| LCOV - code coverage report |
| --- |
|  |
| | Current view: | [top level](http://docs.google.com/index.html) - [LibVsb/src](http://docs.google.com/index.html) - VSBGenericCellReader.cpp (source / [functions](http://docs.google.com/VSBGenericCellReader.cpp.func.html)) |  |  | Hit | Total | Coverage | | --- | --- | --- | --- | --- | --- | --- | | Test: | EBM9K Converter Coverage Report |  | Lines: | 114 | 159 | 71.7 % | | Date: | 2012-03-23 |  | Functions: | 8 | 9 | 88.9 % | | Legend: | Lines: hit not hit | Branches: + taken - not taken # not executed |  | Branches: | 17 | 44 | 38.6 % | |  | |
|  |

|  |
| --- |
| Branch data Line data Source code  1 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   2 : : \* Copyrighted 2007 by Interra Systems, USA  3 : : \* All rights reserved   4 : : \*  5 : : \* Date of Creation: Tue Jul 31 17:03:16 IST 2007  6 : : \* File Name:   7 : : \* Purpose:   8 : : \* Author:   9 : : \*  10 : : \* Revision History:   11 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   12 : :   13 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  14 : : \* Header Include section   15 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  16 : : #include <stdio.h>  17 : : #include <cstring>  18 : : #include "cell.h"  19 : : #include "link.h"  20 : : #include "Porting.h"  21 : : #include "ErrorHandler.h"  22 : : #include "Error.h"  23 : : #include "Logger.h"  24 : : #include "Memory.h"  25 : : #include "cell.h"  26 : : #include "VSBGenericCellReader.h"  27 : : #include "UnCompressVSB12i.h"  28 : : #include "ConverterParams.h"  29 : : using namespace std;  30 : :   31 : : namespace nft {  32 : :   33 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  34 : : \* Function Name: VSBGenericCellReader()   35 : : \* Purpose: Constructor  36 : : \* Parameters: char\* pCFile,  37 : : VSBCellReader \* pCCR,  38 : : VSBLinkReader\* pLinkReader,  39 : : VSBCellRRIDTracker\* ct ,  40 : : uint32\_t targetAU  41 : : \* Return Values:   42 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  43 : : VSBGenericCellReader::VSBGenericCellReader(char\* pCFile,  44 : : VSBCellReader \* pCCR,   45 : : VSBLinkReader\* pLinkReader,  46 : : VSBCellRRIDTracker\* ct ,  47 : : uint32\_t targetAU  48 : : )  49 : 1094 : :mCellFrameReader(NULL), mCellCommonReader(NULL)  50 : 2188 : {  51 : : //we will have both cell.n and cell.common   52 [ # # ][ + - ]: 1094 : if (pCFile)  53 : : {   54 : 1094 : mCellFrameReader = new VSBCellReader(pCFile);  55 : : }  56 : : else  57 : : {  58 : : //no cell.n  59 : : }  60 : 1094 : mCellCommonReader = pCCR;  61 : : //Now initialize the other fields  62 : 1094 : mLinkReader = pLinkReader;  63 : 1094 : mCellTracker = ct;  64 : 1094 : mTargetAU = targetAU;  65 : 1094 : mVSBScellCnt = mVSBAcellCnt = mVSBMDcellCnt = 0;  66 : : }  67 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  68 : : \* Function Name: cellReInit()   69 : : \* Purpose: ReInitialize the Cell Reader  70 : : \* Parameters: char\* pCFile,  71 : : VSBCellReader \* pCCR,  72 : : VSBLinkReader\* pLinkReader,  73 : : VSBCellRRIDTracker\* ct ,  74 : : uint32\_t targetAU  75 : : \* Return Values: void  76 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  77 : : void VSBGenericCellReader::cellReInit (char\* pCFile,  78 : : VSBCellReader \* pCCR,  79 : : VSBLinkReader\* pLinkReader,  80 : : VSBCellRRIDTracker\* ct ,  81 : : uint32\_t targetAU  82 : : )  83 : 0 : {  84 [ # # ]: 0 : if (pCFile)  85 : : {  86 [ # # ]: 0 : if (mCellFrameReader)  87 : : {  88 : 0 : delete mCellFrameReader;  