PF32 CORRELATION SYSTEM: ARTIX 7 USER REFERENCE GUIDE

V1.*1 relating to PF32_USBC[corr_1.08].bit*

WARRANTY AND DISCLAIMERS

Use of this product and the associated software implies acceptance of the Photon Force terms of use.

Bare electronic components are susceptible to electrostatic discharge (ESD). Failure to employ good practice in handling the hardware, or to observe and comply with the warnings and handling precautions stated in this guide will void the product warranty.

This product may not be used in military, aerospace, medical or other safety critical applications without the express written permission of Photon Force Ltd. Any such use is undertaken entirely at the customer's own risk.

Specifications are subject to change without notice.

CONTENTS

Warranty and disclaimers 2

CONTENTS 3

System Overview 4

Introduction 4

Special Cases 4

First Usage 4

Correlation System Software Description 5

New API Methods 5

Data Format 6

Data Footer 6

Correlation FPGA Hardware Description 7

Correlation Processing Phase FSM 7

Correlation Data Readout, G2 Forming and Flush FSM 8

Host PC USB Endpoint Read Example 9

Representative Process Timing 10

Representative Process Timing for Artix 7 Readout 10

FPGA hardware Test modes 11

FPGA Hardware test modes 11

SYSTEM OVERVIEW

INTRODUCTION

The PF32 system and Xilinx Artix 7 FPGA fabric provides sufficient capacity to accommodate and the high-speed calculation and delivery of g2 data to the host PC via USB.

What's included:

- ⇨ Control and readout of PF32 sensor
- ⇨ Optimised hardware DSP and device memory management implementing correlation g2 function
- ⇨ User-programmable correlation period, Tint, from 18-65,535
- ⇨ Fixed 16 Tau data processing
- ⇨ Full hardware calculation and delivery of g2 data in 2.30 format
- ⇨ G2 (2.30) data delivery every Tint, supporting sensor frame rates up to 500kfps
- ⇨ 32bit product and 24-bit normalisation term datapaths able to accommodate up to 65,535 photons/pixel/Tint period
- ⇨ Ability to handle TintFrames up to 65,535
- ⇨ 4 input marker channels which are logged in a packet footer

What's not included quite yet:

- ⇨ Exponential Binning with programmable points
- ⇨ Extended 64 Tau data processing

This document serves as an overview for the user to help appreciate how to control the system and access the g2 data using the system software and hardware.

SPECIAL CASES

In the event that no photons are received for a given pixel during the Tint period, all Tau values for that pixel will be set to 1.

FIRST USAGE

Before using the correlator features on a PF32 camera for the first time, the camera may need to be enabled to run the correlator firmware. To do this, please connect and power on the PF32, and run the PF32_updatePFSerialNumber.exe executable included with the correlator software release.

CORRELATION SYSTEM SOFTWARE DESCRIPTION

NEW API METHODS

The methods for using the correlator have not yet been added to the public API but the following are currently being used internally while the correlator functionality is being developed. Therefore there is the possibility that the method signatures may change or new methods may be added.

void setTintFrames(PF32_HANDLE hnd, unsigned long tintFrames);

Sets the number of target frames for the correlator.

void enableCorrelator(PF32_HANDLE hnd, bool enable, bool testMode);

Pass in enable=true to enable the correlator before performing a read. Pass in enable=false afterwards to disable it again.

The testMode flag can be set true or false – refer to the FPGA Hardware test modes *section for details.*

long readFromCorrelator(PF32_HANDLE hnd, uint8_t * raw, uint8_t * footer);

This method performs the correlator read. The method expects the raw pointer to be allocated sufficient memory to hold the number of bytes returned by getSizeOfCorrelatorData(). The raw data will be written to this location. Similarly, the footer pointer should be allocated SIZE_OF_CORRELATOR_FOOTER bytes (1,024) of memory to receive the correlation packet footer.

unsigned int getSizeOfCorrelatorData(PF32_HANDLE hnd);

*The number of bytes required to contain the correlator data. This is the enabled number of pixels per frame (changes if Regions of Interest is set) * the number of tau values (16 for the current Artix 7 implementation) * 4 bytes per tau value.*

unsigned int getNoOfTauValuesPerPixel();

