Notes on Some CCD Engineering Programs

John Cromer

July 27, 1993

Contents

0.1 Introduction

The purpose of this document is to describe some of the small engineering programs supplied with LRIS CCD control software. This is not a CCD software reference manual. Presumably a CCD software reference manual will come from the HIRES group as they have instituted significant software changes in the last year. This document also does not describe the "testsuite" program which is used mainly to troubleshoot the CCD VME hardware. The programs described here run on the VME system and are located in the `/kroot/kss/lris/vme/ldserv/shell` directory. They are small simple programs which send DSP commands or manipulate DSP memory on the timing or utility boards of the CCD electronics. These programs:

- list and manipulate DSP memory
- send DSP commands with no parameters
- send DSP commands with parameters
- troubleshoot the utility board

These programs were originally put together quickly as one-shot deals to be used by the CCD engineer when testing the camera electronics. As such their structure and design may not be the most consistent and optimal. They do however have the virtue of being simple and easy to understand.

0.2 Functions to Display and Edit DSP Memory

The following functions are used to display and edit DSP memory on the timing and utility boards of the CCD camera electronics.

DSP Memory Programs		
Function	**Module**	**Description**
dspd()	dspd.c	Display timing board dsp memory
dspm()	dspm.c	Modify timing board memory
udspd()	udspd.c	Display utility board DSP memory
udspm()	udspm.c	Modify utility board DSP memory

0.2.1 dspd()–Display Timing Board DSP Memory

Syntax: `dspd` [*space*] [,*astart*] [,*nwords*]

where:

> *space* is the DSP memory space to examine, 'p', 'x' or 'y'.
>
> *astart* is the DSP memory location to begin the display.
>
> *nwords* is the number of 24-bit words to display.

`dspd` is the DSP analog of the VxWorks 'd' command. It allows the display of a given number of words of timing board DSP memory on the VxWorks console. The display consists of the address and the value contained in that address in both hexadecimal and decimal. Parameters are optional. The defaults are:

> *space*='y'
>
> *astart*=0
>
> *nwords*=22

for the first execution after bootup. Afterwards *space* and *nwords* retain the last value specified by the user whereas *astart* becomes the last address displayed.

0.2.2 udspd()–Display Utility Board DSP Memory

Syntax: `udspd` [*space*] [,*astart*] [,*nwords*]

where:

> *space* is the DSP memory space to examine, 'p', 'x' or 'y'.
>
> *astart* is the DSP memory location to begin the display.
>
> *nwords* is the number of 24-bit words to display.

`udspd` is the DSP analog of the VxWorks 'd' command. It allows the display of a given number of words of utility board DSP memory on the VxWorks console. The display consists of the address and the value contained in that address in both hexadecimal and decimal. Parameters are optional. The defaults are:

> *space*='y'
>
> *astart*=0
>
> *nwords*=22

for the first execution after bootup. Afterwards *space* and *nwords* retain the last value specified by the user whereas *astart* becomes the last address displayed.

0.2.3 dspm()–Modify Timing Board DSP Memory

Syntax: `dspm` [*space*] [,*astart*]

where:

> *space* is the DSP memory space to examine, 'p', 'x' or 'y'.
>
> *astart* is the DSP memory location to begin.

`dspm` is the DSP analog of the VxWorks 'm' command. It displays the address and value of the given timing board DSP memory location and waits for user input. Pressing a <return> causes the next memory location to be displayed without any modification to the previous location. Entering a hexadecimal value, alters the DSP memory at the displayed location to the entered value. Entering any non-hex character other than <return> stops the program. Typing "0x" before the entered value is not necessary. Any number typed in is presumed to be in hex. The parameters are optional. Defaults are:

> *space*='y'
>
> *astart*=0

for the first execution after bootup. Afterwards *space* retains the last value specified by the user whereas the default value for *astart* is always zero.

0.2.4 udspm()–Modify Utility Board DSP Memory

Syntax: `udspm` [*space*] [,*astart*]

where:

> *space* is the DSP memory space to examine, 'p', 'x' or 'y'.
>
> *astart* is the DSP memory location to begin.

`udspm` is the utility board DSP analog of the VxWorks 'm' command. It displays the address and value of the given memory location and waits for user input. Pressing <return> causes the next memory location to be displayed without any modification to the previous location. Entering a hexadecimal value, alters the DSP memory at the displayed location to the entered value. Entering any non-hex character other than <return> stops the program. Typing "0x" before the entered value is not necessary. Any number typed in is presumed to be in hex. The parameters are optional. Defaults are:

> *space*='y'
>
> *astart*=0

for the first execution after bootup. Afterwards *space* retains the last value specified by the user whereas the default value for *astart* is always zero.

0.3 Functions to Send Commands to the DSP–No Parameters

The following functions are used to send commands which have no parameters to the the camera electronics:

Programs to Send DSP Commands with No Parameters		
Function	**Module**	**Description**
tb_com()	tb_com.c	Send a command to the timing board DSP
ub_com()	ub_com.c	Send a command to the utility board DSP
idl()	idl.c	Send the IDL command to the timing board DSP
sbv()	sbv.c	Send the SBV command to the timing board DSP

0.3.1 tb_com()–Send a Command to the Timing Board DSP

Syntax: `tb_com` *camera_id*, *command*

where:

> *camera_id* is the ID number of the camera.
>
> *command* is the 3-letter command to the timing board DSP. Must be upper case and in double quotes.

`tb_com` simply sends a command that requires no parameters to the timing board DSP for execution. The function puts the command into a message packet and calls `vmemsgxchng()` to transmit the command over the VME path and optical fiber to the CCD timing board.

