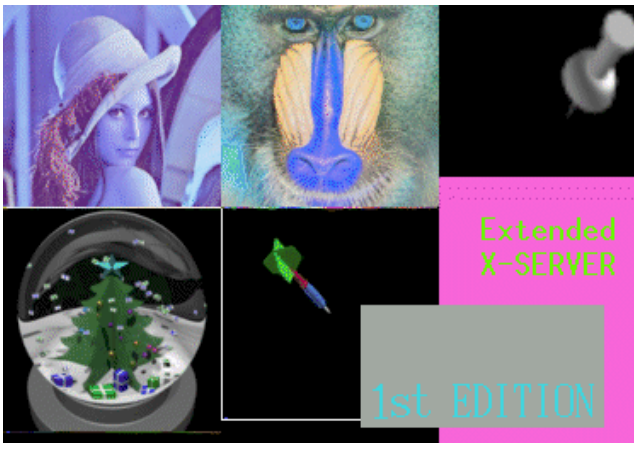
COMPLEX CONVERSION SYSTEM

and

EXTENDED C LIBRARY

HANDBOOK



LAWRENCE BERKELEY LABORATORY

Complex Conversion System and Extended C Library

(CCS-ECL)

Handbook

Copyright (c)

Jin Guojun

Lawrence Berkeley Laboratory

April, 1992

Table of Contents

page

Preface IX

How to Use This Handbook X
Compile and Link CCS-ECL library XII

CCS-ECL Libraries Reference Manual - I

Kernel and Interface Libraries 1-1

1-1

About This Manual 1-2

CalcSubWinMean — calculate mean value in a given area of RLE (interline 24-bit) images. 1-3
CloseColor_in_Map — find a closest color in colormap and return the index. 1-4
DBFourier — Fast Fourier Transform in double floating format 1-5
DBvfft2d 1-5
DBvrft2d 1-5
DBvfft3d 1-5
DBvrft3d — double floating VFFT in 2D and 3D domain 1-5
Fourier — compute one dimensional Fourier Transform 1-6
GetItem — get single variable from a string buffer 1-7
LongSwap — swap the four bytes order in a long integer. 1-8
QuickSort — quick sort procedure 1-9
ReadRGBMap — read RGB binary color map 1-10
ResetLKT — reset lookup table 1-11
ShortSwap — swap two bytes in a short integer 1-12
ShowA — display content of a quick sort array 1-13
To_8 — convert a true color image to a Pseudo color image. 1-14
System variables and some internal routines 1-15

alloc_2d_discrete — allocate 2-dimensional array line by line. 1-17
any_ilc_to_rle — convert any RGB, ARGB, BGR, or BGRA format to ILL (RLE) format. 1-18
any_to_seplane — convert different types of images to a separated plane image. 1-19
arget — return a value if next string in argv array is a number (digital). 1-20
available_type — find image ID by name 1-21
avset — set next available string to satisfy string or number. 1-22
bridge_header_handle — standard header handle interface 1-23
buffer_close — destroy buffer structure created by buffer_create() 1-25
buffer_create — create buffer structure for virtual file reading or writing 1-26
buffer_read — read buffer 1-27
buffer_seek — reposition buffer memory pointer 1-28
buffer_write — write data to a buffer 1-29
build_arg_fmt_list — * build system list from user arg_fmt_list_string 1-30
build_arg_list — build an argv like argument list 1-31
build_ccs_table — allocate CCS table content space 1-32
build_socket — create socket I/O stream for different network application 1-33
bytes_in_colortype — get number of bytes for given color type. 1-34
calibrate_colors — rearrange the color map according to its frequency 1-35
ccs_get_row —input image line by line 1-36
ccs_table_if — CCS internal table interface 1-37
check_host — function returns host type ID 1-38
color_rotate_90 — rotate a color image in 90 degree 1-39
comfirm_host — ** 1-40
core_trace — ** 1-41
create_pr_colormap — * allocate Raster color map space 1-42
dump_tbl — dump any table with LKT type 1-43
dvfft 1-44
dvfft_2d 1-44
dvfftn 1-44
dvrft_2d — Virtual Fast Fourier Transform 1-44
dw_init 1-45
dw_load — double VFFT initialization routines 1-45
error_mesg — error message reporter 1-46
eta_curve — Elastic Tuning curve generator 1-47
exit_timeout — * exit when license time out 1-48
find_input_type — * find input type for given argument format 1-49
fits_convertdata — * 1-50
fits_header_handle — * FITS header handle interface 1-51
fits_transf_data — * FIST data transfer kernel 1-52
fits_uncompress — * FITS image compression kernel 1-53
format_init — image structure initiate 1-54
free_2d_discrete — free a 2-dimensional memory space 1-56
free_arg_fmt_list — * free an argument format list 1-57
free_ccs_table — release CCS table content space 1-58
free_pr_colormap — free color map created by create_pr_colormap 1-59
gauss_mask — generate Gauss masks 1-60
get_addr — get network IP address in long integer 1-61
get_arg_list — get argument format list 1-62
get_args — * get argument lists 1-63
get_fits_head — * FITS header handle kernel 1-64
get_infile — create a readable file by name 1-65
get_outfile — create a writable file by name 1-66
get_port — get network port for transmission 1-67

