SRS - Short User Guide

Contents

1 APV readout - Raw data mode (ADC mode)

1.1 Front-end initialisation

The default power up values of the APV25 and ADC-Card registers do not allow correct operation so an initialization sequence has to be performed:

```
\\ setting ADC C/card registers default values
WITH ADCCARD_PORT                \\ write ADC C-CARD registers

    WRITE HYBRID_RST_N     0x00   \\ (addr 0x00) reset all APVs
    WRITE PWRDOWN_CH0      0x00   \\ (addr 0x01) power on master channels circuitry
    WRITE PWRDOWN_CH1      0x00   \\ (addr 0x02) power on slave channels circuitry
    WRITE EQ_LEVEL_0       0x00   \\ (addr 0x03) equalization set to 0
    WRITE EQ_LEVEL_1       0x00   \\ (addr 0x04) equalization set to 0
    WRITE TRGOUT_ENABLE    0x00   \\ (addr 0x05) when PLL is used disable trg out
    WRITE BCLK_ENABLE      0xFF   \\ (addr 0x06) enable clock&trg outputs

    WRITE HYBRID_RST_N     0xFF   \\ (addr 0x00) deasert APV reset
                                  \\ reset is initally asserted and later deaserted
                                  \\ to create a reset pulse for the APV registers
END WITH
```

```
\\ setting APV registers default values
WITH APV_PORT subaddress = 0xFF03      \\ write to registers of all APVs
    WRITE default values               \\ see default values
                                       \\ in Slow Control Manual
```

```
\\ resync APVs. This will reset the hybrid PLL and readout circuitry of the APVs
WITH APVAPP_PORT
    WRITE (0xFFFFFFFF)     0x0001      \\ force sync reset of all APVs
```

```
\\ optional adjust of PLL phase (for longer HDMI cables)
WITH APV_PORT subaddress = 0xFF00      \\ all PLLs
    WRITE CSR1_FINEDELAY phase_value   \\
```

1.2 Setup run mode

The run mode parameters are setup using the APV APPLICATION port. Writing 1 to the RO_ENABLE register will start the acquisition.

Name	Address (hex)	Test-pulse mode	Run Mode	Description	
BCLK_MODE	00	0x03	0x04	**Test:** testpulse, reset enable	**Run:** external trigger enable
BCLK_TRGBURST	01	n	n	controls how many time slots the APV chip is reading from its memory for each trigger **= (n+1) x 3**	
BCLK_FREQ	02	40000	40000	Period of the trigger sequencer Must be more than the DAQ time (datalength x Nchannels)	Deadtime in run-mode.
BCLK_TRGDELAY	03	256	x	Delay between the FEC trigger and the trigger to APV. Adjusted to match the trigger latency	
BCLK_TPDELAY	04	128	-	Delay between FEC trigger and the APV test-pulse	
BCLK_ROSYNC	05	300	100	Delay between the FEC trigger and the start of data recording. Adjusted to capture correctly the APV data frames	

EVBLD_CHENABLE	08	0xFFFF	0xFFFF	Channel-enable mask for the data transmission. Even bits are masters and odd bits are slaves. If bit is set, corresponding channel is enabled
EVBLD_DATALENGTH	09	y	y	Length of the data capture window. Adjusted to fit all APV data: y > n x 420 + 300 (~)
RO_ENABLE	0F	1	1	Readout Enable register (bit 0). Triggers are accepted for acquisition when this bit is 1