This is hard-coded to be 16 for the current Artix 7 implementation.

void convertCorrelatorOutput(uint8_t * raw, double * converted, unsigned int enabledHeight, unsigned int width);

For convenience, we provide convertCorrelatorOutput method to convert the raw, fixed-point data (32 bits in 2.30 format) produced by the correlator into standard double precision floating point representation. This is achieved by dividing each raw value by 2^30. This functionality has been put into a separate function so it can be called at a later time if need be. The data from the correlator is expected in the raw pointer, as written to by calling readFromCorrelator(). The converted form will be written to the converted pointer. The enabledHeight parameter is the number of enabled rows if Regions of Interest is set. Width is the number of columns (currently always 32) and this will not change if Regions of Interest is set.

DATA FORMAT

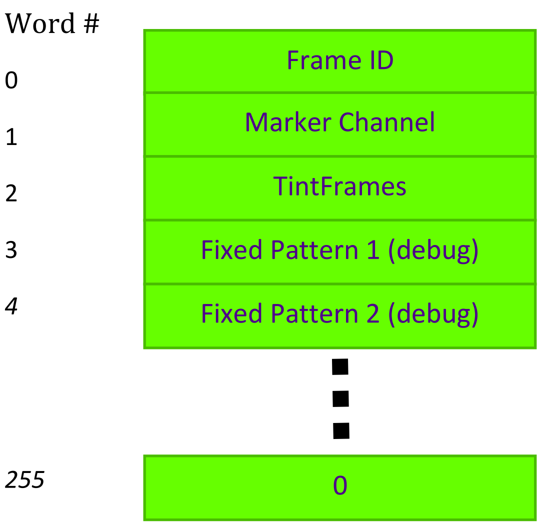
The correlator data is a fixed size regardless of how many frames are correlated over. The data can be iterated over using three for loops:

```
For each row
    For each tau value
        For each column
        End for
    End for
End for
```

DATA FOOTER

Following readout of the correlation data, a special packet is prepared to deliver key system-level status information.

The data footer is presently 256×32 bits (1K bytes) in size and is arranged as follows



Data in each word is little endian.

Frame ID (32 bits) is the frame number associated with the first frame of the correlation sequence.

Marker Channel (4 bits) it is used to indicate if any of the external PF32 sync signals (pixel, line, frame or blanking) were asserted at any point during the correlation process running over TintFrames:

Blanking flag (bit 3), Frame flag (bit 2) , Line flag (bit 1), Pixel flag (bit 0)

Bits 31 to 4 are currently unused.

The marker channel flags are reset by the hardware each time a new correlation process is run.

