| LCOV - code coverage report |
| --- |
|  |
| | Current view: | [top level](http://docs.google.com/index.html) - [LibVsb/src](http://docs.google.com/index.html) - vsb\_patternAR.cpp (source / [functions](http://docs.google.com/vsb_patternAR.cpp.func.html)) |  |  | Hit | Total | Coverage | | --- | --- | --- | --- | --- | --- | --- | | Test: | EBM9K Converter Coverage Report |  | Lines: | 114 | 142 | 80.3 % | | Date: | 2012-03-23 |  | Functions: | 3 | 3 | 100.0 % | | Legend: | Lines: hit not hit | Branches: + taken - not taken # not executed |  | Branches: | 33 | 54 | 61.1 % | |  | |
|  |

|  |
| --- |
| Branch data Line data Source code  1 : : /\*--------------------------------------------------------------------------  2 : : \* Copyright (c) 2003, NuFlare Technology Inc., Digital GlobalSoft Ltd.  3 : : \* All Rights Reserved.  4 : : \* These technical data are subject to export control laws of Japan  5 : : \* regulations, and diversion contrary thereto is prohibited.  6 : : \*   7 : : \* Module : VSB Array Representation Extraction Routines  8 : : \* Design Document : VSB Library - High Level Design (Rev 0.76)  9 : : \* Author : santhosh.kudva@digital.com  10 : : \* Date : 16th April, 2003  11 : : \*   12 : : \* File names prefixed with vsb\_ contain routines used privately by the library   13 : : \*   14 : : \* RCS Keywords  15 : : \* $Id: vsb\_patternAR.cpp,v 1.7 2012/03/02 05:19:43 asarkar Exp $  16 : : \* $Log: vsb\_patternAR.cpp,v $  17 : : \* Revision 1.7 2012/03/02 05:19:43 asarkar  18 : : \* Exception cases  19 : : \*  20 : : \* Revision 1.6 2012/02/10 05:07:19 santanu  21 : : \* Fix for E9K-80  22 : : \*  23 : : \* Revision 1.5 2011/08/18 11:07:00 abua  24 : : \* initial commit for SelfDiagnostics  25 : : \*  26 : : \* Revision 1.4 2011/07/05 11:20:39 asarkar  27 : : \* \*\*\* empty log message \*\*\*  28 : : \*  29 : : \* Revision 1.3 2011/05/23 10:47:40 asarkar  30 : : \* \*\*\* empty log message \*\*\*  31 : : \*  32 : : \* Revision 1.2 2011/05/19 09:23:45 asarkar  33 : : \* Updated  34 : : \*  35 : : \* Revision 1.1 2011/04/26 11:22:49 asarkar  36 : : \* VSB Library  37 : : \*  38 : : \* Revision 1.1 2007/07/05 06:47:42 santanu  39 : : \* Santanu::Second phase megre  40 : : \*  41 : : \* Revision 1.2 2007/02/05 11:46:11 santanu  42 : : \* EBM5K updated to version 1.67  43 : : \*  44 : : \* Revision 1.5 2005/03/15 00:30:15 ram  45 : : \* Added the changes for Self-Diagnostics macro. The SELF\_DIAG is changed to SELFDIAG\_PMD as a part of pattern shift detection.  46 : : \*  47 : : \* Revision 1.4 2004/09/20 21:57:29 ram  48 : : \* Changes related to Self Diagnostics  49 : : \*  50 : : \* Revision 1.3 2003/12/16 21:42:49 gopal  51 : : \* 1. Merged from main CVS after performance related changes  52 : : \*  53 : : \* Revision 1.14 2003/12/11 16:41:33 gopal  54 : : \* 1. Merged performance related changes from exp kit 6-b  55 : : \*  56 : : \* Revision 1.13 2003/11/10 15:58:32 raj  57 : : \* CR PG Keep bug fixes  58 : : \* 1. Termination conditions  59 : : \* 2. Flag resetting  60 : : \* 3. coordinate calculations  61 : : \* 4. counter updates  62 : : \*  63 : : \* Revision 1.12 2003/11/10 14:48:53 raj  64 : : \* CR PG Keep changes done by Santhosh Kudva San after CR PG Keep design  65 : : \*  66 : : \* 1. Complete changes for CR functions.  