Multi-Channel Electronics Command Descriptions

As of 2013-04-22, this document is no longer updated. Instead, see http://e-mode.phas.ubc.ca/mcewiki/index.php/MCE_Commands

Revision History:

Rev	*Date*	*Description of change*
1.0	2007-10-09	MA/added the table to 09/10/07 version
1.1	2007-10-18	BB. Added placeholders for newly supported commands.
1.2	2008-01-30	added scratch and support for slot_id and card_type
1.3	2008-05-16	MA: Updated readout-card command descriptions, references, summary, section 4. Updated command format to not refer to DAS/MAS options as they may change. Added firmware revision information.
1.4	2008-05-20	BB: Added content that I had been working on in my local copy, in CVS. Also BB: Re-integrated information that had been deleted in the update from 1.2 to 1.3. This information has now been reviewed, and is being edited as agreed upon by BB/MA. BB: Added firmware revision columns for all commands BB: Added const_mode and const_data commands for address card. BB: Updated the 'Command Format' section so that it is current MA: added servo_mode per column, resolution for gainpid, details for fw_rev, data_mode and captr_raw commands. MA: merged rcs commands with single-rc commands
1.5	2008-06-30	MA added row_dly minimum=4 Added a note about st and go command only supported for ret_dat so far.
1.6	2009-10-05	MA/BB added following commands: num_rows_reported, num_cols_reported, cards_present, cards_to_report, rcs_to_report_data, stop_dly, bias_start, data_mode 10 to 12, readout_col_index, readout_priority.
1.7	2009-11-13	BB: Added the i_clamp_val command and added a note to the const_mode command. Also added run_id and user_writable commands to the clock card section.
1.8	2010-01-12	BB: Amended i_clamp_val, and added awg_sequence_len, awg_data, awg_addr.
1.9	2010-05-26	MA: added fb_col0 to fb_col29 and enbl_mux to Bias-Card commands In timing diagram, changed SQ1 Bias row_dly to be row_dly=2.
1.10	2010-06-01	MA: added a note about taking into account the mapping of the backplane slot the address-card is plugged into when adding fb_col parameters to the hardware abstraction file (i.e. mce.cfg for mas and xml file for das)
1.11	2010-06-16	BB: added the following commands fltr_type, tdo_sample_dly, tck_half_period, jtag0, jtag1, jtag2, tms_tdi, tdo.
1.12	2010-11-01	MA: added filt_coeff and more info for psc_status, modified fltr_type. Changed i_clamp_val to integral_clamp.
1.13	2011-01-19	MA: added heater_bias, heater_bias_len with enbl_mux=3 definition, added timing diagram, updated section 3.
1.14	2011-02-23	MA: clarified integral_clamp and corrected the calculation guideline
1.15	2011-04-18	MA: updated timing diagram in Figure 1, edited captr_raw description, edited fb_dly minimum
1.16	2011-05-02	BB: updated the timing artefact statement for select_clk, use_sync, use_dv.
1.17	2011-09-21	MA: added pcb_rev to card_type description, marked obsolete and not-implemented commands.
1.18	2011-11-07	MA: added critical_err_rst, dev_clr commands, revised summary!

1.19	2011-11-24	MA: changed dev_clr to fpga_clr , added ramp_step_phase and pterm_decay_bits parameters
1.20	2012-3-23	MA: added enbl_flux_fb_mod, enbl_bias_mod, mod_val to bias card parameters
1.21	2012-04-13	mod_val takes one value and not 32!
1.22	2012-06-25	Moved awg commands next to ramp commands, added a note for fb_dly (changed 18 to 10)
1.23	2013-04-22	Added servo_rst_arm and servo_rst_col commands for Readout Card
1.24	2014-09-11	No longer updated, Moved to wiki instead: http://e-mode.phas.ubc.ca/mcewiki/index.php/MCE_Commands

1. Summary

This document describes the set of commands, or the API, to communicate with the Multi-channel Electronics (MCE) over the fibre-optic interface. A pair of fibre optic cables connects the MCE to a host PC that has a PCI ARC-64 card [1] installed and runs the data acquisition software. Today, there are two such software available: DAS, developed by SCUBA-2 team, and MAS, which can be considered the second generation to DAS and is developed at UBC.
See http://e-mode.phas.ubc.ca/mcewiki/index.php/MAS. The user would use MAS or DAS environment to issue commands to the MCE. In DAS, the MCE commands and their encodings are listed in a file called **mce.xml** while in MAS, such information are in **mce.cfg** (as of 2011-11-06).

2. References

[1] http://www.astro-cam.com/arcpage.php?txt=products.php&cat=Controller%20Boards#ARC-64
[2] SCUBA2 Bus Backplane ISA, SC2_ELE_S580_511
[3] SCUBA2 Data Acquisition Software Overview, Xiaofeng Gao, sc2-sof-s200-014-v1.pdf
[4] SCUBA-2 Data Acquisition to Data Processing Interface, Dennis Kelly, sc2-sof-s200-008rev.pdf
[5] Address-Card Technical Description, SC2_ELE_S584_501
[6] Bias-Card Technical Description, SC2_ELE_S583_501
[7] Clock-Card Technical Description, SC2_ELE_S581_501
[8] Readout-Card Technical Description, SC2_ELE_S582_501

3. Command Format

Commands are issued to the MCE from a host PC with a data-acquisition software installed (MAS or DAS). The nominal command format is the same in either.

In MAS, use *mce_cmd* to issue commands to the MCE.

```
> mce_cmd -x <action> <card_address> <parameter_id> <values>
```

for example:

```
> mce_cmd -x wb cc led 7
```

In the following sections, each of the command fields are described. Note that on the software side, list of card addresses and parameter ids need to match the MCE firmware. In MAS, the list of commands are captured in mce.cfg file and in DAS, they are captured in mce.xml file.

3.1 The <action> Field

In the MCE command formats for DAS and MAS shown above, the <action> field can take the following values:

- **wb**: write block – used to write values to registers in the MCE
- **rb**: read block – used to read values from registers in the MCE
- **go**: go – used to start a process in the MCE that requires several wishbone transactions (i.e. takes and extended period of time)
- **rs**: reset – used to issue commands for which replies must be returned by the MCE before executing the original command
- **st**: stop – used to stop a process that was started with the GO moniker. Note that "st" is implemented for a ret_dat commands as of 20080630.

3.2 The <card_address> Field

The MCE contains different cards. Each command targets either a particular card of the MCE or a group of cards. The <card_address> field can take the following values.

Single Cards:

- psc: Power Supply Card
- cc: Clock Card
- rc1: Readout Card 1
- rc2: Readout Card 2
- rc3: Readout Card 3
- rc4: Readout Card 4
- bc1: Bias Card 1
- bc2: Bias Card 2
- bc3: Bias Card 3
- ac: Address Card

Card Groups:

- rcs: All Readout Cards
- bcs: All Bias Cards
- sys: All Cards

3.3 The <parameter_id> Field

The <parameter_id> field can take the values that are outlined in the section below.

4. Parameter_ID/Command Descriptions

A list of MCE parameters or commands is provided below. The list is broken into sections depending on what card(s) of the MCE is being addressed. Note that RS- and GO-type commands have special reply-data – which are not listed in the "Reply Data" columns below. Keep in mind that each of the card types (i.e. Address/ Bias/ Readout/ Clock Cards) in the MCE run unique firmware. The "firmware revision" column indicates the version that the support for the particular parameter has been added. Dark-shaded rows indicate commands that remain to be implemented as of 16 May 2008.

4.1 General Card Commands

These commands can be issued to any card in the MCE (cc, rc1 to rc4, bc1, bc2, bc3, ac), except the Power Supply Card.

