# Adding Another CCD Camera to the LRIS CCD Control Software

John Cromer

August 18, 1993

# Contents

## 0.1 Introduction

The purpose of this document is to describe the simplest changes to the LRIS/HIRES CCD control software that enable support for two CCD cameras. The system was designed with multiple cameras in mind and each VME system was designed to control two cameras. Relatively simple changes to the software are all that is necessary to bring another camera on line. The changes required to the VME VxWorks functions are described first. Changes to the FIORD library of functions are then described. These changes describe the minimal amount of work neccessary to bring the camera up. The next sections describes work needed to make the system more "user-friendly" and the final section examines the limitations of the system and some possible alternatives.

It is assumed the reader has a familiarity with the design, code and directory structure of the LRIS/HIRES CCD software.

Neglecting changes to the LRIS body and the additional camera hardware, adding a second camera will require that the VME CCD control hardware be updated with a second camera interface board, the so-called Harris-Ricketts board, and also another Ikon DMA controller. The Harris-Ricketts board will have to be built either at CIT or at Lick. (The original board for the LRIS was built at Lick.) This board will be connected to the new camera by a new set of fiber optic cables.

## 0.2 VME VxWorks Software Changes

The changes to the VxWorks software for CCD control are straightfoward. All received control messages already specify a camera number. This number can be used as an index into arrays to select camera-specific options and set camera-specific flags. The major change to the software then is changing certain global variables to arrays and modifying the "set," "show" and auxillary functions to use those arrays rather than the single variables used now. The following step-by-step procedure can be used:

1. In `crate_global_init.c` convert all state-machine variables to arrays whose dimensions are the total number of cameras.
2. Make the same changes in the "set" and "show" functions as well as the functions which manipulate camera semaphores. Modify these functions to use the arrays and the `camera_id` variable to index into each array.
3. Split the `ccdClock()` timing function into `ccd0Clock()` and `ccd1Clock` and modify them to use the appropriate elements in the arrays created in step 1.
4. Modify `broadcast_camera_value()` in `broadcast.c` to accept `camera_id` as a parameter and put it in the broadcast mail message.

The static initialization section of `crate_global_init.c` requiring changes is shown below:

```
unsigned long   start_flag[NUM_CAMERAS];        /* exposure start flag */
unsigned long   stop_flag[NUM_CAMERAS];         /* exposure stop flag */
unsigned long   auto_shutter[NUM_CAMERAS];      /* automatic shutter flag */
unsigned long   auto_erase[NUM_CAMERAS];        /* automatic erase flag */
unsigned long   auto_read[NUM_CAMERAS];         /* automatic readout flag */
unsigned long   elapsed_time[NUM_CAMERAS];      /* exposure elasped time */
unsigned long   total_time[NUM_CAMERAS];        /* exposure total time */
unsigned long   erase_flag[NUM_CAMERAS];        /* erase-in-progress flag */
unsigned long   pause_flag[NUM_CAMERAS];        /* pause-in-progress flag */
unsigned long   erase_count[NUM_CAMERAS];       /* lines to erase on fastwipe */
unsigned long   image_id[NUM_CAMERAS];          /* image identification number */
unsigned long   read_flag[NUM_CAMERAS];         /* readout in progress flag */
unsigned long   camera_status[NUM_CAMERAS];     /* camera status value */
unsigned long   window[NUM_CAMERAS][5]={
                        {1,0,0,2048,2048},      /* camera 0 readout window */
                        {1,0,0,2048,2048}       /* camera 1 readout window */
                        };
unsigned long   binning[NUM_CAMERAS][2]={
                        {1,1},                  /*camera 1 binning */
                        {1,1}                   /*camera 0 binning */
                        };
unsigned long   preline[NUM_CAMERAS];
                            /* lobal number of prescan lines */
unsigned long   preflush[NUM_CAMERAS];
                            /* # of rows flushed after prescan read */
unsigned long   postline[NUM_CAMERAS];
                            /* # of overscan lines */
unsigned long   overflush[NUM_CAMERAS];
                            /* # of rows flushed before postline read */
unsigned long   prepix[NUM_CAMERAS];
                            /* Global number of prescan pixels */
unsigned long   postpix[NUM_CAMERAS];
                            /* Global number of overscan pixels */
unsigned long   keepprepix[NUM_CAMERAS];
                            /* Whether prepix should be saved */
unsigned long   eraseline[NUM_CAMERAS];
                            /* Erase line flag */
long utb_digital_input[NUM_CAMERAS];
                            /* last digital input from utility bd.*/
long utb_digital_output[NUM_CAMERAS];
                            /* last digital output from utility bd.*/
long utb_raw_adc_channel[NUM_CAMERAS][UTB_NUM_ADC_CHAN];
                            /* last ADC readings */
long utb_raw_target_channel[NUM_CAMERAS][UTB_NUM_ADC_CHAN];
```

```
                              /* last ADC readings */
long utb_raw_dac_channel[NUM_CAMERAS][UTB_NUM_DAC_CHAN];
                              /* last DAC readings */
```