67 : : \* 2. related changes in fetching OR , NR and AR  68 : : \* 3. translate function changes  69 : : \* 4. Context data structure changes  70 : : \* ( this code release date 10 Oct 2003)  71 : : \* Revision 1.11 2003/10/06 23:58:27 kudva  72 : : \* Changes for Byte Alignment  73 : : \* 1. Added isAligned flag to context  74 : : \* 2. Added changes for APx and APy processing when isAligned is true  75 : : \* 3. Added changes to CR for APx and APy processing  76 : : \*  77 : : \* Checked in by - Santhosh Kudva  78 : : \*  79 : : \* Revision 1.10 2003/09/08 21:52:37 kudva  80 : : \* Bug ID #24 : Patterns have 5 AU Error (AR-Linear-Term, No Scale, AU Conversion Factor = 10)  81 : : \* Analysis : An Array Representation can have a non integral pitch. So to solve this,  82 : : \* 1. RoundINT was added to calculate pattern position (PP) from pitch.  83 : : \* 2. The PP was Translated. (This involves one more RoundINT)  84 : : \* Due to this the PP which was translated had a lower precision  85 : : \*  86 : : \* Eg. if pattern position was PP.X0 = 123.45 (after using pitch to calculate it)  87 : : \* RoundINT would result in PP.X0 = 123  88 : : \* AU Conversion = PP.X0 \* 10 = 1230  89 : : \*  90 : : \* However, if PP.X0 = 123.45  91 : : \* AU Conversion of this value would be 1234.5 and on round int. PP.X0 = 1235.  92 : : \* This is a 5 AU difference.  93 : : \*  94 : : \* Fix : To remove the RoundINT, the PP must be kept as a double. The same problem was also anticipated for CR. Therefore, vsb now implements a new type vsb\_ep and vsb\_pp, whose members are of primitive type double. EP & PP are now maintained in vsb\_ep and vsb\_pp structures till they have to be returned to the caller. This results in at most 0.5 AU of error from VSB for Pattern positions.  95 : : \*  96 : : \* Reviewed by - Gopal Mor  97 : : \* Checked in by - Santhosh Kudva  98 : : \*  99 : : \* Revision 1.9 2003/08/12 19:15:40 kudva  100 : : \* Bug ID : 36 Reported by : Anpo-san, Script : bug\_036.maf  101 : : \* Analysis : End positions of Matrix, Horizontal and Vertical Terms are less than the start positions  102 : : \* Fix : The X0Offset and Y0Offset of the CR was not being added to Xe and Ye. This is now being added.  103 : : \*  104 : : \* Checked in by - Santhosh Kudva  105 : : \* Reviewed by - Narayana Reddy  106 : : \*  107 : : \* Revision 1.8 2003/07/25 20:29:41 kudva  108 : : \* Bug ID:023 & 024  109 : : \* Linear Term array results in cascaded errors  110 : : \* Analysis : The linear term array has an end position which results in non-integral pitch. Our initial assumption was that pitch would always be integral  111 : : \* Fix : Pitch calculation now casted to double. RoundINT added to ARGetPP  112 : : \*  113 : : \* Reviewed by - Shiva  114 : : \*  115 : : \* Revision 1.7 2003/05/09 01:03:04 kudva  116 : : \* Bug ID : 99  117 : : \* Problem Linear Array Pitch with ep->Y0 > 0 has Ye < ep->Y0  118 : : \* Resolution : Incorrection Calculation of Xe and Ye. the value of ep->X0 and ep->y0 was not added. Also computation of end position from pitch was changed  119 : : \* Xe = (Xe \* (Nx -1)) + ep->X0  120 : : \* Ye = (Ye \* (Ny -1)) + ep->Y0  121 : : \*  122 : : \* Reviewed by - Shivakumar TR  123 : : \* Checked in by - Santhosh Kudva  124 : : \*  125 : : \* Revision 1.6 2003/04/30 21:44:58 raj  126 : : \* Bug ID #34  127 : : \* Calculation of Xe for Matrix TERM returns incorrect value  128 : : \* Resolution  129 : : \* The Extraction of AP for Matrix TERM array is incorrect. Apx, APy was extracted as Nx,Ny,Xe,Ye, instead of Nx,Xe, Ny,Ye. Therefore changed the order of extraction.  