89 : : }  90 : : else  91 : : {  92 : : }  93 : 0 : mCellFrameReader = new VSBCellReader(pCFile);  94 : : }  95 : : else  96 : : {  97 : : //no cell.n file to map  98 : : }  99 : 0 : mCellCommonReader = pCCR;  100 : : //Now initialize the other fields  101 : 0 : mLinkReader = pLinkReader;  102 : 0 : mCellTracker= ct;   103 : 0 : mTargetAU = targetAU;  104 : 0 : mVSBScellCnt = mVSBAcellCnt = mVSBMDcellCnt = 0;  105 : : }   106 : :   107 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  108 : : \* Function Name: cellGetNext()   109 : : \* Purpose: Returns the next Group Context  110 : : \* Parameters: CellContext \*pcl, Cell \*pc  111 : : \* Return Values: GroupContext\*   112 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  113 : : void VSBGenericCellReader::cellGetNext (CellContext \*pcl, Cell \*pc) throw (DPL::Error \*)  114 : 2372 : {  115 : : //Note, we read cell locations from ref.n file.  116 : : //So, given a context, we have a relation record index  117 : 2372 : vsbSingleCell \*vsc ;  118 : 2372 : vsbArrayCell \*vac ;  119 : 2372 : GroupContext \*pgc = NULL ;  120 : 2372 : char \*PatAddress = NULL ;   121 : 2372 : vsbCellPatternDataHeader \*vcph ;  122 : 2372 : vsbPatternSegmentHeader \*vpsh ;  123 : 2372 : vsbRelationRecord \*vRR ;  124 : 2372 : int SkipSize = 0 ;  125 : 2372 : int32\_t RelationRecordIndex ;  126 : :   127 [ + + ]: 2372 : if(pcl->current < pcl->end)  128 : : {  129 : 1261 : pc->refid = -1 ;  130 : :   131 : 1261 : vsbSingleCell Temp\_VSC = \* ((vsbSingleCell \*)pcl->current);   132 : 1261 : Be2Le\_32( (uint32\_t\*)&Temp\_VSC, 5);  133 [ + + ]: 1261 : if(Temp\_VSC.CellType == SINGLE\_CELL)  134 : : {  135 : 1207 : vsc = &Temp\_VSC;  136 : 1207 : pc->RefType = SINGLE\_VALUE ;  137 : :   138 : 1207 : mVSBScellCnt++;  139 : :   140 : 1207 : pc->sc.xb = vsc->Xb ;  141 : 1207 : pc->sc.yb = vsc->Yb ;  142 : 1207 : pc->sc.dx = vsc->Dx ;  143 : 1207 : pc->sc.dy = vsc->Dy ;  144 : 1207 : RelationRecordIndex = vsc->RelationRecordIndex ;  145 : : //#ifdef ERROR\_SEED //<<Test\_Id:VSB\_Single\_Cell\_RelationRecordIndex\_Set\_To\_-1>>  146 : : // RelationRecordIndex = -1;  147 : : //#endif  148 [ - + ]: 1207 : if (RelationRecordIndex<0)  149 : : {  150 : 0 : Logger::instance()->log(LogMessageTable::eUTIL\_ERROR\_GENERAL\_MESSAGE,  151 : : "Negative Relation Record found for VSB-12 cell");  152 : :   153 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  154 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  155 : : ,ErrorHandler::eErrMessageString  156 : : ,pErr->getErrorMsg((ErrorHandler::eErrMessageString)  157 : : ,"Negative Relation Record found for VSB-12 cell"));  158 : : }   159 : 1207 : if(mCellTracker->getCountEncodeVc(vsc->RelationRecordIndex) > 1) //may be optimized away  160 : : {  161 : 220 : pc->RefType = SINGLE\_NEAR ;  162 : 220 : pc->refid = RelationRecordIndex ;  163 : : }  164 : :   165 : 1207 : SkipSize = sizeof(vsbSingleCell) ;  166 : : }  167 : : else  168 : : {  169 : 54 : vsbArrayCell TempVAC = \*((vsbArrayCell\*)pcl->current);  170 : 54 : Be2Le\_32((unsigned int\*)&TempVAC, 8);  171 : 54 : vac = &TempVAC;  172 : 54 : pc->RefType = ARRAY\_VALUE ;  173 : :   174 : 54 : mVSBAcellCnt++;  175 : :   176 : 54 : pc->ac.xb = vac->Xb ;  177 : 54 : pc->ac.yb = vac->Yb ;  178 : 54 : pc->ac.xe = vac->Xe ;  179 : 54 : pc->ac.ye = vac->Ye ;  180 : 54 : pc->ac.dx = vac->Dx ;  181 : 54 : pc->ac.dy = vac->Dy ;  182 : 54 : pc->ac.xr = vac->Nrepx ;  183 : 54 : pc->ac.yr = vac->Nrepy ;  184 : 54 : RelationRecordIndex = vac->RelationRecordIndex ;  185 : : //#ifdef ERROR\_SEED //<<Test\_Id:VSB\_Array\_Cell\_RelationRecordIndex\_Set\_To\_-1>>  186 : : // RelationRecordIndex = -1;  187 : : //#endif  188 [ - + ]: 54 : if (RelationRecordIndex < 0)  189 : : {  190 : 0 : Logger::instance()->log(LogMessageTable::eUTIL\_ERROR\_GENERAL\_MESSAGE,  191 : : "Negative Relation Record found for VSB-12 cell");  192 : :   193 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  194 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  195 : : ,ErrorHandler::eErrMessageString  196 : : ,pErr->getErrorMsg((ErrorHandler::eErrMessageString)  197 : : ,"Negative Relation Record found for VSB-12 cell"));  198 : : }  199 : :   200 : 54 : if(mCellTracker->getCountEncodeVc(vac->RelationRecordIndex) > 1)  201 : : {  202 : 1 : pc->RefType = ARRAY\_NEAR ;  203 : 1 : pc->refid = RelationRecordIndex ;  204 : : }  205 : 54 : SkipSize = sizeof(vsbArrayCell) ;  206 : : }  207 : 1261 : pcl->current += SkipSize ;  208 : 1261 : try  209 : : {  210 : : //#ifdef ERROR\_SEED //<<Test\_Id:link\_RelationRecordIndex\_Set\_To\_65536>>  211 : : // RelationRecordIndex = 65536;  212 : : //#endif  213 : 1261 : mLinkReader->linkGetRelationRecord(  214 : : RelationRecordIndex,   215 : : &vRR ) ;  216 : 1261 : PatAddress = cellGetPatternAddress(vRR) ;  217 : : }  218 [ # # ]: 0 : catch (DPL::Error \*err)  219 : : {  220 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  221 : 0 : pErr->errorAdd(err, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  222 : : ,ErrorHandler::eErrMessageString  223 : : ,pErr->getErrorMsg((ErrorHandler::eErrMessageString)  224 : : ,"VSBGenericCellReader::cellGetNext function failed"));  225 : :   226 : : }  227 : : //Compression Support  228 : 1261 : cellGetPatternSegmentHeader(vRR, &vpsh) ;  229 [ + + ]: 1261 : if(1 == vpsh->compressionType )  230 : : {  231 : 1 : vsbCompCellPatternDataHeader vsbCompCellPatternData;  232 : : //Read the Cell Pattern Data ­ Compressed Type header  233 : 1 : memcpy(&vsbCompCellPatternData, PatAddress ,sizeof(vsbCompCellPatternDataHeader));  234 : 1 : int var = sizeof(vsbCompCellPatternDataHeader);  235 : 1 : NTOHL(vsbCompCellPatternData.compLength);  236 : 1 : NTOHL(vsbCompCellPatternData.unCompLength);  237 : :   238 : : //Call the constructor  239 : 1 : UnCompressVSB12i unCompPatternData(vsbCompCellPatternData.compLength);  240 : :   241 : : //Uncompress the pattern data using compression library  242 : 1 : unCompPatternData.unCompress(PatAddress);  243 : :   244 : : //Point to the pattern data  245 : 0 : PatAddress = unCompPatternData.getUncompPatternData();  246 : : }  247 : : else{  248 : : //Non compressed data  249 : : }  250 : :   251 : 1261 : vcph = (vsbCellPatternDataHeader \*)PatAddress ;  252 : 1261 : vsbCellPatternDataHeader Temp\_vcph = \*vcph;//Local Variable used  253 : 1261 : Be2Le\_32((unsigned int\*)&Temp\_vcph, 3);  254 : 1261 : vcph = &Temp\_vcph;  255 : : /\* Update Group Context \*/  256 : 1261 : pgc = new GroupContext();  257 : 1261 : \*pgc = pcl->pgc;  258 : 1261 : pgc->start = PatAddress ;  259 : 1261 : pgc->current = pgc->start + sizeof(vsbCellPatternDataHeader) ;  260 : 1261 : pgc->end = pgc->start + (vcph->CellPatternDataLength \* WordSize) ;  261 : 1261 : pgc->CellPatDx = vcph->Dx ;  262 : 1261 : pgc->CellPatDy = vcph->Dy ;  263 : :   264 [ - + ]: 1261 : if(vcph->flag2)  265 : 0 : mVSBMDcellCnt++; //Multidefined cells are NOT allowed in PDG-12  266 : : //Estimator.  