Bus Backplane Instructions

CC<->Bus-Backplane Instructions	Paired Instructions	Dennis Kelly's Corresponding Command	Size of Optional Fields in Bytes (4 bits)	Command Code (8 bits) Shown in Hex	Card Address (10 bits) Shown in Binary	Memory Address A (8 bits)	Memory Address B (8 bits)	Data (11 Bytes Max)	Notes	Reply Format
Reserved				0x00					This command code is reserved, because it is the default value for command code register initialization in VHDL	N/A
Address Card Specific										
Read Active Row	Write Active Row	get FPGA memory; select pixel column <number> ???; select all pixel columns ???	1/4	0x01	0011111000	Bit-stuff with 1's	Bit-stuff with 1's	"0x00".."0x28" (row#)	Read/Write which row is currently active. When writing, all other rows will be automatically switched to their 'Off' values, and the designated row will be switched to it's 'On' value. The RC's will also be targeted by this command because their 1st stage feedback may depend on what row is currently active.	Replies take the same format and length as a Write Instruction, with the latest value of the read/written parameter in the Data field
Read Row 'On' Value	Write Row 'On' Value	get FPGA memory; set first stage SQUID bias <row> <voltage number>	2/5	0x02	0010000000	"0x00".."0x28" (row#)	Bit-stuff with 1's	"0x0000".."0x3FFF" (14-bit#)	Read/Write the first stage SQUID 'on' bias for a specific row. This command will not change the which row is currently active. However, if the 'on' bias value is set while the row is active, the DAC output will be updated immediately.	Replies take the same format and length as a Write Instruction, with the latest value of the read/written parameter in the Data field
Read Row 'Off' Value	Write Row 'Off' Value	get FPGA memory	2/5	0x03	0010000000	"0x00".."0x28" (row#)	Bit-stuff with 1's	"0x0000".."0x3FFF" (14-bit#)	Read/Write the first stage SQUID 'off' bias for a specific row. This command will not change the which row is currently inactive. However, if the 'off' bias value is set while the row is inactive, the DAC output will be immediately updated.	Replies take the same format and length as a Write Instruction, with the latest value of the read/written parameter in the Data field
				0x04						
				0x05						
				0x06						
				0x07						
				0x08						
				0x09						
				0x0A						
				0x0B						
				0x0C						
				0x0D						
				0x0E						
				0x0F						
Readout Card Specific										
Read 1st Stage Feedback	Write 1st Stage Feedback	get FPGA memory	3/5	0x10	000????000	"0x00".."0x28" (row#)	"0x00".."0x07" (column#)	"0x0000".."0x3FFF" (14-bit#)	Read/Write the 1st stage feedback value for a specific pixel. The 1st stage feedback may be different for every pixel in a column, and thus may depend on what row is active. If the 1st stage feedback is updated for a pixel that is currently active, the DAC output will be immediately updated.	Replies take the same format and length as a Write Instruction, with the latest value of the read/written parameter in the Data field
Read SA Bias	Write SA Bias	get FPGA memory; set SQUID series array bias <column> <voltage number>	3/5	0x11	000????000	Bit-stuff with 1's	"0x00".."0x07" (column#)	"0x0000".."0xFFFF" (16-bit#)	Read/Write the SA bias value. A written value is output by the SA Bias DAC immediately.	Replies take the same format and length as a Write Instruction, with the latest value of the read/written parameter in the Data field
Read Offset	Write Offset	get FPGA memory	3/5	0x12	000????000	Bit-stuff with 1's	"0x00".."0x07" (column#)	"0x0000".."0xFFFF" (16-bit#)	Read/Write the offset value. A written value is output on the offset DAC immediately.	Replies take the same format and length as a Write Instruction, with the latest value of the read/written parameter in the Data field
