robodoc

J.v.Weert and F.N.C.Slothouber

COLLABORATORS			
	TITLE : robodoc		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	J.v.Weert and F.N.C.Slothouber	February 12, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

robodoc

1.1 robodoc.c.guide

1.2 Autodoc/robodoc.c

```
Autodoc/robodoc.c

NAME
     robodoc.c -- AutoDoc formatter. (v2.0e)

COPYRIGHT
    Maverick Software Development
    This software is public domain and can be freely redistributed as
    long as it is in it's original state.

FUNCTION
     Robodoc.C is indented as a replacement for the original AutoDocs
     program Robodoc.C will extract from a source file the procedure
     comment header and put them in a seperate documentation file.
     There are three different file outputs, ASCII
     HTML Markup langauge mainly used on the Internet
     thus the files can be viewed with a normal html viewer/ and
     finally AmigaGuide format, this format can be viewed by an
     AmigaGuide program (AMIGA ONLY).
```

```
AUTHOR
    Frans Slothouber: Source code and additional extensions
                      from version 2.0 and up.
    <slothoub@xs4all.nl>
    Jacco van Weert:  Original idea and program.
    <weertj@euronet.nl>

CREATION DATE
    20-Dec-94

MODIFICATION HISTORY
    19-Jan-95     -       v0.8:   First test beta-version
    26-Jan-95     -       v0.92:  2nd test beta-version
    2-Feb-95      -       v0.93:  Mungwall hit, solved.
                                  When item headers, are also available
                                  in body then
                                  parts are duplicated solved.
    Mar-95        -       v1.0a:  Final version
    2-Apr-95      -       v1.0b:  Bug fixes
                                  Procedure header search bug solved.
                                  Print 'created procedure' text
    20-Apr-95     -       v1.1a:  INTERNALONLY option added.
                                  Sort problem solved.
  Modifications by FNC Slothouber.
    10-May-1995   -       v2.0a:  * Program completely rewritten
                                  * added SOURCE item and LaTeX output.
                                  * added TAB converter.
    11-May-1995   -       v2.0b:  * Accepts headers that start with
                                    any sequence of non-spaces.
                                    Robodoc should work with any
                                    type of programming language now.
    12-May-1995   -       v2.0c:  * Bug fixes.
    15-May-1995   -       v2.0d:  * New Defaults file.
                                  * Added Verbose option.
    24-May-1995   -       v2.0e:  * Fixed a bug that cause the
                                    CleanUp Routine to lock up.
                                  * Improved the HTML output,
                                    should work faster now.

NOTES
    Has been succesfully compiled with SAS C, the DICE C
    and on a sun system with gcc.

BUGS
    Catch them in a jar and send them to slothoub@xs4all.nl.
```

1.3 Robodoc.c/RB_Add_Link

```
Robodoc.c/RB_Add_Link

NAME
   RB_Add_Link -- add a reference link to the list

SYNOPSIS
```

```
   void RB_Add_Link ()

FUNCTION
   Adds a reference from a xref file to the linked list
   with references.

INPUTS
   Uses the global variable line_buffer and first_link.

NOTES
   Makes sneaky use of the function RB_Insert_In_List.

SEE ALSO
   RB_Analyse_Xrefs, RB_link.

SOURCE


void RB_Add_Link ()
{
  int label_size ;
  int file_size ;
  char *cur_char ;
  char *label_name ;
  char *file_name ;
  struct RB_link *new_link ;

/* Extract Link 1 */
  for (cur_char = line_buffer;
       *cur_char && *cur_char != '\"';
       cur_char++) ;
  cur_char++ ;
  label_name = cur_char ;
  for (label_size = 0 ;
       *cur_char && *cur_char != '\"';
       cur_char++, label_size++) ;
  *cur_char = '\0' ;

/* Extract Link 2 */
  for (cur_char++;
       *cur_char && *cur_char != '\"';
       cur_char++) ;
  cur_char++ ;
  file_name = cur_char ;
  for (file_size = 0 ;
       *cur_char && *cur_char != '\"';
       cur_char++, file_size++) ;
  *cur_char = '\0' ;

  label_size += 2 ;
  file_size += 2 ;
  if (new_link = (struct RB_link *)malloc(sizeof(struct RB_link)))
  {
    if ((new_link->label_name = (char *)malloc(label_size * sizeof(char))) &&
        (new_link->file_name = (char *)malloc(file_size * sizeof(char))))
    {
      strcpy(new_link->label_name, label_name) ;
```

```
      strcpy(new_link->file_name, file_name) ;

/* Sneaky Code so I don't have to write a second insertion routine */
      RB_Insert_In_List ((struct RB_header **)&first_link,
                         (struct RB_header *)new_link) ;
    }
    else
    {
      RB_Panic ("out of memory","RB_link structure") ;
    }
  }
  else
  {
    RB_Panic ("out of memory","RB_link structure") ;
  }
}

/*********/
```

