| LCOV - code coverage report |
| --- |
|  |
| | Current view: | [top level](http://docs.google.com/index.html) - [LibVsb/include](http://docs.google.com/index.html) - vsb\_translate.h (source / [functions](http://docs.google.com/vsb_translate.h.func.html)) |  |  | Hit | Total | Coverage | | --- | --- | --- | --- | --- | --- | --- | | Test: | EBM9K Converter Coverage Report |  | Lines: | 39 | 127 | 30.7 % | | Date: | 2012-03-23 |  | Functions: | 1 | 2 | 50.0 % | | Legend: | Lines: hit not hit | Branches: + taken - not taken # not executed |  | Branches: | 8 | 60 | 13.3 % | |  | |
|  |

|  |
| --- |
| Branch data Line data Source code  1 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   2 : : \* Copyrighted 2007 by Interra Systems, USA  3 : : \* All rights reserved   4 : : \*  5 : : \* Date of Creation: Tue Aug 28 16:59:43 IST 2007  6 : : \* File Name: vsb\_translate.h  7 : : \* Purpose:   8 : : \* Author:   9 : : \*  10 : : \* Revision History:   11 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   12 : :   13 : : #ifndef \_VSB\_TRANSLATE\_H\_  14 : : #define \_VSB\_TRANSLATE\_H\_  15 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  16 : : \* Header Include section   17 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  18 : :   19 : : #include <math.h>  20 : : #include <float.h>  21 : : #include "context.h"  22 : : #include "vsb\_types.h"  23 : : #include "dec\_math.h"  24 : : #include "dec\_types.h"  25 : : #include "SDDB.h"  26 : : #include "print.h"  27 : : #include "HealthCheckCounter.h"  28 : : namespace nft {  29 : : #define mirror(offset,width,bound) ( (bound) - ((offset) + (width)) )  30 : :   31 : : static const int MirrorCode[73]=  32 : : {  33 : : 0x00, 0x02, 0x01, 0x04, 0x03, 0x06, 0x05, 0x07, 0x08,   34 : : 0x0a, 0x09, 0x0c, 0x0b, 0x0e, 0x0d, 0x10, 0x0f, 0x11,  35 : : 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  36 : : 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  37 : : 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  38 : : 0x00, 0x00, 0x00, 0x00, 0x31, 0x33, 0x32, 0x35, 0x34,  39 : : 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  40 : : 0x00, 0x00, 0x43, 0x42, 0x41, 0x40, 0x44, 0x45, 0x46,  41 : : 0x47  42 : : } ;  43 : :   44 : : /\*  45 : : #ifdef \_\_cplusplus  46 : : extern "C" {  47 : : #endif   48 : :   49 : :   50 : : #ifdef \_\_cplusplus  51 : : }  52 : : #endif   53 : : \*/  54 : :   55 : : inline double EPWidth(vsbEP \*vep)  56 : 0 : {  57 [ # # ]: 0 : switch(vep->PatCode)  [ # # # # ]  58 : : {  59 : : case 0x01 :  60 : : case 0x02 :  61 : : case 0x03 :  62 : : case 0x04 :  63 : : case 0x05 :  64 : : case 0x06 :  65 : : case 0x0d :  66 : : case 0x0e :  67 : : case 0x11 :  68 : : case 0x31 :  69 : : case 0x32 :  70 : : case 0x33 :  71 : : case 0x34 :  72 : : case 0x35 :  73 : : case 0x40 :  74 : : case 0x41 :  75 : : case 0x42 :  76 : 0 : case 0x43 : return vep->L1 ;  77 : :   78 : : case 0x07 :  79 : 0 : case 0x08 : return (vep->L1 + vep->L2 + vep->L2) ;  80 : :   81 : : case 0x09 :  82 : : case 0x0a :  83 : : case 0x0b :  84 : : case 0x0c :  85 : : case 0x0f :  86 : 0 : case 0x10 : return (vep->L1 + vep->L2) ;  87 : 0 : default : return vep->L1 ;  88 : : }   89 : : }  90 : :   91 : : inline int EPWidth(EP \*ep)  92 : 68 : {  93 [ + - ]: 34 : switch(ep->PatCode)  [ + - - - ]  [ + - ]  [ + - - - ]  94 : : {  95 : : case 0x01 :  96 : : case 0x02 :  97 : : case 0x03 :  98 : : case 0x04 :  99 : : case 