Parameter ID	Register Address	Possible Actions	Command Parameters	Reply Data	Description	Firmware revision
fpga_temp	0x91	rb	N/A	<fpga temperature>	Read the FPGA temperature of a card.	All
card_temp	0x92	rb	N/A	<card temperature>	Read the card temperature from a IC sensor	All
card_id	0x93	rb	N/A	<card id>	Read Card ID from an on-board silicon-id chip	All
card_type	0x94	rb	N/A	<card type>	Read Card Type and PCB revision. PCB revision is only added in later version of firmware/hardware. Bit 31 to 16: unused Bit 15 to 8: PCB Revision Bit 7 to 0: Card type ▪ Card Type 0: AC , 1: BC, 2: RC, 3: CC, 4: PSUC ▪ PCB Revision 1: A, 2: B, 3: C, 4: D, 5: E, ….., 26: Z	All
slot_id	0x95	rb	N/A	<slot id>	Request Slot ID.	All
fw_rev	0x96	rb	N/A	<RRrrBBBB>	Read firmware revision where the format in hex is RRrrBBBB ▪ RR: major revision number ▪ rr: minor revision number ▪ BBBB: build number	All
dip	0x97	rb	N/A	<dip-switch status>	Request DIP-switch status	None
led	0x99	wb, rb	<status>	<status>	Read/write the LEDs status. When writing to the LEDs, the value written is XOR'ed with the current value of the LED status, to produce the new status.	All
scratch	0x9A	wb, rb	<word0> to <word7>	<word0> to <word7>	These are read/write registers to be used for arbitrarily purposes. An example is to use one to detect undesired reset of MCE.	All
critical_error_rst	0x9B	wb,rb	<word0>	<word0>	writing 1 to this register triggers a reconfiguration of the FPGA from its configuration device and therefore the card would be unresponsive for few seconds.	5.2.0+
fpga_clr	0x9C	wb,rb	<word0>	<word0>	Writing 1 to this register clears all the registers in firmware, but NOT the RAM blocks in the FPGA.	5.2.0+
vfy_eeprom	0x42	wb, rb	<1 = verify>		Not Implemented. Trigger a comprehensive test of the EEPROM on any/all cards it is addressed to	None
clr_error	0x43	wb, rb	<1 = reset>		Not Implemented. Clear error codes that have been recorded on a card.	None
resync	N/A	wb, rb	<1 = reset>		Not Implemented. re-synchronize the start of its internal frame structure with the reception of the next Sync pulse.	None
eeprom	0x41	wb, rb	<word 0> … <word n-1>	<word 0> … <word n-1>	Not Implemented. Read/write 'n' 32-bit words to the EEPROMs on any/all cards starting from the specified starting address (see EEPROM Start).	None
eeprom_srt	0x44	wb, rb	<address>	<address>	Not Implemented. Specify a starting read/write address for the on-board EEPROM. The EEPROMs are AT25128A and are 128 KB in size. A 128 KB device holds 4096 32-bit words, so physical addresses for the EEPROM will range between 0x0000 and 0x1000.	None

4.2 “sys” Commands

These commands affect the timing of the overall system and they must be issued to all cards in the MCE at the same time. Use ‘sys’ card address to address all cards.

Parameter ID	Register Address	Possible Actions	Command Parameters	Reply Data	Description	Firmware revision
row_len	0x30	wb, rb	<# clock cycles>	<# clock cycles>	number of 50MHz clock cycles that are spent per row during multiplexing. Default = 64. Maximum = 65,535.	All
num_rows	0x31	wb, rb	<# rows>	<# rows>	number of rows to be multiplexed. Default = 41. Maximum = 2147483647/row_len.	All
num_rows_reported	0x55	wb, rb	<number of rows>	<number of rows>	number of rows of data that are reported in a data packet. This parameter is set in conjunction with the readout_row_index (for Readout Cards). See also num_cols_reported, readout_col_index. Default = 41.	4.0.0+
num_cols_reported	0xAD	wb	<# of columns>	<# of columns>	number of columns of data that are reported in a data packet. This parameter is set in conjunction with the readout_col_index (a Readout Card command). See also num_rows_reported, readout_row_index. Default = 8.	5.0.0+

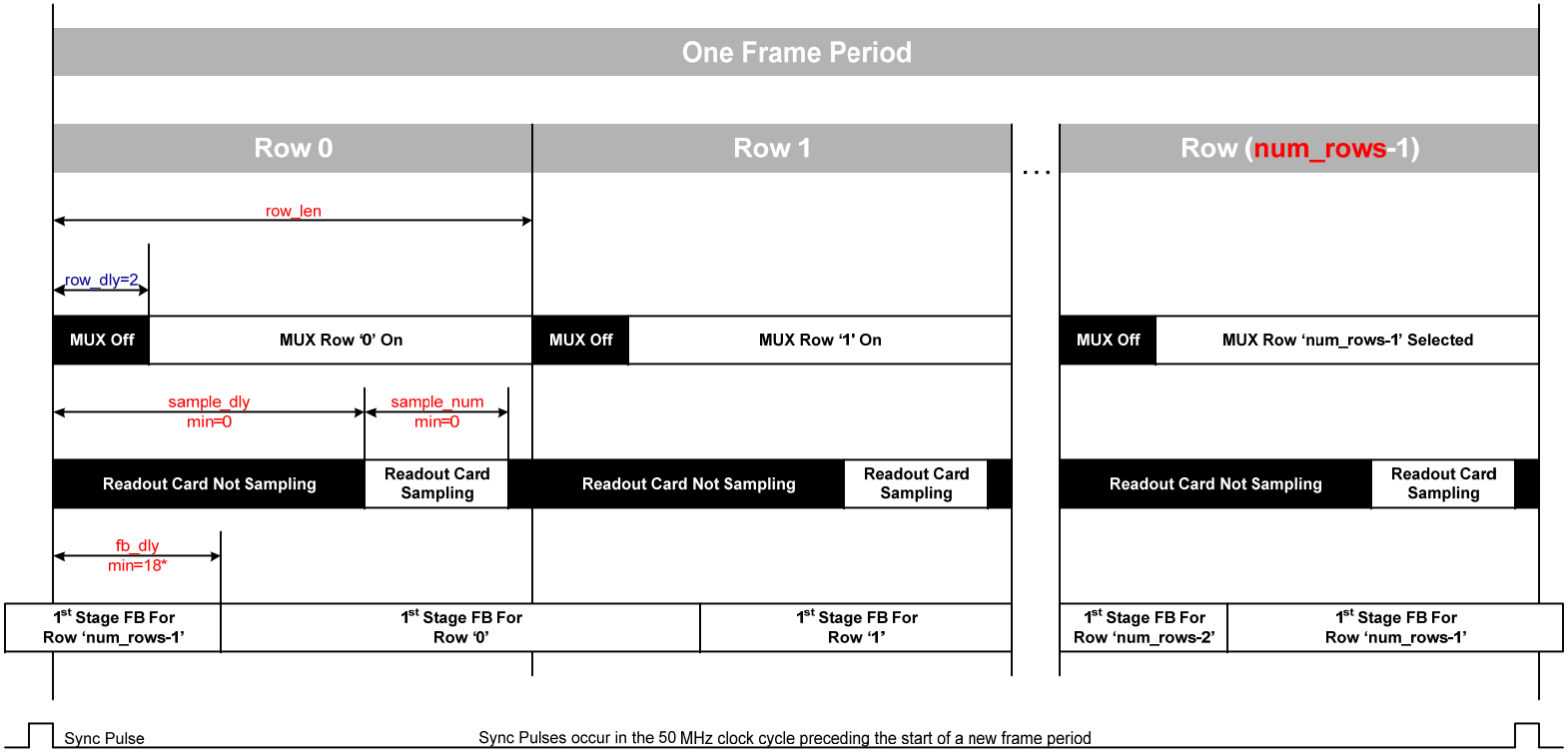

Figure 1. Frame Timing Diagram (Reference: \scuba2_repository\doc\ISA\frame_timing_structure.vsd, CVS version 1.6).

Figure 1 shows the frame-timing structure of the MCE. All the commands that affect timing in the MCE are listed in red. 'row_len' and 'num_rows' are addressable to all cards in the MCE using the 'sys' moniker. 'sample_dly', 'sample_num', 'fb_dly' are addressable to 'rcs'. Note that 'row_dly' is fixed.

Consult RC firmware release notes to find out the effective fb_dly in your firmware. In latest firmware, minimum fb_dly is 7 when flux-jumping off and 10 when flux jumping on.

4.3 "cc" Commands

These commands are for the Clock Card only. Dark-shaded rows indicate commands that remain to be implemented as of 16 May 2008.

