Fun4All 入門 (Fun4All_Intt_Combiner.C)

```
Fun4AllStreamingInputManager in = ...        create an object
SingleInttPoolInput *sngl = ...              create an object

in → registerStreamingInput ( sngl , ... )   register to input manager
```

(ここで m_intt_bco_range と m_intt_negative_bco が作られる。
これらのメンバ変数は FillIntt() の中で処理終了を判定するための条件として使われる)

↑ memo for myself

```
se → run(nEvents)
```

Here, in→run() will be called.
Inside in→run(), in→FillIntt() is called.

This creates an InttRawHit node in the output DST file, and add raw hits to a container.

```
FillIntt()
    FillInttPool()
        sngl → FillPool
```

ここで in のメンバ変数のひとつ m_InttRawHitMap が作られる。

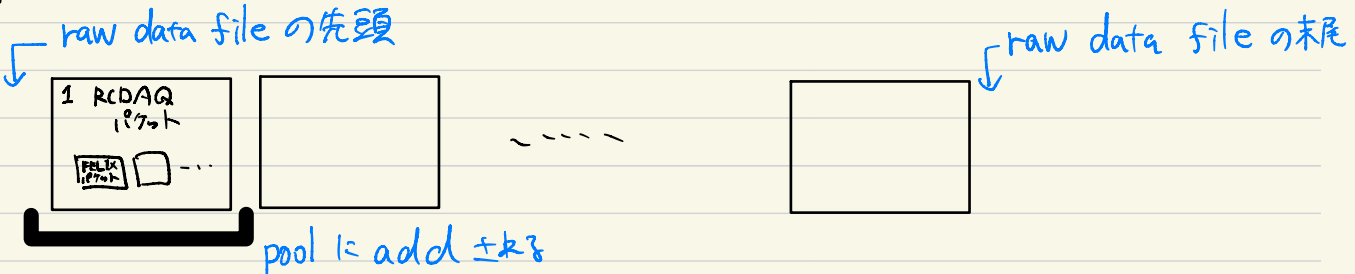
StreamingInputManager() → AddInttRawHit(gtm_bco, newhit.get())

```
    while (                )
    {
      for ( auto intthititer : m_InttRawHitMap )   ← コレ
      {
        inttcont → AddHit (intthititer )
      }                                  ヒットが詰められていく

      全 SingleInttPoolInput に対して
      iter → CleanupUsedPackets

      ここで1イベントをちょこっとだけ clear する

    }
    return 0
```

SingleInttPoolInput :: FillPool 図解

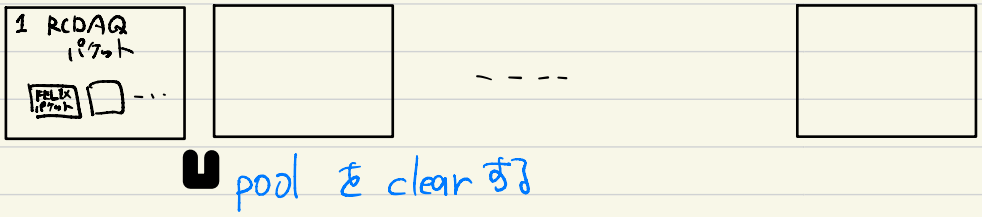
1周目

① GetSomeMoveEvents() で OK が返される (最初は poolmap.empty を満たすので.)
② pool に RCDAQ パケットを1つ add する



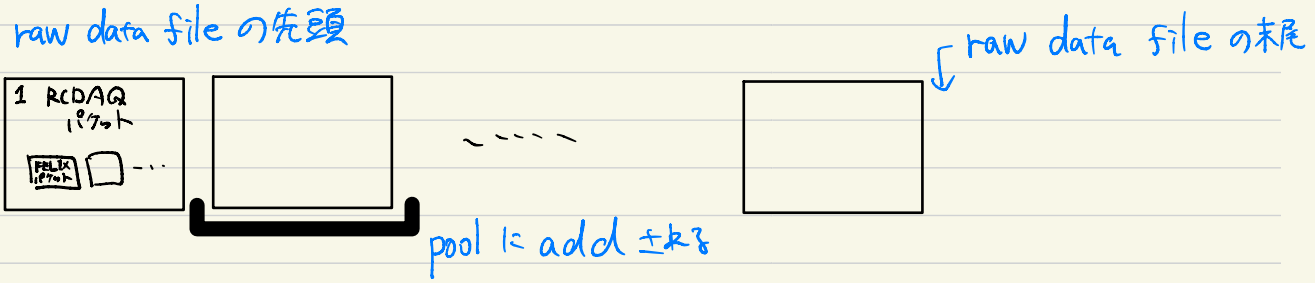
③ pool → depth_ok() のチェック後, (decode が暗に行われてから)
FunAllStreamingInputManager に InttRawHit と gtm_bco の組を渡す

