SBS-offline for SBSGenericDetector

August 16, 2021

1 Introduction

SBSGenericDetector is a THaNonTrackingDetector. For the SBSGenericDetector , a detector can be classified as using ADC and/or TDC information. The SBSGenericDetector is made of SBSElement objects with each element storing the x,y,z position and a row,column,layer location with ADC and/or TDC data and parameters. The constructor sets variables to default values which are given in Table 1 and some variables that can be set by functions that can be called from your replay script are listed in Table 1. You can also set this in the constructor of the detector that inherits from SBSGenericDetector.

The fNrow and fNLayers can be set by parameters read-in during the ReadDatabase. In ReadDatabase, a vector , fElements, is created of SBSElement objects. The fDisableRefADC and/or fDisableRefTDC is set to true in ReadDatabase if the reference ADC and/or TDC time channel is designated in the *chanmap* when that is read-in.

The ADC can be different modes: kNone (no ADC), kADCSimple (like a 792 or fastbus), kADC (F250 in mode 7) and kWaveform (F250 in mode 1). ADC data from a F250 in mode 7 can have up three pulses recorded per channel per event. ADC data from a F250 in mode 1 is a vector of 4ns samples. The size of the vector is fixed in the configuration of the FADC readout list. The size of the vector is determined by the FADC250 module decoder. The TDC can be in modes: kNone (no TDC for detector), kTDCSimple (only leading edge), kTDC (leading and trailing edge).

The tree output is controlled by the Podd output definition file. fStoreEmptyElements set to true means that the fGood output structure is filled for all fElements regardless in the detector element has data for that event. Otherwise, the fGood output structure is filled with the *good hit* for the multi-hit ADC/TDC (this is described later in Section). To save disk

Variable	Default	Functions
fNrows	0	
fNcolsMax	0	
fNlayers	0	
fModeADC	kADCSimple	SetModeADC(SBSModeADC::Mode mode)
fModeTDC	kNone	SetModeTDC(SBSModeTDC::Mode mode)
fDisableRefADC	true	SetDisableRefADC(Bool_t b)
fDisableRefTDC	true	SetDisableRefTDC(Bool_t b)
fConst	0	
fSlope	0	
fAccCharge	0	
fStoreRawHits	false	SetStoreRawHits(Bool_t var)
fStoreEmptyElements	true	SetStoreEmptyElements(Bool_t b)

Table 1: Table of variables in the constructor

space, one can set fStoreEmptyElements is kFALSE so that only elements which had ADC and/or TDC data for the event are written to the output tree. To save disk space, the flag fStoreRawHits is set to kFALSE. If one wants to write out all hits in each channel of the multi-hit ADC/TDC, then set fStoreRawHits to kTRUE. Also if one wants to look at the waveform data in the output tree, the set fStoreRawHits to kTRUE.

The fConst, fSlope and FAccCharge are parameters used by the calorimeter as corrections for the measured energy. Maybe move this to SBSCalorimeter?

2 SBSGenericDetector ReadDatabase

A detector is defined as a grouping of elements which are arranged in rows, cols and layers. The element is defined by the SBSElement class which stores information about the element such as it's x,y,z position and the TDC and/or ADC data. The TDC/ADC data and parameters are stored in structures and classes defined in the SBSData and SBSElement class. There are two numbering schemes for vectors of the elements. A vector , fElements, is created of SBSElement objects. Either by vector index or like a 3-d array with row, col and layer. ncols is a parameter that needs to be set. nrows is optional and the default is 1. ncols is an array. If the size of ncols array is 1,

then all the nrows are given the same number of columns equal to ncols[0]. If the ncols array is nrow size , then each row has a different number of columns. The nlayers is optional and the default is 1. The total number of elements, fNelem, is fNRow×fNCol×fNlayers.

The fElement vector is filled by constructed an SBSElement object by looping through the layers which is inside a loop over the columns which is inside the top loop over the rows. Assume that there is only one layer, the index of the fElement vector runs from 0 (row=0,col=0), 1 (row=0,col=1) ... fNcol-1 (row=0,col=fNcol-1), fNcol (row=1,col=0) ... fNelem-1 (row=fNrow-1,col=fNcol-1). The parameter start_chanmap is optional and is the offset between the element iD used in the *chanmap* and the index of the fElements vector which goes from zero to fNelems-1. The element ID is set for each SBSElement and equals the number used in the chanmap parameter.