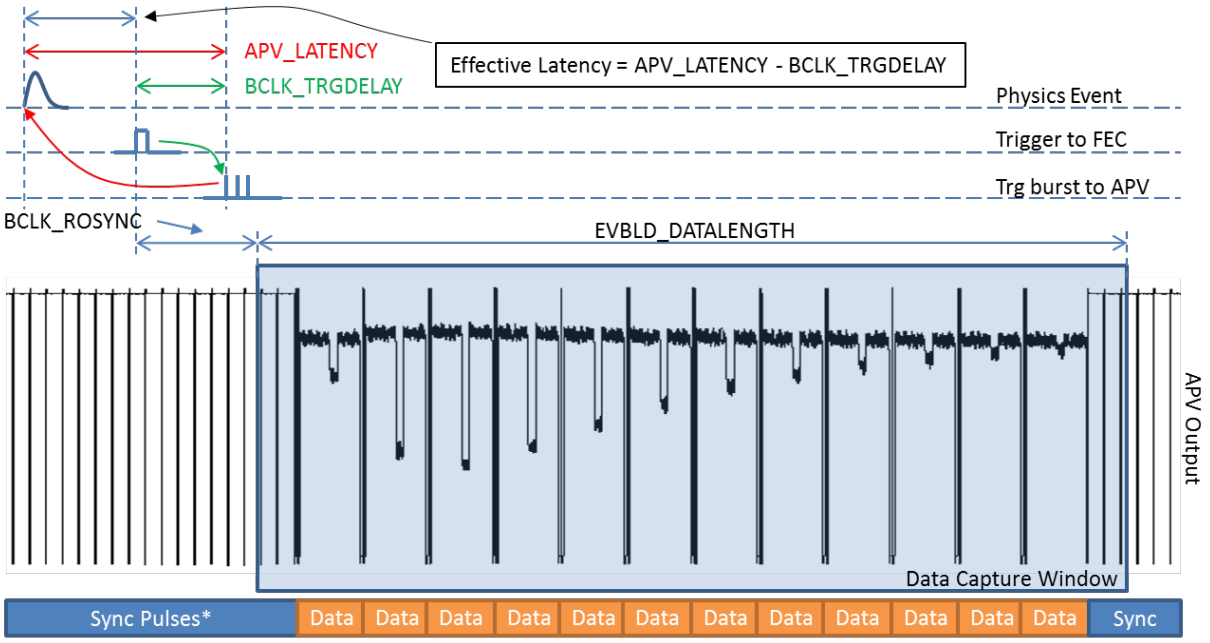


Figure 1. Timing diagram of the APV raw data acquisition. Sync pulses are not synchronized with trigger signal so data position will vary inside the capture window.

The sync pulses and the start of data frames in the APV output is not synchronized with the trigger signal, so the data position in the capture window vary from event to event (the variation has an amplitude of 35 samples). The user must detect off-line the start of APV data frame and decode the APV channel data (see the APV User Manual for more details, or the APV readout using ADC mode chapter in the SRS Data Format document for a short description). The BCLK_ROSYNC parameter has to be adjusted such that the start of the first data frame falls always inside the capture window, and EVBLD_DATALENGTH adjusted such that all data frames fall inside the capture window.

