TECHNICAL REPORT

PIC-Fomat

IPPIC BIBLIOTHEK

VERSION 1.0.7

dkfz **Deutsches Krebsforschungszentrum**

Deutsches
Krebsforschungszentrum

Im Neuenheimer Feld 280
D-69120 Heidelberg
Tel.: (+49) 6221 - 42 2354
Fax (+49) 6221 - 42 2345

Hotline: MBI@DKFZ-Heidelberg.de

Stand 15.9.98
© 1995-1998 Deutsches Krebsforschungszentrum, Heidelberg

Nach dem Stand der Technik ist es nicht möglich, Computersoftware so zu erstellen, daß sie in allen Anwendungen und Kombinationen fehlerfrei arbeitet. Aus diesem Grund übernehmen die Autoren keine Haftung für die Fehlerfreiheit der Software. Insbesondere übernehmen die Autoren keine Gewähr dafür, daß die Software den Anforderungen und Zwecken des Erwerbers genügt oder mit anderen von ihm ausgewählten Programmen zusammenarbeitet. Die Verantwortung für die Folgen der Benutzung der Software, sowie der damit beabsichtigten oder erzielten Ergebnisse trägt der Erwerber.

Die Autoren behalten sich das Recht vor, Änderungen jederzeit ohne Ankündigung vorzunehmen.

Alle Rechte, insbesondere das Recht der Vervielfältigung und Verbreitung, sowie der Übersetzung, bleiben vorbehalten. Kein Teil der Software, bzw. des Benutzerhandbuches darf in irgendeiner Form ohne schriftliche Genehmigung der Autoren reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

TABLE OF CONTENTS

1 PREFACE

1.1 Audience

This document is written for programmers and for curious people interested in the pic image file format.

1.2 Related Documents

None.

1.3 Typographical Conventions

`courier`	Words and letters in courier type font in examples should be typed as is. This font is used for programming examples.
`italics`	Italic (slanted) type indicates variable values, placeholders, and function arguments names.

1.4 Intention

This manual describes the implementation of the C-Language Interface to the PIC3.0-Fileformat. This library is intended to give programmers a platform independent access to the information stored in a PIC-File. The data will be converted from the external representation on the storage device to the internal representation in the memory. These conversions include

for example the conversion to the datatypes and byteorder (endianess) of the machine.

In the current implementation version very little error checking is done!

1.5 The PIC-Format

The PIC-Image-Format is defined for different storage or transmission media. For example an internal (in memory), a network (ipMsg) and an external (on file) representation. The ipPic library is used to convert a PIC-Image between these defined representations.

A PIC Image can be splitted into three main parts:

- Header
- Tag Fields
- Pixel Data

The *Header* describes the *semantic* of the *Pixel Data* for the stored (multidimensional) image (data type, number of dimensions, etc.) which is the *syntax* of the stored image. The *Tag Fields* are a collection of *Tag-Structure-Value* (TSV) elements that store additional data related to the image.

These TAGs can contain structured data, unstructured data or a collection of other TAGs.

1.6 Internal Representation

The internal representation of the PIC-Format is the following C structure declared in *ipPic.h*:

Fig. 1-1: ipPicDescriptor

```
typedef struct
{
        void *data;     /* pointer to pixel data         */
   ipPicInfo *info;     /* pointer to the Info structure */
 ipPicType_t  type;     /* datatype of the pixel data    */
   ipUInt4_t  bpe;      /* bits per element              */
   ipUInt4_t  dim;         /* number of dimensions       */
   ipUInt4_t  n[_ipPicNDIM]/* size of dimension n[i]     */
} ipPicDescriptor;
```

Where *_ipPicNDIM* is *8* and *type* is one of:

```
typedef enum
{
  ipPicUnknown,
  ipPicBool,
  ipPicASCII,
  ipPicInt,
  ipPicUInt,
  ipPicFloat,
  ipPicNonUniform,
  ipPicTSV
};
```

A TSV element is realized as the following C-Structure:

```
typedef struct
  {
           char  tag[_ipPicTAGLEN+1]; /* name of this tag  */
    ipPicType_t  type;            /* datatype of the value */
      ipUInt4_t  bpe;             /* bits per element      */
      ipUInt4_t  dim;             /* number of dimensions  */
        ipUInt4_t  n[_ipPicNDIM]; /* size of dimension n[i]*/
             void *value;         /* the value             */
  } ipPicTSV_t;
```

Fig. 1-2: ipPicTSV_t

Where *_ipPicTAGLEN* is *32* and *_ipPicNDIM* is *8*.

