| LCOV - code coverage report |
| --- |
|  |
| | Current view: | [top level](http://docs.google.com/index.html) - [LibVsb/src](http://docs.google.com/index.html) - ref.cpp (source / [functions](http://docs.google.com/ref.cpp.func.html)) |  |  | Hit | Total | Coverage | | --- | --- | --- | --- | --- | --- | --- | | Test: | EBM9K Converter Coverage Report |  | Lines: | 68 | 121 | 56.2 % | | Date: | 2012-03-23 |  | Functions: | 7 | 8 | 87.5 % | | Legend: | Lines: hit not hit | Branches: + taken - not taken # not executed |  | Branches: | 14 | 56 | 25.0 % | |  | |
|  |

|  |
| --- |
| Branch data Line data Source code  1 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   2 : : \* Copyrighted 2007 by Interra Systems, USA  3 : : \* All rights reserved   4 : : \*  5 : : \* Date of Creation: Fri Jul 20 15:54:06 IST 2007  6 : : \* File Name: ref.cpp   7 : : \* Purpose:   8 : : \* Author:   9 : : \*  10 : : \* Revision History:   11 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   12 : :   13 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  14 : : \* Header Include section   15 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  16 : : #include "ref.h"  17 : : #include "ErrorHandler.h"  18 : : #include "vsb\_types.h"  19 : : #include "Porting.h"  20 : : #include "context.h"  21 : : #include "stdio.h"  22 : : #include "Logger.h"  23 : : #include <iostream>  24 : : using namespace std;  25 : :   26 : :   27 : : namespace nft {  28 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  29 : : \* Function Name: VSBRefReader()  30 : : \* Purpose: This constructor initializes the ref file reader  31 : : \* with a ref file. This means caching  32 : : \* 1. The block size in both directions  33 : : \* 2. The block count in both directions  34 : : \* 3. The decode of the ref file header  35 : : \* Parameters: char\* rfile,  36 : : \* uint32\_t xBlkCnt,  37 : : \* uint32\_t yBlkCnt,  38 : : \* uint32\_t xBlkSz,  39 : : \* uint32\_t yBlkSz,  40 : : \* uint16\_t numCell  41 : : \* Return Values:   42 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  43 : :   44 : : VSBRefReader::VSBRefReader(char\* rfile,   45 : : uint32\_t xBlkCnt,  46 : : uint32\_t yBlkCnt,  47 : : uint32\_t xBlkSz,  48 : : uint32\_t yBlkSz,  49 : : uint16\_t orient  50 : : )   51 : 1094 : :mRefFile(rfile),   52 : : mNumXBlocks(xBlkCnt),  53 : : mNumYBlocks(yBlkCnt),  54 : : mBlockSizeX(xBlkSz),  55 : : mBlockSizeY(yBlkSz),  56 : : mNullFrame(false),  57 : : mOrientation(orient)  58 : :   59 : 2188 : {  60 : : //#ifdef ERROR\_SEED //<<Test\_Id:ref\_orient\_Set\_To\_2>>  61 : : // orient = 2;  62 : : //#endif  63 [ + + ][ # # ]: 1094 : if (orient == 0)  64 : : {  65 : 1 : mOrientation = VERTICAL;  66 : : }  67 [ + - ][ # # ]: 1093 : else if (orient == 1)  68 : : {  69 : 1093 : mOrientation = HORIZONTAL;  70 : : }   71 : : else  72 : : {  73 : 0 : Logger::instance()->log(LogMessageTable::eUTIL\_ERROR\_GENERAL\_MESSAGE,  74 : : "Incorrect frame orientation");  75 : :   76 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  77 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  78 : : ,ErrorHandler::eErrMessageString  79 : : ,pErr->getErrorMsg((ErrorHandler::eErrMessageString)   80 : : ,"Incorrect frame orientation"));  81 : : }   82 : : //Now decode the header of the ref.n file  83 : 1094 : uint32\_t \*length;  84 : 1094 : uint32\_t l\_length[5] = {0, 0, 0, 0, 0}; //length specific to LINUX  85 : : //#ifdef ERROR\_SEED //<<Test\_Id:ref\_rfile\_Set\_To\_NULL>>  86 : : // rfile = NULL;  87 : : //#endif  88 [ - + ][ # # ]: 1094 : if (NULL == rfile)  89 : : {  90 : 0 : Logger::instance()->log(LogMessageTable::eUTIL\_ERROR\_GENERAL\_MESSAGE,  91 : : "VSB Ref Data is (null)");  92 : :   93 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  94 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  95 : : ,ErrorHandler::eErrMessageString  96 : : ,pErr->getErrorMsg((ErrorHandler::eErrMessageString)  97 : : ,"VSB Ref Data is NULL"));  98 : : }  99 : 1094 : refSegment\_[VSB\_REF\_HEADER] = rfile; //refSegment\_[0]  100 : : //Each sub-segment of the header segment is   101 : : //4 bytes long  102 : 1094 : length = (uint32\_t\*)refSegment\_[VSB\_REF\_HEADER];  103 [ + + ][ # # ]: 5470 : for (uint16\_t i=0; i< 5; i++)  104 : : {  105 : 5470 : l\_length[i] = BE2LE\_32(length[i]);  106 : : }  107 : 1094 : length = l\_length;  108 : : //Check if this a null frame  109 : : //Definition: A null frame has only header   110 : : //segment defined  111 [ - + ][ # # ]: 1094 : if ( length[1] == 0 && length[2] == 0   [ # # ][ # # ]  [ # # ][ # # ]  112 : : && length[3] == 0  113 : : )  114 : : {  115 : 0 : mNullFrame = true;  116 : : }  117 : :   118 : :   119 : : //Fetch the segment lengths   120 : : //Skip the magic number  121 : 1094 : refSegLength\_[VSB\_REF\_HEADER] =   122 : : (length[1] \* WordSize) ;  123 : 1094 : refSegLength\_[VSB\_REF\_BLOCK\_POINTER] =   124 : : (length[2] \* WordSize) ;   125 : 1094 : refSegLength\_[VSB\_REF\_EXTENSION] =   126 : : (length[3] \* WordSize) ;   127 : 1094 : refSegLength\_[VSB\_REF\_BLOCK\_DEF] =  128 : : (length[4] \* WordSize);  129 : :   130 : :   131 : : //Get the segment offsets  132 : : //refSegment[VSB\_REF\_HEADER] is   133 : : // already populated above  134 : 1094 : refSegment\_[VSB\_REF\_BLOCK\_POINTER] =  135 : : refSegment\_[VSB\_REF\_HEADER]   136 : : + refSegLength\_[VSB\_REF\_HEADER];  137 : 1094 : refSegment\_[VSB\_REF\_EXTENSION] =  138 : : refSegment\_[VSB\_REF\_BLOCK\_POINTER]  139 : : + refSegLength\_[VSB\_REF\_BLOCK\_POINTER];  140 : 1094 : refSegment\_[VSB\_REF\_BLOCK\_DEF] =  141 : : refSegment\_[VSB\_REF\_EXTENSION]  142 : : + refSegLength\_[VSB\_REF\_EXTENSION];  143 : :   144 : : }  145 : :   146 : :   147 : :   148 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  149 : : \* Function Name: refReInit()  150 : : \* Purpose: Initialize the ref reader  151 : : \* Parameters:  152 : : char\* rfile,  153 : : uint32\_t xBlkCnt,  154 : : uint32\_t yBlkCnt,  155 : : uint32\_t xBlkSz,  156 : : uint32\_t yBlkSz,  157 : : uint16\_t orient  158 : : \* param1 : IN/OUT   159 : : \* param2 : IN/OUT   160 : : \* Return Values: bool  161 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  162 : :   163 : : bool VSBRefReader::refReInit(char\* rfile,  164 : : uint32\_t xBlkCnt,  165 : : uint32\_t yBlkCnt,  166 : : uint32\_t xBlkSz,  167 : : uint32\_t yBlkSz,  168 : : uint16\_t orient)  169 : 0 : {  170 : 0 : mRefFile = rfile;  171 : 0 : mNumXBlocks = xBlkCnt;  172 : 0 : mNumYBlocks = yBlkCnt;  173 : 0 : mBlockSizeX = xBlkSz;  174 : 0 : mBlockSizeY = yBlkSz;  175 : 0 : mNullFrame = false;  176 : 0 : mOrientation = orient;  177 [ # # ]: 0 : if (orient == 0)//Vertical  178 : : {  179 : 0 : mOrientation = VERTICAL;  180 : : }  181 [ # # ]: 0 : else if (orient == 1)  182 : : {  183 : 0 : mOrientation = HORIZONTAL;  184 : : }  185 : : else  186 : : {  187 : 0 : Logger::instance()->log(LogMessageTable::eUTIL\_ERROR\_GENERAL\_MESSAGE,  188 : : "Incorrect frame orientation");  189 : :   