The vector parameters xyz (the x,y and z starting position for the geometry) and dxyz (the dx, dy and dz spacing between elements) need to be in the detector database. When the SBSElement is constructed in the looping over the rows, columns and layers, the position is set to $x = xyz[0] - nr * dxyz[0]$, $y = xyz[0] - nc * dxyz[0]$, $z = xyz[0] - nlayer * dxyz[0]$. Need to modify code so x,y,z position can be set for each element separately.

The detmap gives list of the crate, slot, start channel , end channel and reference time index for each module that is used by the detector. One can also add the module number in the detmap, if one sets the parameter model_in_detmap = 1. By default model_in_detmap = 0 and setting the detector map is probably only necessary for when using the VETROC. When one wants to include the model in the detector map, then the detmap format changes and should be crate, slot, start channel , end channel, model number and reference time index for each module that is used by the detector. The reference time index is an integer that indicates which reference time indicated in the chanmap to use for that module. The detmap vector is an input to the FillDetMap function of the THaDetMap which produces the object fDetMap. To indicate which channel in a module is the reference time in the chanmap, one puts a -1000. The order of the reference times in the chanmap gives the index of reference time.

The chanmap is vector of the element index for the crate, slot and channel. The chanmap order must follow the ordering of the crate and slots in the detmap. It is an optional parameter. Without the chanmap defined, the default is to assign the elements to each channel in the detmap in order. When creating the chanmap, the element ID normally starts at zero and the maximum is the total number of elements (nrow x ncol x nlayer) minus one. If the detector wants the element ID to start at another value (e.g. 1),

then the parameter start_chanmap needs to be set to the starting element ID value () The chanmap format is to match the number of channels in the detmap and in the same order of the detmap. If a channel is not used by the detector then put a -1 in the chanmap. If the channel is a reference time, then put a -1000 in the chanmap.

The code loops through the modules in the fDetMap to create a fChanMap and fRefChanMap based on the chanmap. fChanMap and fRefChanMap are use in Decode method to determine which channel info is loaded into which element. This determines the number of elements, fNelem, and the number of reference time elements for the TDC, NRefADCElem and ADC, NRefADCElem. The vector size of fRefChanMap is equal to the number of -1000 channels that are found and the index of the fRefChanMap vector is the order that it finds the -1000. So in the detmap , the reference time index refers to the order of the -1000 in the chanmap and goes from 0 to the number of chanmap =-1000 minus one. As an example for the FADC, there can be one reference time in one module (reference time index = 0) which is used by all FADC modules (so detmap would have reference time index = 0 for each FADC). Similarly for the TDC, there can be one reference time in one module to be used for all modules or a reference time for each module. For the case of FADC and TDC modules, if the TDC modules are first in the detmap and have one module with chanmap = -1000 then the TDC detmap would use reference time index = 0 and if the FADC modules with chanmap = -1000 then the ADC detmap would use reference time index = 1.

The code then reads in parameters used to process the TDC or ADC raw data into calibrated data for each element. All these parameters are optional with default values in the code. For the TDC, the parameters are given in Table 2. The parameter vector can have all elements set individually or have one value which is used for all elements. The same parameters can be read-in for the TDC reference time but the "tdc" is replaced by "reftdc".

Parameter name	Description	Global variable
tdc.offset	offset (default=0)	tdc_offset (unit channels)
tdc.cal	calibration (default=0)	tdc_cal (ns/channel)
tdc.GoodTimeCut	Expected Good Time peak (ns) (default=0)	tdc_GoodTimeCut

Table 2: Table of TDC parameters

For the ADC, the parameters for the kADCSimple are given in Table 3. For processing the kWaveform ADC data to extract the pulse information ,

Parameter name	Description	Global variable
adc.pedestal	Pedestals (not used for waveform)	adc_ped
adc.gain	gain to convert to GeV	adc_gain

Table 3: Table of kADCSimple parameters

there are the following parameters given in Table 4 are used in addition to the kADCSimple parameters. The parameter vectors can have all elements set individually or have one value which is used for all elements. The same parameters can be read-in for the ADC reference time but the "adc" is replaced by "refadc".

