IO Library

April 25 2017

Data Structures

CDMS_EVENT

eventSizeBytes
triggerID
triggerType
global_timestamp

vector<TRIGPRIMITIVE> primitives

vector<DETECTORS> detectors

TRIGPRIMITIVE

trigStatus
piledUp
triggerID
numPrimsEvent

detectorID
unixtime
rt_time
rt_timefrac
scaler
num_triggers
trigger_time
trigger_timefrac
amplitude
triggerword
maskparis
DCRC

DETECTOR

towerNum
numPhononChannels
numChargeChannels
detectorID
detectorType
dcrcIndex
dcrc0_serial
dcrc0_version
dcrc1_serial
dcrc1_version

readoutStatus
seriesTime
seriesTimefrac

vector<CHANNEL> channels

CHANNEL

prepulseLength
onpulseLength
postpulseLength
pretriggerOffset
samplerateHigh
samplerateLow
channelType
channelNum
*data

Notable Changes to Data Structure

- All prim information is held within a single structure (currently in CDMS DAQ, it is held in a class with 2 sub-structures).
- 0x4 READOUT_REQUEST structure no longer exists. Information is merged with DETECTOR structure.
- CHANNEL structure now holds metadata such as channel number and channel type
  - Currently in CDMS DAQ, there exists separate phonon and charge sub-structures

pack_primitive Function

Inputs:

DWORD *emptybuffer
TRIGPRIMITIVE *prim_ptr
CDMS_EVENT *ev_ptr

Packs:

=1

	0x9	format version=1	total n triggers read

+

		0x5	event size in bytes
		trigger ID	
		trigger type	
		global timestamp low	
		global timestamp high	

+ =1

		0x7	n primitives in event
		length of entry (=0x6 block) in bytes	

+ one of

0x6		trig status	pileup	detector id	index
UT at which rt was issued					
	time fraction rt was run (100nsec/count)				
time of trigger in sec		time rt was run in sec			
mask pairs	time fraction of trigger (100nsec/count)				
trigger word		peak amplitude			

+ =0

igg		0x3	n detectors in event

+

	0x8	total n preceding triggers

pack_primitiveList Function

Inputs:

DWORD *emptybuffer
vector<TRIGPRIMITIVE> *primlist
CDMS_EVENT *ev_ptr

Packs:

=1

	0x9	format version=1	total n triggers read

+ looped over all prims

		0x5	event size in bytes
		trigger ID	
		trigger type	
		global timestamp low	
		global timestamp high	

+

		0x7	n primitives in event
		length of entry (=0x6 block) in bytes	

+

x N prims	0x6		trig status	pileup	detector id	index
	UT at which rt was issued					
		time fraction rt was run (100nsec/count)				
	time of trigger in sec		time rt was run in sec			
	mask pairs	time fraction of trigger (100nsec/count)				
	trigger word		peak amplitude			