267 : 1261 : pgc->mirror = mLinkReader->linkGetMirrorState(vRR) ;  268 : 1261 : pgc->scale = mLinkReader->linkGetScaleFactor(vRR) ;  269 : 1261 : mMirror = pgc->mirror;  270 : 1261 : mScale = pgc->scale;  271 : : //#ifdef ERROR\_SEED //<<Test\_Id:VSBGenericCellReader\_pgc\_scale\_Set\_To\_0>>  272 : : // pgc->scale = 0;  273 : : //#endif  274 [ - + ]: 1261 : if(pgc->scale <= 0)  275 : : {  276 : 0 : Logger::instance()->log(LogMessageTable::eUTIL\_ERROR\_GENERAL\_MESSAGE,  277 : : "Invalid pattern Scale Value");  278 : :   279 : 0 : char buf[BUFSIZ];  280 : 0 : sprintf(buf, "Invalid pattern Scale Value %f\n", pgc->scale);  281 : :   282 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  283 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  284 : : ,ErrorHandler::eErrMessageString  285 : : ,pErr->getErrorMsg((ErrorHandler::eErrMessageString)  286 : : ,buf));  287 : : }  288 : 1261 : pgc->isAligned = vpsh->isAligned;  289 : 1261 : pgc->SourceAU = (int)getAUFactor(vpsh) ;  290 : 1261 : ConverterParams::instance()->setVSBAU(pgc->SourceAU);  291 : 1261 : pc->setSegmentDataLength( vpsh->SegmentDataLength );  292 : : //#ifdef ERROR\_SEED //<<Test\_Id:VSBGenericCellReader\_pgc\_SourceAU\_Set\_To\_0>>  293 : : // pgc->SourceAU = 0;  294 : : //#endif  295 [ - + ]: 1261 : if(pgc->SourceAU <= 0)  296 : : {  297 : 0 : Logger::instance()->log(LogMessageTable::eUTIL\_ERROR\_GENERAL\_MESSAGE,  298 : : "Invalid Segment AU");  299 : :   300 : 0 : char buf[BUFSIZ];  301 : 0 : sprintf(buf, "Invalid Segment AU %d\n", pgc->SourceAU);  302 : :   303 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  304 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  305 : : ,ErrorHandler::eErrMessageString  306 : : ,pErr->getErrorMsg((ErrorHandler::eErrMessageString)  307 : : ,buf));  308 : : }  309 : :   310 [ + - ]: 1261 : pgc->TranslateFactor = ((double)pgc->SourceAU/(double)mTargetAU) \* pgc->scale ;  311 : 1261 : pgc->X0Y0WordLength = getX0Y0WordLength(vpsh) ;  312 : : //#ifdef ERROR\_SEED //<<Test\_Id:VSBGenericCellReader\_pgc\_X0Y0WordLength\_Set\_To\_0>>  313 : : // pgc->X0Y0WordLength = 0;  314 : : //#endif  315 [ - + ]: 1261 : if(pgc->X0Y0WordLength <= 0)  316 : : {  317 : 0 : Logger::instance()->log(LogMessageTable::eUTIL\_ERROR\_GENERAL\_MESSAGE,  318 : : "Invalid word length");  319 : :   320 : 0 : char buf[BUFSIZ];  321 : 0 : sprintf(buf, "Invalid word length %d\n", pgc->X0Y0WordLength);  322 : :   323 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  324 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  325 : : ,ErrorHandler::eErrMessageString  326 : : ,pErr->getErrorMsg((ErrorHandler::eErrMessageString)  327 : : ,buf));  328 : : }  329 : 1261 : pgc->ppc.WhichContext = NoContext ;  330 : 1261 : pgc->ppc.X0Offset = 0 ;  331 : 1261 : pgc->ppc.Y0Offset = 0 ;  332 : :   333 : : }  334 [ # # ]: 2372 : pc->pgc = pgc;  335 : : }  336 : :   337 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  338 : : \* Function Name: cellGetPatternAddress()   339 : : \* Purpose: Returns the Pattern Address of the cell   340 : : \* Parameters: const vsbRelationRecord \*rr  341 : : \* Return Values: char\*   342 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  343 : : char\* VSBGenericCellReader::cellGetPatternAddress(const vsbRelationRecord \*rr) const throw (DPL::Error \*)  344 : 1261 : {  345 : : //int64\_t segOffset = 0;  346 : : //Support for cell.common: In what follows, we assumed  347 : : //we will reading a certain cell.k file which is already mapped  348 : : //to memory.  