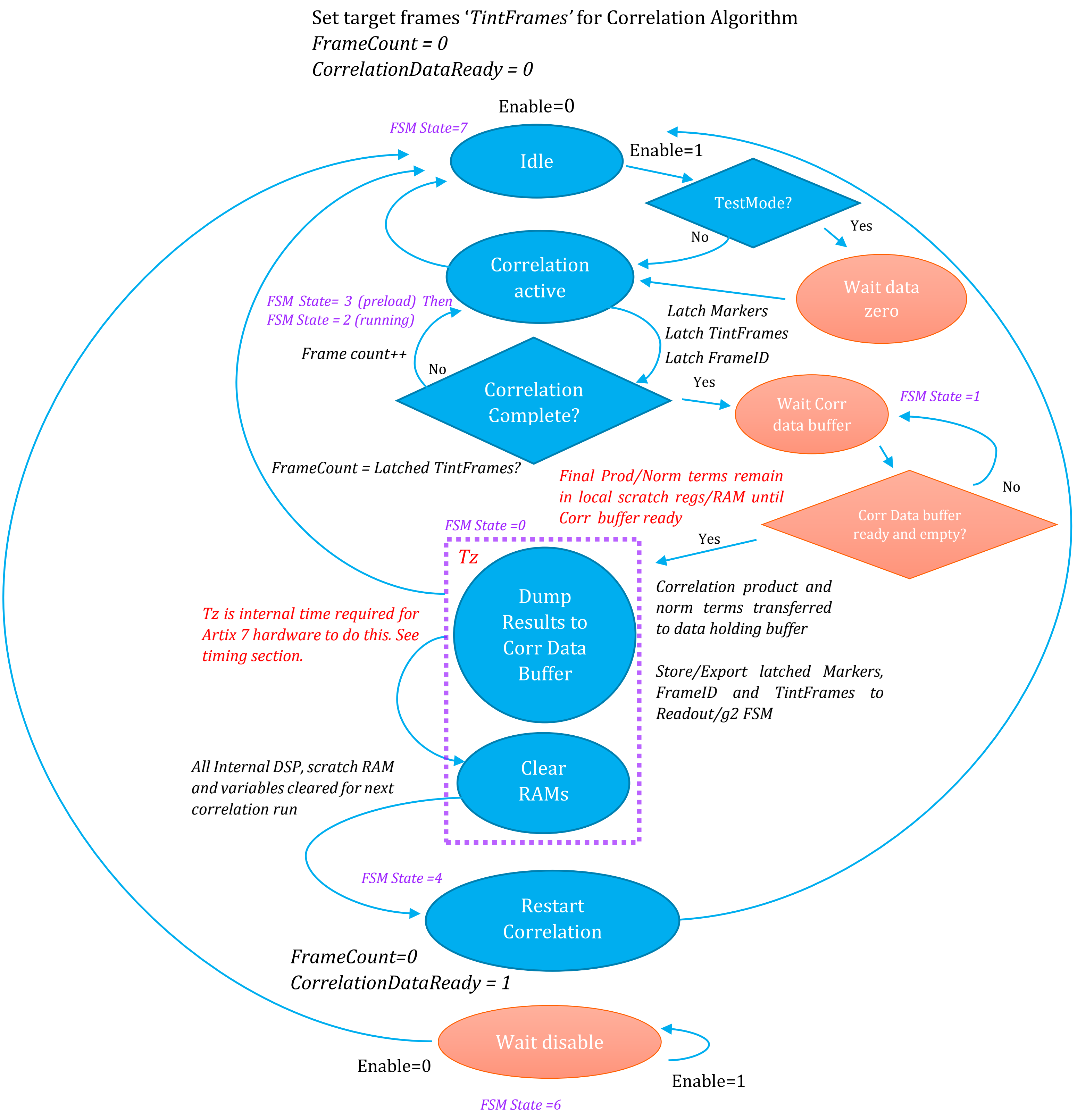
TintFrames is the TintFrames value used for the current data set.

Fixed Pattern 1 is currently for system debug only and is always 0xAAAAAAAA (32 bits)