Parameter ID	Register Address	Possible Actions	Command Parameters	Reply Data	Description	Firmware revision
config_app	0x52	rs	1	N/A	Configure the FPGA with the image in the application configuration device (EPC16).	3.0.1+
config_fac	0x51	rs	1	N/A	Configure the FPGA with the image in the factory configuration device (EPC16)	3.0.1+
ret_dat_s	0x53	wb, rb	<start sequence #> <end sequence #>	<start sequence #> <end sequence #>	Specify the starting and ending sequence-numbers for data frames that the MCE is to return. Each frame of data will have a sequence number affixed to it. Sequence numbers will be assigned to data frames in increasing order, beginning with the starting sequence number, and finishing with the ending sequence number. Both parameters may range between 0x00000000 and 0xFFFFFFFF. If both starting and ending sequence numbers are the same, then a single frame of data will be returned. If the starting sequence number is larger than the ending sequence number, then frames will be returned until the sequence number has wrapped around and reached the stopping sequence number. Starting and stopping sequence numbers are expected for all modes of data acquisition – including for DV pulses.	All
use_dv	0x54	wb, rb	<dv mode>	<dv mode>	Set the dv mode. The Clock Card can return data frames at an internal rate specified by 'data_rate', or in response to Data-Valid pulses from the fibre input or in response to packets from the Manchester input. Default = 0. ▪ dv mode 0: generated by the MCE ▪ dv mode 1: fibre DV input (labeled as DV_IN on the MCE) ▪ dv mode 2: fibre Manchester-encoded input (labeled as SYNC_IN on the MCE) If select_clk=0, then the user must set use_sync=0 and use_dv=0, otherwise timing artifacts will appear in data. The following combinations are OK: [select_clk, use_sync, use_dv] = [0,0,0] or [1,2,2] or [1,0,0] or [1,2,0] or [1,0,2].	All
array_id	0x58	rb		<array id>	Read the id of the sub-array that a MCE box is connected to.	All
box_id	0x59	rb		<id word 1>	Read the identification of the sub-rack. Each production sub-rack will have a unique Silicon ID IC mounted on the Bus Backplane.	All
sram_data	0x5C	wb, rb	<word 0> … <word n>	<word 0> … <word n>	Read/write a block of 32b data to SRAM at address specified by sram_addr parameter.(see sram_addr)	4.0.9+
sram_addr	0x5E	wb, rb	<starting address>	< starting address>	Specify the starting address for subsequent read/write operation to the 1024k x 32-bit SRAM. The, physical addresses range between 0x00000000 and 0x000FFFFF.	4.0.9+
data_rate	0xA0	wb, rb	<# frame periods per data frame>	<# frame periods per data frame>	Specify the rate at which data frames are collected by the MCE if DV pulses are not being used (see "Use DV Pulse"). The rate is specified as the time between data packets measured in frame periods, i.e. a value of 11 will return one data packet every 11 frame periods. A frame period is the amount of time required for the multiplexer to address all the rows on a MUX. Default = 47.	All
mce_bclr	0xAB	Rs	1	N/A	Clear the registers on all the cards in the MCE, including the Clock Card but excluding the PSC.	3.0.4+
cc_bclr	0xAC	rs	1	N/A	Clear the registers on the Clock Card.	3.0.4+
box_temp	0xA8	rb	1	<temperature>	Read the temperature sensor on the Bus Backplane	All

crc_err_en	0xA9	wb, rb	1	<mode>	Force the value of the CRC field in reply packets from the MCE to the PC. The reply packets will have a CRC checksum value of 0xFFFFFFFF. ▪ mode = 0: Normal CRC Calculation performed ▪ mode = 1: CRC word forced to 0xFFFFFFFF	3.0.1+
use_sync	0xA1	wb, rb	<sync mode>	<sync mode>	Specify the sync mode. The Clock Card can return to address 0 using its internal timing, or in response to sync pulses from the fibre input or in response to packets from the Manchester input. Default = 0. ▪ 0: generated by the MCE ▪ 1: fibre DV input (labeled as DV_IN on the MCE) ▪ 2: fibre Manchester-encoded input (labeled as SYNC_IN on the MCE) If select_clk=0, then the user must set use_sync=0 and use_dv=0, otherwise timing artifacts will appear in data. The following combinations are OK: [select_clk, use_sync, use_dv] = [0,0,0] or [1,2,2] or [1,0,0] or [1,2,0] or [1,0,2].	All
select_clk	0xA2	wb, rb	<clock source>	<clock source>	Select the source of the clock input to the Clock Card's PLL. There are two options for this, either the Manchester input clock, or the on-board crystal clock. Both are nominally 25 MHz. Default = 0. ▪ clock source = 0: generated by the MCE ▪ clock source = 1: fibre Manchester-encoded input. (labeled as SYNC_IN on the MCE) Note: when select_clk=1, and the external Manchester Clock disappears, the firmware automatically switches back to the internal clock. After the switch-back, reading back select_clk will return 0. Writing the same value to select_clk as what it is does not cause a clock glitch. Writing a different value causes a realignment of the PLL output clock to the new input, which takes two 25MHz clock cycles, or 80ns. See Altera Application Note 313. If select_clk=0, then the user must set use_sync=0 and use_dv=0, otherwise timing artifacts will appear in data. The following combinations are OK: [select_clk, use_sync, use_dv] = [0,0,0] or [1,2,2] or [1,0,0] or [1,2,0] or [1,0,2].	All
internal_cmd_mode	0xB0	wb, rb	1	<mode>	Enable or disable internal housekeeping/ ramping commands. Default = 0. ▪ mode = 0: Disable internal commands ▪ mode = 1: Enable Housekeeping commands ▪ mode = 2: Enable ramp commands ▪ mode = 3: Enable arbitrary waveform (AWG) commands	4.0.0+
ramp_step_period	0xB1	wb, rb	<# frame periods>	<# frame periods>	Specify the number of frame periods between each ramp step.	4.0.0+
ramp_min_val	0xB2	wb, rb	<min value>	<min value>	Specify the minimum value of the ramp that is to be applied. Must be less than the ramp_max_value	4.0.0+
ramp_step_size	0xB3	wb, rb	<step size in DAC units>	<step size in DAC units>	Specify the ramp step size. If (step_size) = [(max) – (min)] then the ramp will be a square wave.	4.0.0+
ramp_max_val	0xB4	wb, rb	<max value>	<max value>	Specify the maximum value of the ramp that is to be applied. Must be greater than the ramp_min_value	4.0.0+
ramp_param_id	0xB5	wb, rb	<parameter id>	<parameter id>	Specify the parameter ID of the register to be ramped	4.0.0+
ramp_card_addr	0xB6	wb, rb	<card address>	<card address>	Specify the card address of the register to be ramped	4.0.0+
ramp_step_data_num	0xB7	wb, rb	<# of values to ramp>	<# of values to ramp>	Specify the number of data that are to be written per ramp command	4.0.0+
ramp_step_phase	0xBB	wb,rb	<phase offset>	<phase offset>	Specify a phase offset to start the ramp.	5.0.b+
awg_sequence_len	0xB9	wb, rb	<sequence length>	<sequence length>	Specify the length of the Arbitrary Wave Generator (AWG) sequence that is loaded in RAM. This parameter tells the internal commanding FSM how many values to apply before restarting at the beginning.	5.0.3+
awg_data	0xBA	wb, rb	<data word 0> <data word 1> … <data word n>	<data word 0> <data word 1> … <data word n>	Load the Arbitrary Wave Generator (AWG) sequence into RAM on the Clock Card. The RAM has 8192 indexes. This may be expanded later if there are enough resources still available on the Clock Card.	5.0.3+
awg_addr	0xBC	wb, rb	<addr>	<addr>	Specify the starting address for reading from or writing to the Arbitrary Wave Generator (AWG) RAM. This parameter also affects the memory address of the internal commanding FSM.	5.0.3+