Once the arrays are created, the functions that use them must be changed. As an example, consider the functions defined in module `s_expose.c`. The `extern` global declarations should be changed as follows:

```
extern unsigned short   start_flag[];   /* exposure start flag */
extern unsigned long    auto_shutter[]; /* automatic shutter flag */
extern unsigned long    auto_erase[];   /* automatic erase flag */
extern unsigned long    erase_flag[];   /* erase in progress flag */
extern unsigned long    pause_flag[];   /* pause in progress flag */
extern unsigned long    read_flag[];    /* readout in progress flag */
extern unsigned long    image_id[];     /* image id number */
extern unsigned long    elapsed_time[]; /* elapsed integration time */
```

The unpacking of the music message would have to be moved to the very top of the code in order to get the `camera_id` first. Then the remaining code would be changed from, for example:

```
        if (check (start_flag,ERR_ERROR,"E1",CHK_PARAMS,
```

to

```
        if (check (start_flag[camera_id],ERR_ERROR,"E1",CHK_PARAMS,
```

and so on througout the code. These kinds of changes should be made to every module that references the variables changed in `crate_global_init.c`.

The function `ccdClock()` which provides exposure timing and image readout coordination should be split into two functions: `ccd0Clock()` to provide these functions for camera 0 and `ccd1Clock()` for camera 1. In each of these functions the same changes described above should be applied. References to, for example, `start_flag` in `ccd0Clock.c` should be changed to `start_flag[0]` and in `ccd1Clock.c` to `start_flag[1]`. Similarly, all variables changed in `crate_global_init.c` should be changed in these two modules.

The function `broadcast_camera_value()` in the `broadcast.c` module must be modified to accept `camera_id` as a parameter and to include it in the broadcast. The calls to this function will then need to be changed in `ccd0Clock.c`, `ccd1Clock.c`, `s_set_time.c` and `s_expose.c`. This is a bug; `camera_id` was simply left out of the function when it was written. This change also inpacts the host `ktl` software. The functions which receive these broadcasts must be made to unpack `camera_id` from the MUSIC broadcast message.

Finally changes to the startup script should be made so that `ccd0Clock()` and `ccd1Clock()` are spawned instead of `ccdClock()` only.

One problem has been conveniently ignored so far in describing these changes and this is the issue of coordination between the two cameras. A limitation of the system as described is both cameras cannot be read out simulataneously by `rccd()`. So, while one camera is reading out, the other must be blocked from reading out.

(*** I'm not sure if this happens automatically in `rccd()` or not. I imagine the software that calls `rccd()` will have to provide it. ***)

## 0.3 FIORD Software Changes

Changes to the FIORD software are completely straightfoward although cumbersum perhaps. Basically every input and output function must be cloned and where `camera_id` is set to zero, it must be set to one. Where a zero is packed into a MUSIC message for the camera ID, a one must be substituted. Additionally all new keywords must be defined for the new camera. The following steps summarize the process:

1. Clone new keywords for each camera command.
2. Add the new keywords to the keyword database in `fiord.c`
3. Clone all necessary functions: `input_`*keyword*`()` and `output_`*keyword*`()`, changing camera 0 to camera 1.
4. Add the new FIORD functions to the file `fiord_proto.h`.
5. Add new temperature conversion factors, `DEGREES_PER_COUNT` and `COUNTS_AT_ZERO_DEGREES` to the `tempdet.h` file for the second camera.
6. Add any new modules to the `makefile`.