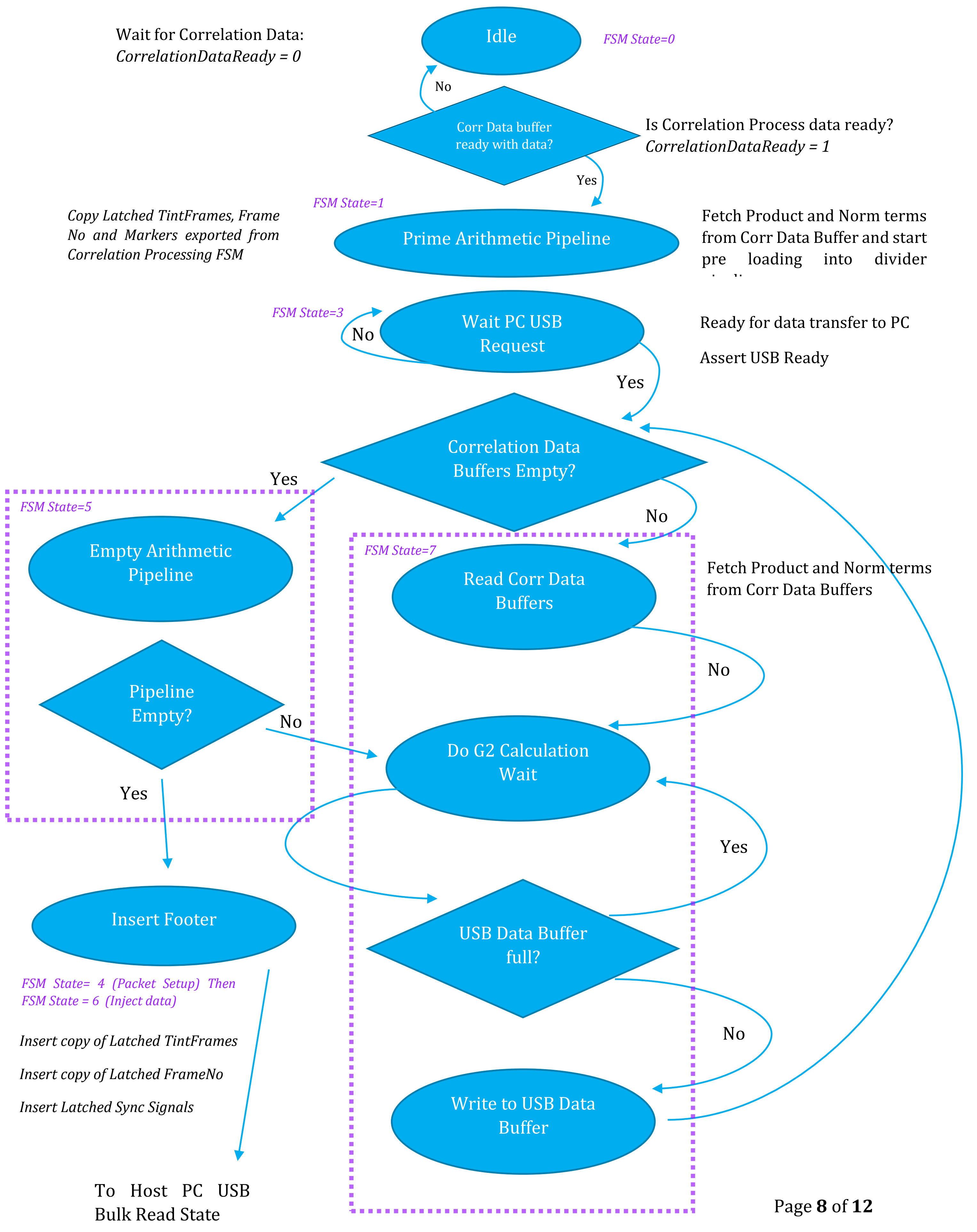
Fixed Pattern 2 is currently for system debug only and is always 0x55555555 (32 bits)

All other words in the footer data are currently set to 0x00000000.