+

=0

igg		0x3	n detectors in event

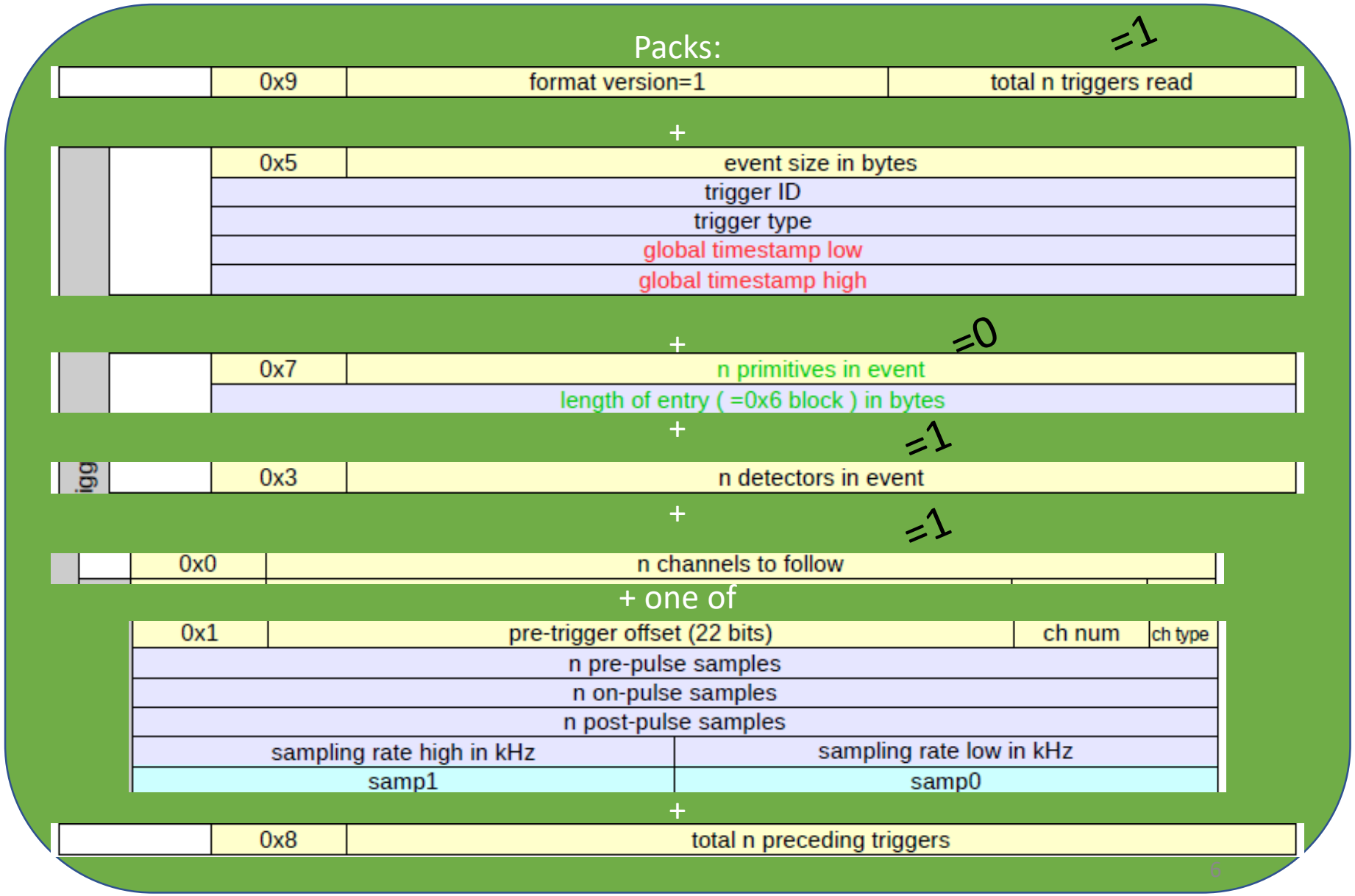
+

	0x8	total n preceding triggers

pack_channel Function

Inputs:

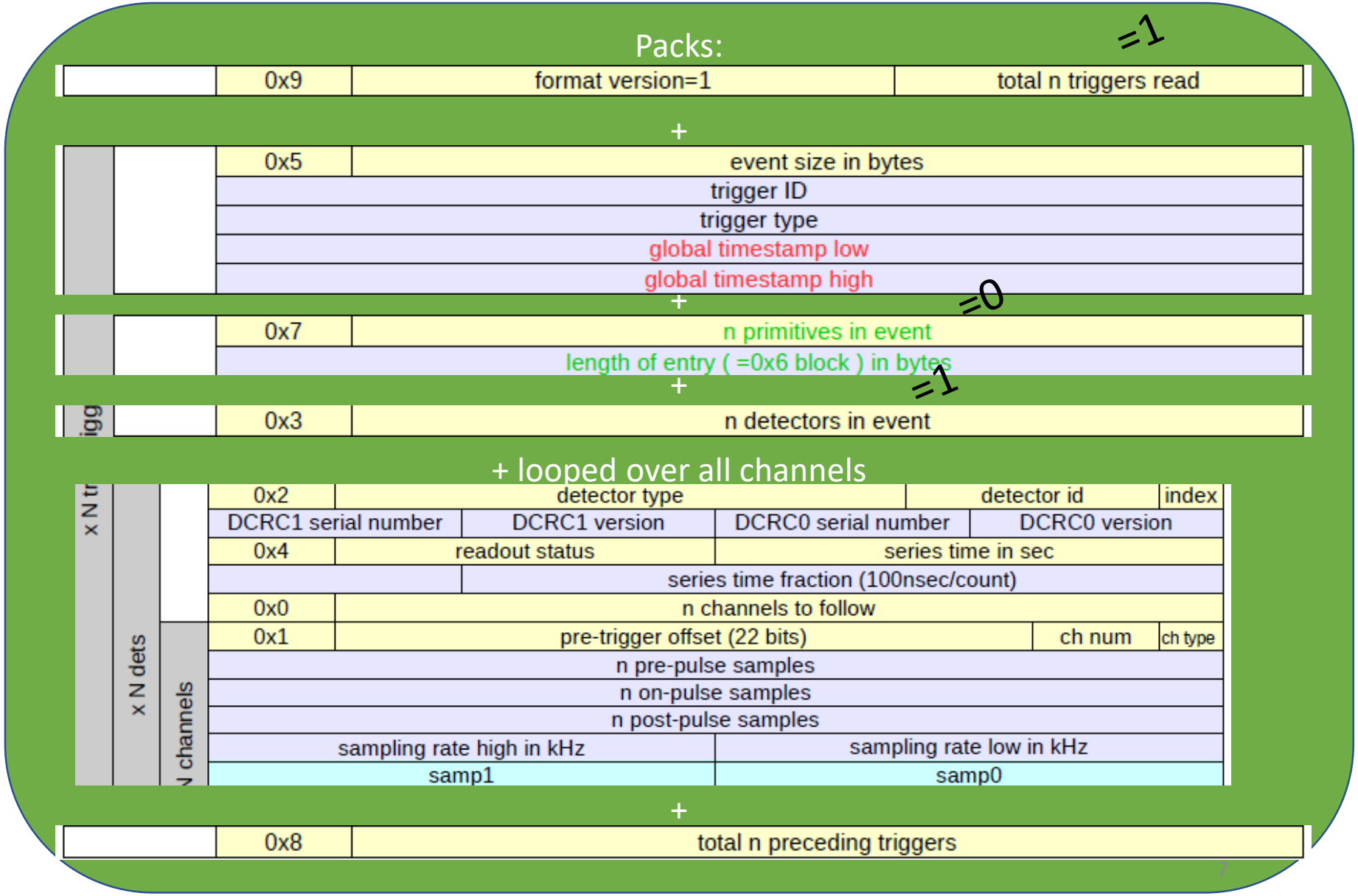
DWORD *emptybuffer
CHANNEL *ch_ptr
CDMS_EVENT *ev_ptr



pack_detector Function

Inputs:

DWORD *emptybuffer
DETECTOR *det_ptr
CDMS_EVENT *ev_ptr



pack_event Function

Inputs:

DWORD *emptybuffer
CDMS_EVENT *ev_ptr

Packs: =1

	0x9	format version=1	total n triggers read

+ looped over all prims, looped over all detectors

x N triggers

		0x5	event size in bytes			
		trigger ID				
		trigger type				
		global timestamp low				
		global timestamp high				
		0x7	n primitives in event			
		length of entry (=0x6 block) in bytes				
	x N prims	0x6		trig status	pileup	detector id / index
		UT at which rt was issued				
			time fraction rt was run (100nsec/count)			
		time of trigger in sec		time rt was run in sec		
		mask pairs	time fraction of trigger (100nsec/count)			
		trigger word		peak amplitude		
		0x3	n detectors in event			
	x N dets	0x2	detector type		detector id	index
		DCRC1 serial number	DCRC1 version	DCRC0 serial number	DCRC0 version	
		0x4	readout status	series time in sec		
			series time fraction (100nsec/count)			
		0x0	n channels to follow			
	x N channels	0x1	pre-trigger offset (22 bits)		ch num	ch type
		n pre-pulse samples				
		n on-pulse samples				
		n post-pulse samples				
		sampling rate high in kHz		sampling rate low in kHz		
		samp1		samp0		

+

	0x8	total n preceding triggers

pack_eventList Function

Inputs:

DWORD *emptybuffer
vector<CDMS_EVENT> *evlist

Packs: =evlist.size()

	0x9	format version=1	total n triggers read

+ looped over all prims, looped over all detectors

x N triggers		0x5	event size in bytes				
		trigger ID					
		trigger type					
		global timestamp low					
		global timestamp high					
		0x7	n primitives in event				
		length of entry (=0x6 block) in bytes					
	x N prims	0x6		trig status	pileup	detector id	index
		UT at which rt was issued					
			time fraction rt was run (100nsec/count)				
		time of trigger in sec		time rt was run in sec			
		mask pairs	time fraction of trigger (100nsec/count)				
		trigger word		peak amplitude			
		0x3	n detectors in event				
	x N dets	0x2	detector type			detector id	index
		DCRC1 serial number	DCRC1 version	DCRC0 serial number	DCRC0 version		
		0x4	readout status	series time in sec			
			series time fraction (100nsec/count)				
		0x0	n channels to follow				
	N channels	0x1	pre-trigger offset (22 bits)			ch num	ch type
		n pre-pulse samples					
		n on-pulse samples					
		n post-pulse samples					
		sampling rate high in kHz		sampling rate low in kHz			
		samp1		samp0			

looped over

+

	0x8	total n preceding triggers

Issue 1: TRIGPRIMITIVE and multiple events

- Having a CDMS_EVENT structure suggests that all the information within that structure corresponds to the same single event.
- However currently in the CDMS DAQ, the TRIGPRIM_BANK_DATA class holds prim information from multiple events.
- As it stands now in the IO Library, using pack_primlist or pack_event will pack prim information from multiple events, using the same decision making as CDMS DAQ:
  - if(i == 0 || (i>0 && primlist[i].triggerID != primlist[i-1].triggerID){ …}

Issue 1: TRIGPRIMITIVE and multiple events

- Option 1: don't have a CDMS_EVENT data structure, keeping TRIGPRIMITIVE and DETECTOR structures separate.
  - Now it doesn't matter if TRIGPRIMITIVE has info from multiple events
  - User will have some event builder to combine banks together
- Option 2: It is expected that the user will populate the vector<TRIGPRIMITIVE> with prims only from one event.
  - Now CDMS_EVENT structure has information corresponding to only 1 event
  - This will require changes to CDMS DAQ
- Option 3: IO Library only packs prim data that matches triggerID with CDMS_EVENT.
  - Eg. If CDMS_EVENT metadata has triggerID = x, then IOLibrary will only pack prim data that has triggerID = x.
  - More work for Library, but would means less changes to CDMS DAQ.
  - CDMS_EVENT structure will still have prim info from multiple event (information will be duplicated to some extent?)

Issue 2: CDMS_EVENT 0x5 data

- The CDMS_EVENT structure holds the 0x5 data as its members:

- Every packing function will pack the 0x5 data.
- This means that all functions must have CDMS_EVENT *ev_ptr pointer to the event structure, in order to pack this info.
- Thus user **must** have CDMS_EVENT structure (instead of only creating the substructures).