1.4 Robodoc.c/RB_Analyse_Arguments

```
Robodoc.c/RB_Analyse_Arguments

NAME
    RB_Analyse_Arguments --

SYNOPSIS
    RB_Analyse_Arguments (argc, argv, file_with_xrefs,
                          output_file_for_xrefs)
    RB_Analyse_Arguments (int, char **, char **, char **)

FUNCTION
    Get and parse the arguments.

SEE ALSO

SOURCE


void RB_Analyse_Arguments (int argc, char **argv,
                           char **file_with_xrefs,
                           char **output_file_for_xrefs)
{
  char **parameter ;
  int  parameter_nr ;

  for (parameter_nr = argc - 3, parameter = argv + 3 ;
       parameter_nr > 0;
       parameter++, parameter_nr--)
  {
    if      (!strcmp (*parameter, "HTML") ||
             !strcmp (*parameter, "html")) output_mode = HTML ;
    else if (!strcmp (*parameter, "GUIDE") ||
             !strcmp (*parameter, "guide")) output_mode = AMIGAGUIDE ;
    else if (!strcmp (*parameter, "LATEX") ||
```

```
         !strcmp (*parameter, "latex")) output_mode = LATEX ;
else if (!strcmp (*parameter, "ASCII") ||
         !strcmp (*parameter, "ascii")) output_mode = ASCII ;
else if (!strcmp (*parameter, "SORT") ||
         !strcmp (*parameter, "sort") ||
         !strcmp (*parameter, "-s"))    course_of_action |= DO_SORT ;
else if (!strcmp (*parameter, "INTERNAL") ||
         !strcmp (*parameter, "internal") ||
         !strcmp (*parameter, "-i"))    course_of_action |=  ↩
            DO_INCLUDE_INTERNAL ;
else if (!strcmp (*parameter, "INTERNALONLY") ||
         !strcmp (*parameter, "internalonly") ||
         !strcmp (*parameter, "-io"))   course_of_action |= DO_INTERNAL_ONLY ;
else if (!strcmp (*parameter, "TOC") ||
         !strcmp (*parameter, "toc") ||
         !strcmp (*parameter, "-t"))    course_of_action |= DO_TOC ;
else if (!strcmp (*parameter, "-v"))    course_of_action |= DO_TELL ;
else if (!strcmp (*parameter, "XREF") ||
         !strcmp (*parameter, "xref") ||
         !strcmp (*parameter, "-x"))
{
  if (parameter_nr > 1)
  {
    course_of_action |= DO_USE_XREFS ;
    *file_with_xrefs = *(parameter+1) ;
    parameter++ ; parameter_nr-- ;
  }
  else
  {
    RB_Panic ("you must specify a xref file with the XREF option.","") ;
  }
}
else if (!strcmp (*parameter, "TABSIZE") ||
         !strcmp (*parameter, "tabsize") ||
         !strcmp (*parameter, "-ts"))
{
  if (parameter_nr > 1)
  {
    tab_size = atoi (*(parameter+1)) ;
    parameter++ ; parameter_nr-- ;
  }
  else
  {
    RB_Panic ("you must specify the number of spaces with the TABSIZE option", ↩
        "") ;
  }
}
else if (!strcmp (*parameter, "GENXREF") ||
         !strcmp (*parameter, "genxref") ||
         !strcmp (*parameter, "-g"))
{
  if (parameter_nr > 1)
  {
    course_of_action |=  DO_MAKE_XREFS ;
    course_of_action &= ~DO_MAKE_DOCUMENT ;
    *output_file_for_xrefs = *(parameter+1) ;
    printf ("%s\n", output_file_for_xrefs) ;
```

```
        parameter++ ; parameter_nr-- ;
      }
      else
      {
        RB_Panic ("you must specify a a xref file with the GENXREF option.", "") ;
      }
    }
    else
    {
      RB_Panic ("unknown option", *parameter) ;
    }
  }

  if ((course_of_action & DO_USE_XREFS) &&
      (output_mode == ASCII || output_mode == LATEX))
  {
    printf ("Robodoc: you can not use xrefs with when you generate documentation\n ↩
        ") ;
    printf ("         in ASCII of LaTeX, discarding switch\n") ;
    course_of_action &= ~DO_USE_XREFS ;
  }
}
```

1.5 Robodoc.c/RB_Analyse_Defaults_File

```
Robodoc.c/RB_Analyse_Defaults_File

NAME
    RB_Analyse_Defaults_file -- read default from defaults file

SYNOPSIS
    RB_Analyse_Defaults_file

FUNCTION
    Read the default vaules from the default file.

SEE ALSO

SOURCE


void RB_Analyse_Defaults_File ()
{
  FILE *defaults_file ;

  if (defaults_file = fopen("robodoc.defaults","r"))
  {
    for (;
         !(feof(defaults_file)) ;
        )
    {
      char *cur_char ;
      *line_buffer = '\0' ;
      fgets (line_buffer, MAX_LINE_LEN, defaults_file) ;
```

```
for (cur_char = line_buffer ;
     *cur_char && !isspace(*cur_char) ;
     cur_char++) ;
if (*cur_char && *cur_char != '\n')
{
  char *values = cur_char+1 ;
  int item_type = NO_ITEM ;
  *cur_char = '\0' ;
  if      (!strcmp("NAME",line_buffer))      item_type = NAME_ITEM ;
  else if (!strcmp("COPYRIGHT",line_buffer)) item_type = COPYRIGHT_ITEM ;
  else if (!strcmp("SYNOPSIS",line_buffer))  item_type = SYNOPSIS_ITEM ;
  else if (!strcmp("FUNCTION",line_buffer))  item_type = FUNCTION_ITEM ;
  else if (!strcmp("AUTHOR",line_buffer))    item_type = AUTHOR_ITEM ;
  else if (!strcmp("CREATION_DATE",line_buffer)) item_type =  ↩
     CREATION_DATE_ITEM ;
  else if (!strcmp("MODIFICATION_HISTORY",line_buffer)) item_type =  ↩
     MODIFICATION_HISTORY_ITEM ;
  else if (!strcmp("INPUTS",line_buffer))    item_type = INPUTS_ITEM ;
  else if (!strcmp("RESULT",line_buffer))    item_type = RESULT_ITEM ;
  else if (!strcmp("EXAMPLE",line_buffer))   item_type = EXAMPLE_ITEM ;
  else if (!strcmp("NOTES",line_buffer))     item_type = NOTES_ITEM ;
  else if (!strcmp("BUGS",line_buffer))      item_type = BUGS_ITEM ;
  else if (!strcmp("SEE_ALSO",line_buffer))  item_type = SEE_ALSO_ITEM ;
  else if (!strcmp("SOURCE",line_buffer))    item_type = SOURCE_ITEM ;
  if (item_type != NO_ITEM)
  {
    cur_char = values ;
    for (; *cur_char && isspace(*cur_char) ; cur_char++) ;
    for (;
         *cur_char;
        )
    {
      for (values = cur_char ;
           *cur_char && !isspace(*cur_char) ;
           cur_char++) ;
      if (*cur_char)
      {
        cur_char = '\0' ;
        if      (!strcmp("NORMAL", values))     item_attributes[item_type] = ↩
            0 ;
        else if (!strcmp("BOLD",   values))     item_attributes[item_type]  ↩
            |= TEXT_BODY_BOLD ;
        else if (!strcmp("ITALICS", values))    item_attributes[item_type]  ↩
            |= TEXT_BODY_ITALICS ;
        else if (!strcmp("SHINE", values))      item_attributes[item_type]  ↩
            |= TEXT_BODY_SHINE ;
        else if (!strcmp("SMALL_FONT", values)) item_attributes[item_type]  ↩
            |= TEXT_BODY_SMALL_FONT ;
        else if (!strcmp("LARGE_FONT", values)) item_attributes[item_type]  ↩
            |= TEXT_BODY_LARGE_FONT ;
        else if (!strcmp("NON_PROP", values))   item_attributes[item_type]  ↩
            |= TEXT_BODY_NON_PROP ;
      }
    for (; *cur_char && isspace(*cur_char) ; cur_char++) ;
    }
  }
}
```

```
    }
    fclose (defaults_file) ;
  }
  else
  {
    printf ("Robodoc: Warning: The robodoc.defaults file was not found.\n" ) ;
    printf ("         You should really use one.\n" ) ;
  }
}
```