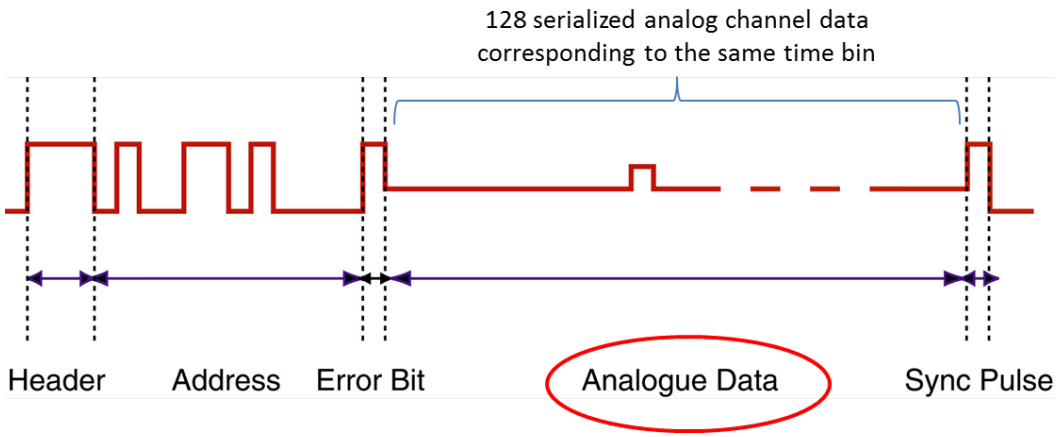


Figure 2. APV data frame structure

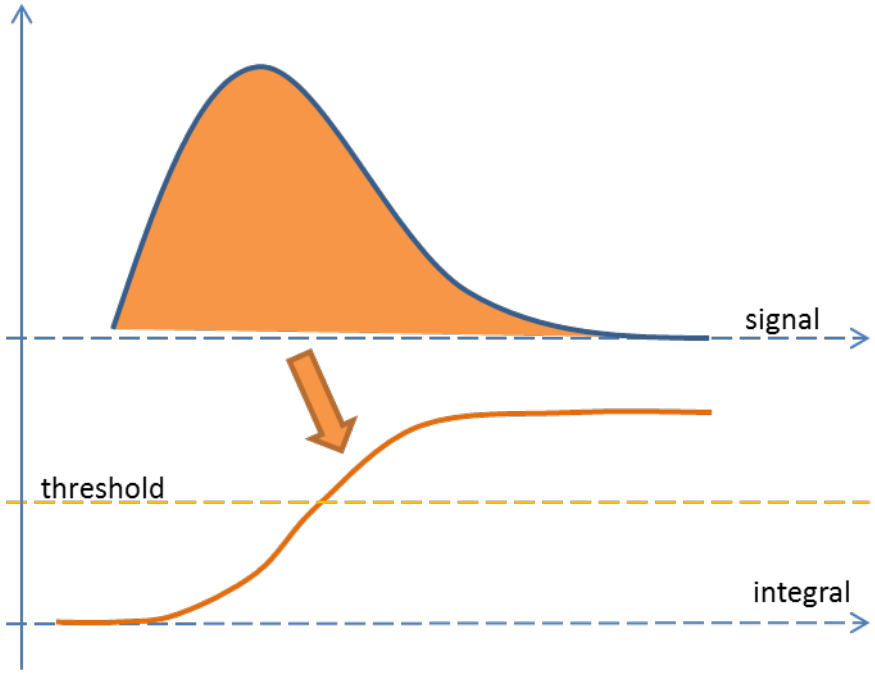
2 APV readout - Zero-suppression mode (APZ)

2.1 Short description

The zero-suppression firmware (code named ***APZ***) detects the APV data frames, decodes the channel data and selects the channels that contain a signal. The zero-suppressed channel data is formatted in a structured way.

The signal condition is given by comparing the integral of the signal in a given channel (sum of the pedestal corrected time samples) with the pedestal variation (sigma) of the same channel times the number of samples.

For the channels which are not suppressed, all samples are acquired.



Name	Address (hex)	Byte count	default	Access Mode	Description	Fw. ver.
APZ REGITERS[1]*:*						
APZ_SYNC_DET	10	2	0	R	Presence of the APV sync pulses on each channel. Read-only	APZ
APZ_STATUS	11	4	0x80	R	Status of the APZ processor. Read-only	APZ
APZ_APVSELECT	12	1	0	RW	Selects one APV channel for single channel commands	APZ
APZ_NSAMPLES	13	1	0	RW	Overrides the *number of samples* parameter. If set to 0 (default) the parameter is calculated internally from **BCLK_TRGBURST**.	APZ
APZ_ZEROSUPP_THR	14	2	0	RW	Zero-suppression threshold Byte 0 = fractional thr part (6 bits, msb)	APZ

[1] These registers are only present in the Zero-suppression (APZ) firmware variant

Name	Address (hex)	Byte count	default	Access Mode	Description	Fw. ver.
					Byte 1 = integer thr part (6 bits, lsb)	
APZ_ZEROSUPP_PRMS	15	2	0	RW	Zero-suppression parameters	APZ
	16 – 1C				Reserved	
APV_SYNC_LOWTHR	1D	2	0	RW	Low threshold for the APV sync-pulse detection. If set to 0 (default) the threshold is internally hard wired (*1100)*	APZ
APV_SYNC_HIGHTHR	1E	2	0	RW	High threshold for the APV sync-pulse detection. If set to 0 (default) the threshold is internally hard wired (*3000)*	APZ
APZ_CMD	1F	1	0	RW	Command register for the APZ processor.	APZ

Table 1. APZ specific registers

The APZ processor needs to learn the values of the channel pedestals and pedestal variations (sigma). A set of calibration procedures is integrated in the code, including clock phase alignment using the on-hybrid PLLs.

