| LCOV - code coverage report |
| --- |
|  |
| | Current view: | [top level](http://docs.google.com/index.html) - [LibVsb/include](http://docs.google.com/index.html) - chipcnf.h (source / [functions](http://docs.google.com/chipcnf.h.func.html)) |  |  | Hit | Total | Coverage | | --- | --- | --- | --- | --- | --- | --- | | Test: | EBM9K Converter Coverage Report |  | Lines: | 39 | 40 | 97.5 % | | Date: | 2012-03-23 |  | Functions: | 1 | 1 | 100.0 % | | Legend: | Lines: hit not hit | Branches: + taken - not taken # not executed |  | Branches: | 53 | 252 | 21.0 % | |  | |
|  |

|  |
| --- |
| Branch data Line data Source code  1 : : /\*------------------------------------------------------------------------------  2 : : \*\* - Copyright (c) 2002, Digital GlobalSoft Ltd, NFT Japan -  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 : : \*\* File : chipcnf.h  8 : : \*\*  9 : : \*\* REVISION HISTORY  10 : : \*\* [1] 07 Apr 2003 Gopal Mor Created the file  11 : : \*\*----------------------------------------------------------------------------\*/  12 : :   13 : : #ifndef \_CHIPCNF\_H  14 : : #define \_CHIPCNF\_H  15 : :   16 : : #include "parse.h"  17 : : #include "inttypes.h"  18 : : #include<stdint.h>  19 : :   20 : : namespace nft{  21 : : #define CHIPCNFFILE "/chip.cnf"  22 : :   23 : : void deleteFrmMatrix (void);  24 : :   25 : : typedef struct \_\_frameInfo  26 : : {  27 : : uint64\_t height; /\* value in terms of blocks \*/  28 : : uint64\_t vsbHeight; /\* value in terms of VSB AU \*/  29 : : uint32\_t subFrms;  30 : : int64\_t frmLocX; /\* frame location in VSB AU \*/  31 : : int64\_t frmLocY;  32 : : uint64\_t wid; /\* value in terms of blocks \*/  33 : : uint64\_t vsbWid; /\* value in terms of VSB AU \*/  34 : : uint32\_t vbgIdx;  35 : : uint32\_t vbgIdy;  36 : : char name[LEN\_NAME] ;  37 : : bool isSelected;  38 : : bool isNull ;  39 : :   40 : : \_\_frameInfo()  41 : 1201 : :height(0),  42 : : vsbHeight(0),  43 : : subFrms(0),  44 : : frmLocX(0),  45 : : frmLocY(0),  46 : : wid(0),  47 : : vsbWid(0),  48 : : vbgIdx(0),  49 : : vbgIdy(0),  50 : : name(),  51 : : isSelected(false),  52 : : isNull(false)  53 : 1201 : {  54 : 1201 : name[0] = 0;  55 : : }  56 : :   57 : :   58 : : } \_frameInfo;  59 : :   60 : : class \_frmMatrix  61 : : {  62 : : public:  63 : : mutable \_frameInfo \*data\_; /\* frames, ordered by row, then by col \*/  64 : : mutable \_frameInfo \*\*frameTbl\_; /\* Pointers to rows \*/  65 : : private:  66 : : uint32\_t numRows\_;  67 : : uint32\_t numCols\_;  68 : : public:  69 : :   70 : : \_frmMatrix() {}  71 : : \_frmMatrix(uint32\_t numRows, uint32\_t numCols)  72 : 1119 : : data\_(NULL),   73 : : frameTbl\_(NULL),  74 : : numRows\_(numRows),  75 : : numCols\_(numCols)  76 : 0 : {  77 : :   78 : : }   79 : :   80 : : ~\_frmMatrix()  81 : 1119 : {  82 : 1119 : releaseMatrixMemory();  83 : : }  84 : :   85 : : // Sending out non-const pointer to private elements is not a good idea  86 : : // However in this case we would have to add a lot of interface   87 : : // functions to do this job. So we take this liberty just once  88 : :   89 : : \_frameInfo \*getFrameInfo(uint32\_t col, uint32\_t row) const  90 : 29319 : {  91 [ + - ][ + - ]: 29319 : if( row < numRows\_ && col < numCols\_ )  [ + - ][ + - ]  [ # # ][ # # ]  [ + - ][ + - ]  [ # # ][ # # ]  [ + - ][ + - ]  [ # # ][ # # ]  [ + - ][ + - ]  [ # # ][ # # ]  92 : : {  93 [ + + ][ - + ]: 29319 : if( !data\_ )  [ # # ][ - + ]  [ # # ][ - + ]  [ # # ][ - + ]  [ # # ]  94 : : {  95 : 1119 : createMatrix();  96 : : }  97 [ + - ][ + - ]: 29319 : return (frameTbl\_) ? &frameTbl\_[col][row] : NULL;  [ # # ][ + - ]  [ # # ][ + - ]  [ # # ][ + - ]  [ # # ]  98 : : }  99 : 20340 : return NULL;  100 : : }  101 : :   102 : : void releaseMatrixMemory()  103 : 1119 : {  104 [ # # ][ + - ]: 1119 : if( NULL != data\_)  [ # # ]  105 : : {  106 : 1119 : delete[] data\_;  107 : : }  108 : 1119 : data\_ = NULL;  109 [ # # ][ + - ]: 1119 : if( NULL != frameTbl\_ )  [ # # ]  110 : : {  111 : 1119 : delete[] frameTbl\_;  112 : : }  113 : 1119 : frameTbl\_ = NULL;  114 : : }  115 : :   116 : :   117 : : uint32\_t get\_numRows\_() const  118 : : {  119 : : return numRows\_;  120 : : }  121 : :   122 : : uint32\_t get\_numCols\_() const  123 [ - + ]: 1128 : {  124 : : return numCols\_;  125 : : }  126 : :   127 : : private:  128 : : void createMatrix() const  129 : 1119 : {  130 [ + - ]: 2320 : data\_ = new(std::nothrow) \_frameInfo[numRows\_ \* numCols\_];  [ + - + + ]  [ # # ]  [ # # # # ]  [ # # ]  [ # # # # ]  [ # # ]  [ # # # # ]  [ # # ]  [ # # # # ]  [ # # ]  [ # # # # ]  [ # # ]  [ # # # # ]  [ # # ]  [ # # # # ]  [ # # ]  [ # # # # ]  131 : 1119 : frameTbl\_ = new(std::nothrow) \_frameInfo \*[numCols\_];  132 : :   133 [ + - ]: 1119 : if ( frameTbl\_ && data\_ )  [ + - # # ]  [ # # # # ]  [ # # # # ]  [ # # # # ]  [ # # # # ]  [ # # # # ]  [ # # # # ]  [ # # # # ]  [ # # ]  134 : : {  135 [ + - ][ + + ]: 29341 : for( size\_t i = 0; i < numCols\_; i++ )  [ # # ][ # # ]  [ # # ][ # # ]  [ # # ][ # # ]  [ # # ][ # # ]  [ # # ][ # # ]  [ # # ][ # # ]  [ # # ][ # # ]  [ # # ][ # # ]  136 : : {  137 : : // Column major  138 : 1141 : frameTbl\_[i] = data\_ + ( i \* numRows\_ );  139 : : }  140 : : }  141 : : else  142 : : {  143 : : #if 0   144 : : Log::logger::logAdd(18, "[ERROR] Error allocating memory");  145 : : errMsgs\* pInstance = errMsgs :: instance();  146 : : #ifdef LINUX  147 : : string sTemp = pInstance->LookUpErrorTable(61,  148 : : "Get\_Mds\_File\_Path" ,(char \*)\_\_func\_\_);  149 : : #else  150 : : string sTemp = pInstance->LookUpErrorTable(61,  151 : : "Get\_Mds\_File\_Pat", NULL);  152 : : #endif  153 : : throw new util::Error(sTemp, (const char\*)\_\_FILE\_\_,  154 : : (unsigned int)\_\_LINE\_\_, NULL);  155 : : #endif  156 : : // Exception handling should be done?   157 : : }  158 : : }  159 : :   160 : : };  161 : :   162 : :   163 : : typedef struct \_\_chipCnfFile  164 : : {  165 : : char \*frmtVersion; /\* version of VSB12 format \*/  166 : : char \*convSystem; /\* Conversion system name (software) \*/  167 : : char \*convMachine; /\* Machine name of conversion system \*/  168 : : char \*convMachineId; /\* Machine ID of conversion system \*/  169 : : char \*convMachineOS; /\* OS of conversion system \*/  170 : : char \*chipName; /\* Name of the chip \*/  171 : : int64\_t chipAddrUnit; /\* AU of cell location data in unit of fm (10e-15m) \*/  172 : : uint64\_t chipSizeX; /\* Chip size in horozontal direction \*/  173 : : uint64\_t chipSizeY; /\* Chip size in vertical