Read Filter Value	Write Filter Value	get FPGA memory; set filter value <filter index> <weight value>	3/5	0x13	000????000	"0x00".."0x03" (filter#)	"0x00".."0x07" (column#)	"0x0000".."0xFFFF" (16-bit#)	Read/Write a filter coefficient of a specific filter, at a specific filter index. There will be 4 filters available in Science Mode, 1 filter availble in Engineering Mode A2, and 1 filter available Engineering Mode B. No other modes require specific filter coefficients. Depending on what filtering techniques are used, the filter indicies for each may be used differently. All possible modes are outlined in MH's document "Functional Description of the Multi-Channel Electronics" available on the SCUBA2 MCE PDR website (http://www.physics.ubc.ca/~scuba2/sc2pdr/index.html).	Replies take the same format and length as a Write Instruction, with the latest value of the read/written parameter in the Data field
				0x14						
				0x15						
				0x16						
				0x17						
				0x18						
				0x19						
				0x1A						
				0x1B						
				0x1C						
				0x1D						
				0x1E						

				0x1F						
Bias Card Specific										
Read Feedback (2nd Stage FB, 2nd Stage Bias, or SA FB)	Write Feedback (2nd Stage FB, 2nd Stage Bias, or SA FB)	get FPGA memory; set second stage SQUID feedback <column> <vltg num>; set second stage SQUID bias <column> <vltg num>; set SQUID series array feedback <column> <vltg num>	3/5	0x20	0000000???	Bit-stuff with 1's	"0x00".."0x07" (column#)	"0x0000".."0xFFFF" (16-bit#)	Read/Write the 2nd Stage FB, 2nd Stage Bias, or SA FB for a specific column. Changing a value that is being output will immediately update the output. Note that each of the three bias cards control one of the 2nd Stage FB, 2nd Stage Bias, or SA FB. Therefore, access to any one of these values is determined by what card is targeted by the Card Address field. During array multiplexing, these parameters must be tuned on the fly - this may be done automatically by the MCE.	Replies take the same format and length as a Write Instruction, with the latest value of the read/written parameter in the Data field
Read Detector Bias or Pixel Heater	Write Detector Bias or Pixel Heater	get FPGA memory; set bolometer bias <voltage number>; set heater level <voltage number>	1/5	0x21	0000000???	Bit-stuff with 1's	Bit-stuff with 1's	"0x0000".."0xFFFF" (16-bit#)	Read/Write the Detector Bias or Pixel Heater value. There is a single channel of Detector Bias circuitry and Pixel Heater circuitry per sub-array, and the capability to drive one or the other on each bias card. It remains to be determined which bias card will drive the Detector Bias and which will drive the Pixel Heater. Writing a value will immediately update the card's DAC-output. Access to either one of these values is determined by what card address is used. During array multiplexing, these parameters must be tuned on the fly - this may be done automatically by the MCE.	Replies take the same format and length as a Write Instruction, with the latest value of the read/written parameter in the Data field
				0x22						
				0x23						
				0x24						
				0x25						
				0x26						
				0x27						
				0x28						
				0x29						
				0x2A						
				0x2B						
				0x2C						
				0x2D						
				0x2E						
				0x2F						
General										
Start Returning Next <n> Data Frames	Stop returning data frames		4/1	0x30	1111111111	Bit-stuff with 1's	Bit-stuff with 1's	"0x0000": Stop returning frames; "0x0001.."0xFFFE": return <n> frames; "0xFFFF": return a continuous stream	Start/Stop returning frames of data. If no parameters are included, then the electronics will stop returning data frames to the BAC. If a parameter between 0x0001..0xFFFE is passed, then that many frames will be returned before stopping. If a "0x0000" parameter is passed, the electronics will stop returning frames. If a "0xFFFF" parameter is passed, frames will be returned indefinately (up to the amount of data that the data-logging buffer can contain on the RC). If no paramaters are passed (except for the Address field), then the instruction will be interpreted as a Stop. Note that the MUX must be functioning for frames to be collected and returned. In general, the MUX will be functioning even when frames aren't being returned. Note that the returned-frame format still needs to be determined.	To acknowledge that a Start/Stop command has been received and that the appropriate action is being taken, the reply will be an echo of the command from the CC. In addition, the CC will expect data to be returned in a special format. These formats are put forward below - see the asterisked lines below.