Name	CMD	Description
CAL_PHASE_SINGLE	0x01	Calibrate clock phase on a single channel.
CAL_PED_SINGLE	0x02	Calibrate pedestal (and sigma), single channel
CAL_FULL_SINGLE	0x03	Calibrate both phase and pedestal (and sigma) values, single channel
CAL_FULL_ALL	0x10	Calibrate all channels enabled by EVBLD_CHENABLE (full calibration). Channels are treated sequentially; current channel being treated is displayed in CALIB_ALL_CRT field of APZ_STATUS register. **Return to run mode (APZ_CMD = 0) after this command is compulsory**
CAL_PHASE_ALL	0x11	As above, phase only
CAL_PED_ALL	0x12	As above, pedestal and sigma only

Notice. *To guarantee correct operation of the zero-suppression algorithm, the APV output signal must be in good shape. In case of faulty connections, excessive channel or common-mode noise or misaligned clock phase there is no guarantee that the resulted data is valid. It is important that calibration results are read and validated before starting a run.*

2.2 Calibration procedures

2.2.1 Definitions

Phase calibration. The relative phase between the APV clock and the ADC clock is varied using the on-hybrid PLL (**Note.** Master and Slave hybrids on the same HDMI port use the same PLL). The calibration routine scans the entire phase space while recording the amplitude of the APV sync pulses. At the end of the routine, the PLL is set with the phase setting corresponding to the maximum sync pulse amplitude.

Pedestal calibration. Pedestal data is acquired for all channels of one APV for a number of internally generated periodic triggers. Mean and rms variation of pedestal data is stored in the pedestal memory.

Note. *In case of faulty connections, excessive channel or common-mode noise or misaligned clock phase, calibration result may be erroneous.*

2.2.2 Single channel calibration using the sync-pulse detector

This procedure calibrates a single channel in two steps (phase calibration and pedestal calibration). The APZ_SYNC_DET status register is used to validate the integrity of the channel. If sync pulses are not detected on the specific channel, pedestal calibration is not run any more (which otherwise will result in a timeout error)

A. Make sure the front-ends are initialized (use Front-end initialisation)
B. Calibrate one channel:

```
\\
WITH APVAPP_PORT                          \\ write to APV APPLICATION registers
    WRITE RO_ENABLE        0x00           \\ disable triggers

    WRITE APZ_APVSELECT    ch_number      \\
    WRITE APZ_CMD          0x01           \\ CAL_PHASE_SINGLE command
    \\ wait until routine is finished
    do { READ APZ_STATUS }
    while (APZ_STATUS.bit5 == 0 )         \\ poll APZ_STATUS.CMD_DONE bit
    if (APZ_STATUS.bit2 == 0 ) break      \\ (optional) abort if routine failed

    \\ check sync pulses
    READ APZ_SYNC_DET
    If ( APZ_SYNC_DET.bit(ch_number) == 0 )   \\ check if sync pulse is present
       Break                                  \\ abort if no sync pulse

    WRITE APZ_CMD          0x02           \\ CAL_PED_SINGLE command
    \\ wait until routine is finished
    do { READ APZ_STATUS }
    while (APZ_STATUS.bit5 == 0 )         \\ poll APZ_STATUS.CMD_DONE bit
    if (APZ_STATUS.bit3 == 1 ) break      \\ abort if routine failed (timeout)
```

C. Repeat routine for all connected channels
D. Setup run mode:

```
WITH APVAPP_PORT                          \\ write to APV APPLICATION registers
    WRITE APZ_CMD          0x00               \\ RUN mode
    WRITE EVBLD_CHENABLE   all_goood_channels  \\ setup all channels
    WRITE RO_ENABLE        0x01               \\ enable triggers
```

2.2.3 Single channel calibration without the sync-pulse detector

This procedure calibrates a single channel in one step (automatic phase and pedestal calibration). In case of bad channel data, the routine will end up with a timeout error.