1.7 External Representation

All data in an external PIC-File is written in little endian format. It basically is a serialization of the in memory representation.

The external representation of the PIC-Format as a PIC-File on a storage media is as follows:

PIC Image File

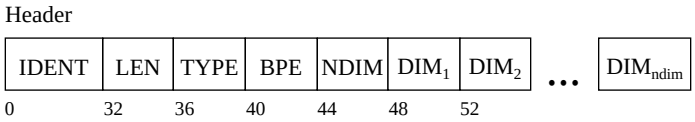
Header	Tag Fields	Pixel Data

Fig. 1-3: Main Parts of the PIC File

The Header describes the *semantic* of the *Pixel Data* for the stored (multidimensional) image (data type, number of dimensions etc.). The *Tag Fields* are a collection of *Tag-Structure-Value* (TSV) elements that store additional data related to the image.

In more detail the header has the following structure:

Fig. 1-4: Structure of the Header

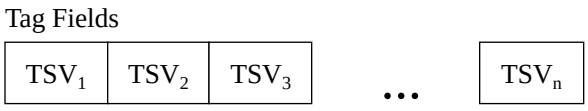


The length of the complete header is 48 + 4 * *NDIM* bytes.

The meaning of the different fields is shown in figure 1-6 at page 1-5.

The Tag Fields contain the TSV elements in sequential order:

Fig. 1-5: Structure of the Tag Fields



A *Tag-Structure-Value* entity (*TSV*) consists of the following items:

- **TAG**: This is an ASCII character string of the length 32. If the TAG has less than 32 characters it is filled up with blanks. The purpose of this string is to describe the *meaning* of the last item (value) of the TSV.
- **STRUCTURE**: This element consists itself of several elements which describe the data type and structure of the data in the VALUE field.The subitems are LENGTH, TYPE, BPE, NDIM, DIM_1, ..., DIM_n
- **VALUE**: This is the data contained of the TSV element written in sequential order without any delimiters.

The meaning of all fields is shown in figure 1-6 at page 1-5.

The Pixel data is written in sequential order without any delimiters.

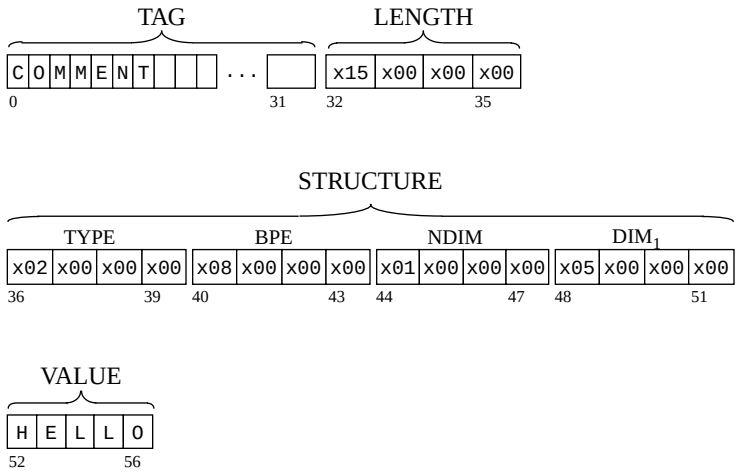
Fig. 1-6: The elements of a PIC-File

Field	Length	Data Type	Meaning
IDENT	32	ASCII	Identification String for the PIC Format and Version (padded with blanks) e.g. "PIC Version 3.00 ..."
TAG	32	ASCII	Identification String for the TAG (padded with blanks).
LENGTH	4	unsigned integer	Number of bytes in the value field
TYPE	4	unsigned integer	data type of the value: 1 = bool 2 = ASCII 3 = Integer 4 = Unsigned Integer 5 = Float 6 = Non Uniform 7 = TSV
BPE	4	unsigned integer	Bits per Element
NDIM	4	unsigned integer	Number of Dimensions
DIM_n	4	unsigned integer	Size of Dimension n

1.7.1 Example 1: TSV Element

This example shows a possible TAG with name ‘COMMENT’, a LENGTH item with the unsigned integer value x1500 000 (dec. 21 = skip 21 bytes to find the next tag) and the VALUE item “HELLO”.