130 : : \*  131 : : \* Checked in by - Santhosh Kudva  132 : : \* Reviewed - Rajanarayana P.K. and Praveen Kumar  133 : : \*  134 : : \* Revision 1.5 2003/04/30 19:14:10 praveen  135 : : \* Changes  136 : : \* Fix for Bug ID : 31  137 : : \* 1. The Ye value was not updated. Because for Horizontal and Vertical Term, only one of the end positions is specified.  138 : : \* Resolution  139 : : \* 1. Xe,Ye is now initialized to ep->X0 and ep->Y0  140 : : \*  141 : : \* Checked in by - Santhosh Kudva  142 : : \* Reviewed by - Praveen Kumar  143 : : \*  144 : : \* Revision 1.4 2003/04/30 18:59:23 praveen  145 : : \* Changes  146 : : \* Fix for Bug ID #31  147 : : \* 1. The VSB Term array begin position is getting added to the end position Xe, Ye  148 : : \* Resolution  149 : : \* 1. Only for Linear Case the End position is the same as the begin position, because the linear array is returned as a EP (Array with 1 repetition) and a set of PP's PP+. Code changed to reflect this case. For all other cases the Array Specification is used to populate Xe,Ye  150 : : \*  151 : : \* Checked in by - Santhosh Kudva  152 : : \* Reviewed by - Praveen Kumar  153 : : \*  154 : : \* Revision 1.3 2003/04/29 22:28:53 narayan  155 : : \* Changes  156 : : \* 1. L1, L2 byte length defined in Ph1 and Ph2 flags start from 0, therefore 1 needs to be added  157 : : \*  158 : : \* Checked in by - Santhosh Kudva  159 : : \* Reviewed by - Narayana Reddy  160 : : \*  161 : : \* Revision 1.2 2003/04/25 17:29:47 narayan  162 : : \* Updated private declerations of function prototypes with extern "C" to  163 : : \* prevent Name Mangling  164 : : \*  165 : : \* Reviewed By - Narayana Reddy  166 : : \* Checked in By - Santhosh Kudva  167 : : \*  168 : : \* Revision 1.1 2003/04/23 21:43:41 kudva  169 : : \* VSB First Code Checkin - First Baseline - Santhosh Kudva  170 : : \*  171 : : \* -------------------------------------------------------------------------\*/  172 : :   173 : : #include <stdio.h>  174 : : #include "dec\_types.h"  175 : : #include "dec\_math.h"  176 : : #include "vsb\_types.h"  177 : : #include "vsb\_utils.h"  178 : : #include "context.h"  179 : : #include "vsb\_translate.h"  180 : : #include "PH4Flag.h"  181 : : #include "SDDB.h"  182 : : namespace nft {  183 : : /\* Private Declerations \*/  184 : :   185 : : extern "C" int ARGetEP(GroupContext \*pgc, EP \*ep, int \*Nx, int \*Ny, int \*Xe, int \*Ye) ;  186 : : extern "C" int ARGetPP(GroupContext \*pgc, PP \*pp) ;  187 : :   188 : :   189 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  190 : : \* Name: ARGetEP  191 : : \* Functionality: Extracts a AR Representation (OR2:PH1(NPEPiPP+)+)  192 : : \* Input: Pointers to GroupContext, EP, Number of X Repetitions, Number of Y   193 : : \* Repetitions, Array end position X direction, Array end position Y direction  194 : : \* Output: Returns an