cards_present	0x5A	rb	1	<cards present>	Read which cards are present in the MCE. Bits are one-hot encoded, active high: Bit 9: Address Card Bit 8: Bias Card 1 Bit 7: Bias Card 2 Bit 6: Bias Card 3 Bit 5: Readout Card 1 Bit 4: Readout Card 2 Bit 3: Readout Card 3 Bit 2: Readout Card 4 Bit 1: Clock Card Bit 0: PSUC	4.0.8+
cards_to_report	0x5B	wb, rb	<cards to report data>	<cards to report data>	Specify which cards are present in the MCE subrack, and will return data over the bus backplane. This command can be used in situations where certain cards are not present in the subrack, and should not return data in replies. The default value for this register is '1' for every possible card. To stop a card from returning data, set its corresponding bit to '0'. Bit 9: Address Card Bit 8: Bias Card 1 Bit 7: Bias Card 2 Bit 6: Bias Card 3 Bit 5: Readout Card 1 Bit 4: Readout Card 2 Bit 3: Readout Card 3 Bit 2: Readout Card 4 Bit 1: Clock Card Bit 0: PSUC	4.0.a+
rcs_to_report_data	0x5F	wb, rb	<card address>	<card address>	This command is NOT redundant with cards_to_report. It applies only to Readout Cards, and only for data taking (ret_dat) commands. This register sets which Readout Cards return data during a data run. This command allows the selective return of data without affecting which cards respond to all other commands, which is useful if a data run fails. Bit 9: -- Bit 8: -- Bit 7: -- Bit 6: -- Bit 5: Readout Card 1 Bit 4: Readout Card 2 Bit 3: Readout Card 3 Bit 2: Readout Card 4 Bit 1: -- Bit 0: --	4.0.a+
stop_dly	0xB8	wb, rb	<timed delay>	<timed delay>	Specify the time delay between the return of the next data frame and the return of the reply to a stop command. The time specified is in microseconds. This command is used to test the robustness of the PCI card to replies following immediately on the heels of a data packet.	4.0.a+
run_id	0x56	wb, rb	<data file name embedded in header>	<data file name embedded in header>	Specify an ID number that will be stored in every data packet header returned to the PCs. In ACT, the ID number corresponds to the c-time at which data acquisition began. ACT's data files are also named by the same c-times, so that the data pipeline can easily track which files the data are from.	4.0.2+
user_writable	0x57	wb, rb	<misc data embedded in header>	<misc data embedded in header>	Specify a value that will be stored in every data packet header returned to the PCs. This register is used by ACT to store a combination of array_id and data_mode information. At Caltech, it is used during I-V curves to store the value of the TES bias applied.	4.0.2+

tms_tdi	0x50	wb	<# TMS/TDI pairs> <16pairs0> <16pairs1> … <16pairs56>		Specify 'n' doublets of packed JTAG data. The first word of the command specifies the total number of [TMS,TDI] doublets contained in the rest of the data = 'n'. The rest of the words consist of 'n' [TMS,TDI] doublets starting from the LSB. The TCK signal is implied. The JTAG FSM automatically runs through a TCK cycle after asserting each new doublet consisting of one TMS and one TDI bit. The 'tms_tdi' command replaces the 'jtag0' command. Command execution: 1- Apply [TMS,TDI] doublet to JTAG interface, 2- Assert TCK (automatic) and wait for tck_half_period. While waiting, store TDO (automatic) after tdo_sample_dly cycles after the assertion of TCK, 3- De-assert TCK (automatic) and wait for tck_half_period.	5.0.5+
tdo	0xAA	rb		<16bits0> <16bits1> … <16bits56>	Read back 'n' bits of packed JTAG data. Unlike the tms_tdi command above, the first word returned does not contain the number of bits read out. It is assumed that following every tms_tdi command, the tdo command will be called and will thus return the same number of single bits as there were doublets of [TMS,TDI] data transmitted. Every new 'tms_tdi' command overwrites the TDO data from the previous 'tms_tdi' command. The MCE Jam Player can ignore TDO data that is not important. The 'tdo' command replaces the 'jtag1' command.	5.0.5+
tdo_sample_dly	0xAE	wb, rb	<sample_dly>	<sample_dly>	Specify the number of clock cycles after the assertion of TCK that TDO should be sampled. Default = 2. The default value is tuned to the MCE, so leave this as is.	5.0.5+
tck_half_period	0xAF	wb, rb	<half_period>	<half_period>	Specify the number of clock cycles that TCK should remain high, and then remain low. Default = 8. The default value is tuned to the MCE, so leave this as is.	5.0.5+
jtag0	0xBD	wb	<word0>	<word0>	Obsolete. Apply the specified values for TDI, TMS, and TCK bits to the JTAG output lines on the FPGA. This command was initially implemented for the debugging MCE JAM Player. Current versions of the MCE JAM Player use 'tms_tdi' instead. This command was typically issued in quartets with the following pattern: 1- wb cc jtag0(TDI, TMS, TCK=0), 2- wb cc jtag0(TDI, TMS, TCK=1), 3- rb cc jtag1(TDO), 4- wb cc jtag0(TDI, TMS, TCK=0). Bit 7: --, Bit 6: TDI, Bit 5: --, Bit 4: --, Bit 3: --, Bit 2: --, Bit 1: TMS, Bit 0: TCK	5.0.4+ (exists, but not to be used)
jtag1	0xBE	Rb	<word1>	<word1	Obsolete. Read back one bit of data from the TDO JTAG input line to the FPGA. This command was initially used for the MCE JAM Player proof-of-concept. Current versions of the MCE JAM Player use 'tdo' instead. Bit 7: TDO, bit 6 to 0 unused.	5.0.4+ (exists, but not to be used)
jtag2	0xBF	wb	<word2>	<word2>	Obsolete. Enable the Clock Card FPGA's control of the JTAG chain. MCE JAM Player scripts must write jtag2 = 2 before executing a script to enable JTAG control, and jtag = 0 after executing a script to relinquish JTAG control. Default = 0. Bit 1: JTAG Control (active high), Bit 0 and Bit 2 to 7 are unused.	5.0.4+ (exists, but not to be used)
tes_tgl_en	0xA3	wb, rb	1	<mode>	obsolete. Enable launching a square wave on TES bias. Clock Card issues internal commands at timed intervals that toggle the TES Bias output between a low value and a high value. mode = 0: Disable TES Bias toggling mode = 1: Enable TES Bias toggling	Until 3.0.8
tes_tgl_max	0xA4	wb, rb	1	<high value>	obsolete. Specify the 'high' value of the TES Bias square wave	Until 3.0.8
tes_tgl_min	0xA5	wb, rb	1	<low value>	obsolete. Specify the 'low' value of the TES Bias square wave	Until 3.0.8
tes_tgl_rate	0xA6	wb, rb	1	<toggle rate>	▪ obsolete. Specify the half-period of the TES Bias square wave, in frame periods	Until 3.0.8
int_cmd_en	0xA7	wb, rb	1	<mode>	obsolete. Enable or disable internal housekeeping commands. Default = 0. mode = 0: Disable internal commandsob mode = 1: Enable internal commands	Until 4.0.0
upload_fw	0x50	wb, rb	<word 0>	<word 0>	Not implemented	None
ret_dat_req	0x5D	wb, rb	<1/0: enable/disable>	<1/0: enable/disable>	Not implemented.	None
config_jtag	0xAA	rs	1	N/A	Not implemented..	None