0x05 :  100 : : case 0x06 :  101 : : case 0x0d :  102 : : case 0x0e :  103 : : case 0x11 :  104 : : case 0x31 :  105 : : case 0x32 :  106 : : case 0x33 :  107 : : case 0x34 :  108 : : case 0x35 :  109 : : case 0x40 :  110 : : case 0x41 :  111 : : case 0x42 :  112 : 34 : case 0x43 : return ep->L1 ;  113 : :   114 : : case 0x07 :  115 : 0 : case 0x08 : return (ep->L1 + ep->L2 + ep->L2) ;  116 : :   117 : : case 0x09 :  118 : : case 0x0a :  119 : : case 0x0b :  120 : : case 0x0c :  121 : : case 0x0f :  122 : 0 : case 0x10 : return (ep->L1 + ep->L2) ;  123 : 0 : default : return ep->L1 ;  124 : : }   125 : : }  126 : :   127 : : inline int BBScale4vsbEP(GroupContext \*pgc, vsbEP \*ep)  128 : : {  129 : : double Bx, By;  130 : : double SBx, SBy;  131 : :   132 : : switch(ep->PatCode)  133 : : {  134 : : case 0x01 :  135 : : case 0x02 :  136 : : case 0x03 :  137 : : case 0x04 :  138 : : case 0x0d :  139 : : case 0x0e :  140 : : Bx = ep->L1 ;  141 : : By = ep->L2 + ep->L1 ;  142 : : SBx = Bx \* pgc->TranslateFactor ;  143 : : SBy = By \* pgc->TranslateFactor ;  144 : : ep->L1 = SBx;  145 : : ep->L2 = SBy - SBx;   146 : : break;  147 : :   148 : : case 0x05 :  149 : : case 0x06 :  150 : : Bx = ep->L1 ;  151 : : By = ep->L2 + 2 \* ep->L1 ;  152 : : SBx = Bx \* pgc->TranslateFactor ;  153 : : SBy = By \* pgc->TranslateFactor ;  154 : : ep->L1 = SBx ;  155 : : ep->L2 = SBy - 2 \* SBx ;  156 : : break;  157 : :   158 : : case 0x07 :  159 : : case 0x08 :  160 : : Bx = ep->L1 + 2 \* ep->L2 ;  161 : : By = ep->L2 ;  162 : : SBx = Bx \* pgc->TranslateFactor ;  163 : : SBy = By \* pgc->TranslateFactor ;  164 : : ep->L1 = SBx - 2 \* SBy ;  165 : : ep->L2 = SBy ;  166 : : break;  167 : :   168 : : case 0x09 :  169 : : case 0x0a :  170 : : case 0x0b :  171 : : case 0x0c :  172 : : case 0x0f :  173 : : case 0x10 :  174 : : Bx = ep->L1 + ep->L2 ;  175 : : By = ep->L2 ;  176 : : SBx = Bx \* pgc->TranslateFactor ;  177 : : SBy = By \* pgc->TranslateFactor ;   178 : : ep->L1 = SBx - SBy ;  179 : : ep->L2 = SBy;  180 : : break;  181 : :   182 : : default :  183 : : ep->L1 = ep->L1 \* pgc->TranslateFactor ;  184 : : ep->L2 = ep->L2 \* pgc->TranslateFactor ;  185 : : break;  186 : : }  187 : : return 0;  188 : : }  189 : :   190 : :   191 : : inline int TranslateEP(GroupContext \*pgc, vsbEP \*vep, double \*vXe, double \*vYe)  192 : 0 : {  193 : :   194 : 0 : HealthCheckCounter::instance()->incrCounter(  195 : : HealthCheckCounter::eUpdateFrequent);  196 : 0 : SDDB \*pSDDB = SDDB::instance();  197 [ # # ]: 0 : if(pgc->mirror)  198 : : {  199 : 0 : double X0 = vep->X0 ;  200 : 0 : double vepWidth = EPWidth(vep) ;  201 : :   202 : 0 : try   203 : : {  204 : 0 : pSDDB->EPReadyForMirror(vep->PatCode, RoundINT(vep->L1),RoundINT(vep->L2),   205 : : RoundINT(vep->X0), RoundINT(vep->Y0), pgc->CellPatDx);   206 : : }  207 [ # # ]: 0 : catch( DPL::Error \*pChild )  208 : : {  209 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  210 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  211 : : ErrorHandler::eErrFunction,  212 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  213 : : "SDDB::EPReadyForMirror"));  214 : : }  215 : :   216 : :   217 [ # # ]: 0 : vep->X0 = mirror(\*vXe, vepWidth, pgc->CellPatDx) ;  218 [ # # ]: 0 : \*vXe = mirror(X0, vepWidth, pgc->CellPatDx) ;  219 : 0 : vep->PatCode = MirrorCode[vep->PatCode] ;  220 : :   221 : 0 : try  222 : : {  223 : 0 : pSDDB->EPMirrorCompleted(vep->PatCode, RoundINT(vep->L1),  