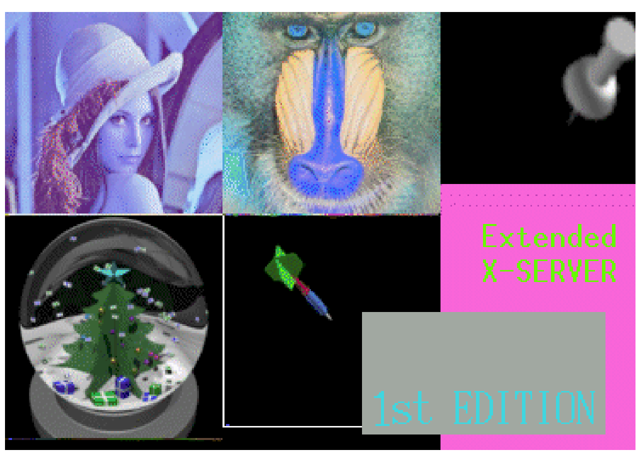
COMPLEX CONVERSION SYSTEM

and

EXTENDED C LIBRARY

HANDBOOK



LAWRENCE BERKELEY LABORATORY

Complex Conversion System and Extended C Library

(CCS-ECL)

Handbook

Jin Guojun

Lawrence Berkeley Laboratory

April, 1992

Table of Contents

CCS-ECL Libraries Reference Manual - II

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.

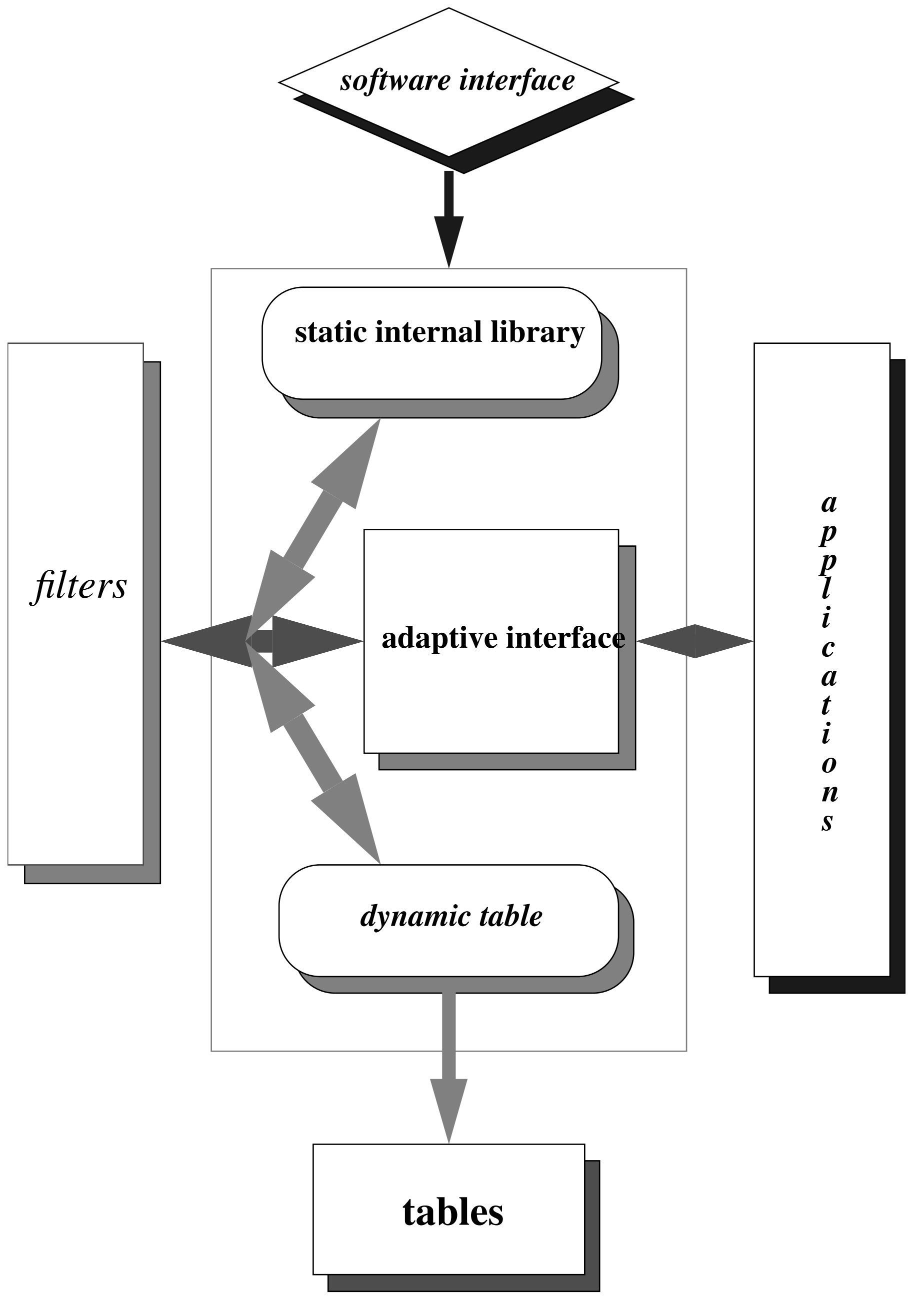

software interface
static internal library
filters
adaptive interface
a p p l i c a t i o n s
dynamic table
tables

Figure 1.0