0_data_q_out0	OUT	ADC0 complex DDC Q sample 0 or real DDC sample 1
adc0_data_i_out0	OUT	ADC0 complex DDC I sample 0 or real DDC sample 3
adc0_data_q_out1	OUT	ADC0 complex DDC Q sample 1 or real DDC sample 0
adc0_data_i_out1	OUT	ADC0 complex DDC I sample 1 or real DDC sample 2
adc0_data_q_out2	OUT	ADC0 complex DDC Q sample 2 or real DDC sample 5
adc0_data_i_out2	OUT	ADC0 complex DDC I sample 2 or real DDC sample 7
adc0_data_q_out3	OUT	ADC0 complex DDC Q sample 3 or real DDC sample 4
adc0_data_i_out3	OUT	ADC0 complex DDC I sample 3 or real DDC sample 6
adc1_data_val_out	OUT	'1' when ADC1 output samples are valid
adc1_data_q_out0	OUT	ADC1 complex DDC Q sample 0 or real DDC sample 1
adc1_data_i_out0	OUT	ADC1 complex DDC I sample 0 or real DDC sample 3
adc1_data_q_out1	OUT	ADC1 complex DDC Q sample 1 or real DDC sample 0
adc1_data_i_out1	OUT	ADC1 complex DDC I sample 1 or real DDC sample 2
adc1_data_q_out2	OUT	ADC1 complex DDC Q sample 2 or real DDC sample 5
adc1_data_i_out2	OUT	ADC1 complex DDC I sample 2 or real DDC sample 7
adc1_data_q_out3	OUT	ADC1 complex DDC Q sample 3 or real DDC sample 4
adc1_data_i_out3	OUT	ADC1 complex DDC I sample 3 or real DDC sample 6
adc2_data_val_out	OUT	'1' when ADC2 output samples are valid
adc2_data_q_out0	OUT	ADC2 complex DDC Q sample 0 or real DDC sample 1
adc2_data_i_out0	OUT	ADC2 complex DDC I sample 0 or real DDC sample 3
adc2_data_q_out1	OUT	ADC2 complex DDC Q sample 1 or real DDC sample 0
adc2_data_i_out1	OUT	ADC2 complex DDC I sample 1 or real DDC sample 2
adc2_data_q_out2	OUT	ADC2 complex DDC Q sample 2 or real DDC sample 5
adc2_data_i_out2	OUT	ADC2 complex DDC I sample 2 or real DDC sample 7

adc2_data_q_out3	OUT	ADC2 complex DDC Q sample 3 or real DDC sample 4
adc2_data_i_out3	OUT	ADC2 complex DDC I sample 3 or real DDC sample 6
adc3_data_val_out	OUT	'1' when ADC3 output samples are valid
adc3_data_q_out0	OUT	ADC3 complex DDC Q sample 0 or real DDC sample 1
adc3_data_i_out0	OUT	ADC3 complex DDC I sample 0 or real DDC sample 3
adc3_data_q_out1	OUT	ADC3 complex DDC Q sample 1 or real DDC sample 0
adc3_data_i_out1	OUT	ADC3 complex DDC I sample 1 or real DDC sample 2
adc3_data_q_out2	OUT	ADC3 complex DDC Q sample 2 or real DDC sample 5
adc3_data_i_out2	OUT	ADC3 complex DDC I sample 2 or real DDC sample 7
adc3_data_q_out3	OUT	ADC3 complex DDC Q sample 3 or real DDC sample 4
adc3_data_i_out3	OUT	ADC3 complex DDC I sample 3 or real DDC sample 6

Note that the 'skarab_adc4x3g14_byp' Yellow Block outputs ten real samples in each DSP clock period. These are labelled as '…out0', '…out1', '…out2', '…out3', '…out4', '…out5', '…out6', '…out7', '…out8', '…out9'. A sample from '…out0' represents the first in time of the ten output samples and '…out9' represents the last in time of the ten output samples.

The ports of the 'skarab_adc4x3g14_byp' Yellow Block are described below:

NAME	**DIR**	**DESCRIPTION**
adc_sync_start_in	IN	'0' to '1' transition starts ADC SYNC
adc_sync_complete_out	OUT	'1' when ADC SYNC complete, output data is now valid
pll_sync_start_in	IN	'0' to '1' transition starts PLL SYNC
pll_sync_complete_out	OUT	'1' when PLL SYNC complete
adc_trigger_out	OUT	'1' when trigger signal to mezzanine module is asserted, '0' otherwise
adc0_data_val_out	OUT	'1' when ADC0 output samples are valid
adc0_data_out0	OUT	ADC0 real sample 0
adc0_data_out1	OUT	ADC0 real sample 1
adc0_data_out2	OUT	ADC0 real sample 2
adc0_data_out3	OUT	ADC0 real sample 3
adc0_data_out4	OUT	ADC0 real sample 4
adc0_data_out5	OUT	ADC0 real sample 5
adc0_data_out6	OUT	ADC0 real sample 6
adc0_data_out7	OUT	ADC0 real sample 7
adc0_data_out8	OUT	ADC0 real sample 8
adc0_data_out9	OUT	ADC0 real sample 9
adc1_data_val_out	OUT	'1' when ADC1 output samples are valid
adc1_data_out0	OUT	ADC1 real sample 0
adc1_data_out1	OUT	ADC1 real sample 1
adc1_data_out2	OUT	ADC1 real sample 2
adc1_data_out3	OUT	ADC1 real sample 3
adc1_data_out4	OUT	ADC1 real sample 4
adc1_data_out5	OUT	ADC1 real sample 5
adc1_data_out6	OUT	ADC1 real sample 6
adc1_data_out7	OUT	ADC1 real sample 7
adc1_data_out8	OUT	ADC1 real sample 8
adc1_data_out9	OUT	ADC1 real sample 9
adc2_data_val_out	OUT	'1' when ADC2 output samples are valid
adc2_data_out0	OUT	ADC2 real sample 0
adc2_data_out1	OUT	ADC2 real sample 1
adc2_data_out2	OUT	ADC2 real sample 2
adc2_data_out3	OUT	ADC2 real sample 3
adc2_data_out4	OUT	ADC2 real sample 4
adc2_data_out5	OUT	ADC2 real sample 5
adc2_data_out6	OUT	ADC2 real sample 6
adc2_data_out7	OUT	ADC2 real sample 7
adc2_data_out8	OUT	ADC2 real sample 8
adc2_data_out9	OUT	ADC2 real sample 9
adc3_data_val_out	OUT	'1' when ADC3 output samples are valid

adc3_data_out0	OUT	ADC3 real sample 0
adc3_data_out1	OUT	ADC3 real sample 1
adc3_data_out2	OUT	ADC3 real sample 2
adc3_data_out3	OUT	ADC3 real sample 3
adc3_data_out4	OUT	ADC3 real sample 4
adc3_data_out5	OUT	ADC3 real sample 5
adc3_data_out6	OUT	ADC3 real sample 6
adc3_data_out7	OUT	ADC3 real sample 7
adc3_data_out8	OUT	ADC3 real sample 8
adc3_data_out9	OUT	ADC3 real sample 9