EP as a Array Representation  195 : : \* Return value: Success or failure of the operation  196 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  197 : : int ARGetEP(GroupContext \*pgc, EP \*ep, int \*Nx, int \*Ny, int \*Xe, int \*Ye)  198 : 267 : {  199 : 267 : ARContext \*arctx = NULL ;  200 : : // vsbPH2 \*ph2 = NULL ;  201 : 267 : vsbPH2 ph2 ;  202 : 267 : int SkipSize = 0 ;  203 : 267 : int NumFieldsInEP = 0 ;  204 : 267 : int L1WordLength = 0 ;  205 : 267 : int L2WordLength = 0 ;  206 : 267 : int Value = 0 ;  207 : 267 : int APSize ;  208 : 267 : unsigned char PatCode ;  209 : 267 : PatternArrangement parrange ;  210 : 267 : ArrayType arrtype ;  211 : 267 : alias\_PH4\_1 alias\_PH4\_1Val;  212 : 267 : alias\_PH4\_2 alias\_PH4\_2Val;  213 : 267 : alias\_PH2 alias\_PH2Val;  214 : :   215 : 267 : arctx = &(pgc->ppc.arctx) ;  216 : :   217 [ + + ]: 267 : if(arctx->isEmpty)  218 : : {  219 : 189 : uint temp ;   220 : : //temp = vsbGetInt(pgc->current) ;  221 : 189 : alias\_PH4\_1Val.value = alias\_PH4\_2Val.value = vsbGetInt(pgc->current) ;  222 : :   223 : : //PH4 handling  224 : 189 : if(1 == PH4Flag::instance()->getValue())  225 : : {  226 : 0 : vsbPH4\_1 ph4;  227 : : //ph4 =\*(vsbPH4\_0 \*)(&temp) ;  228 : 0 : ph4 = alias\_PH4\_1Val.ph4 ;  229 [ # # ][ # # ]: 0 : switch(ph4.b)  230 : : {  231 : : case 0xb1:  232 : : {  233 : 0 : vsbPH4\_1 ph4\_1;  234 : : //ph4\_1=\*(vsbPH4\_1 \*)(&temp) ;  235 : 0 : ph4\_1 = alias\_PH4\_1Val.ph4 ;  236 : 0 : pgc->attributeInfo=ph4\_1.AI1;  237 : 0 : pgc->current = pgc->current + sizeof(vsbPH4\_1);  238 : 0 : break;  239 : : }  240 : : case 0xb2:  241 : : {  242 : 0 : vsbPH4\_2 ph4\_2;  243 : : //ph4\_2 = \*(vsbPH4\_2 \*)(&temp) ;  244 : 0 : ph4\_2 = alias\_PH4\_2Val.ph4 ;  245 : 0 : pgc->attributeInfo = ph4\_2.AI2;  246 : 0 : pgc->current = pgc->current + sizeof(vsbPH4\_2);  247 : 0 : break;  248 : : }  249 : : }  250 : : }  251 : : else  252 : : {  253 : 189 : pgc->attributeInfo=0;  254 : : }  255 : :   256 : : //temp = vsbGetInt(pgc->current) ;  257 : 189 : alias\_PH2Val.value = vsbGetInt(pgc->current) ;  258 : : //ph2 = \*(vsbPH2 \*)(&temp) ; ;  259 : 189 : ph2 = alias\_PH2Val.ph2;  260 : 189 : PatCode = ph2.code1 ;  261 : :   262 : 189 : try{  263 : : //#ifdef ERROR\_SEED //<<Test\_Id:vsb\_patternAR\_PatCode\_Set\_To\_NULL>>  264 : : // PatCode = NULL;  265 : : //#endif  266 : 189 : checkPatternCode(PatCode);  267 : : }  268 [ # # ]: 0 : catch(DPL::Error \*e)  269 : : {  270 : 0 : char msg[1024];  271 : 0 : sprintf(msg,"Invalid Pattern code 0x%x\n",PatCode);  272 : :   273 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  274 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  275 : : ,ErrorHandler::eErrFunction,pErr->getErrorMsg(ErrorHandler::eErrFunction  276 : : ,"Invalid Pattern Code Error while reading Array pattern"));  277 : : }  278 : :   279 : :   280 : 189 : arctx->PatArrangement = parrange = WhichArrangement[ph2.codea] ;  281 : 189 : arrtype = WhichArrayType[ph2.codes % 0x81] ;  282 : :   283 : 189 : L1WordLength = ph2.L1WordLength + 1 ;  284 : 189 : L2WordLength = ph2.L2WordLength + 1 ;  285 : :   286 : 189 : pgc->current += sizeof(vsbPH2) + SizeOfOption[ph2.isOptionAdded] ;  287 : :   288 : 189 : \*Nx = \*Ny = 1 ;  289 : 189 : \*Xe = \*Ye = 0 ;  290 : :   291 : 189 : arctx->CurrLinearReps = arctx->NumLinearReps = 0 ;  292 : :   293 : 189 : APSize = SizeOfAP[pgc->X0Y0WordLength][parrange] ;  294 : :   295 [ - + ]: 189 : if(pgc->isAligned)  296 : : {  297 : 0 : APSize = AlignedSizeOfAP[pgc->X0Y0WordLength][parrange] ;  298 : : }   299 : :   300 : 189 : NumFieldsInEP = vsbNumEPFields(PatCode) ;  301 : 189 : SkipSize = vsbBufferToEP(pgc->current+APSize, NumFieldsInEP, pgc->X0Y0WordLength, L1WordLength, L2WordLength, ep) ;  302 : 189 : ep->PatCode = PatCode ;  303 : 189 : pgc->ppc.ep = \*ep ;  304 : 189 : \*Xe = ep->X0 ;  305 : 189 : \*Ye = ep->Y0 ;  306 : :   307 : : // Copy pre-translated EP to CRContext  308 : : // pgc->ppc.crctx.ReprEP = \*ep ;  309 : :   310 : :   311 [ + + ][ + + ]: 189 : switch(parrange)  [ - + ][ + + ]  [ - + ]  312 : : {  313 : : case Horizontal :   314 : : //\*Nx = \*(short int \*)pgc->current ;   315 : 139 : \*Nx = vsbGetShort(pgc->current) ;   316 : 139 : pgc->current+= sizeof(short int) ;   317 : 139 : pgc->current += vsbBufferToValue(pgc->current, pgc->X0Y0WordLength, pgc->isAligned, &Value) ;  318 : 139 : \*Xe = Value ;  319 [ + + ]: 139 : if(Pitch == arrtype)  320 : 92 : \*Xe = (\*Xe \* (\*Nx -1)) + ep->X0 ;   321 : 92 : break ;  322 : : case Vertical :  323 : : //\*Ny = \*(short int \*)pgc->current ;   324 : 7 : \*Ny = vsbGetShort(pgc->current) ;   325 : 7 : pgc->current+= sizeof(short int) ;   326 : 7 : pgc->current += vsbBufferToValue(pgc->current, pgc->X0Y0WordLength, pgc->isAligned, &Value) ;  327 : 7 : \*Ye = Value ;  328 [ + + ]: 7 : if(Pitch == arrtype)  329 : 5 : \*Ye = (\*Ye \* (\*Ny -1)) + ep->Y0 ;   330 : 5 : break ;  331 : : case Matrix :  332 : : //\*Nx = \*(short int \*)pgc->current ;   333 : 16 : \*Nx = vsbGetShort(pgc->current) ;   334 : 16 : pgc->current+= sizeof(short int) ;   335 : 16 : pgc->current += vsbBufferToValue(pgc->current, pgc->X0Y0WordLength, pgc->isAligned, &Value) ;  336 : 16 : \*Xe = Value ;  337 : 16 : \*Ny = vsbGetShort(pgc->current) ;   338 : 16 : pgc->current+= sizeof(short int) ;   339 : 16 : pgc->current += vsbBufferToValue(pgc->current, pgc->X0Y0WordLength, pgc->isAligned, &Value) ;  340 : 16 : \*Ye = Value ;  341 [ + + ]: 16 : if(Pitch == arrtype)  342 : : {  343 : 4 : \*Xe = (\*Xe \* (\*Nx - 1)) + ep->X0 ;   344 : 4 : \*Ye = (\*Ye \* (\*Ny - 1)) + ep->Y0 ;   345 : : }  346 : 4 : break ;  347 : : case Linear :  348 : : //arctx->NumLinearReps = \*(short int \*)pgc->current ;  349 : 27 : arctx->NumLinearReps = vsbGetShort(pgc->current) ;   350 : 27 : pgc->current += sizeof(short int) ;  351 : 27 : pgc->current += vsbBufferToValue(pgc->current, pgc->X0Y0WordLength, 0, &Value) ;  352 : 27 : arctx->PitchLinearX = Value ;  353 : 27 : pgc->current += vsbBufferToValue(pgc->current, pgc->X0Y0WordLength, 0, &Value) ;  354 : 27 : arctx->PitchLinearY = Value ;  355 : :   356 : 27 : arctx->CurrLinearReps = 1 ;  357 [ + + ]: 27 : if(Term == arrtype)  358 : : {  359 [ + - ]: 17 : arctx->PitchLinearX = (arctx->NumLinearReps > 1) ? (double)(arctx->PitchLinearX - ep->X0)/(double)(arctx->NumLinearReps - 1) : 0 ;  360 [ + - ]: 17 : arctx->PitchLinearY = (arctx->NumLinearReps > 1) ? (double)(arctx->PitchLinearY - ep->Y0)/(double)(arctx->NumLinearReps - 1) : 0 ;  361 : : }  362 : 17 : break ;  363 : : default:  364 : 10 : return -1;   365 : : }  366 : :   367 : 189 : arctx->isEmpty = false ;  368 : 189 : SDDB \*pSDDB = SDDB::instance();  369 [ + + ]: 189 : if(parrange == Linear) {  370 : 27 : try  371 : : {  372 : 27 : pSDDB->EPReadComplete(arctx->NumLinearReps);   373 : : }  374 [ # # ]: 0 : catch( DPL::Error \*pChild )  375 : : {  376 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  377 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  378 : : ErrorHandler::eErrFunction,  379 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  380 : : "SDDB::EPReadComplete"));  381 : : }  382 : :   383 : : } else {  384 : 162 : try  385 : : {  386 : 162 : pSDDB->EPReadComplete((\*Nx) \* (\*Ny));  387 : : }  388 [ # # ]: 0 : catch( DPL::Error \*pChild )  389 : : {  390 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  391 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  392 : : ErrorHandler::eErrFunction,  393 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  394 : : "SDDB::EPReadComplete"));  395 : : }  396 : :   397 : : }  398 : : // Can copy Nx Ny at this point as they are not translated  399 [ + + ]: 189 : if ( Composite == pgc->ppc.WhichContext)  400 : : {   401 : 74 : pgc->ppc.crctx.ReprEP = \*ep ;  402 : 74 : pgc->ppc.crctx.ReprNx = \*Nx ;  403 : 74 : pgc->ppc.crctx.ReprNy = \*Ny ;  404 : 74 : pgc->ppc.crctx.ReprXe = \*Xe ;  405 : 74 : pgc->ppc.crctx.ReprYe = \*Ye ;  406 : : }  407 [ + - ][ - + ]: 115 : else if (( 1.0F != pgc->TranslateFactor || pgc->mirror))   408 : : {  409 [ # # ]: 0 : if( 0 > TranslateEP(pgc, ep, Xe, Ye) )   410 : 0 : return -1 ;  411 : : }  412 : :   413 : 189 : pgc->current += SkipSize ;  414 : 189 : return 0 ;  415 : : }  416 : :   417 : 267 : return -1 ;  418 : : }  419 : :   420 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  421 : : \* Name: ARGetPP  422 : : \* Functionality: Returns one PP from a AR Linear Representation using the   423 : : \* Pitch and Repetitions provided in the ARContext   424 : : \* Input: Pointers to GroupContext and PP  425 : : \* Output: Returns a PP   426 : : \* Return value: Success or failure of the operation  427 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  428 : :   429 : : int ARGetPP(GroupContext \*pgc, PP \*pp)  430 : 74 : {  431 : 74 : ARContext \*arctx = NULL ;  432 : : //int SkipSize = 0 ;  433 : 74 : vsbPP vpp ;  434 : :   435 : 74 : arctx = &(pgc->ppc.arctx) ;  436 : :   437 [ + + ]: 74 : if(arctx->CurrLinearReps < arctx->NumLinearReps)  438 : : {  439 : 47 : vpp.X0 = pgc->ppc.ep.X0 + (arctx->PitchLinearX \* arctx->CurrLinearReps) ;  440 : 47 : vpp.Y0 = pgc->ppc.ep.Y0 + (arctx->PitchLinearY \* arctx->CurrLinearReps) ;  441 : :   442 : : // Copy pre translated PP into CRContext  443 : 47 : pgc->ppc.crctx.ReprPP = vpp ;  444 : :   445 [ + + ][ + - ]: 47 : if ( Composite != pgc->ppc.WhichContext &&  [ - + ]  446 : : ( 1.0F != pgc->TranslateFactor ||  447 : : pgc->mirror))  448 : : {  449 : 0 : if( 0 > TranslatePP(pgc, &vpp) )   450 : 47 : return -1 ;  451 : : }  452 : 47 : dtoi(&vpp, pp) ;  453 : :   454 : 47 : arctx->CurrLinearReps++ ;  455 : 47 : return 0 ;  456 : : }  457 : :   458 : 74 : return -1 ;  459 : : }  460 : :   461 : 1215 : } //namespace nft |

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