Example of a TSV quadruple:



1.7.2 Example 2: Hexdump of an Image

Fig. 1-7: Hexdump of an PIC-File

```
Addr.           Contents hexadezimal                ASCII
-----------------------------------------------------------------
000000 5049 4320 5645 5253 494f 4e20 332e 3030 | PIC VERSION 3.00
000016 2020 2020 2020 2020 2020 2020 2020 2020 |
000032 7400 0000 0300 0000 1000 0000 0200 0000 | t...............
000048 0001 0000 0001 0000 5245 4d41 524b 2020 | ........REMARK
000064 2020 2020 2020 2020 2020 2020 2020 2020 |
000080 2020 2020 2020 2020 3c00 0000 0200 0000 | <.......
000096 0800 0000 0100 0000 2c00 0000 2863 2920 | ........,...(c)
000112 3139 3933 2062 7920 444b 465a 2028 4465 | 1993 by DKFZ (De
000128 7074 2e20 4d42 4929 2048 6569 6465 6c62 | pt. MBI) Heidelb
000144 6572 672c 2046 5247 0000 0000 0000 0000 | erg, FRG........
000160 0000 0000 0000 0000 0000 0000 0000 0000 | ................
000176 0000 0000 0000 0000 0000 0000 0000 0000 | ................
000192 0000 0000 0000 0000 0000 0000 0000 0000 | ................
000208 0000 0000 0000 0000 0000 0000 0000 0000 | ................
000224 0000 0000 0000 0000 0000 0000 0000 0000 | ................
000240 0000 0000 0000 0000 0000 0000 0000 0000 | ................
000256 0000 0000 0000 0000 0000 0000 0000 0000 | ................
000272 0000 0000 0000 0000 0000 0000 0000 0000 | ................
000288 0000 0000 0000 0000 0000 0000 0000 0000 | ................
000304 0000 0000 0000 0000 0000 0000 0000 0300 | ................
000320 0400 0000 0300 0000 0300 0000 0000 0000 | ................
000336 0b00 0000 0400 0000 0600 0500 0000 0700 | ................
000352 0000 0000 0000 0600 0900 0100 0000 0000 | ................
000368 0000 0000 0000 0300 0700 0200 0e00 0000 | ................
000384 0000 0000 0000 0000 0000 0100 0000 0200 | ................
000400 0400 0600 0900 0000 0000 0000 0500 0e00 | ................
.................
...........
```

The hexdump shows in the first column the address of the first byte in the corresponding row. The content is the following:

- The first two lines (32 bytes) contain the *PIC Identification string* “PIC Version 3.00” padded with blanks.
- *LENGTH* field with the content hex 7400 0000 (dec. 116). The current position after reading this field is 36. If the LENGTH of 116 is added to that, the resulting header length will be 152. This means that the first pixel values starts at position 152 (counting starts with 0).
- *TYPE* field (content: hex 0300 0000 = dec. 3 = integer),
- *BPE* (hex 1000 000 = dec. 16 = bits per pixel),
- *NDIM* (hex 0200 0000 = dec. 2 = no. of dimensions),
- DIM_1 (hex 0001 0000 = dec. 256),
- DIM_2 (hex 0001 0000 = dec. 256),

- *TAG* “REMARK” (32 bytes filled up with blanks),
- *LENGTH* field of TSV structure REMARK (x3c00 000 = dec. 51 bytes until next TSV or here: until first pixel value),
- *TYPE* field of TSV structure REMARK (x0200 0000 = dec 2 = data type ASCII),
- *BPE* field of TSV structure REMARK (hex 0800 000, dec. 8 = bits per pixel),
- *NDIM* field of TSV structure REMARK (hex 0100 0000, dec. 1 = no. of dimensions),
- DIM_1 field of TSV structure REMARK which contains the length of the VALUE field (hex 2c00 0000 = dec. 44)
- *VALUE* of the TSV structure REMARK: “(c) 1993 by DKFZ (Dept. MBI) Heidelberg, FRG”
- Pixel *VALUES* start at address 152. The size of the whole file is 131224 bytes (56 bytes Header, 96 bytes Tag Fields, 256 x 256 x 2 (= 131072) bytes image data).

