| LCOV - code coverage report |
| --- |
|  |
| | Current view: | [top level](http://docs.google.com/index.html) - [TimeCalculators/src](http://docs.google.com/index.html) - ShotTimeCalculator.cpp (source / [functions](http://docs.google.com/ShotTimeCalculator.cpp.func.html)) |  |  | Hit | Total | Coverage | | --- | --- | --- | --- | --- | --- | --- | | Test: | EBM9K Converter Coverage Report |  | Lines: | 126 | 132 | 95.5 % | | Date: | 2012-03-23 |  | Functions: | 12 | 13 | 92.3 % | | Legend: | Lines: hit not hit | Branches: + taken - not taken # not executed |  | Branches: | 35 | 62 | 56.5 % | |  | |
|  |

|  |
| --- |
| Branch data Line data Source code  1 : : #include <math.h>  2 : : #include <iostream>  3 : : #include "DPL.h"  4 : : #include "ShotTimeCalculator.h"  5 : : #include "ConverterParams.h"  6 : : #include "DoseHeader.h"  7 : : #include "DoseTable.h"  8 : : #include "AITable.h"  9 : : #include "SimplePattern.h"  10 : : #include "KBRTable.h"  11 : : #include "DOCTable.h"  12 : : #include "UnitConversion.h"  13 : : #include "Logger.h"  14 : : #include "HealthCheckCounter.h"  15 : : #include "ErrorHandler.h"  16 : :   17 : :   18 : :   19 : : namespace nft {  20 : :   21 : :   22 : : class ShotTimePreCalcInfo  23 : : {  24 : : private:  25 : : double preCalcTime;  26 : : bool preCalcDone;  27 : : ShotTimeCalculator::ProxMode proxMode;  28 : :   29 : : ShotTimePreCalcInfo()  30 : 1094 : :preCalcTime(0),  31 : : preCalcDone(false),  32 : : proxMode(ShotTimeCalculator::eProxOff)  33 : 1094 : {  34 : : }  35 : :   36 : : ~ShotTimePreCalcInfo()  37 : 1094 : {  38 : : }  39 : :   40 : : public:  41 : :   42 : : static ShotTimePreCalcInfo \*instance( ) throw()  43 : 8752 : {  44 [ - + ][ # # ]: 4376 : static ShotTimePreCalcInfo object;  [ # # ][ # # ]  [ + + ][ + - ]  45 : : return &object;  46 : 6564 : }  47 : :   48 : : inline void setPreCalcTime( double inValue ) throw()  49 : 1094 : {  50 : 1094 : preCalcTime = inValue;  51 : 1094 : preCalcDone = true;  52 : : }  53 : :   54 : : inline double getPreCalcTime() const throw()  55 : 0 : {  56 : : return preCalcTime;  57 : : }  58 : :   59 : : inline bool getPreCalcDone() const throw()  60 [ + - ]: 1094 : {  61 : : return preCalcDone;  62 : : }  63 : :   64 : : void setProxMode(ShotTimeCalculator::ProxMode value) throw()  65 : 1094 : {  66 : 1094 : proxMode = value;  67 : : }  68 : :   69 : : inline ShotTimeCalculator::ProxMode getProxMode() const throw()  70 : : {  71 : : return proxMode;  72 : : }  73 : : };  74 : :   75 : : /\*\*  76 : : \* Returns the floor form of binary logarithm for a 32 bit integer.  77 : : \* -1 is returned if n is 0.  