224 : : RoundINT(vep->L2), RoundINT(\*vXe), RoundINT(\*vYe), pgc->CellPatDx);  225 : : }  226 [ # # ]: 0 : catch( DPL::Error \*pChild )  227 : : {  228 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  229 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  230 : : ErrorHandler::eErrFunction,  231 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  232 : : "SDDB::EPMirrorCompleted"));  233 : : }  234 : :   235 : :   236 : : }   237 : :   238 : 0 : try  239 : : {  240 : 0 : pSDDB->EPReadyForTransform(RoundINT(vep->X0), RoundINT(vep->Y0),  241 : : RoundINT(vep->L1), RoundINT(vep->L2), vep->PatCode);  242 : : }  243 [ # # ]: 0 : catch( DPL::Error \*pChild )  244 : : {  245 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  246 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  247 : : ErrorHandler::eErrFunction,  248 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  249 : : "SDDB::EPReadyForTransform"));  250 : : }  251 : :   252 : 0 : vep->X0 = vep->X0 \* pgc->TranslateFactor ;  253 : 0 : vep->Y0 = vep->Y0 \* pgc->TranslateFactor ;  254 : : /\* This change is only for consistency. If any performance issues use else part \*/   255 : : #ifdef BB\_SCALE  256 : : BBScale4vsbEP(pgc, vep);  257 : : #else  258 : 0 : vep->L1 = vep->L1 \* pgc->TranslateFactor ;  259 : 0 : vep->L2 = vep->L2 \* pgc->TranslateFactor ;  260 : : #endif  261 : 0 : \*vXe = \*vXe \* pgc->TranslateFactor ;  262 : 0 : \*vYe = \*vYe \* pgc->TranslateFactor ;  263 : 0 : try  264 : : {  265 : 0 : pSDDB->EPScalingCompleted(vep->PatCode);  266 : : }  267 [ # # ]: 0 : catch( DPL::Error \*pChild )  268 : : {  269 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  270 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  271 : : ErrorHandler::eErrFunction,  272 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  273 : : "SDDB::EPScalingCompleted"));  274 : : }  275 : 0 : try  276 : : {  277 : 0 : pSDDB->EPTransformCompleted(RoundINT(vep->X0), RoundINT(vep->Y0),   278 : : RoundINT(vep->L1), RoundINT(vep->L2));  279 : : }  280 [ # # ]: 0 : catch( DPL::Error \*pChild )  281 : : {  282 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  283 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  284 : : ErrorHandler::eErrFunction,  285 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  286 : : "SDDB::EPTransformCompleted"));  287 : : }  288 : :   289 : 0 : return 0 ;  290 : : }  291 : :   292 : : inline int TranslatePP(GroupContext \*pgc, vsbPP \*vpp)  293 : 0 : {  294 [ # # ]: 0 : if(pgc->mirror)  295 : : {  296 : 0 : int epWidth = EPWidth(&pgc->ppc.ep) ;  297 : 0 : SDDB \*pSDDB = SDDB::instance();  298 : 0 : try  299 : : {  300 : 0 : pSDDB->PPReadyForMirror(RoundINT(vpp->X0), pgc->CellPatDx);  301 : : }  302 [ # # ]: 0 : catch( DPL::Error \*pChild )  303 : : {  304 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  305 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  306 : : ErrorHandler::eErrFunction,  307 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  308 : : "SDDB::PPReadyForMirror"));  309 : : }  310 : :   311 [ # # ]: 0 : vpp->X0 = mirror(vpp->X0, epWidth, pgc->CellPatDx) ;  312 : 0 : try  313 : : {   314 : 0 : pSDDB->PPMirrorCompleted(RoundINT(vpp->X0) + epWidth, pgc->CellPatDx);  315 : : }  316 [ # # ]: 0 : catch( DPL::Error \*pChild )  317 : : {  318 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  319 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  