2 FUNCTIONS

This chapter presents an alphabetical list of all functions in the ipPic library.

ipPicGet

Synopsis

```
#include <ipPic.h>
ipPicDescriptor *ipPicGet( char *file_name,
                           ipPicDescriptor *pic );
```

Description

Reads a PIC-File from a filesystem.

Parameters

file_name	The name of the file to read.
pic	If not NULL the file will be read into this picDescriptor if possible.

Return Values

A picDescriptor filled with the data from the file *file_name* or *NULL*.

Example

```
ipPicDescriptor *pic;
...
pic = ipPicGet( "test.pic",
                NULL );
```

ipPicGetSlice

Synopsis

```
#include <ipPic.h>
ipPicDescriptor *ipPicGetSlice( char *file_name,
                                ipPicDescriptor *pic,
                                ipUInt4_t slice );
```

Description

Reads a 2D slice from a 3D PIC-File from a filesystem. The first slice has number 1.

Parameters

file_name	The name of the file to read.
pic	If not NULL the file will be read into this picDescriptor if possible.
slice	the number of the slice to read.

Return Values

A picDescriptor filled with the data from the slice *slice* from the file *file_name* or *NULL*.

Example

```
ipPicDescriptor *pic;
...
pic = ipPicGetSlice( "test.pic",
                     NULL,
                     1 );
```

ipPicPut

Synopsis

```
#include <ipPic.h>
void ipPicPut( char *file_name,
               ipPicDescriptor *pic );
```

Description

Writes a PIC-Descriptor to a filesystem.

Parameters

file_name	The name of the file to write.
pic	The picDescriptor to be written.

Return Values

None.

Example

```
ipPicDescriptor *pic;
...
ipPicPut( "test.pic",
          pic );
```

ipPicPutSlice

Synopsis

```
#include <ipPic.h>
void ipPicPutSlice( char *file_name,
                    ipPicDescriptor *pic
                    ipUInt4_t slice );
```

Description

Writes a 2D PIC-Descriptor as a slice into a 3D PIC-File on a filesystem. The first slice has number 1.

Parameters

file_name	The name of the file to written.
pic	The picDescriptor to be written.
slice	The number of the slice to write.

Return Values

None.

Example

```
ipPicDescriptor *pic;
...
ipPicPutSlice( "test.pic",
               pic,
               1 );
```

ipPicGetHeader

Synopsis

```
#include <ipPic.h>
ipPicDescriptor *ipPicGetHeader( char *file_name,
                                 ipPicDescriptor *pic );
```

Description

Reads the header of a PIC-File from a filesystem.

Parameters

file_name	The name of the file to be read.
pic	If not NULL the file will be read into this picDescriptor if possible.

Return Values

A picDescriptor filled with the header from the file *file_name*.

Example

```
ipPicDescriptor *pic;
...
pic = ipPicGetHeader( "test.pic",
                      NULL );
```

ipPicGetTags

Synopsis

```
#include <ipPic.h>
ipPicDescriptor *ipPicGetTags( char *picfile_name,
                               ipPicDescriptor *pic );
```

Description

Reads the TAGs of PIC-File from a filesystem.

Parameters

file_name	The name of the file to read.
pic	If not NULL the tags will be read into this picDescriptor if possible.

Return Values

A picDescriptor with the header and the tags from the file *file_name*.

Example

ipPicVersionMajor
ipPicVersionMinor

Synopsis

```
#include <ipPic.h>
ipUInt1_t ipPicVersionMajor( ipPicDescriptor *pic );
ipUInt1_t ipPicVersionMinor( ipPicDescriptor *pic );
```

Description

ipPicVersionMajor returns the major version number of the supplied picDescriptor, *ipPicVersionMinor* returns the minor version number.

Parameters

pic The picDescriptor to get info about.

Return Values

ipPicVersionMajor returns the major version number

ipPicVersionMinor returns the minor version number.