1.6 Robodoc.c/RB_Analyse_Document

```
Robodoc.c/RB_Analyse_Document

NAME
   RB_Analyse_Document -- scan document for headers and store them

SYNOPSIS
   RB_Analyse_Document (document)
   RB_Analyse_Document (FILE *)

FUNCTION
   Searches the document for headers. Stores information about
   any headers that are found in a linked list. Information
   that is stored includes, the name of the header, its version
   number, and its contents.

INPUTS
   document - a pointer to a file with the document to be
              analysed
   the gobal buffer line_buffer.

RESULT
   1)   A linked list pointed to by the global variable
        first_header that contains information about each
        header.

NOTE
   Using fseek and ftell cause on the sun unix system that I am
   on the gcc doesn't know fgetpos and fsetpos.

SEE ALSO
   RB_Find_Marker

SOURCE


void RB_Analyse_Document (FILE *document)
{
  int   header_type ;

  for (;
       (header_type = RB_Find_Marker (document)) != NO_HEADER;
      )
```

```
  {
    struct RB_header *new_header ;

    if (new_header = (struct RB_header *)malloc (sizeof (struct RB_header)))
    {
      long cur_file_pos ;

      (*new_header).name          = NULL ;
      (*new_header).function_name = NULL ;
      (*new_header).contents      = NULL ;
      RB_Insert_In_List (&first_header, new_header) ;

      (*new_header).type          = header_type ;
      (*new_header).name          = RB_Find_Header_Name () ;
      RB_Say ("Found header: ", new_header->name) ;
      (*new_header).function_name = RB_Find_Function_Name((*new_header).name) ;
      cur_file_pos = (long)ftell (document) ;
      if (RB_Find_End_Marker (document,&(*new_header).size))
      {
        fseek (document, cur_file_pos ,0) ;
        if ((*new_header).contents = (char *)malloc((*new_header).size*sizeof(char ↩
            )))
        {
          fread ((*new_header).contents,(*new_header).size+1,sizeof(char),document ↩
              ) ;
          *((*new_header).contents+(*new_header).size) = '\0' ;
        }
        else RB_Panic ("no memory!","Alloc Header Contents") ;
      }
      else RB_Panic ("found a header with no end marker","") ;
    }
    else RB_Panic ("no memory","Alloc RB_header structure") ;
  }
}
```

1.7 Robodoc.c/RB_Analyse_Xrefs

```
Robodoc.c/RB_Analyse_Xrefs

NAME
   RB_Analyse_Xrefs -- scan the xref files.

SYNOPSIS
   RB_Analyse_Xrefs (xreffiles_file)
   RB_Analyse_Xrefs (FILE *)

FUNCTION
   Scan the file xreffiles_file. This file contains the
   names of one or more xref files. All the references in the
   files are scaned and stored in a link list of the type
   RB_link.
   These xref files can be generated with robodoc.

INPUTS
```

```
   xreffiles_file - a file pointer to the file with xref file
   names.

RESULTS

BUGS
   Might fail if there are syntax errors in one of the xref
   files.

SEE ALSO
   RB_Generate_xrefs, RB_Add_Link

SOURCE


void RB_Analyse_Xrefs (FILE *xreffiles_file)
{
  for (;
       !(feof(xreffiles_file));
      )
  {
    fgets (line_buffer, MAX_LINE_LEN, xreffiles_file) ;
    if (!feof(xreffiles_file))
    {
      char *cur_char ;

      for (cur_char = line_buffer;
           *cur_char && *cur_char != '\n';
           cur_char++) ;
      if (*cur_char == '\n') *cur_char = '\0' ;
      if (xref_file = fopen(line_buffer,"r"))
      {
        int xrefs_found = FALSE ;
        int end_of_xrefs = FALSE ;

        for (;
             !(feof(xref_file)) && !xrefs_found ;
            )
        {
          fgets (line_buffer, MAX_LINE_LEN, xref_file) ;
          if (!feof(xref_file))
          {
            if (!strncmp("XREF:",line_buffer,5)) xrefs_found = TRUE ;
          }
        }

        for (;
             !(feof(xref_file)) && !end_of_xrefs ;
            )
        {
          fgets (line_buffer, MAX_LINE_LEN, xref_file) ;
          if (!feof(xref_file))
          {
            for (cur_char = line_buffer;
                 *cur_char && *cur_char != '\"';
                 cur_char++) ;
            if (*cur_char == '\"') RB_Add_Link () ;
```

```
            else                    end_of_xrefs = TRUE ;
          }
        }
        fclose (xref_file) ;
        xref_file = NULL ;
      }
      else
      {
        RB_Panic ("could not open xref file", line_buffer) ;
      }
    }
  }
}

/*********/
```

1.8 Robodoc.c/RB_Close_The_Shop

```
Robodoc.c/RB_Close_The_Shop

NAME
   RB_Close_The_Shop -- free resources.

SYNOPSIS
   void RB_Close_The_Shop ()

FUNCTION
   Frees all resources used by robodoc.
```

1.9 Robodoc.c/RB_Find_End_Marker

```
Robodoc.c/RB_Find_End_Marker

NAME
   RB_Find_End_Marker -- Search for end marker in document.

SYNOPSIS
   result = RB_Find_End_Marker (document, total_size)
        int RB_Find_End_Marker (FILE *,int *)

FUNCTION


INPUTS
   document   - pointer to the file to be searched.
   total_size - pointer to an integer that will store the number of
                characters that have been searched till the end marker
                was found.
   the gobal buffer line_buffer.

RESULT
   TRUE   - end marker was found
```

```
   FALSE  - no end marker was found

SEE ALSO
   RB_Find_Marker

SOURCE


int RB_Find_End_Marker (FILE *document,int *total_size)
{
  int marker_found ;

  for (marker_found = FALSE, *total_size = 0;
       !(feof(document)) && !marker_found;
      )
  {
    int line_length ;
    char *cur_char ;

    *line_buffer = '\0' ;
    fgets (line_buffer, MAX_LINE_LEN, document) ;
    line_length = strlen(line_buffer) ;
    *total_size += line_length ;
    for (cur_char = line_buffer + 1;
         *cur_char && *cur_char != '*' && !isspace(*cur_char) ;
         cur_char++) ;
    if (isspace(*cur_char))
    {
      for (;
           *cur_char && isspace(*cur_char) ;
           cur_char++) ;
    }
    if (*cur_char)
    {
      if (!(strncmp(cur_char,"***",3))) marker_found = TRUE ;
    }
  }
  return marker_found ;
}
```