78 : : \*/  79 : : static inline int32\_t log2(uint32\_t n)  80 : : {  81 : : int32\_t pos = 0;  82 : : if (n >= 1<<16) { n >>= 16; pos += 16; }  83 : : if (n >= 1<< 8) { n >>= 8; pos += 8; }  84 : : if (n >= 1<< 4) { n >>= 4; pos += 4; }  85 : : if (n >= 1<< 2) { n >>= 2; pos += 2; }  86 : : if (n >= 1<< 1) { pos += 1; }  87 : : return ((n == 0) ? (-1) : pos);  88 : : }  89 : :   90 : : const int32\_t FloatReadTable[] =  91 : : {  92 : : 0x00000fff, /\* parameter m = 12 figure cut mask \*/  93 : : 0x00001fff, /\* parameter m = 13 figure cut mask \*/  94 : : 0x00003fff, /\* parameter m = 14 figure cut mask \*/  95 : : 0x00007fff /\* parameter m = 15 figure cut mask \*/  96 : : };  97 : :   98 : : const int32\_t FloatReadTableBase = 12;  99 : :   100 : : //Implementation function that returns the Dose value for this pattern  101 : : #ifdef INTEGER\_CALC // The following code is not tested  102 : : uint16\_t  103 : : ShotTimeCalculator::calculateDoseValue(  104 : : SimplePattern & inoutPattern,  105 : : const PatternPosition & inPosition) {  106 : :   107 : : const DoseHeader \*pDoseHeader = mDoseTable->getHeader();  108 : : ConverterParams \*pParams = ConverterParams::instance();  109 : : static const uint32\_t meshSize = pParams->getColumnParams()->prox.mesh\_size;  110 : : static const int32\_t paramM = log2(meshSize);  111 : : int64\_t lenX = inoutPattern.getSizeX();   112 : : int64\_t lenY = inoutPattern.getSizeY();  113 : : int64\_t positionX = inPosition.getX();  114 : : int64\_t positionY = inPosition.getY();  115 : :   116 : : switch (inoutPattern.getCode())  117 : : {  118 : : case 0x01:  119 : : case 0x04:  120 : : positionX = positionX + (lenX >> 1);  121 : : positionY = positionY + (lenY >> 1);  122 : : break;  123 : : case 0x02:  124 : : positionX = positionX - (lenX >> 1);  125 : : positionY = positionY - (lenY >> 1);  126 : : break;  127 : : case 0x03:  128 : : positionX = positionX - (lenX >> 1);  129 : : positionY = positionY + (lenY >> 1);  130 : : break;  131 : : case 0x05:  132 : : positionX = positionX + (lenX >> 1);  133 : : positionY = positionY - (lenY >> 1);  134 : : break;  135 : : }  136 : : const DPMANAGER\_MESSAGE\_CONVERTER \*pStartMsg =   137 : : pParams->getStartupMessage();  138 : : int64\_t vchipX = pStartMsg->llVchipOriginX;  139 : : int64\_t vchipY = pStartMsg->llVchipOriginX;  140 : :   141 : : //NOTE::Pattern position is already w.r.t DPB  142 : : //For Dose table lookup it is translated w.r.t vchip  143 : :   144 : : int32\_t iIndexX = positionX  145 : : + ( vchipX  146 : : - static\_cast<int32\_t>(pDoseHeader->mOriginX) ) ;  147 : : int32\_t iIndexY = positionY  148 : : + ( vchipY  149 : : - static\_cast<int32\_t>(pDoseHeader->mOriginY) ) ;  150 : :   151 : : //printf("iIndexX=%ld iIndexY=%ld ", iIndexX, iIndexY );  152 : : int32\_t indexX = static\_cast<int32\_t>( iIndexX ) >> paramM;  153 : : int32\_t indexY = static\_cast<int32\_t>( iIndexY ) >> paramM;  154 : :   155 : : int32\_t decX = iIndexX & FloatReadTable[ paramM - FloatReadTableBase];  156 : : int32\_t decY = iIndexY & FloatReadTable[ paramM - FloatReadTableBase];  157 : :   158 : : const FloatTableReader \*pDoseTable = mDoseTable->getReader();  159 : :   160 : : //printf("%d %d ", indexX, indexY );  161 : : float D1 = pDoseTable->get( indexY, indexX );  162 : : float D2 = pDoseTable->get( indexY, indexX + 1 );  163 : : float D3 = pDoseTable->get( indexY + 1, indexX );  164 : : float D4 = pDoseTable->get( indexY + 1, indexX + 1 );  165 : :   166 : : int32\_t decXY = ( decX \* decY ) >> paramM ;  167 : : //printf("VALUE=%d %d %d %d %d %d %d ",  168 : : // iIndexX, iIndexY, indexX, indexY, decX, decY, decXY );  169 : :   170 : : float doseValue = D1 \* ( meshSize - decX - decY + decXY )  171 : : + D2 \* ( decX - decXY )  172 : : + D3 \* ( decY - decXY )  173 : : + D4 \* decXY;  174 : :   175 : : uint16\_t dose = static\_cast<int32\_t>(doseValue) >> paramM;  176 : : Logger::instance()->log(LogMessageTable::eSHOT\_TIME\_DOSE\_PARAM,  177 : : D1, D2, D3, D4, dose);  178 : : //printf("D1=%f D2=%f D3=%f D4=%f dose= %u\n", D1, D2, D3, D4, dose );  179 : : //printf(" dose= %u ", dose );  180 : :   181 : : //Convert and return  182 : : return dose;  183 : : }  184 : : #else  185 : : uint16\_t  186 : : ShotTimeCalculator::calculateDoseValue(  187 : : SimplePattern & inoutPattern,  188 : 2505 : const PatternPosition & inPosition ) {  189 : :   190 : 2505 : const DoseHeader \*pDoseHeader = mDoseTable->getHeader();  191 : 2505 : ConverterParams \*pParams = ConverterParams::instance();  192 [ + + ][ + - ]: 2505 : static const uint32\_t meshSize = pParams->getColumnParams()->prox.mesh\_size;  193 : 2505 : int64\_t lenX = inoutPattern.getSizeX();   194 : 2505 : int64\_t lenY = inoutPattern.getSizeY();  195 : 2505 : int64\_t positionX = inPosition.getX();  196 : 2505 : int64\_t positionY = inPosition.getY();  197 : :   198 : 2505 : switch (inoutPattern.getCode())  199 : : {  200 : : case 0x01:  201 : : case 0x04:  202 : 1941 : positionX = positionX + (lenX >> 1);  203 : 1941 : positionY = positionY + (lenY >> 1);  204 : 1941 : break;  205 : : case 0x02:  206 : 187 : positionX = positionX - (lenX >> 1);  207 : 187 : positionY = positionY - (lenY >> 1);  208 : 187 : break;  209 : : case 0x03:  210 : 180 : positionX = positionX - (lenX >> 1);  211 : 180 : positionY = positionY + (lenY >> 1);  212 : 180 : break;  213 : : case 0x05:  214 : 197 : positionX = positionX + (lenX >> 1);  215 : 197 : positionY = positionY - (lenY >> 1);  216 : 2505 : break;  217 : : }  218 : :   219 : : const DPMANAGER\_MESSAGE\_CONVERTER \*pStartMsg =   220 : 2505 : pParams->getStartupMessage();  221 : 2505 : int64\_t vchipX = pStartMsg->llVchipOriginX;  222 : 2505 : int64\_t vchipY = pStartMsg->llVchipOriginX;  223 : :   224 : : //NOTE::Pattern position is already w.r.t DPB  225 : : //For Dose table lookup it is translated w.r.t vchip  226 : :   227 : : double posX = translatePositionToDoseMap(  228 : : positionX,  229 : 2505 : vchipX, pDoseHeader->mOriginX, meshSize );  230 : :   231 : : double posY = translatePositionToDoseMap(  232 : : positionY,  233 : 2505 : vchipY, pDoseHeader->mOriginY, meshSize );  234 : :   235 : : //printf("%lf %lf\n", posX, posY);  236 : :   237 [ + - ]: 2505 : uint64\_t indexX = static\_cast<uint64\_t>( posX );  238 [ + - ]: 2505 : uint64\_t indexY = static\_cast<uint64\_t>( posY );  239 [ + - ]: 2505 : double decX = posX - static\_cast<double>(indexX);  240 [ + - ]: 2505 : double decY = posY - static\_cast<double>(indexY);  241 : :   242 : 2505 : const FloatTableReader \*pDoseTable = mDoseTable->getReader();  243 : 2505 : double D1 = pDoseTable->get( indexY, indexX );  244 : 2505 : double D2 = pDoseTable->get( indexY, indexX + 1 );  245 : 2505 : double D3 = pDoseTable->get( indexY + 1, indexX );  246 : 2505 : double D4 = pDoseTable->get( indexY + 1, indexX + 1 );  247 : :   248 : 2505 : double tmpX = 1 - decX;  249 : 2505 : double tmpY = 1 - decY;  250 : 2505 : double tmpXY = decX \* decY;  251 : : double doseValue = D1 \* tmpX \* tmpY  252 : : + D2 \* decX \* tmpY  253 : : + D3 \* tmpX \* decY  254 : 2505 : + D4 \* tmpXY;  255 : :   256 : : #ifdef TESTING\_THROUGH\_LOG  257 : : Logger::instance()->log(LogMessageTable::eSHOT\_DOSE\_DATA, static\_cast<float>(doseValue));  258 : : #endif  259 : :   260 : : //printf("%lu %lu %lf %lf %lf %lf %lf %lf %lf %lf \n", indexX, indexY, decX, decY, decX \* decY, D1, D2, D3, D4, doseValue );  261 : 2505 : Logger::instance()->log(LogMessageTable::eSHOT\_TIME\_DOSE\_VAL,  262 : : D1, D2, D3, D4, doseValue);  263 : :   264 : 2505 : return static\_cast<uint16\_t>( doseValue + 0.5 );  265 : : }  266 : : #endif  267 : :   268 : : void   269 : : ShotTimeCalculator::precalculate(ShotTimeCalculator::ProxMode inProxMode,  270 : : double inBaseDose,   271 : : double inCurrDensity,   272 : : double inStdEta) throw()  273 : 1094 : {  274 : 1094 : ShotTimePreCalcInfo \*pTimeInfo = ShotTimePreCalcInfo::instance();  275 : :   276 : 1094 : if( false == pTimeInfo->getPreCalcDone())  277 : : {  278 [ + + ]: 1094 : if( eProxOn == inProxMode )  279 : : {  280 [ + - ]: 197 : mPreCalcTime = (1.0 / C\_SHOT\_TIME) \* ( 1.0 + 2.0 \* inStdEta );  281 : : }  282 : : else  283 : : {  284 : 897 : mPreCalcTime = 1;  285 : : }  286 : :   287 : 1094 : mPreCalcTime \*= inBaseDose \* 1000 / inCurrDensity  288 : : / ConverterParams::instance()->  289 : : getStartupMessage()->dShotTimeResolution;  290 : : //mPreCalcTime is in dShotTimeResolution units (3.125ps/bit typically)  291 : :   292 : 1094 : pTimeInfo->setPreCalcTime( mPreCalcTime );  293 : 1094 : pTimeInfo->setProxMode( inProxMode );  294 : : }  295 : : else  296 : : {  297 : 0 : mPreCalcTime = pTimeInfo->getPreCalcTime();  298 : :   299 : : }  300 [ # # ]: 1094 : Logger::instance()->log(LogMessageTable::eSHOT\_TIME\_PRECALC,  301 : : mPreCalcTime);  302 : : }  303 : :   304 : : void  305 : : ShotTimeCalculator::calculateShotTime( SimplePattern &inoutPattern,  306 : : const PatternPosition &inPosition,  307 : : const int16\_t inDxpValue,  308 : : const int16\_t inSflNo,  309 : : uint32\_t inAI) throw(DPL::Error \*)  310 : 15960 : {  311 : :   312 : 15960 : HealthCheckCounter::instance()->incrCounter(  313 : : HealthCheckCounter::eUpdateFrequent);  314 : 15960 : double doseValueFromCalc = 