ClearPET Project

LMF Gate Digi Record Implementation
Version 1.1

CONFIDENTIEL

Luc SIMON[1]
Daniel STRUL[1]
Christian MOREL[1]
May 7, 2003
[1]*IPHE/UNIL*

Introduction

This document describes the implementation of a new record type for List Mode Format (LMF) : the gate digi record.

This new gate digi record keeps the philosophy of the two first LMF implemented records:
- event record
- count rate record
described in *ClearPET list mode format implementation*, *LMF specifications* and *Software Design*.

The only difference is that it is impossible to find a LMF file containing only this record: the gate digi record is always attached to the event record. You can consider it like an extension of the event record, able to store some specific GATE simulation informations.

Consequently, the gate digi record has no specific tag (like event and count rate record), and always follows an event record.

A flag is reserved in the event record header to specify if the event record is followed by a gate digi record.

1 Gate digi header

1.1 Format

The gate digi header structures describes what is contained in the gate digi record. This header is stored in the head of the LMF binary file (*.ccs* file) as a two bytes pattern located just after the event record pattern.
The 16 bits of the gate digi header pattern are:

```
TTTT cSpe rGMR RRRR
```

where:
TTTT = 1100 (tag of gate digi header)
c = 1 if the number of compton interactions is stored (else 0)
S = 1 if the source ID is stored (else 0)
p = 1 if the source decay XYZ positions are stored (else 0)
e = 1 if the event ID is stored (else 0)
r = 1 if the run ID is stored (else 0)
G = 1 if the global digi XYZ positions in detector are stored (else 0)
M = 1 if the multiple ID generated by LMF coincidence sorter is stored (else 0)
R RRRR = 0 0000 are reserved bits

1.2 Implemented C-like structure

All informations needed to build the gate digi header are stored in the following structure:

```
struct LMF_ccs_gateDigiHeader
{
  unsigned char comptonBool;
  unsigned char sourceIDBool;
  unsigned char sourceXYZPosBool;
  unsigned char eventIDBool;
  unsigned char runIDBool;
  unsigned char globalXYZPosBool;
  unsigned char multipleIDBool;
};
typedef struct LMF_ccs_gateDigiHeader GATE_DIGI_HEADER;
```

2 Gate digi record

2.1 Format

The gate digi record's format depends of the gate digi header, but also of the event header. If the *event header's coincidence bool* (C) is TRUE the gate digi record stores informations concerning two photons (one photon otherwise). If the *event header's neighbour bool* (n) is TRUE, the gate digi record stores informations concerning the XYZ positions of the neighbouring crystals of first photon (and also of second one if C). The number of neighbours k(NN) depends of the two NN bits of the event header: (cf *Software Design*)
$k(00) = 0$
$k(01) = 4$
$k(10) = 8$
$k(11) = 20$

These records are stored in the body of the LMF binary (*.ccs* file) in the following format:

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx	run ID if r
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx	event ID of 1st photon if e
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx	event ID of 2nd photon if e && C
xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx	multiple ID of coincidence if C && M
xxxx xxxx xxxx xxxx	source ID of 1st photon if S
xxxx xxxx xxxx xxxx	source ID of 2nd photon if S && C
xxxx xxxx xxxx xxxx	source X position of 1st photon if p
xxxx xxxx xxxx xxxx	source Y position of 1st photon if p
xxxx xxxx xxxx xxxx	source Z position of 1st photon if p
xxxx xxxx xxxx xxxx	source X position of 2nd photon if p && C
xxxx xxxx xxxx xxxx	source Y position of 2nd photon if p && C
xxxx xxxx xxxx xxxx	source Z position of 2nd photon if p && C
xxxx xxxx xxxx xxxx	digi X position of 1st photon if G
xxxx xxxx xxxx xxxx	digi Y position of 1st photon if G
xxxx xxxx xxxx xxxx	digi Z position of 1st photon if G
....	digi XYZ position of 1st photon neighbours:
....	6 bytes repeated k(NN) times
....	if G && n
xxxx xxxx xxxx xxxx	digi X position of 2nd photon if G
xxxx xxxx xxxx xxxx	digi Y position of 2nd photon if G
xxxx xxxx xxxx xxxx	digi Z position of 2nd photon if G
....	digi XYZ position of 2nd photon neighbours:
....	6 bytes repeated k(NN) times
....	if G && n && C
xxxx xxxx	number of compton of 1st photon if c && !C
xxxx	number of compton of 1st photon if c && C
xxxx	number of compton of 2nd photon if c && C

2.2 Implemented C-like structure

All informations needed to build the gate digi record are stored in the following structure:

```
struct LMF_ccs_gateDigiRecord
{

  unsigned long runID; // 4 bytes
  unsigned long eventID[2]; // 4 bytes
  unsigned short sourceID[2];
  struct LMF_ccs_XYZpos sourcePos[2];
  struct LMF_ccs_XYZpos globalPos[42];
  unsigned char numberCompton[2];
  unsigned long multipleID;
};
typedef struct LMF_ccs_gateDigiRecord GATE_DIGI_RECORD;
```

where the *LMF_ccs_XYZpos* structure is:

```
struct LMF_ccs_XYZpos
{
  short X;
  short Y;
  short Z;
};
```

Contents