320 : : ErrorHandler::eErrFunction,  321 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  322 : : "SDDB::PPMirrorCompleted"));  323 : : }  324 : :   325 : : }  326 : :   327 : :   328 : 0 : vpp->X0 = vpp->X0 \* pgc->TranslateFactor ;  329 : 0 : vpp->Y0 = vpp->Y0 \* pgc->TranslateFactor ;  330 : :   331 : : return 0 ;  332 : : }  333 : :   334 : : inline int BBScale4EP(GroupContext \*pgc, EP \*ep)  335 : : {  336 : : int Bx, By;  337 : : double SBx, SBy;  338 : :   339 : : switch(ep->PatCode)  340 : : {  341 : : case 0x01 :  342 : : case 0x02 :  343 : : case 0x03 :  344 : : case 0x04 :  345 : : case 0x0d :  346 : : case 0x0e :  347 : : Bx = ep->L1 ;  348 : : By = ep->L2 + ep->L1 ;  349 : : SBx = Bx \* pgc->TranslateFactor ;  350 : : SBy = By \* pgc->TranslateFactor ;  351 : : ep->L1 = RoundINT(SBx);  352 : : ep->L2 = RoundINT(SBy) - ep->L1;   353 : : break;  354 : :   355 : : case 0x05 :  356 : : case 0x06 :  357 : : Bx = ep->L1 ;  358 : : By = ep->L2 + 2 \* ep->L1 ;  359 : : SBx = Bx \* pgc->TranslateFactor ;  360 : : SBy = By \* pgc->TranslateFactor ;  361 : : ep->L1 = RoundINT(SBx);  362 : : if(0 != ep->L2)  363 : : ep->L2 = RoundINT(SBy) - 2 \* ep->L1;  364 : : break;  365 : :   366 : : case 0x07 :  367 : : case 0x08 :  368 : : Bx = ep->L1 + 2 \* ep->L2 ;  369 : : By = ep->L2 ;  370 : : SBx = Bx \* pgc->TranslateFactor ;  371 : : SBy = By \* pgc->TranslateFactor ;  372 : : ep->L2 = RoundINT(SBy);  373 : : if(0 != ep->L1)  374 : : ep->L1 = RoundINT(SBx) - 2 \* ep->L2 ;  375 : : break;  376 : :   377 : : case 0x09 :  378 : : case 0x0a :  379 : : case 0x0b :  380 : : case 0x0c :  381 : : case 0x0f :  382 : : case 0x10 :  383 : : Bx = ep->L1 + ep->L2 ;  384 : : By = ep->L2 ;  385 : : SBx = Bx \* pgc->TranslateFactor ;  386 : : SBy = By \* pgc->TranslateFactor ;   387 : : ep->L2 = RoundINT(SBy);  388 : : ep->L1 = RoundINT(SBx) - ep->L2 ;  389 : : break;  390 : :   391 : : default :  392 : : ep->L1 = RoundINT(ep->L1 \* pgc->TranslateFactor );  393 : : ep->L2 = RoundINT(ep->L2 \* pgc->TranslateFactor );  394 : : break;  395 : : }  396 : : return 0;  397 : : }  398 : :   399 : :   400 : : inline int TranslateEP(GroupContext \*pgc, EP \*ep, int \*Xe, int \*Ye)  401 : 54 : {  402 : 54 : SDDB \*pSDDB = SDDB::instance();  403 : :   404 [ + + ]: 54 : if(pgc->mirror)  405 : : {  406 : 21 : int X0 = ep->X0 ;  407 : 21 : int epWidth = EPWidth(ep) ;  408 : :   409 : 21 : try  410 : : {  411 : 21 : pSDDB->EPReadyForMirror(ep->PatCode, ep->L1, ep->L2, ep->X0,   412 : : ep->Y0, pgc->CellPatDx);  413 : : }  414 [ # # ]: 0 : catch( DPL::Error \*pChild )  415 : : {  416 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  417 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  418 : : ErrorHandler::eErrFunction,  419 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  420 : : "SDDB::EPReadyForMirror"));  421 : : }  422 : :   423 : :   424 : 21 : ep->X0 = mirror(\*Xe, epWidth, pgc->CellPatDx) ;  425 : 21 : \*Xe = mirror(X0, epWidth, pgc->CellPatDx) ;  426 : 21 : ep->PatCode = MirrorCode[ep->PatCode] ;  427 : :   428 : 21 : try  429 : : {  430 : 21 : pSDDB->EPMirrorCompleted(ep->PatCode, ep->L1, ep->L2, \*Xe, \*Ye, pgc->CellPatDx);  431 : : }  432 [ # # ]: 0 : catch( DPL::Error \*pChild )  433 : : {  434 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  435 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  436 : : ErrorHandler::eErrFunction,  437 