adc.conv	conversion factors (waveform is mV/chan)	adc_conv
adc.thres	threshold (mV)	adc_thres
adc.GoodTimeCut	Expected Good Time peak (ns) (default=0)	adc_GoodTimeCut
adc.FixThresBin	fixed threshold crossing bin	adc_FixThresBin
adc.NSB	# of integration samples before threshold crossing	adc_NSB
adc.NSA	# of integration samples after threshold crossing	adc_NSA
adc.NPedBin	# of samples for pedestal determination	adc_NPedBin

Table 4: Table of ADC parameters

The code creates a vector of elements called fElements based on the SBSElement class. The element number is incremented by looping through number of columns inside the looping through rows. The element is created and loads the x,y,z locations along with the row, column, layer and element number. The element can be have ADC and/or TDC information depending on how the GenericDetector is defined. If the GenericDetector has TDC info, then the element is set to have an SBSData::TDC data function and the offset, calibration and GoodTimeCut are loaded. If the GenericDetector has ADC info, then it depends on the ADC mode. If the ADC mode is kWaveform, the element is set to have an SBSData::Waveform data function and the pedestal (which is not used), gain, conversion factor and good time cut are loaded. For the kWaveform there is an additional call to load the parameter used to determine ADC signal in the waveform. If the ADC mode is kADC, the element is set to have an SBSData::ADC data function and the pedestal (which is not used), gain, conversion factor and good time cut are loaded. For the kADC there is an additional call to load the parameters used

to determine ADC signal in the F250 module. If the ADC mode is neither kWaveform nor kADC , the element is set to have an SBSData::ADC and the pedestal and gain are loaded.

The code creates a vector of reference time elements called fRefElements based on the SBSElement class. It follows the same logic for creating the reference time elements as the detector elements, except that the reference time element will either be a TDC or an ADC reference time element. When the reference time element is created the row number and element number are both the reference time index.

3 SBSGenericDetector Decode

First the ClearEvent method is called. This clears the ADC and/or TDC data in all the elements.

If the reference times are in the channel map, then the code looks for the reference time in the data. The code loops the channels in each module. If there is event data for the reference time channel, then the data is processed for the fRefElement with DecodeADC or DecodeTDC depending if the module is an ADC or TDC. Only expect ADC reference times for the FADCs. For ADC waveform, the time for the pulse in the reference time is determined the same way as the time a regular detector element. The TDC reference times are processed the same as a regular detector element. The total number of ADC and TDC reference channels hit in the Decode method is fNRefhits.

The code loops through the channels in each module. If there is event data for the detector channel, then the data is processed for that fElement with DecodeADC or DecodeTDC depending if the module is an ADC or TDC. The total number of ADC and TDC channels hit in the Decode method is fNhits. The ADC and TDC channels are not counted separately. Look into adding counter for the ADC and TDc separately and put in the tree.

3.1 SBSGenericDetector DecodeTDC

The DecodeTDC method calls the SBSData method TDC::Process or TDC::ProcessSimple for each detector element that has a TDC. TDC::Process fills the fTDC which is a SBSData::TDCData container for that detector or reference time element. TDCData holds the offset, calibration, GoodTimeCut and the good_hit index. The offset, calibration, GoodTimeCut are set in SBSGenericDetector::ReadDatabase. good_hit index is an integer variable which holds

the index of the good TDC hit that is determined by the SBSGenericDetector::FindGoodHit method that is described later. TDCData also contains a vector, "hits" of the SBSData::TDCHit structure. The SBSData::TDCHit structure contains the "le" SingleData structure, "te" SingleData structure, "ToT" SingleData structure and "elem ID" ,the integer of the element ID. The SBSData::SingleData structure contains the "raw" and "cal" (the raw and calibrated values). For a TDC, the raw value is reference time subtracted and in units of channels and the calibrated value is the (raw value - offset)*cal.

The reference time elements are processed first in the Decode method with DecodeTDC called with IsRef flag set to kTRUE. For either a reference or detector time, the TDC data extraction is the same.

If the TDC mode is kTDCSimple, then the SBSData TDC::ProcessSimple is called and the vector of TDCData objects, fTDC, is filled for each hit in the TDC channel. If the TDC mode is kTDC, the method loops through the hits for that element and first determines whether the hit is leading (LE) or trailing edge (TE). If the first hit is a TE, then it skips that hit. If the last hit is a LE, then that it skips that hit. If it has passed the first two tests, then the hit is processed using the SBSData method TDC::Process. TDC::Process has the arguments of the element ID, raw time value and edge value (LE = 0, TE = 1). The raw time value is the data from the channel minus reftime (local variable determined earlier in DecodeTDC and described in the next paragraph). When the reference time is being processed reftime is set to zero.