direction \*/  174 : : int32\_t chipBSizeX; /\* Block size in horizontal direction \*/  175 : : int32\_t chipBSizeY; /\* Block size in vertical direction \*/  176 : : int32\_t chipFrames; /\* Number of frames in chip \*/  177 : : char \*frmOrientation; /\* Orientation of frames \*/  178 : : int32\_t frmHeight; /\* frame height \*/  179 : : int32\_t frmWid; /\* frame width \*/  180 : : char \*multiPlacement; /\* Whether multiple placement rule is applied \*/  181 : : int32\_t cellMaxSizeX; /\* Maximum size of cell in horizontal \*/  182 : : int32\_t cellMaxSizeY; /\* Maximum size of cell in vertical \*/  183 : : int cellArrMaxSizeX; /\* Maximum size of array cell in horizontal \*/  184 : : int cellArrMaxSizeY; /\* Maximum size of array cell in vertical \*/   185 : : char \*patArrayCompress; /\* Pattern compression by AR \*/  186 : : char \*patCompress; /\* Pattern compression by OR2 \*/  187 : : int32\_t patSet; /\* Set of patterns used when formatted \*/  188 : : int32\_t maxDataRef; /\* Limit of frame reference data \*/  189 : : int32\_t maxDataCell; /\* Limit of pattern data \*/  190 : : int32\_t maxDataCommon; /\* Limit of data amount of common data \*/  191 : : int32\_t maxDataFrame; /\* Max frame data size \*/  192 : : uint32\_t maxNumCellLoc; /\* Max number of cell locations \*/  193 : : int32\_t frameTranslateY; /\* frame translation in y direction \*/  194 : : \_frameInfo \*\*frame; /\* pointer points to frame matrix \*/  195 : : \_frmMatrix \*pMatrix\_;  196 : : \_\_chipCnfFile()  197 : 1125 : : frmtVersion (NULL),  198 : : convSystem(NULL),  199 : : convMachine(NULL),  200 : : convMachineId(NULL),  201 : : convMachineOS(NULL),  202 : : chipName(NULL),  203 : : chipSizeX(0),  204 : : chipSizeY(0),  205 : : chipBSizeX(0),  206 : : chipBSizeY(0),  207 : : chipFrames(0),  208 : : frmOrientation(NULL),  209 : : multiPlacement(NULL),  210 : : cellMaxSizeX(0),  211 : : cellMaxSizeY(0),  212 : : cellArrMaxSizeX(0),  213 : : cellArrMaxSizeY(0),  214 : : patArrayCompress(NULL),  215 : : patCompress(NULL),  216 : : patSet(0),  217 : : maxDataRef(0),  218 : : maxDataCell(0),  219 : : maxDataCommon(0),  220 : : maxDataFrame(0),  221 : : maxNumCellLoc(0),  222 : : frameTranslateY(0),  223 : : pMatrix\_(NULL)  224 : 1125 : {  225 : : }  226 : :   227 : : ~\_\_chipCnfFile()  228 : 1125 : {  229 [ + + ][ # # ]: 1125 : delete [] frmtVersion;  230 [ + + ][ # # ]: 1125 : delete [] convSystem;  231 [ + + ][ # # ]: 1125 : delete [] convMachine;  232 [ + + ][ # # ]: 1125 : delete [] convMachineId;  233 [ + + ][ # # ]: 1125 : delete [] convMachineOS;  234 [ + + ][ # # ]: 1125 : delete [] chipName;  235 [ + + ][ # # ]: 1125 : delete [] frmOrientation;  236 [ + + ][ # # ]: 1125 : delete [] multiPlacement;  237 [ + + ][ # # ]: 1125 : delete [] patArrayCompress;  238 [ + + ][ # # ]: 1125 : delete [] patCompress;  239 : : //delete frame; // Is deleted from inside parser  240 : : }  241 : :   242 : : } \_chipCnfFile;  243 : :   244 : : extern int extern\_parseChipCnf(char \*path) ;  245 : : double getVSBtoPDGConvFact(void) ;  246 : : void cleanupChipCnf(void) ;  247 : : extern \_chipCnfFile\* getChipCnfTbl(void) ;  248 : :   249 : : } //namespace nft  250 : : #endif /\* \_CHIP\_H \*/ |

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