349 : : //To support cell.common, we must check if cell.common is present  350 : 1261 : int16\_t fileLocatorIndex;  351 : :   352 : 1261 : fileLocatorIndex = BE2LE\_16((int16\_t)rr->FileLocatorIndex);   353 : : //We assume that file locator type == 0. So we switch directly  354 : : //on the basis of file loactor index  355 : : //#ifdef ERROR\_SEED  356 : : // fileLocatorIndex = 0; //<<Test\_Id:VSBGenericCellReader\_fileLocatorIndex\_Set\_To\_0>>  357 : : // fileLocatorIndex = 1; //<<Test\_Id:VSBGenericCellReader\_fileLocatorIndex\_Set\_To\_1>>  358 : : //#endif  359 [ - + ]: 1261 : if (fileLocatorIndex==0)  360 : : {  361 : : //#ifndef ERROR\_SEED //<<Test\_Id:VSBGenericCellReader\_fileLocatorIndex\_Set\_To\_0>>  362 [ # # ]: 0 : if (mCellCommonReader)  363 : : {  364 : 0 : return mCellCommonReader->cellGetPatternAddress(rr);  365 : : }  366 : : else  367 : : //#endif  368 : : {  369 : 0 : Logger::instance()->log(LogMessageTable::eUTIL\_ERROR\_GENERAL\_MESSAGE,  370 : : "cell.common not found while processing pattern");  371 : :   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 : : ,"cell.common not found while processing pattern"));  377 : : }  378 : :   379 : : }  380 : : else  381 : : {  382 : : //#ifndef ERROR\_SEED //<<Test\_Id:VSBGenericCellReader\_fileLocatorIndex\_Set\_To\_1>>  383 [ + - ]: 1261 : if (mCellFrameReader)  384 : : {   385 : 1261 : return mCellFrameReader->cellGetPatternAddress(rr);  386 : : }  387 : : else  388 : : //#endif  389 : : {  390 : 0 : Logger::instance()->log(LogMessageTable::eUTIL\_ERROR\_GENERAL\_MESSAGE,  391 : : "Frame specific cell file not found while processing pattern");  392 : :   393 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  394 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  395 : : ,ErrorHandler::eErrMessageString  396 : : ,pErr->getErrorMsg((ErrorHandler::eErrMessageString)  397 : : ,"Frame specific cell file not found while processing pattern"));  398 : : }  399 : : }  400 [ # # ]: 1261 : return NULL;  401 : :   402 : : }  403 : :   404 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  405 : : \* Function Name: cellGetPatternSegmentHeader()   406 : : \* Purpose: Sets the PatternSegmentHeader  407 : : \* Parameters: const vsbRelationRecord \*rr,  408 : : \* vsbPatternSegmentHeader \*\*SegmentHeader  409 : : \* Return Values: void  410 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  411 : : void VSBGenericCellReader::cellGetPatternSegmentHeader(  412 : : const vsbRelationRecord \*rr,  413 : : vsbPatternSegmentHeader \*\*SegmentHeader) const  414 : 1261 : {  415 : : //We assume that file locator type == 0. So we switch directly  416 : : //on the basis of file loactor index  417 : 1261 : int16\_t fileLocatorIndex;  418 : 1261 : fileLocatorIndex = BE2LE\_16((int16\_t)rr->FileLocatorIndex);  419 : : //Access the cell file specific function  420 [ # # ][ - + ]: 1261 : if (fileLocatorIndex==0)  421 : : {  422 : 0 : mCellCommonReader->cellGetPatternSegmentHeader(rr, SegmentHeader);  423 : : }  424 : : else  425 : : {  426 : 1261 : mCellFrameReader->cellGetPatternSegmentHeader(rr, SegmentHeader);  427 : : }  428 : : }  429 : :   430 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  431 : : \* Function Name: ~VSBGenericCellReader()  432 : : \* Purpose: Destructor  433 : : \* Parameters:  434 : : \* Return Values:   435 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  436 : :   437 : : VSBGenericCellReader::~VSBGenericCellReader()  438 : 2188 : {  439 : 1094 : delete mCellFrameReader;  440 : : }  441 : :   442 : 1215 : } //namespace nft |

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