④ pool → next()



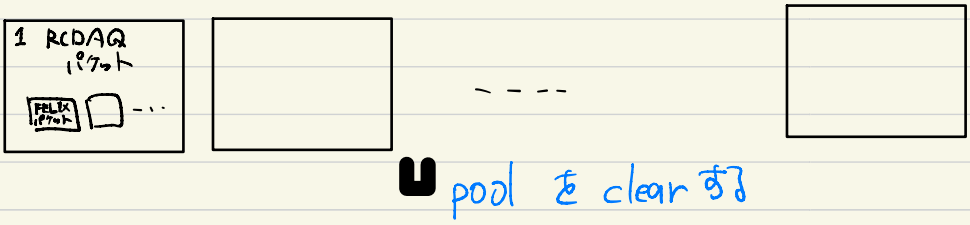
2周目

① GetSomeMoveEvents() で OK が返される (AllDone になるまでは基本 OK (true))
② pool に RCDAQ パケットを1つ add する



③ pool → depth_ok() のチェック後, (decode が暗に行われてから)
FunAllStreamingInputManager に InttRawHit と gtm_bco の組を渡す

④ pool → next()



N周目

① GetSomeMoreEvents() で OK が返される (AllDone になるまでは基本 OK (true))

② pool に RCDAQ パケットを 1つ add する

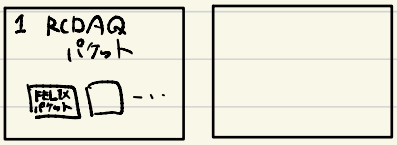
raw data file の先頭

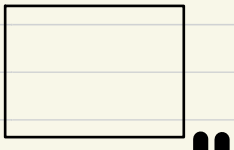
raw data file の末尾

pool に add される

③ pool → depth_ok() のチェック後, (decode が皆に行われてから)
Fun4AllStreamingInputManager に InitRawHit と gtm_bco の値を渡す

④ pool → next()





pool を clear する

最後

① GetSomeMoreEvents() で false が返される (AllDone の判定で)

SingleInttPoolInput :: FillPool

FillPool

```
while ( GetSomeMoreEvents(0) )        ← RCDAQ パケットに相当 たぶん
{
  Event  *evt = GetEventIterator() -> getNextEvent()

  int  npacket = evt -> getPacketList (plist, 1)
                                       ← 実は 1

  npacket の数だけ(実際には1度だけ)以下を実行

  poolmap[ plist[i]-> getIdentifier() ] = new  InttPool (1000, 100)

  poolmap[ plist[i]-> getIdentifier() ] -> addPacket ( plist[i] )

        packetID → RCDAQパケットというmap.   pool に( packetData に)
        felixのID                            packet (たぶん RCDAQ パケット)
                                             を通 future (何個?)

  poolmap の各要素で順番に以下を実行

  InttPool  pool = ---

  if ( pool -> depth_ok() )
  {
    int  num_hits  =  pool -> iValue (        )
    int  numBCOs   =  pool -> iValue (        )

    for ( int j=0 ; j < numBCOs ; j++ )
    {
      uint64_t  bco  =  pool -> lValue( j, "BCOLIST" )   ← j種類目の BCOをとってくる

      largest_bco と比較して大きいならすばやく
      if( bco < minBCO )  continue
      if( !IsStandAloneMode() && bco > minBCO * 2 ) continue

      skip this = false        セレクションをとおったのでキロクする flag
      m_BclkStack.insert( bco )
      m_BclkStackPacketMap[packet_id].insert( bco )
    }
```

```
if (skipthis)
{
    ...

}
else
{
    for (int j=0; j< num_hits; j++)
    {
        auto newhit = std::make_unique<IntttRawHitv2>()

        newhit -> set_packetid(...)
        newhit -> set_...
            ;

        StreamingInputManager()->AddHit(gtm_bco, newhit.get())
            ;

    }

pool -> next()
}
```

highest_bclk - mInttRawHitMap.begin()->first < MaxBclkDiff()

= default 1M