1.10 Robodoc.c/RB_Find_Function_Name

```
Robodoc.c/RB_Find_Function_Name

NAME
   RB_Find_Function_Name -- search for function name

SYNOPSIS
   result = RB_Find_Function_Name (header_name)
      char *RB_Find_Function_Name (char *)

FUNCTION
   Searches the line buffer for the header name.
   It assumes that the header name is constructed is follows
```

```
     <hiarachy name>/<function name>
   It allocates an string and copies the name to this string.

INPUTS
   a pointer to the header name.

RESULT
   pointer to the string with the name of the function
   terminated with a '\0'.
   NULL if no header name was found.

SEE ALSO
   RB_Find_Function_Name

SOURCE


char *RB_Find_Function_Name (char *header_name)
{
  char *function_name ;

  if (header_name)
  {
    for (function_name = header_name ;
         *function_name && *function_name != '/';
         function_name++) ;
    if (!*function_name || !*(function_name+1))
    {
      printf ("Robodoc: Warning function name missing from header name\n") ;
      printf ("  in %s\n", header_name) ;
      function_name = NULL ;
    }
    else
    {
      function_name++ ;
    }
  }
  else
  {
    function_name = NULL ;
  }
  return function_name ;
}
```

1.11 Robodoc.c/RB_Find_Header_Name

```
Robodoc.c/RB_Find_Header_Name

NAME
   RB_Find_Header_Name -- search for header name

SYNOPSIS
   result = RB_Find_Header_Name ()
      char *RB_Find_Header_Name ()
```

```
FUNCTION
   Searches the line buffer for the header name.
   It assumes that the header name follows after the
   header marker, seperated by one or more spaces, and terminated
   by one or more spaces or a '\n'.
   It allocates an array of chars and copies the name to this array.

INPUTS
   the gobal buffer line_buffer.

RESULT
   pointer to the allocated array of chars that contains the name,
   terminated with a '\0'.
   NULL if no header name was found.

SEE ALSO
   RB_Find_Function_Name

SOURCE


char *RB_Find_Header_Name ()
{
  char *cur_char ;
  char *header_name ;

  for (cur_char = line_buffer;
       *cur_char != '*';
       cur_char++) ;
  for (;
       *cur_char && !isspace(*cur_char) ;
       cur_char++) ;
  for (;
       *cur_char && isspace(*cur_char);
       cur_char++) ;
  if (!(*cur_char))
  {
    printf ("Robodoc: Warning, found header marker but no name\n") ;
    printf ("Line:  %s",line_buffer) ;
    header_name = NULL ;
  }
  else
  {
    int name_length ;

    for (name_length = 0 ;
         *cur_char && !isspace(*cur_char);
         cur_char++, name_length++) ;
    if (header_name = (char *)malloc((name_length+2)*sizeof(char)))
    {
      strncpy(header_name,cur_char-name_length,name_length) ;
      *(header_name+name_length) = '\0' ;
    }
    else RB_Panic ("no memory!","Alloc Header Name") ;
  }
  return header_name ;
```

```
}
```

1.12 Robodoc.c/RB_Find_Item

```
Robodoc.c/RB_Find_Item

NAME
    RB_Find_Item -- find item in header contents.

SYNOPSIS
    item_type = RB_Find_Item (next_line,item_line)

            int RB_Find_Item (char **,  char **)

FUNCTION
    Searches the header contents line by line, looking
    for an item Indicator.

INPUTS
    next_line  - pointer to a pointer that points to line
                 at which the search will start.

RESULTS
    next_line  - pointer to a pointer that points to begin of the line
                 after the line the item was found on.
    item_line  - pointer to a pointer that points to the line the item
                 was found on.
    item_type  - one of possible items indicators:
      NAME_ITEM, COPYRIGHT_ITEM, SYNOPSIS_ITEM,
      FUNCTION_ITEM, AUTHOR_ITEM, CREATION_DATE_ITEM,
      MODIFICATION_HISTORY_ITEM, INPUTS_ITEM, RESULT_ITEM, EXAMPLE_ITEM,
      NOTES_ITEM, BUGS_ITEM, SEE_ALSO_ITEM, OTHER_ITEM
      or NO_ITEM when no item could be found.

SOURCE



int RB_Find_Item (char **next_line,char **item_line)
{
  char *cur_char = *next_line ;
  int item_type ;

  for(item_type = NO_ITEM;
      *cur_char && (item_type == NO_ITEM);
     )
  {
    *item_line = cur_char ;

/* Skip Comment marker */
    for (;
         !isspace(*cur_char) && *cur_char && *cur_char != '\n';
         cur_char++) ;
```

```
    for (;
         isspace(*cur_char) && *cur_char && *cur_char != '\n';
         cur_char++) ;
    if (isupper(*cur_char))
    {
      char *item_begin = cur_char ;
      char *item_end ;

      for (; isupper(*cur_char) && *cur_char; cur_char++) ;
      item_end = cur_char ;

      if (isspace(*cur_char) && *cur_char)
      {
        for (; isspace(*cur_char) && *cur_char != '\n'; cur_char++) ;

/* Item consists of two words ? */
        if (isupper(*cur_char) && *cur_char)
        {
          for (; isupper(*cur_char) && *cur_char; cur_char++) ;
          item_end = cur_char ;
          for (; isspace(*cur_char) && *cur_char != '\n'; cur_char++) ;
        }

        if (*cur_char == '\n')
        {
          char old_char = *item_end ;

          *item_end = '\0' ;
          if      (!strcmp (item_begin, "NAME"))     item_type = NAME_ITEM ;
          else if (!strcmp (item_begin, "FUNCTION")) item_type = FUNCTION_ITEM ;
          else if (!strcmp (item_begin, "SYNOPSIS")) item_type = SYNOPSIS_ITEM ;
          else if (!strcmp (item_begin, "INPUTS"))   item_type = INPUTS_ITEM ;
          else if (!strcmp (item_begin, "RESULT"))   item_type = RESULT_ITEM ;
          else if (!strcmp (item_begin, "SEE ALSO")) item_type = SEE_ALSO_ITEM ;
          else if (!strcmp (item_begin, "EXAMPLE"))  item_type = EXAMPLE_ITEM ;
          else if (!strcmp (item_begin, "NOTES"))    item_type = NOTES_ITEM ;
          else if (!strcmp (item_begin, "BUGS"))     item_type = BUGS_ITEM ;
          else if (!strcmp (item_begin, "SOURCE"))   item_type = SOURCE_ITEM ;
          else if (!strcmp (item_begin, "AUTHOR"))   item_type = AUTHOR_ITEM ;
          else if (!strcmp (item_begin, "COPYRIGHT")) item_type = COPYRIGHT_ITEM ;
          else if (!strcmp (item_begin, "CREATION DATE")) item_type =  ↩
             CREATION_DATE_ITEM ;
          else if (!strcmp (item_begin, "MODIFICATION HISTORY")) item_type =  ↩
             MODIFICATION_HISTORY_ITEM ;
          else item_type = OTHER_ITEM ;
          *item_end = old_char ;
          cur_char++ ;
        }
      }
    }
    if (item_type == NO_ITEM)
    {
      for (; *cur_char && (*cur_char != '\n'); cur_char++) ;
      if (*cur_char) cur_char++ ;
    }
  }
  *next_line = cur_char ;
```

```
  return item_type ;
}

/******/
```