4.4 "ac" Commands

Parameter ID	Register Address	Possible Actions	Command Parameters	Reply Data	Description	Firmware revision
row_order	0x01	wb, rb	<1st row> <2nd row> … <41st row>	<1st row> <2nd row> … <41st row>	This command is relevant when enbl_mux = 1 or 2. Read/write the row-addressing order. A sequence of up to 41 rows is specified at once. The row numbers that are specified with this command refer to the physical channel numbers on the Address Card. Note: After this command is issued, a "flx_lp_init" command must be issued to all the readout cards to discard previous PID-loop calculations and to clean out the rest of the data pipeline. Default = 0,0,0,…	All
on_bias	0x02	wb, rb	<bias row 0> <bias row 1> … <bias row 39> <bias dark row>	<bias row 0> <bias row 1> … <bias row 39> <bias dark row>	Applies when enbl_mux = 1 or 3. Read/write the14-bit on-bias values for all 41 channels of the DACs on the Address Card. Bias values are stored in the order of physical channels on the Address card – which is not necessarily the multiplexing order specified with row_order. This command is nominally is used to specify the 1st Stage SQUID "on" biases, and are applied one at a time in the sequence specified by row_order. Default = 0,0,0,…	All
off_bias	0x03	wb, rb	<bias row 0> <bias row 1> … <bias row 39> <bias dark row>	<bias row 0> <bias row 1> … <bias row 39> <bias dark row>	Applies when enbl_mux = 1 or 3. Read/write the 14-bit off-bias values for all 41 channels of the row-addressing DACs on the Address Card. Bias values are stored in the order of physical channels on the Address card – which is not necessarily the multiplexing order specified with row_order. This command is nominally is used to specify the 1st Stage SQUID "off" biases. Default = 0,0,0,…	All
enbl_mux	0x05	wb, rb	<mode>	<mode>	Start/stop array multiplexing. Default: mode = 0. ▪ mode = 0: Multiplexing off. This is the default value. ▪ mode = 1: (changed starting in v2.0.5+) Multiplexing on. The Address Card turns one DAC 'on' and one DAC 'off' at every row switch. The multiplexed values are specified by on_bias and off_bias. In v2.0.4-, the values were applied starting at row_dly, and done by row_dly+4. Starting in v2.0.5+, the values are applied in the first two clock cycles after a row switch, and row_dly is obsolete. ▪ mode = 2: Multiplexing on. The Address Card changes the values on all the DACs at every row switch. The multiplexed values are specified by fb_colx. This mode is implemented starting in v2.0.5+. All 41 DAC values are applied in the first 4 clock cycles after a row switch, and row_dly is obsolete. ▪ mode=3: special multiplexing on. In the first heater_bias_len clock cycles of each row visit, the heater_bias values are applied to the DACs, the remaining time, the on_bias values are applied. See Figure 2 for timing.	All
bias_start	0x09	wb, rb	<start row 0> <start row 1> … <start row 39> <start dark row>	<start row 0> <start row 1> … <start row 39> <start dark row>	Applies only when enbl_mux = 1. Specify the point during each row dwell period when the 1st Stage Bias is applied. This is specified on a row-by-row basis so that SCUBA2 can differential bias heating across the rows of their arrays.	2.0.8; 5.0.1+
heater_bias	0x0A	wb, rb	<bias row 0> <bias row 1> … bias row 40>	<bias row 0> <bias row 1> … bias row 40>	When enbl_mux=3, specify the magnitude of the SQ1-bias heating pulse in DAC units (0 to 16,383) during the initial heater_bias_len period of each row visit. See Figure 2 for timing.	2.0.9, 5.0.2
heater_bias_len	0x0B	wb, rb	<# clock cycles>	<# clock cycles>	When enbl_mux=3, specifies the length of the 1st stage bias heating pulses at the beginning of every new row period. See Figure 2 for timing.	2.0.9, 5.0.2
row_dly	0x35	wb, rb	<# clock cycles>	<# clock cycles>	Obsolete. This command was in use until v2.0.4 for enbl_mux = 1 only. Specify the number of 50MHz clock cycles between the de-assertion of the previous row at a row-switch, and the start of the assertion of the current row on the Address Card (multiplexer). Minimum row_dly is 4. This command is not supported after 2.0.5 version.	Until2.0.4-;
fb_col0 … fb_col40 (See *)	0xC0 … 0xE8 (See *)	wb, rb	<fb_row_0> … <fb_row_39> <fb_row_dark>	<fb_row_0> … <fb_row_39> <fb_row_dark>	This command is relevant when enbl_mux = 2. Specifies the 41 multiplexing values for each DAC channel on the Address Card, and for each row that the channel is to multiplex. There are 41 channels on the Address Card, and up to 41 rows per channel that need to be specified. This command was implemented to support using the Address Card to multiplex the 2nd Stage Feedback from a Bias Card slot.	2.0.5+

const_mode	0x06	wb, rb	<mode_dac_0> <mode_dac_1> … <mode_dac_39> <mode_dac_dark>	<mode_dac_0> <mode_dac_1> … <mode_dac_39> <mode_dac_dark>	Specify which DAC channels are held at a constant value regardless of the value of enbl_mux. A constant value is applied once only, and is specified by const_val. In addition, DACs for which const_mode=1 do not receive data strobes during multiplexing – so that there are no glitches or excess noise on the DAC outputs. ▪ mode_dac_x = 0: this DAC channel is multiplexed when enbl_mux = 1, or enbl_mux = 2, or enbl_mux=3 ▪ mode_dac_x = 1: the corresponding DAC channel is held at a constant value and not strobed when enbl_mux = 1, or enbl_mux = 2.	2.0.6+
const_val	0x07	wb, rb	<val_dac_0> <val_dac_1> … <val_dac_39> <val_dac_dark>	<val_dac_0> <val_dac_1> … <val_dac_39> <val_dac_dark>	Specify values for the DACs that are held constant. <val_dac_n> is asserted by DAC 'n' when const_mode[n] = 1. In addition, all constant values are applied to each respective DAC when enbl_mux = 0. When <wb enbl_mux>, <wb const_mode>, <wb const_val> or <wb const_val39> commands are executed, all the DACs that are being held constant have their values re-strobed. This command was implemented to avoid strobing DACs that are connected to dead rows or columns, while multiplexing the rest. This effectively eliminates the noise in the cryostat associated with strobing DACs.	2.0.6+
const_val39	0x08	wb, rb	<val_dac_39>	<val_dac_39>	Specify the value that DAC 39 is to be held constant at. This value will be asserted if const_mode[39] = 1, or if enbl_mux = 0. This command was created to allow internal commands to ramp this DAC directly, and to avoid excess strobe noise, because it is connected to the TES Bias in some ACT cryostats.	2.0.7+

* Note that if Address-Card is plugged into a Bias-Card slot, then mapping of column 0 of Bias Card into column 0 of Address Card should be accounted for in the hardware-abstraction file, i.e., mce.cfg when running MAS and mce.xml when running DAS. For example, when Address-Card is plugged into BC2 slot to provide sq2fb values, the following entries are captured in mce.cfg:

```
{ name = "fb_col0"; id = 0xE3; count = 41; },
{ name = "fb_col1"; id = 0xE1; count = 41; },
{ name = "fb_col2"; id = 0xDF; count = 41; },
{ name = "fb_col3"; id = 0xDD; count = 41; },
{ name = "fb_col4"; id = 0xDB; count = 41; },
{ name = "fb_col5"; id = 0xD9; count = 41; },
{ name = "fb_col6"; id = 0xD7; count = 41; },
{ name = "fb_col7"; id = 0xD5; count = 41; },
{ name = "fb_col8"; id = 0xD3; count = 41; },
{ name = "fb_col9"; id = 0xD1; count = 41; },
{ name = "fb_col10"; id = 0xCF; count = 41; },
{ name = "fb_col11"; id = 0xCD; count = 41; },
{ name = "fb_col12"; id = 0xCB; count = 41; },
{ name = "fb_col13"; id = 0xC9; count = 41; },
{ name = "fb_col14"; id = 0xC7; count = 41; },
{ name = "fb_col15"; id = 0xC5; count = 41; },
{ name = "fb_col16"; id = 0xE2; count = 41; },
{ name = "fb_col17"; id = 0xE0; count = 41; },
{ name = "fb_col18"; id = 0xDE; count = 41; },
{ name = "fb_col19"; id = 0xDC; count = 41; },
{ name = "fb_col20"; id = 0xDA; count = 41; },
{ name = "fb_col21"; id = 0xD8; count = 41; },
{ name = "fb_col22"; id = 0xD6; count = 41; },
{ name = "fb_col23"; id = 0xD4; count = 41; },
{ name = "fb_col24"; id = 0xD2; count = 41; },
{ name = "fb_col25"; id = 0xD0; count = 41; },
{ name = "fb_col26"; id = 0xCE; count = 41; },
{ name = "fb_col27"; id = 0xCC; count = 41; },
{ name = "fb_col28"; id = 0xCA; count = 41; },
{ name = "fb_col29"; id = 0xC8; count = 41; },
```

{ name = "fb_col30"; id = 0xC6; count = 41; },
{ name = "fb_col31"; id = 0xC4; count = 41; }