Example

ipPicDR

Synopsis

```
#include <ipPic.h>
ipUInt4_t ipPicDR( ipUInt2_t type,
                   ipUInt2_t bpe );
```

Description

Combines a type and bpe parameter into a 32 bit data representation code. This code can for example be used in a switch statement to branch dependent on the real pixel data type.

Parameters

type	a ipPic type code.
bpe	The bits per element value.

Return Values

The combined 32 bit data representation code.

Example

```
ipPicDescriptor *pic;
...
switch( ipPicDR( pic->type, pic->bpe) )
  {
    case( ipPicDR( ipPicUInt, 8 ):   /*  8bit uint */
      ...
      break;
    case( ipPicDR( ipPicInt, 16 ):   /* 16bit int  */
      ...
      break;
    .
    .
    .
    default:
      fprintf( stderr, datatype (%i/%i) not supported for",
               pic->type, pic->bpe );
      break;
  }
```

ipPicNew

Synopsis

```
#include <ipPic.h>
ipPicDescripor *ipPicNew( void );
```

Description

Allocates and initializes a new picDescriptor.

Parameters

None.

Return Values

A new empty picDescriptor.

Example

```
ipPicDescriptor *pic;
...
pic = ipPicNew( );
```

ipPicClone

Synopsis

```
#include <ipPic.h>
ipPicDescriptor *ipPicClone( ipPicDescriptor *pic );
```

Description

This function creates an exact copy of a picDescriptor. All data is duplicated.

Parameters

pic The picDescriptor to duplicate.

Return Values

The duplicated picDescriptor.

Example

ipPicCopyHeader

Synopsis

```
#include <ipPic.h>
ipPicDescriptor *ipPicCopyHeader( ipPicDescriptor *pic,
                                  ipPicDescriptor *pic_new );
```

Description

Makes a copy of the header of a picDescriptor.

Parameters

pic	The picDescriptor that should his header copied.
pic_new	If not NULL this picDescriptor will be overwritten.

Return Values

A picDescriptor with the header filled from *pic*. Only the header is copied. Not the TAGs or the pixel data.

Example

ipPicClear

Synopsis

```
#include <ipPic.h>
void ipPicClear( ipPicDescriptor *pic );
```

Description

Empties a picDescriptor. All tags and the pixel data is cleared.

Parameters

pic The picDescriptor that should be cleared.

Return Values

None.

Example

ipPicFree

Synopsis

```
#include <ipPic.h>
void ipPicFree( ipPicDescriptor *pic );
```

Description

Frees all memory occupied by a picDescriptor.

Parameters

pic The picDescriptor that should be freed.

Return Values

None.

Example

ipPicAddTag
ipPicAddSubTag

Synopsis

```
#include <ipPic.h>
void ipPicAddTag( ipPicDescriptor *pic, ipPicTSV_t *tsv );
void ipPicAddSubTag( ipPicTSV_t *parent, ipPicTSV_t *tsv );
```

Description

Adds a new TAG or subTAG to a picDescriptor. It is not checked if a TAG with the same name already exists.

Parameters

pic	The picDescriptor where the new TAG should be added.
parent	The parent TSV where the new TAG should be added.
tsv	The TSV to add.

Return Values

None.

Example

```
ipPicDescriptor *pic;
ipPicTSV_t tsv;
...
tsv = malloc( sizeof(ipPicTSV_t) );
strcpy( tsv->tag, "PATIENT NAME" );
tsv->type = ipPicASCII;
tsv->bpe = 8;
tsv->dim = 1;
tsv->value = strdup( "TEST PATIENT" );
tsv->n[0] = strlen(tsv->value);

ipPicAddTag( pic, tsv );
```

ipPicDelTag
ipPicDelSubTag

Synopsis

```
#include <ipPic.h>
ipPicTSV_t *ipPicAddTag( ipPicDescriptor *pic, char *name );
ipPicTSV_t *ipPicAddSubTag( ipPicTSV_t *parent, char *name );
```

Description

Removes a TAG from a picDescriptor. No memory is freed.

Parameters

pic	The picDescriptor where the named TAG should be removed.
parent	The parent TSV where the named TAG should be removed.
name	The name of the TSV to remove.

Return Values

A pointer to the removed TSV or NULL.