1.13 Robodoc.c/RB_Find_Link

```
Robodoc.c/RB_Find_Link

NAME
    RB_Find_Link -- try to match word with a link

SYNOPSIS
    result = RB_Find_Link (word_begin, label_name, file_name)
    int      RB_Find_Link (char *,     char **,    char **)

FUNCTION
    Searches for the given word in the list of links.

INPUTS
    word_begin  - pointer to a word (a string).
    label_name  - pointer to a pointer to a string
    file_name   - pointer to a pointer to a string

RESULTS
    TRUE or FALSE.

BUGS

SOURCE


int RB_Find_Link (char *word_begin,
                  char **label_name,
                  char **file_name)
{
  struct RB_link *cur_link ;
  struct RB_link *matching_link ;
  struct RB_header *cur_header ;
  struct RB_header *matching_header ;

  int match_found = FALSE ;

  for (cur_header = first_header ;
       cur_header && !match_found ;
       cur_header = cur_header->next_header)
  {
    char *cur_char1, *cur_char2 ;

    matching_header = cur_header ;
    for (cur_char1 = word_begin,
         cur_char2 = cur_header->function_name,
         match_found = TRUE ;
         *cur_char1 &&
         *cur_char2 &&
```

```
          (isalpha(*cur_char1) || *cur_char1 == '_') &&
          match_found ;
          cur_char1++, cur_char2++)
    {
      if (*cur_char1 != *cur_char2) match_found = FALSE ;
    }
    if (isalpha(*cur_char1) || *cur_char1 == '_' ||
        *cur_char2) match_found = FALSE ;
  }
  if (match_found)
  {
    *label_name = matching_header->function_name ;
    *file_name  = NULL ;
  }
  else
  {
    for (cur_link = first_link ;
         cur_link && !match_found ;
         cur_link = cur_link->next_link)
    {
      char *cur_char1, *cur_char2 ;

      matching_link = cur_link ;
      for (cur_char1 = word_begin,
           cur_char2 = cur_link->label_name,
           match_found = TRUE ;
           *cur_char1 &&
           *cur_char2 &&
           (isalpha(*cur_char1) || *cur_char1 == '_') &&
           match_found ;
           cur_char1++, cur_char2++)
      {
        if (*cur_char1 != *cur_char2) match_found = FALSE ;
      }
      if (isalpha(*cur_char1) || *cur_char1 == '_' ||
          *cur_char2) match_found = FALSE ;
    }
    if (match_found)
    {
      *label_name = matching_link->label_name ;
      *file_name  = matching_link->file_name ;
    }
  }
  return match_found ;
}
```

1.14 Robodoc.c/RB_Find_Marker

```
Robodoc.c/RB_Find_Marker

NAME
   RB_Find_Marker -- Search for header marker in document.

SYNOPSIS
```

```
   header_type = RB_Find_Marker (document)
             int RB_Find_Marker (FILE *)

FUNCTION
   Read document file line by line, and search each line for the
   following combinations of characters:
     "x****h*" = MAIN Header
     "x******" = Generic header
     "x****i*" = Internal header
     Where 'x' is any sequence of non space characters.

INPUTS
   document - pointer to the file to be searched.
   the gobal buffer line_buffer.

RESULT
   header type
   can be:
   (1) NO_HEADER - no header found, end of file reached
   (2) MAIN_HEADER
   (3) GENERIC_HEADER
   (4) INTERNAL_HEADER

SEE ALSO
   RB_Find_End_Marker

SOURCE


int RB_Find_Marker (FILE *document)
{
  int header_type ;

  for (header_type = NO_HEADER;
       !(feof(document)) && (header_type == NO_HEADER); )
  {
    *line_buffer = '\0' ;
    fgets (line_buffer, MAX_LINE_LEN, document) ;
    if (strlen(line_buffer) > 7)
    {
      char *cur_char ;

      for (cur_char = line_buffer + 1;
           *cur_char && *cur_char != '*' && !isspace(*cur_char) ;
           cur_char++) ;
      if (isspace(*cur_char))
      {
        for (;
             *cur_char && isspace(*cur_char) ;
             cur_char++) ;
      }
      if (strlen(cur_char) >= 6)
      {
        if (*(cur_char+5) == '*')
        {
          if (!(strncmp(cur_char,"****",4)))
          {
```

```
            switch (*(cur_char+4))
            {
              case 'h' :  header_type = MAIN_HEADER     ; break ;
              case '*' :  header_type = GENERIC_HEADER  ; break ;
              case 'i' :  header_type = INTERNAL_HEADER ; break ;
              default  :  printf ("Robodoc: Warning, Undefined headertype, using  ↩
                  GENERIC\n") ;
                          header_type = GENERIC_HEADER  ; break ;
            }
          }
        }
      }
    }
  }
  return header_type ;
}
```