get_soaddr — get socket information of connection. 1-68
get_superimpose_param — * 1-69
getbyte — ** 1-70
gif_header_handle — * GIF header handle interface 1-71
gray_to_rle — * RLE gray scale image to ILL (RLE) format convertor 1-72
hips_header_handle — * HIPS header handle interface 1-73
histogram — computer histogram 1-74
histogram_calc — histogram calculation 1-75
icc_header_handle — * ICC header handle interface 1-76
if_time_exp — * check if license time expired 1-77
ilc_to_gray — ILC (Raster) color image to gray scale image convertor 1-78
ill_to_gray — ILL (RLE) or SEPLANE image to gray scale image format convertor 1-79
ilc_to_sep— ILC (RAS) to SEPLANE image convertor 1-80
ilc_transfer — ILC format converter 1-81
ill_to_sep — ILL (RLE) to SEPLANE image converter 1-82
(*img->errors)() 1-83
(*img->header_handle)() 1-83
(*img->read)() 1-83
(*img->seek)() 1-83
(*img->std_swif)() 1-83
(*img->write)() 1-83
(*img->table_if)() 1-83
(*img->map_scanline)() 1-83
(*img->MAG_scanline)() 1-83
U_IMAGE internal function pointers — interface handler 1-83
init_FS_tables — * initialize Floyd-Steinberg tables for quantization 1-86
isColorImage — check if the format represent a color image 1-87
jpeg_header_handle — * JPEG image header handle interface 1-88
last_pointer_pos — get last memory pointer position 1-89
line_to_cell_color — ILL (RLE) to ILC (RAS) image converter 1-90
line_to_sep_color — ILL (RLE) to SEPLANE image converter 1-91
link_buffer — set buffer handle routines to U_IMAGE structure 1-92
load_DBw, lload_w — * Double FFT initial routines 1-93
load_rastfile — * load Sun raster files 1-94
map8_gray — map 8-bit color image to gray scale image 1-95
message — print information to stderr channel 1-96
min_bits — minimum bits for represent a number 1-97
nzalloc — allocate non-clean memory 1-98
parse_argus — parse argument list for command handling 1-99
parse_usage — display program usage and option formats 1-106
parserr — * report parse_argus() errors 1-107
peekbyte — ** peek a byte from a file stream 1-108
pict_header_handle — * Mac PICT image header handle interface 1-109
pnm_header_handle — * PNM header handle 1-110
pointer_buffer_size 1-111
p_buffer_size — get buffer size by passing buffer pointer 1-111
prgmerr — program error handler 1-112
pseudo_to_sep — * 8-bit color image to separated plane image converter 1-113
pull_Itype — check the image type by looking the first few bytes in header 1-114
put_fits_head — write FITS header 1-115
put_superimpose_param — insert image superimpose information into HIPS header 1-116
quant_to_8 — quantizing 24-bit color image to 8-bit color image 1-117
ras8_to_rle — 8-bit Raster image to ILL (RLE) converter 1-118
rast_header_handle — ** Raster image header handle interface 1-119