1; //aitable structure change  315 : 15960 : uint16\_t doseValue = 0;  316 : 15960 : int32\_t tmpT = 0;  317 : :   318 : : //Shot Time calculated values  319 : 15960 : ShortTimeInfo shotTime();  320 : 15960 : ConverterParams \*pParams = ConverterParams::instance();  321 : 15960 : bool aidr = false;  322 : :   323 [ + - ]: 15960 : if(pParams)  324 : : {  325 : 15960 : const sgs\_column\_t \*secParam = pParams->getColumnParams();  326 [ + - ]: 15960 : if(secParam)  327 : : {  328 : 15960 : aidr = pParams->getColumnParams()->prox.mode & 0x08;  329 : : }  330 : : }  331 : :   332 : : //#ifdef ERROR\_SEED //<<Test\_Id:AITbl\_GetDoseForAI>>  333 : : // aidr=true;  334 : : //#endif  335 : 15960 : double ratio = 1.0; //structure change  336 [ + + ]: 15960 : if(aidr)  337 : : {  338 : 431 : try  339 : : {  340 : 431 : ratio = pParams->getAITable()->getDoseForAI(  341 : : inAI);  342 : : }  343 [ + - ]: 4 : catch(DPL::Error \*pChild)  344 : : {  345 : 4 : ErrorHandler \*pErr = ErrorHandler::instance();  346 : 4 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_,  347 : : \_\_LINE\_\_, ErrorHandler::eErrFunction,  348 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  349 : : "ShotTimeCalculator::calculateShotTime"));  350 : : }  351 : : }  352 [ + + ]: 15956 : if(eProxOn == mProxMode)  353 : : {  354 : 2505 : uint16\_t doseVal = calculateDoseValue(inoutPattern, inPosition);  355 [ + + ]: 2505 : if( NULL != mKBRTable ) // Get Dose correction value  356 : : {  357 : 2475 : doseValue = mKBRTable->getReader()->get(  358 : : static\_cast<uint64\_t>(doseVal),  359 : : static\_cast<uint64\_t>(inDxpValue));  360 : : }  361 : : else  362 : : {  363 : 30 : doseValue = doseVal;  364 : : }  365 : 2505 : doseValueFromCalc = ratio \* doseValue;  366 : : }  367 : : else  368 : : {  369 [ + + ]: 13451 : if(aidr)  370 : : {  371 : 5 : doseValueFromCalc = ratio;  372 : : }  373 : : else  374 : : {  375 : 13446 : doseValueFromCalc = 1;  376 : : }  377 : : }  378 : : const DPMANAGER\_MESSAGE\_CONVERTER \*pStartMsg  379 : 15956 : = pParams->getStartupMessage();  380 : 15956 : tmpT = static\_cast<int32\_t>(doseValueFromCalc \* mPreCalcTime  381 : : \* pStartMsg->dShotTimeGain  382 : : + 0.5);  383 [ + - ]: 15960 : inoutPattern.setShotTime(tmpT);  384 : : }  385 : :   386 : : uint32\_t   387 : : ShotTimeCalculator::calculateShotTimeRepetition(const SimplePattern \*pattern,  388 : : uint32\_t repetition, uint32\_t inSFLNo) throw(DPL::Error \*)  389 : 15978 : {  390 : 15978 : ConverterParams \*pParams = ConverterParams::instance();  391 : 15978 : int32\_t tmpT = pattern->getShotTime();  392 : : const DPMANAGER\_MESSAGE\_CONVERTER \*pStartMsg  393 : 15978 : = pParams->getStartupMessage();  394 : 15978 : uint32\_t repCount = pStartMsg->uiRepCount;  395 : 15978 : uint32\_t totalCount = pStartMsg->uiMaxSFLID \* repCount;  396 : 15978 : uint32\_t rem = tmpT % totalCount;  397 : : const DPMANAGER\_MESSAGE\_SFL\_INFO \*pInfo  398 : 15978 : = pParams->getLayerParams() + inSFLNo;  