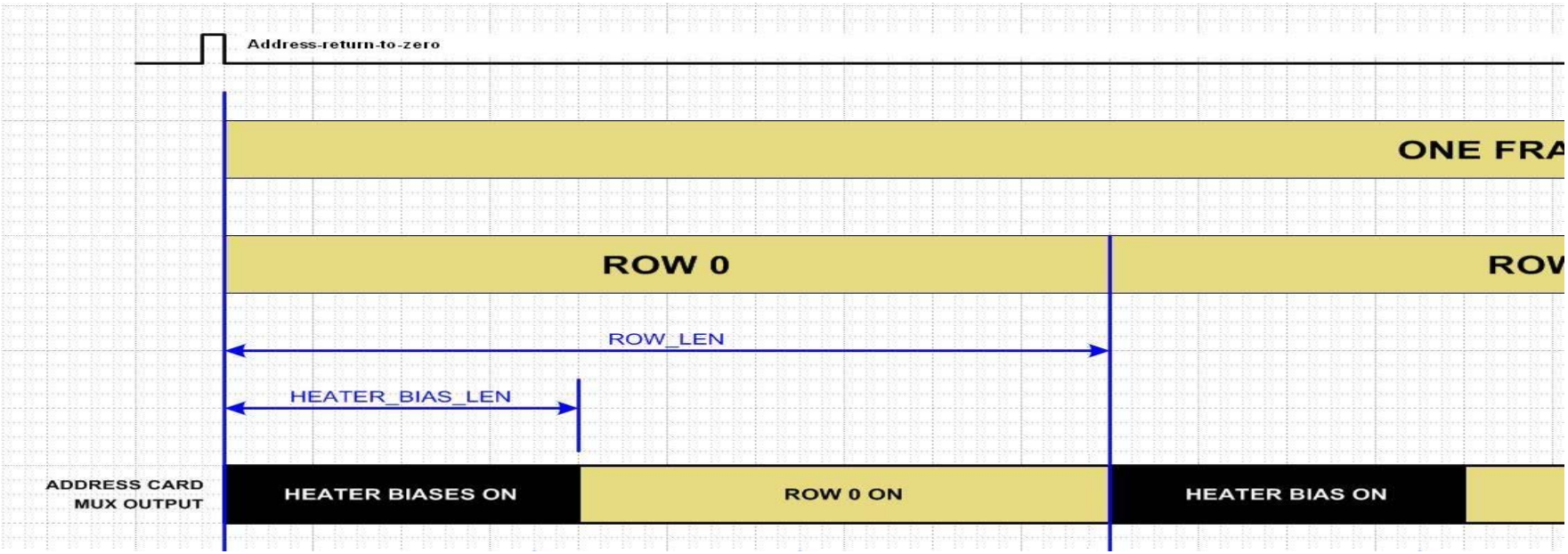

Figure 2. Timing diagram of AC bias lines when enbl_mux=3

4.5 "rc1, rc2, rc3, rc4" Commands

These commands target the Readout Cards only.

Parameter ID	Register Address	Possible Actions	Command Parameters	Reply Data	Description	Firmware Revision
sa_bias	0x10	wb, rb	<column 0 SA bias> <column 1 SA bias> … <column 7 SA bias>	<column 0 SA bias> <column 1 SA bias> … <column 7 SA bias>	Read/write the SQUID Series Array Bias values for all columns on a Readout Card.	All
offset	0x11	wb, rb	<column 0 offset> <column 1 offset> … <column 7 offset>	<column 0 offset> <column 1 offset> … <column 7 offset>	Read/write the SQUID Series Array Offset values for all columns on a Readout Card.	All
gainp0 gainp1 gainp2 gainp3 gainp4 gainp5 gainp6 gainp7	0x70 0x71 0x72 0x73 0x74 0x75 0x76 0x77	wb, rb*	<column x, gainp 0> <column x, gainp 1> … <column x, gainp 38> <column x, gainp 39> <column x, gainp dark>	<column x, gainp 0> <column x, gainp 1> … <column x, gainp 38> <column x, gainp 39> <column x, gainp dark>	Read/write the Readout Card's 41 P coefficients for any given column. The <x> in the command moniker refers to which column the values are being specified for, from 0 to 7. The Parameter IDs are listed in order of column, from 0 to 7. gainp are specified as "signed 10-bit" values as of revision 4.0.2. (These were 8-bit values prior to revision 4.0.2.)	All *
gaini0 gaini1 gaini2 gaini3 gaini4 gaini5 gaini6 gaini7	0x78 0x79 0x7A 0x7B 0x7C 0x7D 0x7E 0x7F	wb, rb*	<column x, gaini 0> <column x, gaini 1> … <column x, gaini 38> <column x, gaini 39> <column x, gaini dark>	<column x, gaini 0> <column x, gaini 1> … <column x, gaini 38> <column x, gaini 39> <column x, gaini dark>	Read/write the Readout Card's 41 I coefficients for any given column. The <x> in the command moniker refers to which column the values are being specified for, from 0 to 7. The Parameter IDs are listed in order of column, from 0 to 7. gaini are specified as "signed 10-bit" values as of revision 4.0.2. (These were 8-bit values prior to revision 4.0.2.)	All *
flx_quanta0 flx_quanta1 flx_quanta2 flx_quanta3 flx_quanta4 flx_quanta5 flx_quanta6 flx_quanta7	0x80 0x81 0x82 0x83 0x84 0x85 0x86 0x87	wb, rb	<column x, quanta 0> <column x, quanta 1> … <column x, quanta 38> <column x, quanta 39> <column x, quanta drk>	<column x, quanta 0> <column x, quanta 1> … <column x, quanta 38> <column x, quanta 39> <column x, quanta drk>	Read/write the Readout Card's 41 Flux Quanta values for any given column. The Flux Quanta are used to calculate the 1st Stage Feedback value. The <x> in the command moniker refers to which column the values are being specified for, from 0 to 7. The Parameter IDs are listed in order of column, from 0 to 7.	All
gaind0 gaind1 gaind2 gaind3 gaind4 gaind5 gaind6 gaind7	0x88 0x89 0x8A 0x8B 0x8C 0x8D 0x8E 0x8F	wb, rb*	<column x, gaind 0> <column x, gaind 1> … <column x, gaind 38> <column x, gaind 39> <column x, gaind dark>	<column x, gaind 0> <column x, gaind 1> … <column x, gaind 38> <column x, gaind 39> <column x, gaind dark>	Read/write the Readout Card's 41 D coefficients for any given column. The <x> in the command moniker refers to which column the values are being specified for, from 0 to 7. The Parameter IDs are listed in order of column, from 0 to 7. gaind are specified as "signed 10-bit" values as of revision 4.0.2. (These were 8-bit values prior to revision 4.0.2.)	All *