read_fits_image — * FITS image read interface 1-120
read_hex_rgbmap — read hex RGB color map 1-121
read_pgm_image — * PGM image read interface 1-122
read_pict_image — * Mac PICT image read interface 1-123
read_pnm_image — * PNM images read interface 1-124
read_ras_cmap — read regular color map 1-125
read_rast_image — * Raster image read interface 1-126
read_rle_image — * RLE image read interface 1-127
read_sepplane_image — * separated plane image read interface 1-128
read_tiff_image — * TiFF image read interface 1-129
read_var — * read variable length records from input file 1-130
readpipe — ** pipe reading handler 1-131
regmap_to_rlemap — regular color map to RLE colormap converter 1-132
rgbmap_to_othermap — Raster color map to other map converter 1-133
rle_header_handle — * RLE header handle interface 1-134
rlemap_to_regmap — RLE color map to regular color map converter 1-135
rotate90 — rotate an image in 90 degree 1-136
rtcp_getoptions — get RTP control header and options 1-137
rtcp_setoption — set RTCP option header 1-138
rtp_close — close real time protocol (RTP) socket 1-139
rtp_open — open real time protocol socket 1-140
rtp_read — read RTP frame 1-142
rtp_receive_cmd — retrieve RTP communication command 1-143
rtp_recvbygroup — RTP group receiver 1-144
rtp_release_channel — RTP release channel resource 1-145
rtp_request_channel — RTP request channel resource 1-146
rtp_sendbygroup — RTP group sender 1-147
rtp_setoption — set RTP transmission options 1-148
rtp_write — write RTP frame 1-149
s_buf_size — * change socket buffer 1-150
select_color_form — * select right color format for type conversion 1-151
set_pr_colormap — ** map regular color map to a Raster color map 1-152
set_gs_colormap — ** set a linear Raster color map 1-153
set_time_limit — * set time limit for license time out 1-154
snf_to_rle — convert ILC (Raster) image to ILL (RLE) image 1-155
socket_connect — make connection after built a socket 1-156
std_interface — * standard interface for image data handling 1-157
syserr — system error handle 1-159
tiff_header_handle — * TiFF image header handle interface 1-160
tvadd, tvsub— time value addition and substraction 1-161
tvdiff — difference of time value 1-162
ungetbyte — ** 1-163
usage_n_options — print usage options 1-164
verify_buffer_size — get the correct size buffer 1-165
vff_def_cmap — generate default VFF color map 1-166
vfft 1-167
vfft2d 1-167
vfft3dre 1-167
vfft_2d 1-167
vfftn 1-167
vrft2d 1-167
vrft3d 1-167
vrft_2d — virtual Fast Fourier Transform 1-167
w_init 1-169

w_load — * 1-169
write_rast — write image to a Sun Raster image file 1-170
write_rle — write image to a RLE image file 1-171
x_extender — extended X server 1-172
x_extender_init — X extended server initial routine 1-173
zalloc — allocate memory with all cells cleaned to zero 1-174