If a detector time is being processed, then the method first determines the good reference time to use for the detector element based on the module's reference time index. It loops through the reference time TDC hits and selects the reference time hit that has the smallest abs(Tdc -GoodTimeCut). The GoodTimeCut is in ns. This good reference time is the raw time in units of channels and is locally stored in the variable reftime. If a reference time is being processed, then reftime is set to zero.

TDC::Process has a counter array fEdgeIdx[edge] with edge = 0 (LE) or 1 (TE). If the TDC detects LE and TE, then the goal is to first find a LE hit and match it with a TE hit which usually follows the LE. If the counter fEdgeIdx[edge] is larger or equal to the size of the TDCHit hits vector for that element, then TDC::Process increments the size of the hits vector and the fEdgeIdx[edge] counter is incremented. The element ID is loaded into the hits vector. If the edge is LE then the "le" structure is filled. If the edge is TE then the "te" structure is filled. If the fEdgeIdx[0] equals fEdgeIdx[1] then the "ToT" structure is filled with the "te" - "le". If the fEdgeIdx[1] is

larger than fEdgeIdx[0], then fEdgeIdx[0] is set equal to fEdgeIdx[1]. This is done to insure that if the next hit is a LE then the size of the hits vector will be increased whether the next hit is a LE or TE. The TDCHit hits vector is ideally assuming that it will have an LE and TE (or just LE if the TDC module was programmed that way). If there is only a LE or TE filled bu a hit, then the missing information will have a value of zero. After looping through all the hits for that element, the element will have an TDC data structure that contains the TDCHit hits vector.

3.2 SBSGenericDetector DecodeADC

It fills the fADC which is a SBSData::ADCData container for that detector or reference time element. The SBSData ADCData structure holds the pedestal, calibration, time calibration (for FADC). In addition, ADCData contains vector of the PulseADCData structure, hits, and the good_hit which is the index number of the good hit which are needed for multihit FADC. If ADC mode is kADCSimple, the SBSData::ADC Process method is called with a single argument. For a kADCSimple mode , only the fADC hits vector will just be the single hit. The PulseADCData is a structure containing the integral, time and amplitude which are all SingleData structures having the "raw" and "val" (calibrated value). For the kADCSimple mode, the ADC::Process with single argument fills the SingleData integral with the raw data and the (raw-ped)*cal with the time and amplitude set to zero and sets fHasData to true.

When ADC is in waveform mode, a sample vector is filled with ADC value for each 4ns bin. The SBSData::Waveform::Process is called. A vector with the raw ADC converted to mV is filled. The pedestal (in mV) is determined by averaging the first NPedBin ADC values. A vector with the pedestal subtracted ADC is filled. The threshold crossing bin is found by looping through the sample raw ADC vector and finding the first bin which has a raw ADC value larger than the pedestal plus the threshold. If a threshold crossing bin, TC, is found, then the ADC samples are integrated from TC-NSB to TC+NSA-1. If no threshold crossing bin, TC, is found, then the ADC samples are integrated from FixThresBin-NSB to FixThresBin+NSA-1. The ADC integration is converted to pC using the 4ns bin width and 50 ohm resistance.

If a threshold crossing bin is found, then the peak amplitude is found within the integration region. The peak, Vpeak, is the ADC value in the sample which within the integration window (starting from the threshold crossing) the following bin has a smaller ADC value than previous sample.

To calculate the pulse time, first $V - Mid = (V_{peak} + \text{pedestal})/2$ is calculated. Then loops through the integration region and finds the sample, i, with which has value greater than Vmid and the next sample, i+1, is less than Vmid. The time is $4\text{ns} * [(i + (V_{Mid} - V_i)/(V_{i+1} - V_i)]$. If no threshold crossing bin is found, then the time and amplitude are set to zero. For now the code only find the first pulse that is above threshold.

Need to implemented the FADC reference time subtraction. Need to get multiple hits from the FADSC waveform.