A. Make sure the front-ends are initialized (use Front-end initialisation)

B. Calibrate one channel:

```
WITH APVAPP_PORT                          \\ write to APV APPLICATION registers
    WRITE RO_ENABLE        0x00           \\ disable triggers

    WRITE APZ_APVSELECT    ch_number      \\
    WRITE APZ_CMD          0x03           \\ CAL_FULL_SINGLE command
    \\ wait until routine is finished
    do { READ APZ_STATUS }
    while (APZ_STATUS.bit5 == 0 )         \\ poll APZ_STATUS.CMD_DONE bit
    if (APZ_STATUS.bit2 == 0 ) break      \\ abort if phase routine failed
    if (APZ_STATUS.bit3 == 1 ) break      \\ abort if pedestal routine failed
                                          \\ (timeout flag)
```

C. Repeat for all connected channels

D. Setup run mode:

```
WITH APVAPP_PORT                          \\ write to APV APPLICATION registers
    WRITE APZ_CMD          0x00               \\ RUN mode
    WRITE EVBLD_CHENABLE   all_goood_channels  \\ setup all channels
    WRITE RO_ENABLE        0x01               \\ enable triggers
```

2.2.4 All channels calibration using the sync-pulse detector

A. Make sure the front-ends are initialized (use Front-end initialisation)

B. Calibrate all channels:

```
\\
WITH APVAPP_PORT                          \\ write to APV APPLICATION registers
    WRITE RO_ENABLE        0x00           \\ disable triggers

    WRITE EVBLD_CHENABLE   0xFFFF         \\ setup all channels
    WRITE APZ_CMD          0x11           \\ CAL_PHASE_ALL command
    \\ wait until routine is finished
    do { \\ READ APZ_STATUS
         print "Crt. Ch = " APZ_STATUS.byte1   \\ (optional) display curent channel
       }
    while (APZ_STATUS.bit4 == 0 )         \\ poll APZ_STATUS.CALIB_ALL_DONE bit

    \\ check sync pulses
    READ APZ_SYNC_DET
    Print (APZ_SYNC_DET)                       \\ (optional) report good channels
    WRITE EVBLD_CHENABLE   (APZ_SYNC_DET)      \\ setup ped calib for good channels

    WRITE APZ_CMD          0x12           \\ CAL_PED_SINGLE command
    \\ wait until routine is finished
    do { \\ READ APZ_STATUS
         print "Crt. Ch = " APZ_STATUS.byte1   \\ (optional) display curent channel
       }
    while (APZ_STATUS.bit4 == 0 )         \\ poll APZ_STATUS.CALIB_ALL_DONE bit
    print (APZ_STATUS.byte(3..2))         \\ report result
```

C. Setup run mode:

```
WITH APVAPP_PORT                          \\ write to APV APPLICATION registers
    WRITE APZ_CMD          0x00               \\ RUN mode
    WRITE EVBLD_CHENABLE   all_goood_channels  \\ setup all channels
    WRITE RO_ENABLE        0x01               \\ enable triggers
```

2.2.5 All channels calibration without the sync-pulse detector

A. Make sure the front-ends are initialized (use Front-end initialisation)

B. Calibrate all channels:

```
\\
WITH APVAPP_PORT                          \\ write to APV APPLICATION registers
    WRITE RO_ENABLE        0x00           \\ disable triggers

    WRITE EVBLD_CHENABLE   0xFFFF         \\ setup all channels
    WRITE APZ_CMD          0x10           \\ CAL_FULL_ALL command
    \\ wait until routine is finished
    do { \\ READ APZ_STATUS
         print "Crt. Ch = " APZ_STATUS.byte1   \\ (optional) display curent channel
       }
    while (APZ_STATUS.bit4 == 0 )         \\ poll APZ_STATUS.CALIB_ALL_DONE bit

    print (APZ_STATUS.byte(3..2))         \\ report result
```

C. Setup run mode:

```
WITH APVAPP_PORT                          \\ write to APV APPLICATION registers
    WRITE APZ_CMD          0x00               \\ RUN mode
    WRITE EVBLD_CHENABLE   all_goood_channels  \\ setup all channels
    WRITE RO_ENABLE        0x01               \\ enable triggers
```