Index **1-175**

CCS-ECL Libraries Reference Manual - II

X Panel and Interface Libraries 2-1

About This Manual 2-2

BuildColorImage — build a color image in X window 2-3
ButtonPressed — check if a pressbutton is pressed 2-4
ChangeSliderScale — change slider scale range 2-5
CreateButton — create a set of rectangle buttons on a control panel 2-6
CreateCLT — create the Color Lookup Table 2-7
CreateImage — create image structure with X window 2-8
CreatePanel — create a control panel 2-9
CreatePopMenu — create a poping menu on a control panel 2-10
CreatePressButton — create a press button on a control panel 2-11
CreateScrollBar — create a scroll bar on any window 2-12
CreateSlider — create a slider on a control panel 2-13
CreateWindow — create a X window for any purpose 2-14
Delay_Clear — * flushing cursor delay routine 2-15
DestroyImage — destroy image content and free memory space 2-16
DispInfo — * display information on any panel or window 2-17
DrawButton — show button on its panel 2-18
DrawPixWindow — display pixel information on image window 2-19
DrawPressButton — draw pressbutton on its panel 2-20
DrawScrollBars — draw scrollbar on its parent window 2-21
DrawSlider — draw slider on its panel 2-22
DrawSpeedWindow — display movie speed on image window 2-23
DrawVMark — draw a vertical mark in an image window 2-24
Draw_ImageScrollBars — show scrollbars on an image window 2-25
Draws — draw patterns in an image window by given type 2-26
DumpScan_to_dpy — dump image content to window display 2-27
EraseSlider — hide a slider from its panel 2-28
FillDGT — ** fill Digtizer Table (Quantization Table) 2-29
Find_3_min_max — find minimum and maximum value in color histogram 2-30
Find_min_max — find minimum and maximum value in image data 2-31
FlushingCursor — flushing cursor generator 2-32
GetCloseColor — get close color from image color map 2-33
GetVctEntry — get the 1st available entry in given color map 2-34
LinearQuantization — generate linear color map table 2-35
LoadGXImage — load an image from a file and display it on X window 2-36
LoadIcon — load image icon X window data buffer 2-37
Maintain_Flush — * keep all image dumped to screen to be flushed 2-38
MapPixWindow — show pixel information window 2-39
MapRGB — map an image line into X window data buffer and maintain the display 2-40
Map_Scanline — map ILL scan line into X window data buffer 2-41
OnButton — check if mouse is clicked on a button 2-42
OnScrollBar — if mouse is clicked on a scrollbar 2-43
OnSliderBar — if mouse is clicked on a slider 2-44
PanelMessage — display message onto right area of panel parts or onto given place 2-45
ParameterWin — display pixel information in an image window 2-46
PopingMenu — pop a pop menu 2-47

ReadButton — get button name 2-48
ReadSlider — read slider bar position (value) 2-49
ResetPressButton — make pressed pressbutton unpressed 2-50
ResizeWindow — resize window size 2-51
SetFontGetSize — set font in a window, get size into img structure and return font ID 2-52
SetParameterWin — set pixel information window 2-53
SetSBarPos — set Slider Bar position by point on panel 2-54
SetSBarRPos — set Slider Bar in value (p) position 2-55
SetScrollBar — set scrollbar position 2-56
SetScrollBarLength — set scrollbar length 2-57
Set_Panel — set panel part properties 2-58
SliderInfo — draw slider information message on the slider 2-59
TextLine — text line editor for panel input 2-60
TopWindow — put window on the top of the display stack 2-61
TrackSubWin — Crop a sub-window in different sharp 2-62
WhichImage — see cursor is in which image window 2-63
XCopyImage — copy a X image region to another region 2-64
check_pixmap_allocation — set error handler 2-65
choose_scanline_converter — find appropriate scan line routine to use 2-66
create_windows — create X window for image display 2-67
eq_cmap — compare two color maps 2-68
exposure_r — exposure event handler 2-69
find_appropriate_visual — find appropriate visual type 2-70
find_min_max — find minimum and maximum value in a region 2-71
free_unique_colors — * free system color map space 2-72
get_X_image — create X image for image display 2-73
get_cursors — create cursors for a window 2-74
get_dither_arrays — allocate image dither arrays 2-75
get_dither_colors — sets in_cmap, cmlen, ncmap and mono_color in an image structure 2-76
get_iconsize — get appropriate icon size 2-77
get_pic — build a image and its display window from open files 2-78
get_x_colormap — create X color map 2-79
handle_exposure — image exposure handler 2-80
init_color — initial color map 2-81
init_img_info — initial image structure content 2-82
interpolation — interpolate an image by regions 2-83
mag_pan — zoom an image 2-84
map_1_to_3 — map 8-bit color image to a ILL (RLE) image 2-85
map_rgb_to_bw — map RGB to black-and-white through NTSC transform 2-86
map_rgb_to_rgb — convert RGB to RGB through a colormap 2-87
new_curve — ETA curve generator for X program 2-88
on_superimpose_elem — if mouse clicked on a superimposed element 2-89
set_button_color — set button colors 2-90
set_circle_cursor — set window cursor to circle cursor 2-91
set_left_ptr_cursor — set window cursor to left pointer cursor 2-92
set_pressbutton_color — set pressbutton color 2-93
set_slider_color — set slider colors 2-94
set_timer — set timer for wait_timer() 2-95
set_watch_cursor — set window cursor to watch cursor 2-96
set_window_cursor — set window cursor 2-97
superimpose_add_elem — add draw element to an image 2-98
superimpose_handle — superimposed element moving handler 2-99
superimpose_images — draws pattern onto an image 2-100

wait_timer — wait time out set by set_timer)(2-101
win_exposure — image window exposure handler 2-102