3.3 SBSGenericDetector FindGoodHit

The final part of the Decode method is to loop through all the fELements and call the SBSGenericDetector::FindGoodHit for each element. If the element that has TDC data and the hit which is the "good" hit is marked by setting the variable fTDC.good_hit which is the hit index ofthe TDC hit which has the closest time to the GoodTimeCut parameter (comparison in units of ns). If one wants to access the TDCData structure of the good hit, then the SBSData function GetGoodHit() can be called. If the TDC channel has any hits then the good_hit will be set.

If the element has ADC data and the mode is kADCSimple, them the fADC.good_hit is set to zero. If the mode is kADC, then the code loops through all hits in the channel and marks good_hit index for the hit that is closest to the GoodTimeCut parameter. For now only one hit is found for the kWaveform mode so good_hit index set to zero if the ADC time > 0

4 SBSGenericDetector::CoarseProcess

The CoarseProcess method separately loops through the reference time elements and the detector elements. The goal is to fill the fRefGood and the fGood SBSGenericOutputData structures for the output tree. The SBSGenericOutputData is a structure the contains vectors of the row,col,layer, element ID. SBSGenericOutputData contains vectors of the ADC integral, pedestal subtracted ADC integral, the calibrated pedestal subtracted ADC integral, FADC pulse amplitude, FADC pedestal subtracted pulse amplitude and the FADC pulse time. SBSGenericOutputData contains vectors of the TDC LE time, TE time and Tot (TE-LE) time. SBSGenericOutputData contains vectors of number of samples, index for the start of samples and the samples of the full FADC waveform.

The method first loops through the reference time elements. The fRefGood structure is filled with row, column, layer and element ID. If the refer-

ence time element is a TDC and has TDC data, then the TDCHit structure for the good reference time hit is retrieved and the fRefGood TDC structure is filled. By default SBSGenericDetector sets fStoreEmptyElements to true, but the detector class that inherits could set fStoreEmptyElements to false or this can be set in the replay script. If fStoreEmptyElements is true then the TDC structure is filled with kBig.

If the element has TDC data then it is possible have the code store the TDC hit information for all hits in the reference time elements in the fRefRaw SBSGenericOutputData structure. By default SBSGenericDetector sets fStoreRawHits to false but the detector class that inherits could set fStorerawHits to true or this can be set in the replay script. If true, then the fRefRaw struture is filled with the element ID and the calibrated LE, TE and Tot values for all hits in an element for all elements in the loop that have TDC data.

Need to add code for processing the ADC reference times

The method then loops through all the detector elements. If the element has TDC data, then fGood is filled with TDC information. If fStoreEmptyElements is true then the TDC information is filled with kBig. If fStoreRawHits is true, then all the hits in the element is loaded into the fRaw SBSGenericOutputData structure.

Next, if the element has ADC data (fHasData is true), then the "good" ADC hit is used to fill the fGood with ADC information. If the ADC-mode is kADCSimple, then the fGood pedestal, raw integral ("a"), raw -pedestal ("a_p") and the calibrated value ("a_c") are loaded. For kADC mode, the amplitude ("a_amp"), amplitude - pedestal ("a_amp_p") and pulse time ("a_time") are loaded. If fStoreRawHits is true (by default it is false) then the element ID, calibrated integral, calibrated amplitude and pulse time are loaded into fRaw structure. If fStoreEmptyElements is true (it is set true by default) then the ADC information is filled with zero. If the ADC mode is Waveform, then the fGood pedestal, raw integral ("a"), raw -pedestal ("a_p"), the calibrated value ("a_c"), the amplitude ("a_amp"), amplitude - pedestal ("a_amp_p") and pulse time ("a_time") are loaded. If fStoreRawHits is true, the index of the sample data in the "samps" vector ("sidx"), the number of samples per element ("nsamps") and the calibrated sample data ("samps") are loaded into a vector for all elements with data.

In general, a detector class that inherits from the GenericDetector should first call GenericDetector::CoarseProcess.

5 SBSGenericDetector::FineProcess

Does nothing.

