ClearPET Project

LMF specifications

February 2005

C. Morel[1]
L. Simon[1]
M. Krieguer[2]
M. Rey[1]
[1]LPHE/EPFL
[2]IIHE/VUB

1. Introduction

This document describes the specifications of the List Mode Format (LMF) defined for the ClearPET project to store singles and coincidences on an event-by-event basis.

2. List Mode Format (LMF) specifications

3.1 The LMF header file

The LMF header file is in ASCII format. The size of the LMF header file is not fixed and it can include any relevant information about the scanner and the scan.

3.2 The LMF binary file

The LMF binary file contains singles or coincidences on an event-by-event basis. It starts with a binary encoding header of variable length, followed itself by binary records of predefined lengths such as the event record, the count rate record, the simulation record, or any auxiliary information record outsourced from an external device. Each record always starts with a tag of 4 bits merged with the time stamp of the record.

3.2.1 The LMF encoding header

The LMF format allows to encode detector IDs over 16-bits unsigned short– (`u16`), 32-bits unsigned long– (`u32`) or 64-bits unsigned long long– integers (`u64`). The LMF encoding header sets the number of bits used to store detector ID and contains a description of each record type encoded over a `u16` whose first 4 bits are used to assign a tag to the record type which is described. It also contains a description of the scanner topology which consists of a cylindrical arrangement of flat detectors named *rsectors* regularly distributed around the scanner axis. *rsectors* are subdivided into two-dimensional *module*, *sub-module*, and *crystal* arrays that can have several *layers*.

The first `u16` is used to set the number of bits used to store detector ID. Integer values of 0, 1, and 2 are used to set it to 16-, 32- and 64-bits, respectively. If the first bit of the LMF encoding header is equal to 1, detector ID is encoded over 16 bits by default. In this case, the LMF encoding header starts directly by the rule used to store the encoding rule for detector ID described below.

The second `u16`, `u32` or `u64` is used to store the encoding rule for detector ID (see 3.2.2). For example, using `u16` ID encoding, `sssMMmmcccccccl` will be stored as `1110011000000001`, where the allowed maximum number of rings and sectors, i.e. *rsectors*, is 8 (encoded as `111`, i.e. *rsectors* numbered from 0 to 7), *modules* 4 (encoded as `00`), *sub-modules* 4 (encoded as `11`), *crystals* 256 (encoded as `00000000`) and *layers* 2 (encoded as `1`). The next `u16`, `u32` or `u64` of the LMF encoding header are used to

store a description of the scanner which specifies the numbers of *rsectors*, *modules*/*rsector*, *sub-modules*/*module*, *crystals*/*sub-module* and *layers*/*crystal*, according to the encoding rule described above. The next and over next `u16`, `u32` or `u64` describe the tangential and axial topology of the scanner structure, using the same encoding pattern as for the previous `u16`, `u32` or `u64` information. For example, using `u16` ID encoding, the tangential number *n* of *crystals*/*sub-module* is encoded in the next `u16`, and the axial number *m* of *crystals*/*sub-module* is encoded in the over next `u16`. Consistency of the scanner description implies that *n* □ *m* gives the numbers of *crystals*/*sub-module* encoded in the second `u16`, `u32` or `u64`. Moreover, since the number of rings is given by the number of axial *rsectors*, and the number of sectors by the number of tangential *rsectors*, the number of *rsectors* is given by the number of rings multiplied by the number of sectors.

Finally, the last `u16` of the encoding header is used to store the number of different record types which are described. Then, encoding of each type of record is stored using `u16`.

3.2.2 The LMF event record

The LMF event record allows to record singles or coincidences associated by the acquisition electronics. The LMF event record contains the following information:

- record tag and time stamp of the event (1+23 = 24 bits, 23 bits available for time stamp since tag `0` is exclusively used for event record)
- TOF (`u8`, only used for coincidences)
- additional `u32` for singles time stamp (thus singles time stamps are encoded over a total of 63 bits)
- detector ID and DOI (`u16`, `u32` or `u64`)
- energy deposited into the detector *module* (`u8`)
- energy and DOI of neighbouring *crystals* (`u8`)
- gantry position (`u16`)
- bed position (`u16`)
- external source angle position (`u16`)
- external source axial position (`u16`)

The encoding pattern describing the event record (event record header) is defined by `TTTTcdEnNNgbsGZR` (using `u16` ID encoding), with the following bit correspondence scheme:

`TTTT` : tag of the event record (i.e. `0xxx`, with all other tags starting by `1xxx`)
`c` : 0 = singles event, 1 = coincidence event
`d` : detector ID and DOI

`E` : energy deposited in the detector *module*

`n` : energy and DOI of neighbouring *crystals*

`NN` : order of neighbourhood

`g` : azimuth position of the gantry

`b` : axial position of the bed or the gantry

`s` : external source angle and axial positions

`G` : Monte Carlo simulated data

`Z` : neighbour information given by the FPGA readout

`R` : reserve

Consequently, the event format is coded as:

`0xxx xxxx xxxx xxxx xxxx xxxx`	tag followed by the time stamp of the record
`xxxx xxxx`	TOF if `c`, otherwise time stamp of the record
`xxxx xxxx xxxx xxxx`	time stamp of the record if `!c`
`xxxx xxxx xxxx xxxx`	
`xxxx xxxx xxxx xxxx`	detector ID and DOI of 1st single if `d`
`xxxx xxxx xxxx xxxx`	detector ID and DOI of 2nd single if `d&&c`
`xxxx xxxx xxxx xxxx`	gantry position if `g`
`xxxx xxxx xxxx xxxx`	bed position if `b`
`xxxx xxxx xxxx xxxx`	external source angle position if `s`
`xxxx xxxx xxxx xxxx`	external source axial position if `s`
`xxxx xxxx`	energy deposited into detector *module* of 1st single if `E`
`xxxx xxxx`	energy and DOI of neighbouring *crystals* of 1st single if `n`
`.... ....`	
`.... ....`	
`.... ....`	repeated V(`NN`) times
`xxxx xxxx`	energy deposited into detector module of 2nd single if `E&&c`
`xxxx xxxx`	energy and DOI of neighbouring crystals of 2nd single if `n&&c`
`.... ....`	
`.... ....`	
`.... ....`	repeated V(`NN`) times
`xxxx xxxx`	FPGA readout neighbour information if `Z`

`xxxx xxxx`	FPGA readout neighbour information for 2nd single if `Z&&C`

V(`NN`) is the size of the neighbourhood which depends on the neighbourhood order, as shown in Fig. 1. Crystal cells are scanned in a predefined order within a given neighbourhood, considering a toroidal crystal matrix topology to ensure that V(`NN`) remains independent of detector ID.

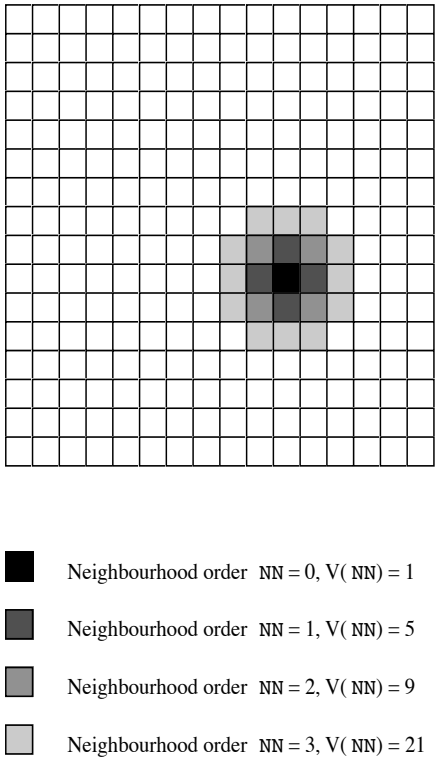

Figure 1: Definition of V(`NN`)

Detector ID and DOI allow to locate the interaction position within the detector architecture. In the example given above, it is encoded as `sssMMmmcccccccc1`, where:

`sss`	: *rsector* (0-7)
`MM`	: *module* (0-3)
`mm`	: *sub-module* (0-3)
`cccccccc`	: *crystal* (0-255)
`1`	: *layer* (0-1)

Using this encoding hierarchy, *rsectors* are divided into *modules*, *modules* comprise a number of *sub-modules*, *sub-modules* comprise a number of *crystals*, and *crystals* comprise a number of *layers*.

The *FPGA neighbour information* (`u8`) specifies which neighbour crystal of the main crystal has been activated. The value is 99 if no neighbour crystals have been detected by the FPGA readout. Otherwise, we use the following numbering convention where X represents the triggering crystal:

0	1	2
3	X	4
5	6	7

3.2.3 The LMF count rate record

A count rate record is regularly issued at a predefined rate (typically of the order of 1 second). This record stores count rates for singles summed over rings, sectors or modules, as well as the total singles, coincidence, and random count rates. It also stores the detector rotation, and bed axial speeds.

The encoding pattern describing the event count rate record (count rate header) is defined by `TTTTsSScFrbRRRRR` (`u16`), with the following bit correspondence scheme:

`TTTT` : tag of the event record (e.g. set to 1000)
`s` : singles count rate
`SS` : 00 = total only, 01 = *rsector*, 10 = *module*, 11 = *sub-module*
`c` : total coincidence count rate
`F` : total random count rate
`r` : rotation speed
`b` : bed or gantry axial speed
`RRRRR` : reserve

Consequently, the count rate format is coded as:

`1000 xxxx xxxx xxxx xxxx xxxx`	tag followed by the time stamp of the record
`0000 0000`	not used
`xxxx xxxx xxxx xxxx`	
`xxxx xxxx xxxx xxxx`	total singles rate if `s`
`.... .... .... ....`	
`.... .... .... ....`	

`.... .... .... ....`	repeated as function of `SS` if `s&&(SS!=00)`
`xxxx xxxx xxxx xxxx`	total coincidence rate if `c`
`xxxx xxxx xxxx xxxx`	total random rate if `F`
`xxxx xxxx`	rotation speed if `r`
`xxxx xxxx`	bed axial speed if `b`