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  438 : : "SDDB::EPMirrorCompleted"));  439 : : }  440 : :   441 : : }  442 : :   443 : 54 : try  444 : : {  445 : 54 : pSDDB->EPReadyForTransform(ep->X0, ep->Y0, ep->L1, ep->L2, ep->PatCode);  446 : : }  447 [ # # ]: 0 : catch( DPL::Error \*pChild )  448 : : {  449 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  450 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  451 : : ErrorHandler::eErrFunction,  452 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  453 : : "SDDB::EPReadyForTransform"));  454 : : }  455 : :   456 : 54 : ep->X0 = RoundINT(ep->X0 \* pgc->TranslateFactor) ;  457 : 54 : ep->Y0 = RoundINT(ep->Y0 \* pgc->TranslateFactor) ;  458 : : #ifdef BB\_SCALE  459 : : BBScale4EP(pgc, ep);  460 : : #else  461 : 54 : ep->L1 = RoundINT(ep->L1 \* pgc->TranslateFactor) ;  462 : 54 : ep->L2 = RoundINT(ep->L2 \* pgc->TranslateFactor) ;  463 : : #endif  464 : 54 : \*Xe = RoundINT(\*Xe \* pgc->TranslateFactor) ;  465 : 54 : \*Ye = RoundINT(\*Ye \* pgc->TranslateFactor) ;  466 : 54 : try  467 : : {  468 : 54 : pSDDB->EPScalingCompleted(ep->PatCode);  469 : : }  470 [ # # ]: 0 : catch( DPL::Error \*pChild )  471 : : {  472 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  473 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  474 : : ErrorHandler::eErrFunction,  475 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  476 : : "SDDB::EPScalingCompleted"));  477 : : }  478 : 54 : try  479 : : {   480 : 54 : pSDDB->EPTransformCompleted(ep->X0, ep->Y0, ep->L1, ep->L2);  481 : : }  482 [ # # ]: 0 : catch( DPL::Error \*pChild )  483 : : {  484 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  485 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  486 : : ErrorHandler::eErrFunction,  487 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  488 : : "SDDB::EPTransformCompleted"));  489 : : }  490 : :   491 : 54 : return 0 ;  492 : : }  493 : :   494 : : inline int TranslatePP(GroupContext \*pgc, PP \*pp)  495 : 14 : {  496 [ + + ]: 12 : if(pgc->mirror)  497 : : {  498 : 13 : int epWidth = EPWidth(&pgc->ppc.ep) ;  499 : :   500 : 13 : SDDB \*pSDDB = SDDB::instance();  501 : 13 : try  502 : : {   503 : 13 : pSDDB->PPReadyForMirror(pp->X0, pgc->CellPatDx);  504 : : }  505 [ # # ]: 0 : catch( DPL::Error \*pChild )  506 : : {  507 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  508 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  509 : : ErrorHandler::eErrFunction,  510 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  511 : : "SDDB::PPReadyForMirror"));  512 : : }  513 : :   514 : 13 : pp->X0 = mirror(pp->X0, epWidth, pgc->CellPatDx) ;  515 : :   516 : 13 : try  517 : : {  518 : 13 : pSDDB->PPMirrorCompleted(pp->X0 + epWidth, pgc->CellPatDx);  519 : : }  520 [ # # ]: 0 : catch( DPL::Error \*pChild )  521 : : {  522 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  523 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  524 : : ErrorHandler::eErrFunction,  525 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  526 : : "SDDB::PPMirrorCompleted"));  527 : : }  528 : :   529 : : }   530 : :   531 : 14 : pp->X0 = RoundINT(pp->X0 \* pgc->TranslateFactor) ;  532 : 28 : pp->Y0 = RoundINT(pp->Y0 \* pgc->TranslateFactor) ;  533 : :   534 : : return 0 ;  535 : : }  536 : :   537 : : } //namespace nft  538 : : #endif /\* \_VSB\_TRANSLATE\_H\_ \*/ |

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