Index 2-103

CCS-ECL Libraries

Macros, Header Files, and Table Format 3-1

Macros 3-3

Library Header Files and Major Structures 3-5

Table Format for Table Interface 3-13

Program Guide 4-1

Example 1: 4-3
toany.c - torle and tohips 4-3
toany . c — enhanced program 4-4
toany . c — completed program 4-6
Example 2: 4-8
display-image - a simple image display program 4-8
getx . c — prototype 4-9
getx . c — a color image tuning and analysis system - ETA 4-12
getx .c — ETA + Extended X server 4-15

how to add a new image type handler into CCS 5-1

CCS-ECL Future and Limitations 6-1

Preface

The complex conversion system uses the standard software interface to handle different types of images in image processing. During the development of the CCS, many useful and high performance functions and subroutines have been programmed. These functions and subroutines generates an extended C language library (CCS-ECL).

What is the standard software interface? Compatibility and exchangeability are perennial problems in the computer field, in both the software and hardware areas. Some very good operating systems and compilers are limited in use because they are not compatible and exchangeable with all machines. Software interface increases image processing tool compatibility and exchangeability. Many image systems have different and incompatible formats. Many our programs handle only the HIPS [Landy, 1984] image, but we want to handle many other type of images, especially in the interactive programs. Even though software engineers expend much labor designing filters to convert x to y, y to x, y to z, ..., but these filters cause the complexity for users and still cannot satisfy all of them. Many of these converting filters are not reversible. So, for each different image system, we have to rebuild hundreds of image processing tools (including converting filters). In the face of established diversity of machines and software, it is almost impossible to imagine standardizing image systems, the answer is standard software interfaces. This includes static and dynamic handling. Interface is a type of hardware technique used to enhance and improve the CPU performance and increase the CPU power. The software interface is a type of the software technique used to coordinate the different types of systems, allowing machines to have compatibility and improving efficiency through exchangeability. The basic software interface scheme is shown in Figure 1.0.

Our software interface is implemented by 3 modules. They are internal library, dynamic table, and adaptive interface. The internal library collects often-used image conversion tools for common image types, and builds certain program sub-routines. These libraries must be compiled and linked with the main filter programs. The conversion library interface is a static interface, and has high efficiency. The dynamic table interface, which is built on the top of the internal library, uses a well-designed formatted table-file to obtain information telling how to read an image header and how to decode the image data. This interface may require that the input images have a special symbol at the beginning of the image file. This special symbol is usually called a "magic number". When a filter equipped with a dynamic table interface recognizes the input image, it will try to find the magic number in the input image to match one in the interface table file. Once a matching symbol is found, the filter can retrieve the information in that related table fields to get the header information and to decode the image

data. Another requirement of the dynamic table interface is that the encoding method of an input image must be known by the internal library. Otherwise, even though a filter can read the image header, but will not be able to decode the image data. The dynamic table interface advantages are flexibility, convenience, and speed. Users can add new image types into the dynamic table file using the text editor. The adaptive interface is a general software interface for image processing. The adaptive interface invokes other programs in run time which are used to handle the image header recognition and the image decoding. This differs from the internal library interface in two aspects: First, once an internal library interface has been changed, all the filters using this library must be re-compiled and linked to this library again, unless operating system allows to use dynamic linking. If we want to add a new image type into an adaptive interface based image processing system, however, the only thing that needs to be done is to install the new header handling or data decoding program into the particular area for the adaptive interface to search. Secondly, in the conversion library interface, all the data transfers are done in the local memory since all the sub-routines are linked at compile time. In the adaptive interface, image data transfer uses either pipe, shared memory, or other data communication schemes. Because of the interactions between the adaptive interface and handling programs through the operating system, the time delay is considerable in the adaptive interface technique.

Comparing these three types of interfaces, we prefer to use the conversion library interface for simple functional filter design. The adaptive interface is good for programs that are not very time sensitive to handle very different images, such as window based image analysis systems. The dynamic table interface may be used in both case to remedy their defects and to enhance the performance. Currently, the CCS kernel can handle FITS, GIF, HIPS, ICC, PICT, PNM, RLE, SUN-Raster, TIFF, and JPEG images. With our experience of image processing, use of the software interface for image type conversion in image processing is imperative.