For the new keywords, I propose the keywords for the red camera be modified in the following way: If the keyword has less than 8 characters, the FITS standard, simply add a '1' to it. If the keyword has 8 characters, change the last character two a '1'. With these changes, the new keywords for both red and blue cameras are:

| Red Camera Keywords | Blue Camera Keywords |
|---|---|
| AUTOERAS | AUTOERA1 |
| AUTOREAD | AUTOREA1 |
| AUTOSHUT | AUTOSHU1 |
| EXPOSIP | EXPOSIP1 |
| PAUSEIP | PAUSEIP1 |

| | |
|---|---|
| ERASEIP | ERASEIP1 |
| TEMPDET | TEMPDET1 |
| WCRATE | WCRATE1 |
| WDISK | WDISK1 |
| ERASECNT | ERASECN1 |
| TTIME | TTIME1 |
| ELAPTIME | ELAPTIM1 |
| EXPOSE | EXPOSE1 |
| PAUSE | PAUSE1 |
| RESUME | RESUME1 |
| ABORTEX | ABORTEX1 |
| STOPEX | STOPEX1 |
| CSHUTTER | CSHUTTE1 |
| ADDFRAME | ADDFRAM1 |
| DFORM | DFORM1 |
| FRAMENO | FRAMENO1 |
| OUTDIR | OUTDIR1 |
| OBJECT | OBJECT1 |
| OUTFILE | OUTFILE1 |
| TODISK | TODISK1 |
| TOTAPE | TOTAPE1 |
| TAPEDEV | TAPEDEV |
| WINDOW | WINDOW1 |
| BINNING | BINNING1 |
| PREPIX | PREPIX1 |
| POSTPIX | POSTPIX1 |
| DATAPIX | DATAPIX1 |
| PRELINE | PRELINE1 |
| DATALINE | DATALIN1 |
| ERASLINE | ERASLIN1 |
| POSTLINE | POSTLIN1 |
| PREFLUSH | PREFLUS1 |
| OVRFLUSH | OVRFLUS1 |
| KEEPPREP | KEEPPRE1 |

The long versions of the keywords can be changed similarly although these are rarely used. Since the FIORD function names are of the form `output_`*keyword*`()`, etc., creating the new keywords determines the function names. The modification of the input and output functions is simple; All instances of `camera_id` or and `cam_id` should be assigned 1 instead of 0.

In the HIRES software, the use of macros to define functions of a repetitive nature is employed to decrease the amount of replicate code included in the system. This has been done to a certain extent with the CCD software as the HIRES group has continued to develop the CCD software while the LRIS group "froze" their effort around the time of the LRIS preshipment review. As a result the the CCD software is out of sync between the two groups. Making these changes to the FIORD library might be a good oportunity to "macroize" the rest of the CCD code (especially the duplicate functions for the LRIS blue camera) and bring the two systems closer together.

## 0.4 Further Work

The procedures described above will get a minimal system with two cameras up and running. Nothing was mentioned about the windows user interface. Either the current **`xpose`** display will have to be modified to display two cameras or another **`xpose`** display will have to be created to control the blue camera separately. It is trivial to add the second temperature display to **`xshow`**.

Image display is also an area in need of further work. The current version of the image server should be able to handle more than one image with no modifications. It will not, however, display more than one image simultaneously, that is, it cannot talk to two **`figdisp`** displays at once. This program should be modified to display two images simultaneously.

## 0.5 Other Options

As noted previously the procedures described above lead to a system in which two cameras cannot be read out simultaneously. The additional 67 seconds of readout time (assuming a two-amplifier CCD configuration) should not be significant for long exposures, but could be painful for short exposures and test situations. Two options are considered to remedy this problem:

1. Add a second complete VME system that would operate totaly independent of the first CCD VME system.

2. Modify **`rccd()`** to read out more than one camera simultaneously.

The first option provides the most flexibility and is simpler in software terms, but the hardware expense is considerable. It also contradicts the original idea that two cameras could be controlled from one VME system. It would require no significant new software development, only the change to the image server described in the previous section to display more than one image at once. Actually it would require less software work than described in this document since the VME software essentially would not need to be modified, just duplicated for the additional VME system. A number of less complex systems being easier to maintain than a single more complex system, this options has advantages for maintenance.

In a private communication with Sam Southard, he estimated an additional month of software effort on his part to implement option two above, that is, change **`rccd()`** so it could read out two cameras simultaneously. This coupled with changes to the image server on the host computer to display images simultaneously constitute significant software changes in addition to those previously specified. No additinal hardware costs would be necessary.

As a personal note, I tend to come down on the side of option one for its simplicity with respect to the software, however, the additional complexity of option 2 may not be significant enough to produce the ease-of-maintenance benefits.