190 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  191 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  192 : : ,ErrorHandler::eErrMessageString  193 : : ,pErr->getErrorMsg((ErrorHandler::eErrMessageString)   194 : : ,"Incorrect frame orientation"));  195 : : }   196 [ # # ]: 0 : if (NULL == rfile)  197 : : {  198 : 0 : Logger::instance()->log(LogMessageTable::eUTIL\_ERROR\_GENERAL\_MESSAGE,  199 : : "VSB Ref Data is (null)");  200 : :   201 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  202 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  203 : : ,ErrorHandler::eErrMessageString  204 : : ,pErr->getErrorMsg((ErrorHandler::eErrMessageString)  205 : : ,"VSB Ref Data is NULL"));  206 : : }  207 : :   208 : : //Now decode the header of the ref.n file  209 : 0 : uint32\_t \*length;  210 : : //length specific to LINUX   211 : 0 : uint32\_t l\_length[5] = {0, 0, 0, 0, 0};   212 : :   213 : 0 : refSegment\_[VSB\_REF\_HEADER] = rfile;  214 : : //Each sub-segment of the header segment is   215 : : //4 bytes long  216 : 0 : length = (uint32\_t\*)refSegment\_[VSB\_REF\_HEADER];  217 : :   218 [ # # ]: 0 : for (int i=0; i<5; i++)  219 : : {  220 : 0 : l\_length[i] = BE2LE\_32(length[i]);  221 : : #ifdef DEVEL\_DEBUG  222 : : printf("Before conversion : %d\n", length[i]);  223 : : printf("After conversion : %d\n", l\_length[i]);  224 : : #endif  225 : : }  226 : 0 : length = l\_length;  227 : : //Check if this a null frame  228 : : //Definition: A null frame has only header   229 : : //segment defined  230 [ # # ][ # # ]: 0 : if ( length[2] == 0 && length[3] == 0  [ # # ]  231 : : && length[4] == 0  232 : : )  233 : : {  234 : 0 : mNullFrame = true;  235 : 0 : return false;  236 : : }  237 : :   238 : :   239 : : //Fetch the segment lengths   240 : : //Skip the magic number  241 : : #ifdef DEVEL\_DEBUG  242 : : cout<<"LENGTHS";  243 : : cout<<endl;   244 : :   245 : : cout<<"Start address of header seg. "; printf("%p\n",length + 1);  246 : : cout<<"Start address of BPointer seg. = "; printf("%p\n",length+2);  247 : : cout<<"header segment length = "; printf("%u\n", length[1]);  248 : : cout<<"Extn segment length in WORDS = "; printf("%u\n", length[3]);  249 : : cout<<"Block def. segment length in WORDS = "; printf("%u\n",length[4]);  250 : :   251 : : #endif  252 : 0 : refSegLength\_[VSB\_REF\_HEADER] =  253 : : (length[1] \* WordSize) ;  254 : 0 : refSegLength\_[VSB\_REF\_BLOCK\_POINTER] =  255 : : (length[2] \* WordSize) ;  256 : 0 : refSegLength\_[VSB\_REF\_EXTENSION] =  257 : : (length[3] \* WordSize) ;  258 : 0 : refSegLength\_[VSB\_REF\_BLOCK\_DEF] =  259 : : (length[4] \* WordSize);  260 : :   261 : :   262 : : //Get the segment offsets  263 : : //refSegment[VSB\_REF\_HEADER] is   264 : : // already populated above  265 : 0 : refSegment\_[VSB\_REF\_BLOCK\_POINTER] =  266 : : refSegment\_[VSB\_REF\_HEADER]  267 : : + refSegLength\_[VSB\_REF\_HEADER];  268 : 0 : refSegment\_[VSB\_REF\_EXTENSION] =  269 : : refSegment\_[VSB\_REF\_BLOCK\_POINTER]  270 : : + refSegLength\_[VSB\_REF\_BLOCK\_POINTER];  271 : 0 : refSegment\_[VSB\_REF\_BLOCK\_DEF] =  272 : : refSegment\_[VSB\_REF\_EXTENSION]  273 : : + refSegLength\_[VSB\_REF\_EXTENSION];  274 : :   275 : 0 : return true;  276 : : }  277 : :   278 : :   279 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  280 : : \* Function Name: refNearReferenceTableCreate()  281 : : \* Purpose: Identify the reference cells  282 : : \* Parameters: VSBCellRRIDTracker& cellCounter  283 : : \* Return