CORRELATION PROCESSING PHASE FSM

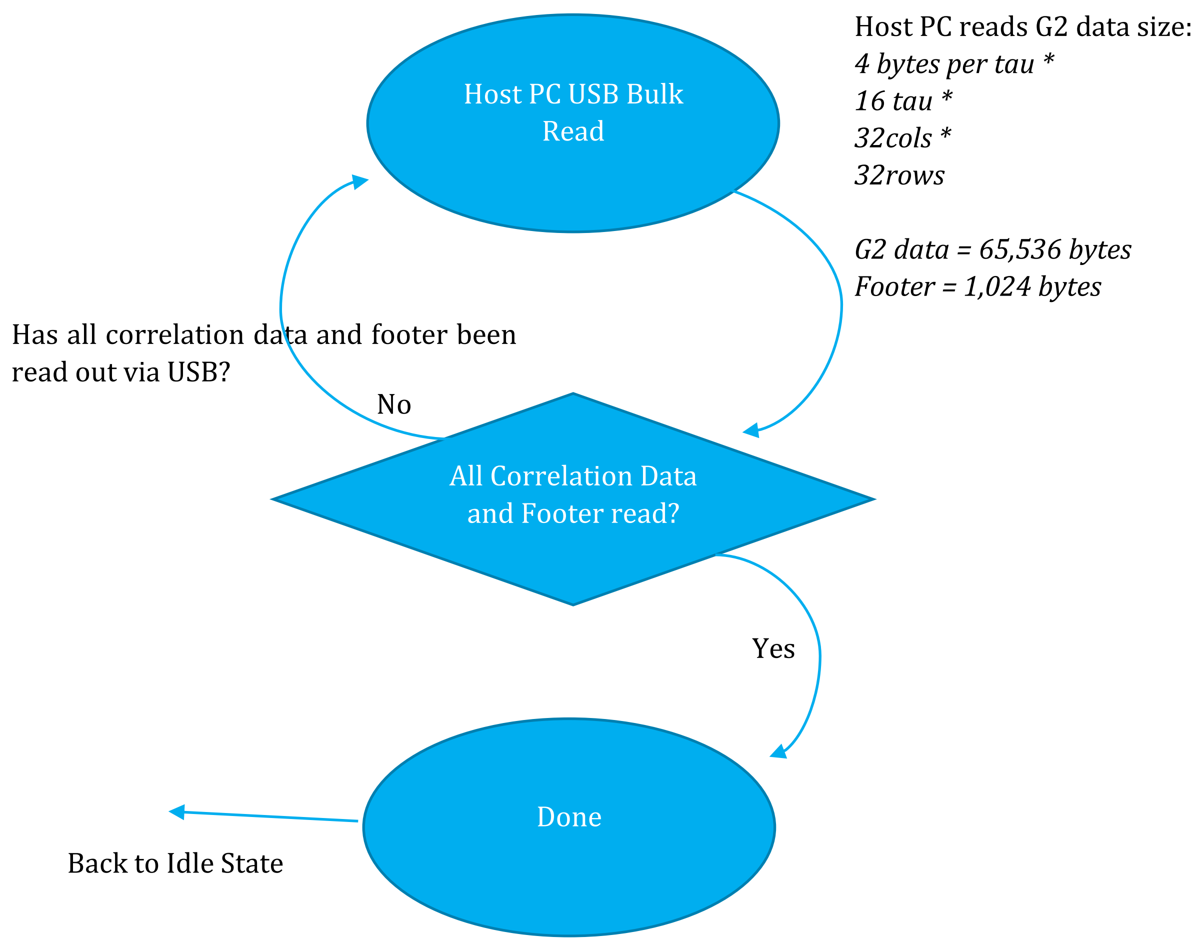

Set target frames '*TintFrames'* for Correlation Algorithm
FrameCount = 0
CorrelationDataReady = 0
Enable=0
FSM State=7
Idle
Enable=1
TestMode?
Yes
No
Wait data zero
Correlation active
FSM State= 3 (preload) Then
FSM State = 2 (running)
Latch Markers
Latch TintFrames
Latch FrameID
Frame count++
No
Correlation Complete?
Yes
Wait Corr data buffer
FSM State =1
FrameCount = Latched TintFrames?
Final Prod/Norm terms remain in local scratch regs/RAM until Corr buffer ready
Corr Data buffer ready and empty?
No
Yes
FSM State =0
Tz
Dump Results to Corr Data Buffer
Correlation product and norm terms transferred to data holding buffer
Tz is internal time required for Artix 7 hardware to do this. See timing section.
Store/Export latched Markers, FrameID and TintFrames to Readout/g2 FSM
Clear RAMs
All Internal DSP, scratch RAM and variables cleared for next correlation run
FSM State =4
Restart Correlation
FrameCount=0
CorrelationDataReady = 1
Wait disable
Enable=0
Enable=1
FSM State =6