adc_offset0 adc_offset1 adc_offset2 adc_offset3 adc_offset4 adc_offset5 adc_offset6 adc_offset7	0x68 0x69 0x6A 0x6B 0x6C 0x6D 0x6E 0x6F	wb, rb	<column x, offset 0> <column x, offset 1> … <column x, offset 38> <column x, offset 39> <column x, offset dark>	<column x, offset 0> <column x, offset 1> … <column x, offset 38> <column x, offset 39> <column x, offset dark>	Read/write the Readout Card's 41 ADC Offsets for any given column. The ADC Offset coefficients are used to digitally offset each 50 MHz ADC sample on the Readout Cards. The offset is subtracted from each 50 MHz ADC sample before being co-added. The <x> in the command moniker refers to which column the values are being specified for, from 0 to 7. The Parameter IDs are listed in order of column, from 0 to 7.	All
readout_row_index	0x13	wb, rb	<starting row index>	<starting row index>	Specify the starting row index of data to be returned in a data block. Default = 0. Note: if num_rows_reported > num_rows – readout_row_index – 1, then the readout row index will wrap accordingly to 0 during readout.	4.0.0+
ret_dat	0x16	go, st	<1 = return frames>		Request data frames. Prior to a ret_dat command, the Clock Card expects a "ret_dat_s" command to set up how many frames are to be collected and what sequence numbers to assign. Following that, a single "ret_dat" command triggers the return of n frames of data.	All
en_fb_jump	0x15	wb, rb	<1 = enable flux jumping>	<1 = enable flux jumping>	Enable/disable flux jumping on the Readout Cards. The size of flux jumps is specified by the 'flux_quanta' command.	All
data_mode	0x17	wb, rb	<data mode>	<data mode>	data read-out mode. Various data can be read out: the coadded error, the SQ1 feedback, the filtered feedback, the flux-jump counter, the raw 50MHz data, or a combination of the above. (Consult readout card description document [8] or wiki for more information.) 0: Signed 32-bit co-added error data: error[31:0] 1: signed 32-bit SQ1 feedback (*fb*) , when servo_mode=3, $fb_{DAC}=fb/2^{12}$ is applied to the DAC. 2: signed 32-bit low-pass filtered SQ1feedback: fltr[31:0] 3: raw 50MHz error data from all 8 columns: raw[13:0] 4: mixed data: 18b SQ1_fb + 14b coadded error: fb[31] & fb[29:12] & coadd[31] & coadd[12:0] 5: mixed data: 24b SQ1_fb + 8b flux-jump counter: fb[31:8] & fj[7:0] 6: mixed data: 18b filtered fb + 14b coadded error: fltr[28:11] & error[31] & error[12:0] 7: mixed data: 22b filtered fb + 10b coadded error: fltr[28:7] & error[31] & error[12:4] 8: mixed data: 24b filtered fb + 8b flux-jump counter: fltr[31:8] & fj[7:0] 9: mixed data: 24b filtered fb + 8b flux-jump counter: fltr[31] & fltr[23:1] & fj[7:0] 10: mixed data: 25b filtered fb + 7b flux-jump counter: fltr[27:3] & fj[6:0] 11: pixel addresses: 6 bits of row index + 3 bits of column index: row[8:3] & col[2:0] 12: raw data sampled at 50MHz (see captr_raw command): raw[13:0]	0: All 1: All 2: 1.4.1+ 3: Special Req. 4: All 5: All 6: 3.0.30 – 4.0.6 7: 4.0.2+ 8: 4.0.4 only 9: 4.0.5 – 4.0.a 10: 4.0.b+ 11: 5.0.0+ 12: 4.0.d, 4.0.e, 5.0.1+
captr_raw	0x18	wb, rb	<1 = capture data>	N/A	Captures a 64k timestream snapshot of the error signal when Address-Return-to-Zero is asserted. 65,536 50MHz ADC samples from the channel specified by readout_col_index,are stored in the raw-data buffer on the Readout Card. This command was only available in special builds of firmware, but after v5.0.1 it is included in all revisions.	3.0.19, 4.1.7, 4.2.7, 4.3.7, 4.0.d, 4.0.e, 5.0.1+
servo_mode	0x1B	wb, rb	<column 0> ….. <column 7>	<column 0> …. <column 7>	Set the servo on the Readout Cards. Servo modes include: ▪ Servo mode 0 or 1: constant feedback mode as specified by fb_const parameter ▪ Servo mode 2: ramp mode ▪ Servo mode 3: PID-loop lock mode	All
ramp_dly	0x1C	wb, rb	<# frames>	<# frames>	Specify the number of frame-periods before the next step in a ramp function. This delay can be tuned to the data-frame readout rate (data_rate), so that one frame of data is read out per ramp step.	All
ramp_amp	0x1D	wb, rb	<ramp amplitude>	<ramp amplitude>	Specify the maximum ramp value, in 14-bits	All
ramp_step	0x1E	wb, rb	<ramp step size>	<ramp step size>	Specify what the step size of each step in the ramp is, max 14-bit	All

fb_const	0x1F	wb, rb	<col0> <col1> <col2> <col3> <col4> <col5> <col6> <col7>	<col0> <col1> <col2> <col3> <col4> <col5> <col6> <col7>	Specify the constant feedback DAC value for a specific column when the servo is in constant mode (see servo_mode). Applied to SQ1_FB. Max =14-bit (-8192 to 8191). The constant value is applied to the Readout Card DAC repetitively for every new row. (	All
sample_dly	0x32	wb, rb	<# clock cycles>	<# clock cycles>	Specify the number of 50MHz clock cycles from the start of a row during which ADC samples do not contribute to a co-added value on the Readout Cards.	All
sample_num	0x33	wb, rb	<# samples>	<# samples>	Specify the number of 50MHz samples co-added during a row-period on the Readout Cards. Co-addition begins after the sample delay, and one value is co-added per 50MHz clock cycle. Note that default is 0 and therefore you need to explicitly set this parameter.	All
fb_dly	0x34	wb, rb	<# clock cycles>	<# clock cycles>	Specify the number of 50MHz clock-cycles between the start of a row and the assertion of the row's feed-back value on the Readout Cards. (minimum is 7 with flux-jumping off, 10 or 18 (depending on firmware revision) when flux-jumping is on).	All
flx_lp_init	0x37	wb, rb	<1 = initialize>	N/A	Initialize/reset the PID loop calculations on the Readout Cards for new PID parameters to take effect.	All
readout_col_index	0x19	wb, rb	<starting column index>	<starting column index>	Specify the starting column index of data to be returned in a data block. Default = 0. Note: if num_cols_reported > num_cols – readout_col_index – 1, then the readout column index will wrap accordingly to 0 during readout. During raw readout, this parameter specifies the index of the column for which raw data is stored.	4.0.d+ (except 5.0.0)
readout_priority	0x67	wb, rb	<priority>	<priority>	Specify whether readout should be column-major or row-major. Default = 0. ▪ Priority 0: Data readout is in row-major order – all the pixels in a row are read out before moving on to the next row. ▪ Priority 1: Data readout is in column-major order – all the pixels in a column are read out before moving on to the next column.	5.0.0+
Integral_clamp	0x66	wb, rb	<clamp_value>	<clamp_value>	This parameter is used to clamp the integral term used in SQ1 feedback calculation. Once this value is reached, the integral term is permanently clamped and can only be reset by issuing a flx_lp_init command or reset. This is invented to prevent ramping of unlocked pixels and mitigate the adverse effects of the ramping pixels on locked pixels. To calculate the appropriate integral_clamp value for your experiment, use: integral_clamp ~ 0.9 * I_{max} = 8192 * 4096 / gaini = 30,000,000 / gaini (flux-jump disabled) 127 * 4096 * flux_quantum / gaini = 470000 * flux_quantum / gaini (flux-jump enabled) Note: The default value is 0 or no clamping is in effect. (see FSFB Clamping Commands on mcewiki)	5.0.9+
servo_rst_arm	0xf8	wb,	<1=reset specified servos>	<1=reset servo for flagged detectors>	The flux-loop servo can be reset on a per-detector basis without having to stop the data acquisition. When servo_rst_arm is set to 1, the flux-loop servo on all detectors flagged by servo_rst_col are reset.	

servo_rst_col0 servo_rst_col1 servo_rst_col2 servo_rst_col3 servo_rst_col4 servo_rst_col5 servo_rst_col6 servo_rst_col7	0xf0 0xf1 0xf2 0xf3 0xf4 0xf5 0xf6 0xf7	wb,rb	<row0,... row40> <row0,... row40> <row0,... row40> <row0,... row40> <row0,... row40> <row0,... row40> <row0,... row40> <row0,... row40>	<1=reset servo when servo_rst_arm is issued> <0= do not reset servo when servo_rst_arm is issued>	The flux-loop servo can be reset on a per-detector basis without having to stop the data acquisition. A per-detector flag is set using servo_rst_colx parameters and then a servo_rst_arm is set to 1. To reset servo on col2 row 2 and 4: wb rc1 servo_rst_col2 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 wb rc1 servo_rst_arm 1	
fltr_rst	0x14	wb	<1 =reset filterpipeline>	-	Reset the filter data pipeline. The flx_lp_init command automatically triggers a fltr_rst.	All
fltr_coeff	0x1A	wb, rb	<b11, b12, b21, b22, k1, k2>	<b11, b12, b21, b22, k1, k2 >	Specifies a set of 6 filter parameters for the low-pass Butterworth filter, the first 4 are the biquad coefficients followed by k1 and k2. The low-pass filter is implemented as two cascaded Butterworth biquads where each biquad is a 2nd order polynomial: $H(z) = (1 + 2*z^{-1} + z^{-2}) / (1 + b11*z^{-1} + b12*z^{-2}). 2^{-k1}. (1 + 2*z^{-1} + z^{-2}) / (1 + b21*z^{-1} + b22*z^{-2}). 2^{-k2}$ b11, b12, b21, b22: butterworth coefficients k1: number of bits to trim from the output of the first biquad before feeding to the 2nd biquad k2: number of bits to trim from the filter output before reading out. See details of these coefficients on mce wiki.	5.1.0
fltr_type	0x65	rb	-	<filter_type>	Up to version 5.0.7 all firmware implemented : low-pass filter: f_{3dB}=122Hz for f_{sample}=15kHz. Version 5.0.7 implemented f_{3dB}=75Hz for f_{sample}=30kHz. To identify different versions, fltr_type parameter is introduced starting 5.0.a. ▪ filter_type = 1 f_{cutoff} / f_{samp} = 122.226Hz / 15151Hz ▪ filter_type = 2 f_{cutoff} / f_{samp} = 75Hz / 30000Hz ▪ filter_type = 255 configurable coefficients	5.0.a+
pterm_decay_bits	0x64	rb, wb		<number of bits>	The p-term is now implemented as p(n) = er(n) + b*p(n-1) where b=(1-1/2^k). k is specified by the parameter pterm_decay_bits where k=0 is the default and behaves the same as older firmware.	5.1.a+

*In Pre-4.0.e firmware that has raw-mode enabled read back of gaini, gainp, gaind parameters is disabled to save memory for raw mode.