1.15 Robodoc.c/RB_Gen_Doc_End

```
Robodoc.c/RB_Gen_Doc_End

NAME
     RB_Gen_Doc_End -- generate document trailer.

SYNOPSIS
     RB_Gen_Doc_End (dest_doc, name)

     RB_Gen_Doc_End (FILE *, char *)

FUNCTION
     Generates for depending on the output_mode the text that
     will be at the end of a document.

INPUTS
     dest_doc - pointer to the file to which the output will
                be written.
     name     - the name of this file.
     output_mode - global variable that indicates the output
                   mode.

NOTES
     Doesn't do anything with its arguments, but that might
     change in the future.

BUGS
     RB_Gen_Doc_Start
```

1.16 Robodoc.c/RB_Gen_Doc_Start

```
Robodoc.c/RB_Gen_Doc_Start

NAME
     RB_Gen_Doc_Start -- Generate document header.

SYNOPSIS
     RB_Gen_Doc_Start (dest_doc, name)

     RB_Gen_Doc_Start (FILE *, char *)

FUNCTION
     Generates for depending on the output_mode the text that
     will be at the start of a document.
     Including the table of contents.

INPUTS
     dest_doc - pointer to the file to which the output will
                be written.
     name     - the name of this file.
     output_mode - global variable that indicates the output
                   mode.

SEE ALSO
     RB_Gen_Doc_End
```

1.17 Robodoc.c/RB_Gen_Header_End

```
Robodoc.c/RB_Gen_Header_End

NAME
     RB_Gen_Header_End --

SYNOPSIS
   void RB_Gen_Header_End (dest_doc, cur_header)

   void RB_Gen_Header_End (FILE *, struct RB_header *)

FUNCTION
   Generates for depending on the output_mode the text that
   will be at the end of a header.

INPUTS
   dest_doc - pointer to the file to which the output will
              be written.
   cur_header - pointer to a RB_header structure.

SEE ALSO
   RB_Gen_Header_Start
```

1.18 Robodoc.c/RB_Gen_Header_Start

```
Robodoc.c/RB_Gen_Header_Start

NAME
   RB_Gen_Header_Start -- generate header start text.

SYNOPSIS
  void RB_Gen_Header_Start (dest_doc, cur_header)

  void RB_Gen_Header_Start (FILE *, struct RB_header *)

FUNCTION
   Generates depending on the output_mode the text that
   will be at the end of each header.

INPUTS
   dest_doc - pointer to the file to which the output will
              be written.
   cur_header - pointer to a RB_header structure.

SEE ALSO
   RB_Gen_Header_End
```

1.19 Robodoc.c/RB_Generate_Documentation

```
Robodoc.c/RB_Generate_Documentation

NAME
   RB_Generate_Documentation --

SYNOPSIS
   RB_Generate_Documentation (dest_doc, name)

   RB_Generate_Documentation (FILE *, char *)

FUNCTION
   Generates the autodoc documentation from the list of
   function headers that has been created by
   RB_Analyse_Document.

INPUTS
   dest_doc - Pointer to the file to which the output will be written.
   dest_name     - The name of this file.

BUGS
   There might be plenty.

SEE ALSO
   RB_Gen_Doc_Start, RB_Gen_Doc_End, RB_Gen_Header_Start,
   RB_Gen_Header_End, RB_Generate_Header_Name,
   RB_Generate_Item_Name, RB_Generate_Item_Doc,
   RB_Generate_Item_Body.

SOURCE
```

```
void RB_Generate_Documentation (FILE *dest_doc, char *dest_name)
{
  struct RB_header *cur_header ;

  RB_Gen_Doc_Start(dest_doc, dest_name) ;

  for (cur_header = first_header;
       cur_header;
       cur_header = cur_header->next_header
      )
  {
    int  item_type ;
    char *next_line ;
    char *item_line ;
    int header_type ;

    header_type = cur_header->type ;
    if ( ((header_type == INTERNAL_HEADER) &&
          ((course_of_action & DO_INCLUDE_INTERNAL) ||
           (course_of_action & DO_INTERNAL_ONLY))) ||
         ((header_type != INTERNAL_HEADER) &&
          !(course_of_action & DO_INTERNAL_ONLY)))
    {
      RB_Say ("generating documentation for", cur_header->name) ;
      RB_Gen_Header_Start (dest_doc, cur_header) ;
      RB_Generate_Header_Name (dest_doc, cur_header->name) ;
      next_line = cur_header->contents ;
      item_type = RB_Find_Item(&next_line,&item_line) ;
      if (item_type != NO_ITEM)
      {
        int old_item_type ;
        char *old_next_line, *old_item_line ;
        for (;
             item_type != NO_ITEM ;)
        {
           RB_Generate_Item_Name (dest_doc, item_line,next_line, item_type) ;
           old_next_line = next_line ;
           old_item_line = item_line ;
           old_item_type = item_type ;
           item_type = RB_Find_Item(&next_line,&item_line) ;
           RB_Generate_Item_Doc(dest_doc, dest_name, old_next_line,
                                item_line, old_item_type) ;
        }
      }
      else
      {
        printf ("Robodoc: Warning, header has no items\n") ;
      }
      RB_Gen_Header_End(dest_doc, cur_header) ;
    }
  }
  RB_Gen_Doc_End (dest_doc, dest_name) ;
}
```

1.20 Robodoc.c/RB_Generate_Header_Name

```
Robodoc.c/RB_Generate_Header_Name

NAME
    RB_Generate_Header_Name --

SYNOPSIS
    RB_Generate_Header_Name (dest_doc, name)

    RB_Generate_Header_Name (FILE *, char *)

FUNCTION

INPUTS
   dest_doc - pointer to the file to which the output will
              be written.
   name - pointer to the header name.

RESULTS

BUGS

SEE ALSO
```

1.21 Robodoc.c/RB_Generate_Item_Body

```
Robodoc.c/RB_Generate_Item_Body

NAME
   RB_Generate_Item_Body --

SYNOPSIS

FUNCTION

INPUTS
     dest_doc  NMNM - pointer to the file to which the output will
                 be written.
     dest_name    - the name of this file.
     begin_of_item -
     end_of_item   -
     item_type -

RESULTS

BUGS

SEE ALSO
```