0.3.2 ub_com()–Send a Command to the Utility Board DSP

Syntax: `ub_com` *camera_id*, *command*

where:

> *camera_id* is the ID number of the camera.
>
> *command* is the 3-letter command to the utility board DSP. Must be upper case and in double quotes.

`ub_com` sends a command that requires no parameters to the utility board DSP for execution. The function puts the command into a message packet and calls `vmemsgxchng()` to transmit the command over the VME path and optical fiber to the CCD timing board which forwards it on to the utility board.

0.3.3 idl()–Send the Idle command (IDL) to the Timing Board DSP

Syntax: `idl` *camera_id*

where:

> *camera_id* is the ID number of the camera.

This function was written as a convenience. It sends the IDL command to the timing board DSP on the selected camera. This starts the DSP clocking the CCD in the idle mode. Note this is the same as typing `tb_com 0,"IDL"`.

0.3.4 sbv()–Send the SBV command to the Timing Board DSP

Syntax: `sbv` *camera_id*

where:

> *camera_id* is the ID number of the camera.

This function It sends the SBV command to the timing board DSP on the selected camera. This applies the current bias voltages to the two CCD amplifiers. Note this is the same as typing `tb_com 0,"SBV"`. Note also this program should be run any time the power to the camera electronics is cycled.

0.4 Functions to Send Commands with Parameters to the DSP

The following functions send commands with parameters to the DPSs on the utility and timing boards:

Programs to Send DSP Commands with Parameters		
Function	**Module**	**Description**
cbv()	cbv.c	Change bias voltages
lda()	lda.c	Load a utility board program
tdl()	tdl.c	Test data links

0.4.1 cbv()–Change Bias Voltages

Syntax: `cbv` *v0*, *v1*

where:

> *v0* DAC value (0-255) for the bias voltage on board 0.

v1 DAC value (0-255) for the bias voltage on board 1.

This function changes the values of two DSP Y memory locations on the CCD timing board. These two locations contain the values that are loaded into the DACs which control the bias voltages applied to the two CCD amplifiers. The program displays the current values and the values after the change. Typing `-1` for v0 causes the values to be displayed only. Typing `cbv` with no parameters will cause the two values to be set to 0! This probably should be avoided.
The `sbv` command must be used after the `cbv` command in order to apply the new DAC values to the CCD bias voltage circuits.

0.4.2 lda()–Load a Program into Utility Board DSP Memory

Syntax: `lda` *camera_id*, *n*

where:

camera_id is the ID number of the camera.

n is the number of the program to load.

This function causes the selected program to be copied from ROM to DSP RAM on the utility board. See the Leach electronics documentation for the utility board for more information. Typing `lda 0,0` loads the shutter functions. Note if power is cycled to the camera electronics the shutter will not operate until this command is typed on the VxWorks console.

0.4.3 tdl()–Test Data Link

Syntax: `tdl` *camera_id*, *board_id*

where:

camera_id is the ID number of the camera.

board_id is the number of the board to which the data link is tested, 2=timing board, 3=utility board.

This function sends the TDL command to the specified board DSP along with the parameter 0xffffff. This tests all 24-bits of the data path. The TDL command operates by simply echoing the parameter back down the link after it is recieved on the destination board. If the reply value is not the same as the value sent, the data link is presumed to be at fault. Note: this command will not show a problem with a bit being stuck high.

0.5 Functions to Troubleshoot the Utility Board

The following functions are used to troubleshoot various functions of the Leach electronics utility board:

Utility Board Troubleshooting Functions		
Function	**Module**	**Description**
read_util()	read_util.c	Continuously read a memory location
read_temp()	read_temp.c	Continuously read CCD temperature

0.5.1 read_util()–Continuously Read a DSP Memory Location

Syntax: `read_util` *camera_id*, *uadr*, *dt*, *max_readings*

where:

camera_id is the ID number of the camera.

uadr is the DSP Y memory location on the utility board to read.

dt is the delay time in seconds between each reading.

max_readings is the total number of readings to make.

This function loops monitoring a specific location in the DSP Y memory space on the utility board. The utility board is capable of controlling and monitoring a variety of sensors. These Sensor readings are generally converted to a digital number via an ADC and placed in some location in DSP Y memory. From there it may be read by VME software. `read_util()` takes are reading of the value in location `uadr` every `dt` seconds until a counter reaches `max_readings`. The readings are displayed on the VxWorks console and also written to the file `/kroot/kss/lris/vme/ldserv/shell/util.dat`.

0.5.2 read_temp()–Continuously Read a DSP Memory Location

Syntax: `read_temp` *camera_id*, *dt*, *max_readings*

where:

camera_id is the ID number of the camera.

dt is the delay time in seconds between each reading.

max_readings is the total number of readings to make.

This function is identical to `read_util()` except the DSP Y memory location which is monitored is hard coded to be 0xc, the location where the CCD temperature ADC counts are stored. Readings are taken every `dt` seconds until a counter reaches `max_readings`. The readings are displayed on the VxWorks console and also written to the file `/kroot/kss/lris/vme/ldserv/shell/temp`.

0.6 CCD Electronics Power Cycle–What to do

Use the following procedure if power is cycled to the camera electronics and the VME crate is not rebooted. On the VxWorks console:

1. Type `tdl 0,2` to test the data link to the timing board.
2. Type `tdl 0,3` to test the data link to the utility board.
3. Run `cbv` to change the bias voltages if you are not happy with the default values.
4. Run `sbv 0` to apply bias voltages. Do this regardless of whether you changed the bias voltages in the previous step.