How to Use This Handbook

This handbook has four parts:

(1) reference manual - I — CCS-ECL kernel and interface;

(2) reference manual - II — CCS X panel and interface;

(3) macro and header file — macros defined in library header files, library structures, and library header files;

(4) program guide — build programs step by step to show the simplicity of using CCS libraries and to build complicated programs in a few steps.

(5) how to add a new image type handler into CCS

(6) future and limitations for CCS-ECL

All tables of contents in this book are alphabetically ordered, and indices are sort in subjects. The tables of contents for reference manuals also contain simple descriptions for each function and subroutine. The indices, however, do not have any further explanations for details. Therefore, to find a new function or subroutine, you may need both subject index and alphabetic table of contents.

Two reference manuals describe the CCS-ECL library kernel and interface routine calls with parameters in details. The manual - I covers CCS-ECL kernel and interface functions and subroutines, and the manual - II covers CCS panel and interface functions and subroutines. These two manuals describe the usage and parameters in details for each function and subroutine if they are recommend for user programming, as well in some usage examples. Otherwise, only a brief description is given. The most useful categories in these manuals are:

CCS —

colors:
: color conversion, color map and mapping, quantization

error:
: error report and control

image:
: math function, histogram, convolution, Fourier transform, rotation, superimpose

interface:
: header handle, image data read and write, user interface

input/output:
: file control, buffer

memory:
: management, debug

network:
: socket, RTP, TCP, UDP, extended server

others:
: argument handle, bytes swap, host_check, timer

table:
: table interface

X window —

panel:
: basic window, panel, button, press button, slider, scroll bar, pop menu, message window, note window

image display and control

and the indices in these two manuals are sorted by these categories.

The third chapter, Marco and Header Files, lists a number of macros defined in CCS library header files and explains some important library structures, such as U_IMAGE.

The next chapter is the program guide. Two programs are used to illustrate how to program in CCS-ECL for regular and X window programs. They are tonay.c and getx.c. The toany.c is a more general purpose tool in CCS programming for image type conversion. The getx.c is the X window programming in CCS; its binary and manual page are available for Sun 4 workstations, but its source code is not released to public. The getx.c source code is available to only HIPS and LBL users now. In these examples, both programs are build in server times from very simple case, but very useful, to a complicated, in functionality not in programming, program.

In Chapter 5, how to extend a new image type for CCS library is introduced for every one who wants to add his/her own stuff into CCS-ECL package.

The last thing in this handbook is giving the goal and limitation about this library.

Compile and Link CCS-ECL Libraries

The CCS-ECL package is well formatted for users to use. The config.xxxy files in ccs-lib directory are for different machine architectures. The xxxy is the machine type. If you find any xxxy is same as your machine type or close to your machine type, type Configure xxxy, or just Configure in other cases, to configure CCS Makefiles. Before you configure CCS, you may need to do a minor modification in your config.xxxy file to change the default library and binary destination paths. To do so:

change	LIBDIR = $(TOPDIR)/sun4/lib
to	LIBDIR = your_lib
e.g.	LIBDIR = /usr/local/lib

change	DESTDIR = $(TOPDIR)/sun4/bin
to	DESTDIR = dest-binary-dir
e.g.	DESTDIR = /usr/local/bin

After configuring CCS, type make to build libraries. On IBM/PC system, special makefiles are needed for compiling CCS-ECL. These makefile.src files are in directory makefile.bcc for Borland C or Turbo C compiler. If these files are not available, contact me by email.

The syntax to link program to CCS libraries on unix system is:

cc -o $(DESTDIR)/filte $(OBJS) -lscs# -lccs $(OTHER_LIBS)

Look at the makefile.src in convert directory for more different usage.

For different programming requirement, the CCS-ECl libraries have six interface levels: libscs1 - libscs6. Application programs are linked to CCS kernel by these interfaces to suit different demands. The level 1 only reads and writes HIPS and FITS images. The level 2 can read and write one more type image RLE. The level 3 reads most of the supporting type images in CCS, except PICT & JPEG. The level 4 can read PICT image. The level 5 reads all of supporting type images. The level 6 can write Sun Raster image, as well other CCS supported image types (not guaranteed). The level 1 - 4 is stable, and any new image type should be added into level 5 and 6 unless it has to be split to a higher level.