Example

```
ipPicDescriptor *pic;
ipPicTSV_t tsv;
...
tsv = ipPicDelTag( pic, "MY TAG" );

if( tsv )
  {
    if( tsv->value )
      free( tsv->value )

    free( tsv );
  }
```

ipPicQueryTag
ipPicQuerySubTag

Synopsis

```
#include <ipPic.h>
ipPicTSV_t *ipPicQueryTag( ipPicDescriptor *pic, char *name );
ipPicTSV_t *ipPicQuerySubTag( ipPicTSV_t *parent, char *name);
```

Description

Retrieves a TAG.

Parameters

pic	The picDescriptor to be queried.
parent	The parent TSV to be queried.
name	The name of the TAG that should be queried.

Return Values

A pointer to the found TSV or NULL.

Example

```
ipPicDescriptor *pic;
ipPicTSV_t tsv;
...
tsv = ipPicQueryTag( pic, "PATIENT NAME" );

if( tsv
    && tsv->type == ipPicASCII )
  printf( "%*s\n", tsv->n[0], (char *)tsv->value );
```

_ipPicInfo

Synopsis

```
#include <ipPic.h>
void _ipPicInfo( FILE *stream,
                 ipPicDescriptor *pic,
                 ipUInt4_t flags );
```

Description

Prints information about a picDescriptor and the attached TAGs. A *flag* value of *_ipPicInfoNORMAL* will produce a long listing of all attached TAGs, a value of *_ipPicInfoSHORT* will only list the name of the attached tags.

Parameters

stream	The stream where the information should be printed to. This can be *stdout*.
pic	The picDescriptor to print.
flags	one of *_ipPicInfoNORMAL* or *_ipPicInfoSHORT*.

Return Values

None.

Example

ipPicTagName

Synopsis

```
#include <ipPic.h>
char *ipPicTypeName( ipUInt4_t type );
```

Description

Gives the name of pic type.

Parameters

type The type to convert to the ASCII name.

Return Values

A pointer to internal memory that holds the string that is the ASCII name of the type.

Example

```
ipPicDescriptor *pic;
...
printf( "This Pic-Image is of type %i byte %s\n",
        pic->bpe / 8,
         ipPicTagName(pic->type) );
```

ipPicFORALL_x

Synopsis

```
#include <ipPic.h>
ipPicFORALL_x( _ipPIC_CMD, _ipPIC_PIC, ... );
```

Description

Expands the code from *_ipPic_CMD* so that it can execute on all possible data types of *_ipPIC_PIC*.

Parameters

_ipPIC_CMD	The type to convert to the ASCII name.
_ipPIC_PIC	The pic to execute *_ipPIC_CMD* on.

Return Values

None.

Example

```
ipPicDescriptor *pic;
...
double min;
double max;

#define GET_MIN_MAX( type, pic, PIXEL_MIN, PIXEL_MAX ) \
  {                                                     \
    ipUInt4_t picsize = pic->n[0] * pic->n[1];          \
    type *pixel_ptr = (type *)pic->data;                \
    int i;                                              \
                                                        \
    PIXEL_MIN = PIXEL_MAX = *pixel_ptr++;               \
    for( i = 1; i < picsize; i++ )                      \
      {                                                 \
        type pixel = *pixel_ptr++;                      \
                                                        \
        if( pixel < PIXEL_MIN )                         \
          PIXEL_MIN = pixel;                            \
        else if( pixel > PIXEL_MAX )                    \
          PIXEL_MAX = pixel;                            \
      }                                                 \
    }

ipPicFORALL_2( GET_MIN_MAX, pic, min, max );
```

```
#undef GET_MIN_MAX

printf( "This images has pixels in the range from %d to %d\n",
        window_min,
        window_max );
```

yy

```
extern ipUInt4_t _ipPicSize(const
                            ipPicDescriptor *pic );

extern ipUInt4_t _ipPicElements( ipPicDescriptor *pic );

extern ipUInt4_t _ipPicTSVSize( ipPicTSV_t *tsv );
extern ipUInt4_t _ipPicTSVElements( ipPicTSV_t *tsv );
```

3 LIST OF FUNCTIONS

4 INDEX

5 LIST OF FIGURES