399 : 15978 : uint32\_t sfgCount = pInfo->uiSFCountX;  400 : 15978 : uint32\_t sflId = pInfo->uiSFLID;  401 : 15978 : uint32\_t threshold = sfgCount \* (sflId - 1);  402 : :   403 : :   404 : 15978 : uint32\_t shotTime = 0;  405 : :   406 [ + + ]: 15978 : if(rem >= (threshold + repetition + 1))  407 : : {  408 : 15544 : shotTime = (tmpT / totalCount) + 1;  409 : : }  410 : : else  411 : : {  412 : 434 : shotTime = (tmpT / totalCount);  413 : : }  414 : : //#ifdef ERROR\_SEED //<<Test\_Id:shotTime\_Set\_To\_65537>>  415 : : // shotTime = 65537;  416 : : //#endif  417 [ + - ]: 15978 : if(NULL != mDOCTable)   418 : : {  419 : 15978 : const int16\_t \*docTblParam = mDOCTable->getDOCData();  420 [ + + ]: 15978 : if(shotTime >= 65536)   421 : : {  422 : 6 : ErrorHandler \*pErr = ErrorHandler::instance();  423 : 6 : pErr->errorAdd(NULL, \_\_FILE\_\_, \_\_FUNCTION\_\_,  424 : : \_\_LINE\_\_, ErrorHandler::eErrFunction,  425 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  426 : : "in calculateShotTimeRepetition - array index out of bound"));  427 : :   428 : : }  429 : 15972 : shotTime = docTblParam[shotTime];  430 : : }  431 [ - + ]: 15972 : shotTime = (shotTime > MIN\_TP) ? shotTime : ADJ\_TP;  432 [ + - ]: 15972 : shotTime = static\_cast<uint32\_t>(shotTime \* pStartMsg->dShotTimeResolution);  433 : :   434 : 15972 : Logger::instance()->log(LogMessageTable::eSHOT\_TIME,  435 : : repetition+1, shotTime);  436 [ + - ]: 15978 : return shotTime;  437 : : }  438 : :   439 : : const uint32\_t ShotTimeCalculator::C\_SHOT\_TIME = 2048;  440 : : const double ShotTimeCalculator::BLK\_UNIT = 0.0125; //ns  441 : : uint16\_t ShotTimeCalculator::MIN\_TP;  442 : : uint16\_t ShotTimeCalculator::ADJ\_TP;  443 : :   444 : : ShotTimeCalculator::ShotTimeCalculator( ProxMode inMode,  445 : : const DoseTable \*inDoseTable,  446 : : const KBRTable \*inKBRTable,  447 : : const DOCTable \*inDOCTable)  448 : 3282 : :mPreCalcTime( ShotTimePreCalcInfo::instance()->getPreCalcTime() ),  449 : : mProxMode( inMode ),  450 : : mDoseTable( inDoseTable ),  451 : : mKBRTable( inKBRTable ),  452 : 3282 : mDOCTable( inDOCTable ) {  453 : : }  454 : :   455 : : ShotTimeCalculator::~ShotTimeCalculator()  456 : 6612 : {  457 : : }  458 : :   459 : : ShotTimeCalculator::ShotTimeCalculator(const ShotTimeCalculator & source)   460 : 3330 : :mPreCalcTime( source.mPreCalcTime ),  461 : : mProxMode( source.mProxMode ),  462 : : mDoseTable( source.mDoseTable ),  463 : : mKBRTable( source.mKBRTable ),  464 : : mDOCTable( source.mDOCTable )  465 : 3330 : {  466 : : }  467 : :   468 : : ShotTimeCalculator & ShotTimeCalculator::operator=(const ShotTimeCalculator & source)   469 : 0 : {  470 [ # # ]: 0 : if( this != &source )  471 : : {  472 : 0 : mPreCalcTime = source.mPreCalcTime;  473 : : }  474 : 0 : return \*this;  475 : : }  476 : :   477 : 1215 : } //namespace shot  478 : 2430 : |

|  |
| --- |
| Generated by: [LCOV version 1.9](http://ltp.sourceforge.net/coverage/lcov.php) |