4.6 "bc1, bc2, bc3" Commands

These commands are for Bias Cards only. Dark-shaded rows indicate commands that remain to be implemented as of 2008-05-16.

Parameter ID	Addr	Actions	Command Parameters	Reply Data	Description	Firmware revision
flux_fb	0x20	wb, rb	<flux fb, column 0> <flux fb, column 1> … <flux fb, column 31>	<flux fb, column 0> <flux fb, column 1> … <flux fb, column 31>	Specify the flux feedback for all 32 16-bit DAC channels on the Bias Cards. In MCEv1, the nominal mappings are: ▪ Bias Card 1 ↔ SQUID Series Array Feedback (32 channels), ▪ Bias Card 2 ↔ 2nd Stage SQUID Feedback (32 channels), ▪ Bias Card 3 ↔ 2nd Stage SQUID Bias (32 channels). In MCEv2, the nominal mappings are: ▪ Bias Card 1 ↔ SQUID Series Array Feedback (lower 16 channels), ▪ Bias Card 1 ↔ 2nd Stage SQUID Bias (upper 16 channels), ▪ Bias Card 2 ↔ 2nd Stage SQUID Feedback (lower 16 channels), ▪ Bias Card 3 ↔ TES Bias (lower 16 channels).	All
bias	0x21	wb, rb	<bias value>	<bias value>	Read/Write the low-noise LVDS 16-bit DACs that are typically used to supply TES bias or pixel heater currents. In early MCE/Bias Card versions, nominal mappings are: Bias Card 1 ↔ Pixel Heater, Bias Card 2 ↔ TES Bias. The latest Bias Cards have 12 low-noise bias lines.	All
flux_fb_upper	0x24	Wb,rb	<flux fb, column 16> <flux fb, column 17> … <flux fb, column 31>	<flux fb, column 16> <flux fb, column 17> … <flux fb, column 31>	Specify the flux feedback for 16 upper 16-bit DACs, i.e. channel 16 to 31 on bias cards. This command was implemented to match the cryostat wiring for the MCEv2 subracks, which use the upper 16 channels of BC1 to supply the SQ2 Bias.	1.4.0 +
fb_col0 … fb_col31	0xC0 … 0xDF	wb, rb	<fb_row_0> … <fb_row_40>	<fb_row_0> … <fb_row_40>	When enbl_mux = 1 or multiplexing mode is enabled, then these commands specify the 41 multiplexing values for each of the 32 DAC channels (fb_colx) on Bias Card.	5.0.3+
enbl_mux	0x05	wb, rb	<col0 mux mode> ….. <col31 mux mode>	<col0 mux mode> ….. <col31 mux mode>	enable /disable multiplexing DAC values. Default: mode = 0 (disabled). ▪ mode = 0 disables multiplexing on the particular column. The column DAC is updated only once when a new value is commanded by issuing a flux_fb command. This is the legacy behavior prior to introducing enbl_mux and fb_col parameters. ▪ mode = 1 enables multiplexing on the particular column. The column DAC is updated at every row visit by values specified through fb_col parameters. This mode is implemented starting in v5.0.3+. All 32 DAC values are applied one clock cycles after a row switch. But obviously there is additional hardware delay .	5.0.3+
enbl_flux_fb_mod	0x25	wb,rb	<col0 mod enable> … <col31 mod enable>	<col0 mod enable> … <col31 mod enable>	Enable/disable the addition of *mod_val* to *flux_fb* values for regular DACs. • 1 : DAC is loaded with (*flux_fb* + *mod_val*) • 0 : DAC is loaded with value specified by *flux_fb* parameter	5.3.0+
enbl_bias_mod	0x26	wb,rb	<ln_bias0 mod enable> … <ln_bias11 mod enable>	<ln_bias0 mod enable> … <ln_bias11 mod enable>	Enable/disable the addition of mod_val to bias values for LN_BIAS DACs • 1 : ln_bias DAC is loaded with value specified by (*bias* + *mod_val*) • 0 : DAC is loaded with value specified by *bias* parameter	5.3.0+
mod_val	0x27	wb,rb	<val>	<val>	Modulation value to be added to either of *flux_fb* or *bias* values depending on whether modulation is enabled or not. (see enbl_flux_fb_mod and enbl_bias_mod)	5.3.0+
sa_htr0	0x22	wb, rb	<heater value>	<heater value>	Not Implemented. Read/write a from/to the Series Array Heather channel 0. This is used to release trapped flux from the Series Array Modules.	None
sa_htr1	0x23	wb, rb	<heater value>	<heater value>	Not Implemented. Read/write a heater value from/to the Series Array Heather channel 1. This is used to release trapped flux from the Series Array Modules.	None

4.7 "psc" Commands

These commands are for the Power Supply Card (a switching power supply designed at UBC), only. The Power Supply Card firmware blocks are integrated in the Clock Card firmware as a virtual card. This is why the firmware revision refers to the Clock Card. *Most Modern MCEs are not using a PSC anymore and therefore these commands can be considered* ***obsolete****.*

Parameter ID	Register Address	Possible Actions	Command Parameters	Reply Data	Description	Firmware revision
brst	0x60	rs	1	N/A	Cause the Power Supply Card to assert the RBst line (active low) on the bus backplane to reset all FPGA devices in the MCE. Asserting this signal causes all FPGA devices in the MCE to be reconfigured, including the Clock Card FPGA. The Clock Card FPGA will be reconfigured by the Factory reconfiguration device.	2.0.c+
cycle_pow	0x61	rs	1	N/A	Cause the Power Supply Card to cycle power to the MCE. All power supplies will be switched off and then back on again in proper sequence. This will cause all FPGA devices in the MCE to be reconfigured, including the Clock Card FPGA. The Clock Card FPGA will be reconfigured by the Factory configuration device.	2.0.c+
psc_status	0x63	rb	N/A	<status word 0> <status word 1> … <status word 35>	Report power-supply-card status information. Only available when UBC-supplied switching supplies are present. The 36 bytes (octets) are defined as follows: SILICON_ID 0,1,2,3 // Read from DS18S20 LS 32 bits of 48 SOFTWARE_VERSION 4 // Software Version FAN1_TACH 5 // Fan 1 speed /16 FAN2_TACH 6 // Fan 2 speed /16 PSU_TEMP_1 7 // temperature 1 from DS18S20 PSU_TEMP_2 8 // temperature 2 from DS18S20 PSU_TEMP_3 9 // temperature 3 from DS18S20 ADC_OFFSET 10,11 // Grounded ADC input channel reading V_VCORE 12,13 // +Vcore supply scaled 0 to +2V V_VLVD 14,15 // +Vlvd supply scaled 0 to +2V V_VAH 16,17 // +Vah supply scaled 0 to +2V V_VA_PLUS 18,19 // +Va supply scaled 0 to +2V V_VA_MINUS 20,21 // -Va supply scaled 0 to +2V I_VCORE 22,23 // Current +Vcore supply scaled I_VLVD 24,25 // Current +Vlvd supply scaled I_VAH 26,27 // Current +Vah supply scaled I_VA_PLUS 28,29 // Current +Va supply scaled I_VA_MINUS 30,31 // Current -Va supply scaled STATUS_WORD 32,33 // undefined place for status word ACK_NAK 34 // either ACK or NAK CHECK_BYTE 35 // checksum byte	2.0.c+
cut_pow	0x62	rs	1	N/A	Cause the Power Supply Card to turn off all power bricks.	2.0.c+