1.22 Robodoc.c/RB_Generate_Item_Doc

```
Robodoc.c/RB_Generate_Item_Doc

NAME
     RB_Generate_Item_Doc --

SYNOPSIS

FUNCTION

INPUTS

RESULTS

BUGS
```

1.23 Robodoc.c/RB_Generate_Item_Name

```
Robodoc.c/RB_Generate_Item_Name

NAME
     RB_Generate_Item_Name --

SYNOPSIS

FUNCTION

INPUTS

RESULTS

BUGS
```

1.24 Robodoc.c/RB_Generate_xrefs

```
Robodoc.c/RB_Generate_xrefs

NAME
   RB_Generate_xrefs --

SYNOPSIS
   RB_Generate_xrefs (dest_doc, source_name, dest_name)

   RB_Generate_xrefs (FILE *, char *, char *)

FUNCTION
   Generates a xref file for the document that has been
   analysed by robodoc.
```

```
INPUTS
   dest_doc    - pointer to the file to which the xrefs will be
                 written.
   source_name - pointer to the name of the document that has
                 been analysed by robodoc
   dest_name   - pointer to the name of the document robodoc will
                 write the documentation to.
   first_heade - global variable, the list with function
                 headers.

SEE ALSO


SOURCE


void  RB_Generate_xrefs (FILE *dest_doc, char *source_name, char *dest_name)
{
  struct RB_header *cur_header ;

  fprintf (dest_doc, "/* XREF-File generated by ROBODoc v2.0 */\n") ;
  fprintf (dest_doc, "\nXREF:\n") ;
  fprintf (dest_doc, " \"%s\" \"%s\" 0 0\n",source_name, dest_name) ;
  for (cur_header = first_header;
       cur_header;
       cur_header = cur_header->next_header
      )
  {
    if (cur_header->function_name)
       fprintf (dest_doc, " \"%s\" \"%s\" 0 0\n",
                cur_header->function_name, dest_name) ;
  }
  fprintf (dest_doc, "\n/* End of XREF-File /*\n") ;
}

/*********/
```

1.25 Robodoc.c/RB_Insert_In_List

```
Robodoc.c/RB_Insert_In_List

NAME
   RB_Insert_In_List -- Insert a header in a list.

SYNOPSIS
   RB_Insert_In_List (anchor,new_header)

   RB_Insert_In_List (struct RB_header **, struct RB_header *)

FUNCTION
   Insert a node in a doubly linked list.

INPUTS
   anchor     - pointer to the first node in the list.
```

```
   new_header - node to be inserted.
```

1.26 Robodoc.c/RB_Panic

```
Robodoc.c/RB_Panic

NAME
   RB_Panic -- free resources and shut down

SYNOPSIS
   RB_Panic (cause, add_info)
   RB_Panic (char *, char *)

FUNCTION
   Print error message.
   Frees all resources used by robodoc.
   Terminates program

INPUTS
   cause    - pointer to a string with the cause of the error.
   add_info - pointer to a string with additional info.

SEE ALSO
```

1.27 Robodoc.c/RB_Print_Header_List

```
Robodoc.c/RB_Print_Header_List

NAME
   RB_Print_Header_List -- Display the complete header list

SYNOPSIS
   RB_Print_Header_List (cur_header)

FUNCTION

NOTE
   For debug purposes.
```

1.28 Robodoc.c/RB_Print_Link_List

```
Robodoc.c/RB_Print_Link_List

NAME
   RB_Print_Link_List -- Display the complete Link list

SYNOPSIS
   RB_Print_Link_List (cur_Link)
   RB_Print_Link_List (struct RB_link *)
```

```
FUNCTION
   Print all information in the list of links.

NOTE
   For debug purposes.
```

1.29 Robodoc.c/RB_Remove_From_List

```
Robodoc.c/RB_Remove_From_List

NAME
   RB_Remove_From_List -- remove a header from a list.

SYNOPSIS
   RB_Remove_From_List (anchor, old_header)
   RB_Remove_From_List (struct RB_header **, struct RB_HEADER **)

FUNCTION


INPUTS


NOTE
```

1.30 Robodoc.c/RB_Say

```
Robodoc.c/RB_Say

NAME
   RB_Say -- Say what we're doing.

SYNOPSIS

FUNCTION
```

1.31 Robodoc.c/RB_Slow_Sort

```
Robodoc.c/RB_Slow_Sort

NAME
   RB_Slow_Sort -- sort list of headers alphabetically

SYNOPSIS
   RB_Slow_Sort ()

FUNCTION
   Sorts the list of headers according to the header name
```

```
   in alphabetically fashion.

NOTE
   This isn't a particularly speedy way of sorting.

SOURCE


void RB_Slow_Sort ()
{
  struct RB_header *cur_header ;
  struct RB_header *unsorted_headers ;
  struct RB_header *bigger_header ;

  unsorted_headers = first_header ;
  first_header = NULL ;

  for (;
       unsorted_headers->next_header;
      )
  {
    for (bigger_header = unsorted_headers,
         cur_header = bigger_header->next_header;
         cur_header;
         cur_header = cur_header->next_header)
    {
      if (strcmp(cur_header->name, bigger_header->name) > 0) bigger_header =  ↩
          cur_header ;
    }
    RB_Remove_From_List (&unsorted_headers, bigger_header) ;
    RB_Insert_In_List (&first_header, bigger_header) ;
  }
  RB_Insert_In_List (&first_header, unsorted_headers) ;
}
```