6 SBSGenericDetector Tree variables

For the tree variables, most are arrays of variable size depending on the number of hits. The total number of ADC and TDC channels hit in the Decode method is fNhits. The ADC and TDC channels are not counted separately. Need to add ADC reference time variables. Need to add hits.ADCElemID

Tree name	Description	Global variable
nhits	Number of total hits	fNHits
nrefhits	Number of total reference time hits	fNRefHits
ngoodTDChits	Number of total good TDC hits	fNGoodTDChits
ngoodADChits	Number of total good ADC hits	fNGoodADChits

The good hit information is the following variables. If fStorerawHits is set true, then the TDC hit info is stored. The TDC reference times have the same tree variable names with *Ref.* in front of the name.

Tree name	Description	Global variable
tdcelemID	Element ID in chanmap	fGood.TDCelemID
tdcrow	Row	fGood.TDCrow
tdccol	Column	fGood.TDCcol
tdclayer	Layer	fGood.TDClayer
tdc_mult	# of TDChits in channel	fGood.t_mult
tdc	Calibrated Good hit Leading edge TDC	fGood.t
tdc_te	Calibrated Good hit Trailing Edge TDC	fGood.t_te
tdc_tot	Calibrated Good hit Time over Threshold	fGood.t_Tot
hits.elemID	all hit's Element ID	fRaw.elemID
hits.t	all hit's Calibrated Leading edge TDC	fRaw.t
hits.t_te	all hit's Calibrated Trailing edge TDC	fRaw.t_te
hits.t_tot	all hit's TDC Time Over Threshold	fRaw.t_ToT

Tree name	Description	Global variable
adcelemID	Element ID in chanmap	fGood.ADCelemID
adcrow	Row	fGood.ADCrow
adccol	Column	fGood.ADCcol
adclayer	Layer	fGood.ADClayer
ped	ADC pedestal	fGood.ped
a	ADC integral of the good hit (units pC)	fGood.a
a_p	Pedestal subtracted ADC integral of the good hit (units pC)	fGood.a_p
a_c	Energy Calibrated Pedestal subtracted ADC integral of the good hit (units GeV)	fGood.a_c
adc_mult	# of ADChits in channel	fGood.a_mult
a_amp	ADC peak of the good hit(units mV)	fGood.a_amp
a_amp_p	Pedestal subtracted ADC peak of the good hit (units mV)	fGood.a_amp_p
a_time	ADC time of the good hit (units ns)	fGood.a_time
a_amp	ADC peak of the good hit (units mV)	fGood.a_amp
hits.a	all ADC integral of the all hits (units pC)	fRaw.a
hits.a_amp	all ADC peaks of the all hits (units mV)	fRaw.a_amp
hits.a_time	all ADC time of the all hits (units mV)	fRaw.a_time

Tree name	Description	Global variable
samp_idx	Index in the samples vector for given row-col element	fGood.sidx
samp_nsamps	Number of samples for a given row-col element	fGood.nsamps
samp	Calibrated Pedestal subtracted ADC samples (units mV) in 4ns bins	fGood.samps

7 Short description of FADC250

The FADC250 is a fast data acquisition electronics module which is able to sample a signal pulse at 250MHz (i.e. 4ns time sample). It has two main data acquisition modes:

- Mode 1: all samples within a defined time window are recorded; this allows to perform the waveform reconstruction offline.
- Mode 7: if the recorded pulse passes a defined threshold, the following quantities are recorded: the pedestal, the pulse time, the pulse amplitude, and the pulse integral over a defined time window; Up to 3 pulses can be recorded the trigger window

For the SBS hadron calorimeter (HCal) and SBS BBShower/PreSHower, the FADC will be used with acquisition mode 1.

Details on the algorithm of the FADC is given in the manual. Here is a short description. In mode 7, a threshold is defined in the ROL FADC configuration file for each channel in the FADC. When a pulse has one sample that passes the threshold, then that sample is marked as the threshold crossing sample, TC. The pulse is integrated from NSB samples before the TC and NSA samples after the TC. NSB and NSA are programmed in the ROL FADC configuration file. The average of first four samples in the trigger window are used to determine the pedestal or VMIN. The peak amplitude (VPEAK) is determined by finding a sample beyond TC for which the sample value first decreases. The fine time, TF, is determined in bins of 62.5ps. First the time bin is roughly determined by the time bin of the half amplitude (VMID = (VPEAK + VMIN) / 2)). Then the fine time is determined as TF = 64*(VMID – V(N1)) / (V(N1+1) – V(N1)). The sample number N1 is found on the leading edge of the pulse that satisfies V(N1) $\leq$ VMID $<$ V(N1+1).