Read Data Mode	Write Data Mode	get FPGA memory; select full time resolved data; select filtered data <filter number>; select sync filtered data <filter number>	1/4	0x31	1111111111	Bit-stuff with 1's	Bit-stuff with 1's	"0x00": Science Mode; "0x01": Engineering Mode A1; "0x02": Engineering Mode A2; "0x03": Engineering Mode B; "0x04": Engineering Mode C;	Read/Write the setting that determines what mode the MCE operates in. All possible modes are outlined in MH's document "Functional Description of the Multi-Channel Electronics" available on the SCUBA2 MCE PDR website (http://www.physics.ubc.ca/~scuba2/sc2pdr/index.html). Each mode may have specific settings associated with it, but these still need to be determined. The Data Mode may be changed while frames are being returned, and will take effect on the frame that follows the issue of this command. Note also that the MCE may carry with them different versions of settings for each mode, with each version being altered depending on what mode one is in (i.e. filter coeffiecients).	Replies take the same format and length as a Write Instruction, with the latest value of the read/written parameter in the Data field
Start MUX	Stop MUX	start servo; stop servo	4/4	0x32	1111111111	Bit-stuff with 1's	Bit-stuff with 1's	"0x00": MUX off; "0xAA": MUX on	Start/Stop the array multiplexing. Multiplexing is automated within the MCE, and the ability of the MCE to successful multiplex depends on several initial settings and the ability of the MCE to maintian a lock on the transition point of the SQUIDs.	Replies will be an echo of the Start/Stop Instruction received

Read Frame Row-Order	Write Frame Row-Order		2/14	0x33	1111111111	"0x00": rows 0-9; "0x01": rows 10-19; "0x02": rows 20-29; "0x03": rows 30-40 (sequence#)	Bit-stuff with 1's	11X "0x00.."0x28" (a series of 11 row #s)	Read/Write the row-addressing order. The frame pixel order is specified as a series of rows because all columns are sampled simultaneously. In each command of this type, a sequence of up to 11 rows will be specified at once, so 4 individual commands are necessary to specify a full row-addressing sequence of 41 rows. One byte per row is used in each sequence of 11 rows. Three commands each containing a sequence of 10 rows will specify the order of rows 0-29, and a fourth command will specify the order of rows 30-40. If an 11th row is included in sequences 0, 1 or 2, it will be disregarded. Note that regardless of the row-addressing order used, frames will be returned with data ordered by increasing row-number.	Replies take the same format and length as a Write Instruction, with the latest value of the read/written parameter in the Data field
Reset System			4	0x34	1111111111	Bit-stuff with 1's	Bit-stuff with 1's	"0x00": Reset registers; "0x01": Load CC Factory Configuration; "0x02": Load Application Configuration; "0x03" Reset Power	Reset the MCE electronics. For further discussion of resets, see paragraph 3.2.19 of the "Clock Card Technical Description" on the SCUBA2 MCE PDR website (http://www.physics.ubc.ca/~scuba2/sc2pdr/index.html).	Replies will be an echo of the Start/Stop Instruction received
				0x35						
				0x36						
				0x37						
				0x38						
				0x39						
				0x3A						
				0x3B						
				0x3C						
***Science Mode Data Packet (this is not an instruction; this is a specialized reply) In this mode, each RC returns 32 bits of data/pixel for 8X41 pixels @200Hz.			11	0x3D	000????000	"0x00.."0x28" (row#)	"0x00.."0x07" (column#)	2 pixels of data * 32 bits/pixel = 8 bytes		This is the format for data packets returned from the RCs to the CC, in Science Mode. This format assumes that data from two consecutive pixels are returned in each packet. This means that the row and column parameters (Memory Address A/B) will index groups of two pixels. For example, consecutive data packets could be indexed with the following row:column pairs: 0:0; 0:2; 0:4; 0:6; 1:0; 1:2; etc.. This protocol will use about 17% of the serial return line bandwidth between the RC<->CC, with ~20 bits of overhead per pixel.
***Engineering Mode 'A1', 'A2' and 'B' Data Packet (this is not an instruction; this is a specialized reply) In this mode, each RC returns 24 bits of data/pixel			12	0x3E	000????000	"0x00.."0x28" (row#)	"0x00.."0x07" (column#)	3 pixels of data * 24 bits/pixel = 9 bytes		This is the format for data packets returned from RC->CC in Engineering Mode 'A1', 'A2' or 'B'. For descriptions of these modes, see http://www.physics.ubc.ca/~scuba2/sc2pdr/doc/functional_desc.pdf. Each of the three readings returned per packet will be consecutive pixel indices. Note that the serial RC->CC bandwidth is not large enough to prevent eventual buffer overruns on the RC for some engineering modes.
***Engineering Mode 'C' Data Packet (this is not an instruction; this is a specialized reply) In this mode, each RC returns 16 bits of data/pixel for 1X1 pixels @50MHz			11	0x3F	000????000	"0x00.."0x28" (row#)	"0x00.."0x07" (column#)	4 pixels of data * 16 bits/pixel = 8 bytes		This is the packet-format for data returned from the RCs to the CC, in Engineering Mode C. This format assumes that the data from four consecutive pixels are returned in each packet. This means that the row and column parameters are used to index pairs of pixels. For example, consecutive data packets may be indexed with the row:column pairs of <0:0>;<0:4>; <1:0>; etc.. This protocol will not be fast enought to support continuous streaming without buffer overruns.
Housekeeping										
Read Status (fibre optic flags, card status etc)			1	0x40	0111111111				Read a message that relates the current state of the MCE. The message will consist of a set of flags and fields of data yet to be determined. If need be, this command will also serve to let other cards in the MCE know that the clock card is online and functional. Returns 16 bits.	Replies will echo the Read instruction, with some extra fields added. Both the Address fields will be bit-stuffed with null values, and the requested data will be returned in the 'Data' field.