1.32 Robodoc.c/_main

```
Robodoc.c/_main

NAME
    _main -- Entry point of robodoc.c

SYNOPSIS
    _main (int argc, char **argv)

FUNCTION
    Get and parse the arguments.
    Analyse document.

SOURCE


int main (int argc, char **argv)
{
```

```
char *file_with_xrefs ;
char *output_file_for_xrefs ;

if ((argc < 3) || (*argv[1] == '?'))
{
  printf ("\nROBODoc v2.0e, autodocs formatter\n\n" );
  printf ("(c) 10-5-95, Maverick Software Development\n" );
  printf ("Original idea and program:  Jacco van Weert   <weertj@euronet.nl>\n") ;
  printf ("Version 2.0 and up:         Frans Slothouber <slothoub@xs4all.nl>\n") ;
  printf ("\nFORMAT\n" );
  printf ("  robodoc filename docfilename\n") ;
  printf ("\nIn addition you can use one or more of the following options:\n") ;
  printf ("  GENXREF <xref_filename>  - to generate an xref file.\n") ;
  printf ("  XREF <xreflist_filename> - if you want to use xref files to create\n") ;
  printf ("                             cross links\n") ;
  printf ("  TABSIZE <nr_sp>          - convert each TAB to nr_sp of spaces.\n") ;
  printf ("  TOC          - a table of contents will be generated.\n") ;
  printf ("  SORT         - the headers will be sorted.\n") ;
  printf ("  -v           - tell robodoc to tell you all about it.\n") ;
  printf ("  INTERNAL     - headers marked internal will also be included.\n") ;
  printf ("  INTERNALONLY - only headers marked internal will be included.\n") ;
  printf ("\nThe type of output is selected with one of the following switches:\n") ;
  printf ("  ASCII, GUIDE, HTML, or LATEX\n") ;
  printf ("\nThe following abbreviations are also allowed:\n") ;
  printf ("  TOC = -t  XREF = -x   SORT = -s  INTERNAL = -i \n") ;
  printf ("  GENXREF = -g  INTERNALONLY = -io  TABSIZE = -ts\n") ;
}
else
{
  RB_Analyse_Defaults_File () ;
  RB_Analyse_Arguments (argc, argv, &file_with_xrefs, &output_file_for_xrefs) ;

  RB_Say ("Trying to open source file", argv[1]) ;
  if (document = fopen (argv[1], "r"))
  {
    RB_Say ("analysing source file", "") ;
    RB_Analyse_Document (document) ;

    if (course_of_action & DO_SORT)
    {
      RB_Say ("Sorting headers", "") ;
      RB_Slow_Sort () ;
    }

    if (course_of_action & DO_USE_XREFS)
    {
      if (xreffiles_file = fopen(file_with_xrefs, "r"))
      {
        RB_Analyse_Xrefs (xreffiles_file) ;
      }
      else
      {
```

```
          RB_Panic ("can't open file with xref files", file_with_xrefs) ;
        }
      }

      if (course_of_action & DO_MAKE_DOCUMENT)
      {
        RB_Say ("trying to open destination file", argv[2]) ;
        if (dest_doc = fopen (argv[2], "w"))
        {
          RB_Say ("generating documentation", "") ;
          RB_Generate_Documentation (dest_doc, argv[2]) ;
          fclose (dest_doc) ;
          dest_doc = NULL ;
        }
        else RB_Panic ("can't open destination file",argv[2]) ;
      }

      if (course_of_action & DO_MAKE_XREFS)
      {
        RB_Say ("trying to open xref destiation file", output_file_for_xrefs) ;
        if (dest_doc = fopen (output_file_for_xrefs, "w"))
        {
          RB_Say ("Generating xref file","") ;
          RB_Generate_xrefs (dest_doc, argv[1], argv[2]) ;
          fclose (dest_doc) ;
          dest_doc = NULL ;
        }
        else RB_Panic ("error, can't open destination file", output_file_for_xrefs ↩
            ) ;
      }
    }
    else
    {
      RB_Panic ("error, can't open source file","") ;
    }
  }
  RB_Say ("Ready.", "") ;
  RB_Close_The_Shop () ;
  return (0) ;
}

/*********/
```

1.33 Robodoc.c/course_of_action

```
Robodoc.c/course_of_action

NAME
   course_of_action --

FUNCTION
   Global Variable that defines the course of action.

SOURCE
```

```
#define DO_SORT              1
#define DO_MAKE_XREFS        2
#define DO_USE_XREFS         4
#define DO_TOC               8
#define DO_MAKE_DOCUMENT    16
#define DO_INCLUDE_INTERNAL 32
#define DO_INTERNAL_ONLY    64
#define DO_TELL            128

int course_of_action = DO_MAKE_DOCUMENT ;
```

1.34 Robodoc.c/item_attributes

```
Robodoc.c/item_attributes

NAME
   item_attributes -- attributes of the various items

FUNCTION
   links each item type with a text attribute.

SOURCE


#define ITEM_NAME_LARGE_FONT      1
#define TEXT_BODY_LARGE_FONT      2
#define TEXT_BODY_ITALICS         4
#define TEXT_BODY_NON_PROP        8
#define TEXT_BODY_SMALL_FONT     16
#define TEXT_BODY_BOLD           32
#define TEXT_BODY_UNDERLINE      64
#define TEXT_BODY_SHINE         128

long item_attributes[NUMBER_OF_ITEMS] =
{
  0,    /* NO_ITEM */
  TEXT_BODY_SHINE,                          /* NAME_ITEM */
  TEXT_BODY_ITALICS | TEXT_BODY_SHINE,      /* COPYRIGHT_ITEM */
  0,                                        /* SYNOPSIS_ITEM */
  0,                                        /* FUNCTION_ITEM */
  TEXT_BODY_SHINE | TEXT_BODY_ITALICS,      /* AUTHOR_ITEM */
  TEXT_BODY_SHINE,                          /* CREATION_DATE_ITEM */
  0,                                        /* MODIFICATION_HISTORY_ITEM */
  0,                                        /* INPUTS_ITEM */
  0,                                        /* RESULT_ITEM */
  TEXT_BODY_SHINE,                          /* EXAMPLE_ITEM */
  TEXT_BODY_SHINE,                          /* NOTES_ITEM */
  TEXT_BODY_SHINE,                          /* BUGS_ITEM */
  0,                                        /* SEE_ALSO_ITEM */
  0,                                        /* SOURCE_ITEM */
  0                                         /* OTHER_ITEM */
} ;
```

1.35 Robodoc.c/line_buffer

```
Robodoc.c/line_buffer

NAME
   line_buffer -- global line buffer

FUNCTION
   Temporary storage area for lines that a read from a
   input file.

SOURCE


#define MAX_LINE_LEN 512
char line_buffer[MAX_LINE_LEN] ;

/*********/
```

1.36 Robodoc.c/output_mode

```
Robodoc.c/output_mode

NAME
   output_mode -- the mode of output

FUNCTION
   Controls which type of output will be generated.

SOURCE


/* Output Modes */
enum { ASCII = 0, AMIGAGUIDE, HTML, LATEX, SIZE_MODES } ;

/* Global Variable that defines the output mode */
int output_mode = ASCII ;
```