CORRELATION DATA READOUT, G2 FORMING AND FLUSH FSM



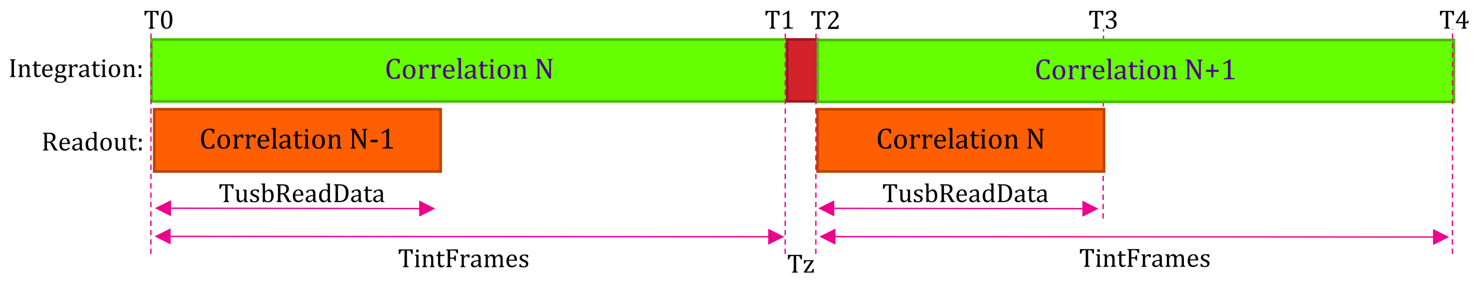
Wait for Correlation Data:
CorrelationDataReady = 0
Idle
FSM State=0
No
Corr Data buffer ready with data?
Is Correlation Process data ready?
CorrelationDataReady = 1
Yes
FSM State=1
Copy Latched TintFrames, Frame No and Markers exported from Correlation Processing FSM
Prime Arithmetic Pipeline
Fetch Product and Norm terms from Corr Data Buffer and start pre loading into divider
FSM State=3
No
Wait PC USB Request
Ready for data transfer to PC
Assert USB Ready
Yes
Correlation Data Buffers Empty?
Yes
No
FSM State=5
Empty Arithmetic Pipeline
Pipeline Empty?
No
Yes
FSM State=7
Read Corr Data Buffers
Fetch Product and Norm terms from Corr Data Buffers
No
Do G2 Calculation Wait
Yes
USB Data Buffer full?
No
Write to USB Data Buffer
Insert Footer
FSM State= 4 (Packet Setup) Then FSM State = 6 (Inject data)
Insert copy of Latched TintFrames
Insert copy of Latched FrameNo
Insert Latched Sync Signals
To Host PC USB Bulk Read State

HOST PC USB ENDPOINT READ EXAMPLE



REPRESENTATIVE PROCESS TIMING

REPRESENTATIVE PROCESS TIMING FOR ARTIX 7 READOUT



TintFrames
Frame Integration Time * the number of Frames required for the correlation process. The correlator integration time is always composed of contiguous frames – there will be no gaps or dropped frames.

TusbReadData
USB Data Readout Time = time required to readout all of the correlation G2 data via USB.

*For the current Artix 7 implementation, G2 data is presented as 32 bits (in 2.30 fixed point format), hence 4 bytes. The total correlator packet data payload size is therefore = bytes per tau (4) * Number of tau (16) * Number of cols (32) * Number of rows (32) = 65,536 bytes. Note that this followed by a 1,024 byte packet footer, as discussed in the* Data Footer *section.*

⇨ *TusbReadData = Datasize(e.g.65536)/(1024 bytes***)** *= 64µs (assumes 100% USB bandwidth with no down time – in real world, this will not be absolutely correct).*

Tz
Time required for internal hardware to safely transfer product and normalisation data for all pixels and for all tau to the data storage buffer, after which a new correlation process can be restarted. Typically Tz required will be:

For the current Artix 7 implementation, number of tau (16) x Num of rows (16) x Fsys_clk (e.g. 160MHz) = 1.6µs*

**NB for timing calculation, correlation top and bottom blocks run in parallel with 16 rows each*

Assumption

In order for the current Artix 7 implementation of the correlation process to complete, and accommodate time Tz to transfer data internally between processing units and data buffers, it is recommended that a minimum value of *TintFrames* should be set to being at least *n x TusbReadData (n=2) to allow the USB data to be fully transferred beforehand e.g. from above, 2 x 64µs/Tint (e.g. 1µs) so 128 frames (minimum) ** to be confirmed.*

FPGA HARDWARE TEST MODES

The FPGA includes some test modes allowing the hardware to deliver deterministic data patterns which can be useful for arithmetic test and system integration.

The test modes are likely to be extended in sophistication as the design evolves.

Both test modes may be set using the enableCorrelator API method.

At present we have:

1. TestMode

When Testmode is asserted, the correlation module waits for a known pixel data value (presently hard-coded to zero) to trigger the start of data processing.

To do this, the FPGA includes a means to swap out the PF32 sensor data with a fixed test pattern where every pixel in the array is forced to be the same value, and that this value increases by one each frame.

The internal correlator pixel data bus width is 8 bits, so data increments from 0 – 255 then repeats.

Photon Force Ltd.

2nd Floor,
Murchison House,
10 Max Born Crescent
Edinburgh,
EH9 3BF,
Scotland, UK.