Issue 2: CDMS_EVENT 0x5 data

- Option 1: Leave it as is – this is fine.

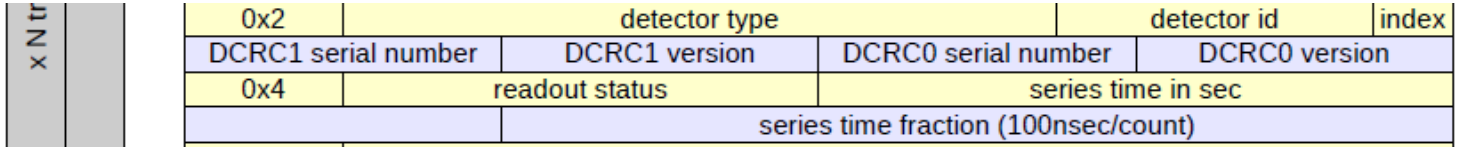
- Option 2: The substructure will also hold the 0x5 data, so if you want to pack a substructure, you can still pack 0x5 data without needing to point to a CDMS_EVENT structure.
  - Eg. Each channel will hold this 0x5 data. So to pack channel, all you need is pack_channel(DWORD *emtpybuffer, CHANNEL *ch_ptr){…}
  - This is already done at the detector level for triggerID, but not the channel level. What about the other 0x5 data?
  - Is this possible? Would it be a lot of data duplication?

Issue 3: Header values

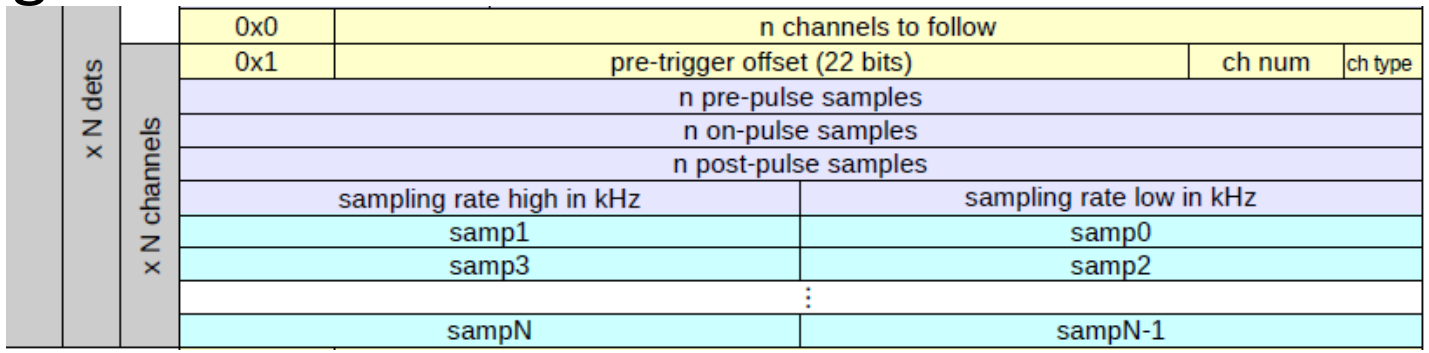
- The 0x9, 0x5, 0x7, 0x3, and 0x8 headers are packed for all packing functions.
- Should the value of the headers reflect the actual event information, or the information that is being packed?
- For example, if I was using pack_primitiveList (so no detector information included), but the event has 5 detectors (hypothetically) would the 0x3 (numDetectors) read:
  - 0, since there is no detector information being packed
  - 5, since there are 5 detectors with this event. However, the actual detector information has not been packed.

Issue 4: pack_channel and detector metadata

- If I wanted to pack data for a single channel, should the packed data include the detector metadata or not?
- I.e. If I call pack_channel(…){…}, which packs the info for just one channel, should it include:



along with:



DATA FORMAT VERSION 1: Created: 05. Feb. '16, Last updated: 30. Jun. '16