Weekly Meeting

May 11th 2017

- Large updates to IO Library
  - Working on status handling
  - Meeting with Belina today
- Thesis outline
- CAP Poster
  - Abstract still has “submitted” tag.
  - Starting to work on poster.

IO Library Updates:

- Updated packing functions. User’s only use pack_event and pack_eventList functions.
  - Packing functions are now more streamlined.
- Added unpacking functions.
- Added get_size functions.

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

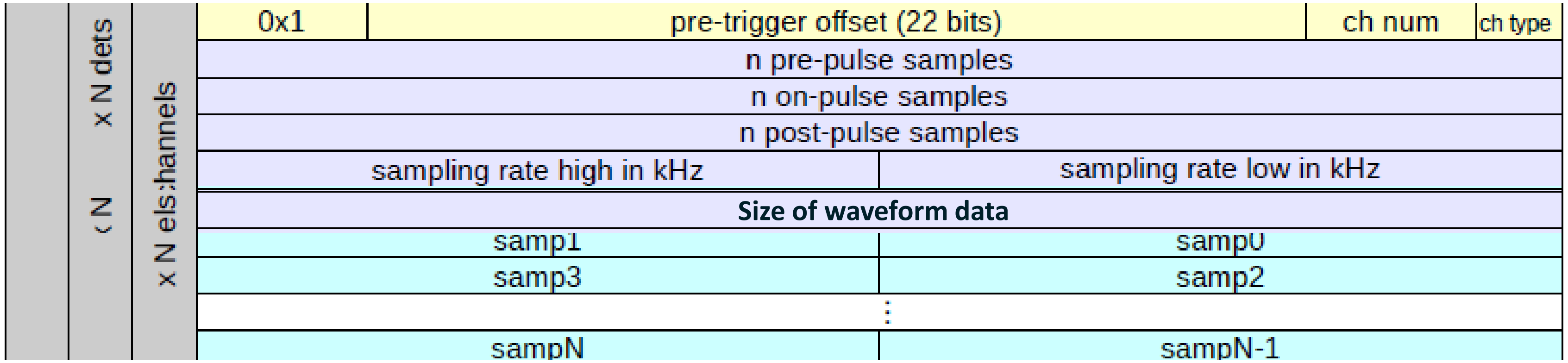
vector<CHANNEL> channels

CHANNEL

prepulseLength
onpulseLength
postpulseLength
pretriggerOffset
samplerateHigh
samplerateLow
channelType
channelNum
waveformSize
vector<WORD>*data

Change to the Data Structure

- I propose adding an integer variable "waveformSize" which is the size of the data (waveform) vector.
- Having this variable is extremely helpful for unpacking the data.
- In the IO Library, I added an additional row to the format (but that can be changed):



The Updated Functions

pack_event:

Inputs:
DWORD *emptybuffer
CDMS_EVENT *ev_ptr

Goes through CDMS_EVENT, packs accordingly

Outputs:
int status
Filled emptybuffer

pack_eventList:

Inputs:
DWORD *emptybuffer
vector<CDMS_EVENT> *evlist

Goes through vector of CDMS_EVENTs, packs accordingly

Outputs:
int status
Filled emptybuffer

The Updated Functions

unpack_event:

Inputs:
DWORD *databuffer
CDMS_EVENT *ev_ptr

→

Goes through databuffer, unpacks into CDMS_EVENT

→

Outputs:
int status
Filled CDMS_EVENT

unpack_eventList:

Inputs:
DWORD *databuffer
vector<CDMS_EVENT> *evlist

→

Goes through databuffer, unpacks into vector of CDMS_EVENTs

→

Outputs:
int status
Filled vector of CDMS_EVENTS

The Updated Functions

get_eventsize:

Inputs:
CDMS_EVENT *ev_ptr

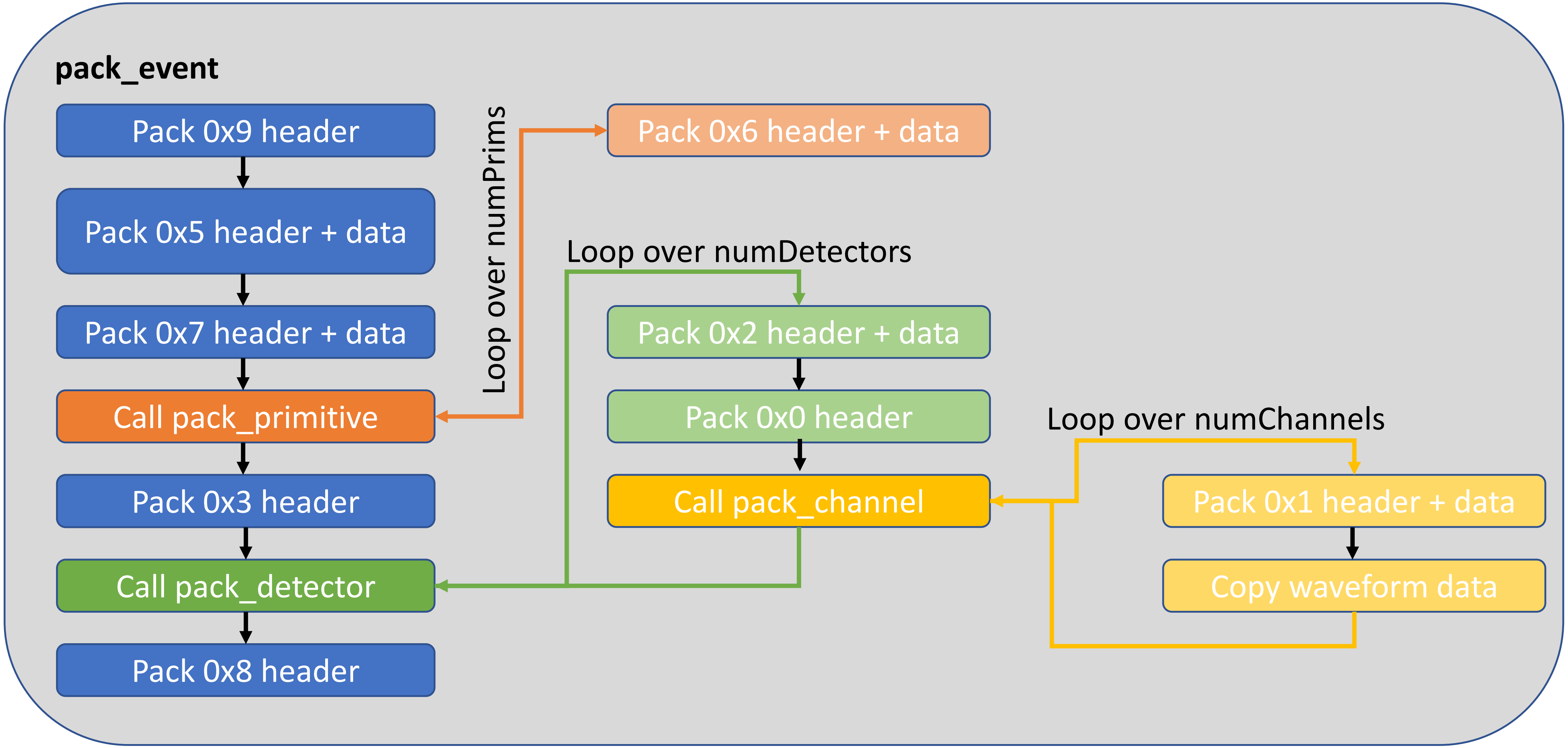
Outputs:
Size of array needed

get_eventListsize:

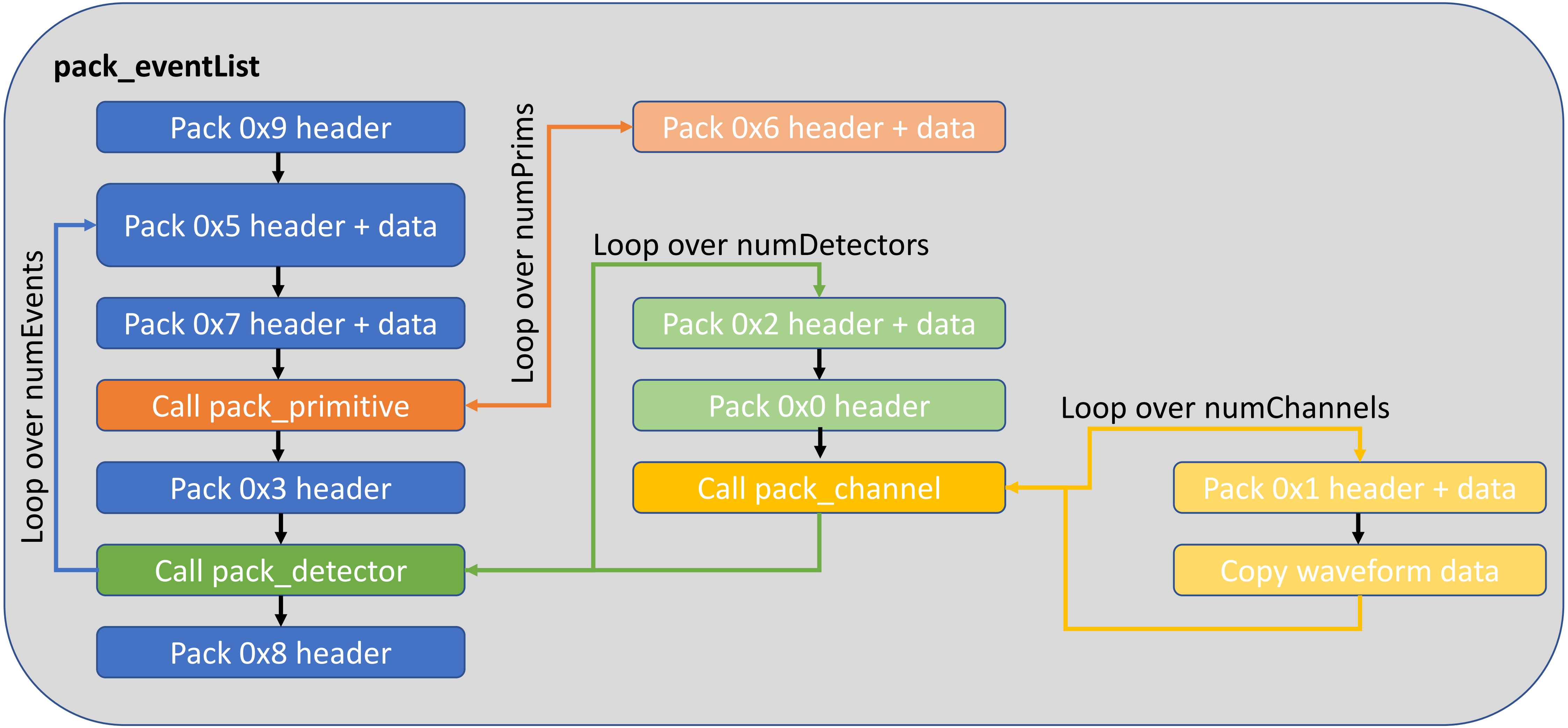
Inputs:
vector<CDMS_EVENT> *evlist

Outputs:
Size of array needed

Internal Functionality: pack_event



Internal Functionality: pack_eventList



Internal Functionality: unpacking

- The internal functionality of the unpacking functions is analogous to the packing functions.

The Updated Functions

- User’s will use pack_event function, even if just to pack subset of information.
  - E.g. pack prim information….would still use pack_event.
  - Because in CDMS_EVENT, detectors.size() = 0, so no information about detectors is packed.
  - Event builder would combine packed buffers

Error Handling

- Ben thinks we should stick to ‘status’ method, instead of c++ exceptions (easier to implement).
- My takeaway from Ben’s comments:

- Have an enum to differentiate types of errors.
- CDMS_EVENT has a new field for errors. I think this could be a vector of errors.
- IO Library can have a PrintErrors function, which user can decide to use.
- Int status will just be used to indicate success or failure.
- Handling errors will be determined by the users.

Error Handling

```
int status = unpack_event(buffer, &myevent);
if (status !=  IOLIBRARY::STATUS_OK){
       PrintErrors(&myevent); //users can decide to print errors
       //users decide how to handle errors
}

void IOLIBRARY::PrintErrors(CDMS_EVENT *ev_ptr){
       numErrors = errors.size();
       for(int i = 0; i<numErrors; i++){
              //print errors[i];
              //can also have ifdef MIDAS ...
       }
}
```

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

bits	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0					
	0x9	format version=1	total n triggers read			
x N triggers	0x5	event size in bytes				
x N triggers	trigger ID					
x N triggers	trigger type					
x N triggers	global timestamp low					
x N triggers	global timestamp high					
x N triggers	0x7	n primitives in event				
x N triggers	length of entry (=0x6 block) in bytes					
x N triggers / x N prims	0x6		trig status	pileup	detector id	index
x N triggers / x N prims	UT at which rt was issued					
x N triggers / x N prims		time fraction rt was run (100nsec/count)				
x N triggers / x N prims	time of trigger in sec	time rt was run in sec				
x N triggers / x N prims	mask pairs	time fraction of trigger (100nsec/count)				
x N triggers / x N prims	trigger word	peak amplitude				
x N triggers	0x3	n detectors in event				
x N triggers / x N dets	0x2	detector type	detector id	index		
x N triggers / x N dets	DCRC1 serial number	DCRC1 version	DCRC0 serial number	DCRC0 version		
x N triggers / x N dets	0x4	readout status	series time in sec			
x N triggers / x N dets		series time fraction (100nsec/count)				
x N triggers / x N dets	0x0	n channels to follow				
x N triggers / x N dets / x N channels	0x1	pre-trigger offset (22 bits)	ch num	ch type		
x N triggers / x N dets / x N channels	n pre-pulse samples					
x N triggers / x N dets / x N channels	n on-pulse samples					
x N triggers / x N dets / x N channels	n post-pulse samples					
x N triggers / x N dets / x N channels	sampling rate high in kHz	sampling rate low in kHz				
x N triggers / x N dets / x N channels	samp1	samp0				
x N triggers / x N dets / x N channels	samp3	samp2				
x N triggers / x N dets / x N channels	⋮					
x N triggers / x N dets / x N channels	sampN	sampN-1				
	0x8	total n preceding triggers				