Values: void  284 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  285 : :   286 : : void VSBRefReader::refNearReferenceTableCreate(  287 : : VSBCellRRIDTracker& cellCounter  288 : : )  289 : 1094 : {  290 : 1094 : vsbBlockDefinitionHeader \*vbdh ;  291 : 1094 : CellContext pcl ;  292 : 1094 : int refID = 0;  293 : :   294 : 1094 : pcl.end = refSegment\_[VSB\_REF\_BLOCK\_DEF];   295 : : //Now iterate over all blocks in this frame  296 : : // Each BLOCK DEFINITION:  297 : : // +-----------+ ^  298 : : // | vbdh | |  299 : : // | | |  300 : : // +-----------+ block data length   301 : : // | cell loc | |  302 : : // +-----------+ |  303 : : // | cell loc | |  304 : : // +-----------+ V  305 : : //   306 : : //  307 : : /\* The main algo. is:  308 : : for each cell c (over all blocks)  309 : : {  310 : : refID = c->relationRecordIndex;  311 : : refCountArr\_[refID] ++;  312 : : if (refCountArr\_[refID] == 1)  313 : : {  314 : : //store the block it belongs to  315 : : refBlkMapArr\_[refID].{IDx, IDy} = {blkIDX, blkIDY}  316 : : }  317 : : if (refCountArr\_[refID] == 2)  318 : : {  319 : : //this is a reference cell  320 : : //insert it into reference cell map  321 : : refCellMap\_[refID] = NULL;   322 : : }  323 : : }   324 : : \*/  325 : :   326 : :   327 [ + - ]: 1094 : while(pcl.end != refSegment\_[VSB\_REF\_BLOCK\_DEF] +   328 : : refSegLength\_[VSB\_REF\_BLOCK\_DEF])  329 : : {  330 : 1113 : vbdh = (vsbBlockDefinitionHeader \*)pcl.end ;  331 : 1113 : setNumCellLocations(vbdh->NumCellLocations);  332 : :   333 : : //We move to the first cell location  334 : 1113 : pcl.start = pcl.end + sizeof(vsbBlockDefinitionHeader) ;   335 : 1113 : pcl.current = pcl.start ;  336 : : //We move to the end of the block  337 : 1113 : pcl.end += (BE2LE\_32(vbdh->BlockDataLength) \* WordSize) ;  338 : 1113 : while( (refID = pcl.getRefIDNext()) != -1)  339 : : {  340 : 1261 : cellCounter.incrCountEncodeVc(refID);  341 : : }  342 : : }  343 : : }   344 : :   345 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  346 : : \* Function Name: refGetBlock  347 : : \* Purpose: Return the Cell Context for the current block.  348 : : \* Parameters: Context \*pctx,   349 : : \* int ibx,  350 : : \* int iby,   351 : : \* Block \*pb  352 : : \* Return Values: CellContext\*  353 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  354 : :   355 : : CellContext\* VSBRefReader::refGetBlock(Context \*pctx, int ibx,  356 : : int iby, Block \*pb)  357 : 1127 : {  358 : 1127 : Byte \*blockPointerAddress ;  359 : 1127 : vsbBlockPointer \*vbp ;  360 : 1127 : vsbBlockDefinitionHeader \*vbdh ;  361 : 1127 : CellContext \*pcl ;  362 : :   363 : 1127 : blockPointerAddress = refIndexToBlockOffset(ibx, iby) ;  364 : : //#ifdef ERROR\_SEED //<<Test\_Id:ref\_blockPointerAddress\_Set\_To\_NULL>>  365 : : // blockPointerAddress = NULL;  366 : : //#endif  367 [ - + ]: 1127 : if(NULL == blockPointerAddress)  368 : : {  369 : 0 : Logger::instance()->log(LogMessageTable::eUTIL\_ERROR\_GENERAL\_MESSAGE,  370 : : "Invalid Block Pointer Address");  371 : 0 : pb->IDx = pb->IDy = -1 ;  372 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  373 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  374 : : ,ErrorHandler::eErrMessageString  375 : : ,pErr->getErrorMsg((ErrorHandler::eErrMessageString)  376 : : ,"Invalid Block Pointer Address"));  377 : :   378 : 0 : return NULL; //No more blocks  379 : : }  380 : :   381 : 1127 : vbp = (vsbBlockPointer \*)blockPointerAddress ;  382 : :   