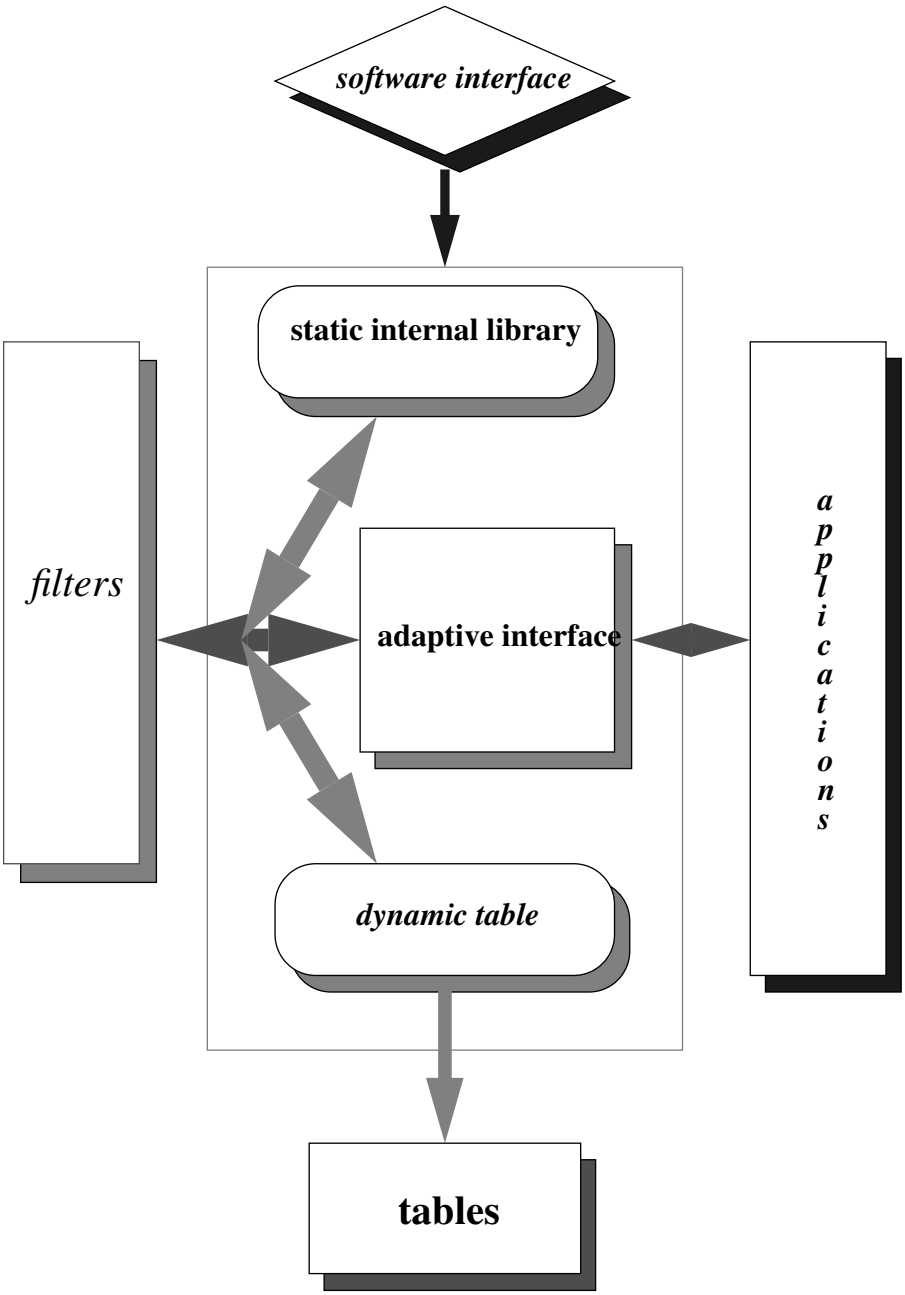

software interface
static internal library
filters
adaptive interface
applications
dynamic table
tables

Figure 1.0