383 : : //Porting Updation - 07 May 2004  384 : 1127 : pb->IDx = BE2LE\_16(vbp->IDx) ;  385 : 1127 : pb->IDy = BE2LE\_16(vbp->IDy) ;  386 : : //Porting Updation - 07 May 2004  387 : 1127 : pb->Dx = mBlockSizeX ;  388 : 1127 : pb->Dy = mBlockSizeY ;  389 : :   390 : : #ifndef SOLARIS //default LINUX  391 : :   392 : 1127 : int nTemp\_BlockPointer = BE2LE\_32(vbp->BlockPointer);  393 [ + + ]: 1127 : if(-1 == nTemp\_BlockPointer)  394 : : {  395 : : /\*  396 : : logAdd(glog, LOG\_LV4, "WRN:%d,%d is a Null block (BlockPointer = -1)\n",ibx,iby) ;  397 : : logAdd(glog, LOG\_LV4, "WRN:Block Pointer Structure (IDx, IDy) = (%d,%d)\n",pb->IDx, pb->IDy) ;  398 : : \*/  399 : 14 : return NULL ;  400 : : }  401 : :   402 : 1113 : pcl = &(pctx->pcl) ;  403 : 1113 : pcl->start = refSegment\_[VSB\_REF\_BLOCK\_DEF] +  404 : : (nTemp\_BlockPointer \* WordSize) ;  405 : 1113 : pcl->current = pcl->start + sizeof(vsbBlockDefinitionHeader) ;  406 : :   407 : 1113 : vbdh = (vsbBlockDefinitionHeader \*)(refSegment\_[VSB\_REF\_BLOCK\_DEF] + (nTemp\_BlockPointer \* WordSize)) ;  408 : 1113 : setNumCellLocations(vbdh->NumCellLocations);  409 : 1113 : pcl->end = pcl->start + (BE2LE\_32(vbdh->BlockDataLength) \* WordSize) ;  410 : :   411 : : #else // SOLARIS  412 : :   413 : : if(-1 == vbp->BlockPointer)  414 : : {  415 : : /\*  416 : : logAdd(glog, LOG\_LV4, "WRN:%d,%d is a Null block (BlockPointer = -1)\n",ibx,iby) ;  417 : : logAdd(glog, LOG\_LV4, "WRN:Block Pointer Structure (IDx, IDy) = (%d,%d)\n",pb->IDx, pb->IDy) ;  418 : : \*/  419 : : return NULL ;  420 : : }  421 : :   422 : : pcl = &(pctx->pcl) ;  423 : : pcl->start = refSegment\_[VSB\_REF\_BLOCK\_DEF] + (vbp->BlockPointer \* WordSize) ;  424 : : pcl->current = pcl->start + sizeof(vsbBlockDefinitionHeader) ;  425 : :   426 : : vbdh = (vsbBlockDefinitionHeader \*)(refSegment\_[VSB\_REF\_BLOCK\_DEF] + (vbp->BlockPointer \* WordSize)) ;  427 : : setNumCellLocations(vbdh->NumCellLocations);  428 : : pcl->end = pcl->start + (vbdh->BlockDataLength \* WordSize) ;  429 : : #endif  430 : :   431 : 1127 : return pcl ;  432 : : }  433 : :   434 : :   435 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  436 : : \* Function Name: refIndexToBlockOffset()   437 : : \* Purpose: Return the block pointer corresponding to the  438 : : \* block id's.  439 : : \* Parameters: int ibx,   440 : : \* int iby  441 : : \* Return Values: char \*  442 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  443 : : char\* VSBRefReader::refIndexToBlockOffset (int ibx, int iby) const  444 : 2254 : {  445 : 1127 : unsigned int bi ;  446 : 1127 : char \*bsl ;  447 [ # # ][ + + ]: 1127 : if (mOrientation == HORIZONTAL)  448 : : {  449 : : //BLOCKS ARE STORED IN COLUMN-MAJOR FORM  450 : 1111 : bi = ((ibx \* mNumYBlocks) + iby) ;  451 : : }  452 : : else   453 : : {  454 : : //BLOCKS ARE STORED IN ROW-MAJOR FORM  455 : 16 : bi = ((iby \* mNumXBlocks) + ibx) ;  456 : : }  457 : :   458 : :   459 : 1127 : bsl = refSegment\_[VSB\_REF\_BLOCK\_POINTER] + (sizeof(vsbBlockPointer) \* bi);  460 : :   461 [ # # ][ - + ]: 1127 : if(bsl >= refSegment\_[VSB\_REF\_BLOCK\_POINTER] +   462 : : refSegLength\_[VSB\_REF\_BLOCK\_POINTER]  463 : : )  464 : 0 : {  465 : 0 : return NULL ;  466 : : }  467 : 0 : return bsl ;  468 : : }  469 : :   470 : : void VSBRefReader::setNumCellLocations(uint32\_t value)  471 : 2226 : {  472 : 0 : mNumCellLocations = value;  473 : : }  474 : :   475 : 1215 : } //namespace nft |

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