Read FPGA Temperature			1	0x41	0?????????				Read the internal temperature of an FPGA. Returns 2 bytes containing a 9-bit number.	Replies will echo the Read instruction, with some extra fields added. Both the Address fields will be bit-stuffed with null values, and the requested data will be returned in the 'Data' field.
Read Card Temperature			1	0x42	0?????????				Read a card temperature from the on-board silicon ID chip. Returns a 16-bit number. If the value retreived from hardware is erroneous, the value returned will be all 1's.	Replies will echo the Read instruction, with some extra fields added. Both the Address fields will be bit-stuffed with null values, and the requested data will be returned in the 'Data' field.
Read Card Silicon Serial Number			1	0x43	0?????????				Read a card serial number from an on-board silicon ID chip. Returns a 48-bit serial number. If the value retreived from hardware is erroneous, the value returned will be all 1's.	Replies will echo the Read instruction, with some extra fields added. Both the Address fields will be bit-stuffed with null values, and the requested data will be returned in the 'Data' field.
Read Voltage Levels			1	0x44	??????????				Read the voltage flags on a card to determine whether they're within tolerance. Returns ?? bytes with voltage statuses.	Replies will echo the Read instruction, with some extra fields added. Both the Address fields will be bit-stuffed with null values, and the requested data will be returned in the 'Data' field.
Read Faceplate LEDs Status	Write Faceplate LEDs Status		1/4	0x45	0?????????	Bit-stuff with 1's	Bit-stuff with 1's	0000????; bit#3: Top LED; bit#2: 2nd LED from top; bit#1: 3rd LED from top; bit#0: 4th LED from top	There are up to 4 LEDs per faceplate. Each bit position corresponds to an LED. Bit values: 0- Inactive; 1- Active	Replies will echo the Read instruction, with some extra fields added. Both the Address fields will be bit-stuffed with null values, and the requested data will be returned in the 'Data' field.
Read Card Type			1	0x46	0?????????				Read the card-type of a card. Returns a byte containing ?? bits of information.	Replies will echo the Read instruction, with some extra fields added. Both the Address fields will be bit-stuffed with null values, and the requested data will be returned in the 'Data' field.
Read Slot ID			1	0x47	0?????????				Read the slot number of a card that is inserted in a subrack. Returns a byte containing 4 bits of information.	Replies will echo the Read instruction, with some extra fields added. Both the Address fields will be bit-stuffed with null values, and the requested data will be returned in the 'Data' field.
Read Firmware Version #			1	0x48	0?????????				Read the version number of firmware that a card is currently using. Returns a byte.	Replies will echo the Read instruction, with some extra fields added. Both the Address fields will be bit-stuffed with null values, and the requested data will be returned in the 'Data' field.
Read Array ID			1	0x49	0000000000				Read the identification of the array that the MCE subrack interfaces to. Returns a byte containing 3 bits of information	Replies will echo the Read instruction, with some extra fields added. Both the Address fields will be bit-stuffed with null values, and the requested data will be returned in the 'Data' field.
Read Box Silicon Serial Number (Box ID)			1	0x4A	0000000000				Read the subrack serial number from a silicon ID chip mounted on the bus backplane. Returns a 48-bit number. If the value retreived from hardware is erroneous, the value returned will be all 1's.	Replies will echo the Read instruction, with some extra fields added. Both the Address fields will be bit-stuffed with null values, and the requested data will be returned in the 'Data' field.
Read EEPROM				0x4B						
				0x4C						
				0x4D						
				0x4E						
				0x4F						
Clock Card Specific										
Start Verifying SRAM	Stop Verifying SRAM		1	0x50	0?????????				Veryify that no memory locations in the SRAM are bad sectors, so that access faults can be avoided during operation. This instruction will require the card targeted to keep a record of bad sector so that they aren't used.	Replies will echo the Start instruction
Read Configuration Data	Write Configuration Data	set FPGA program memory	3/11	0x51	0?????????	Memory Address A & B will form a 16-bit configuration data sequence number	Memory Address A & B will form a 16-bit configuration data sequence number	8 bytes of configuration data	Read/Write the configuration data for a specific device in the MCE to the RAM on the CC. This command will allow the download of configuration data in contiguous chuncks to a RAM on the clock card; and will also allow the BAC to read data back in contiguous chunks to the real-time linux computers. Note that there are nine configuration devices which can be configured from the RAM: 1 on the CC, 1 on the AC, 4 on the RCs and 3 on the BCs.	Replies take the same format and length as a Write Instruction, with the latest value of the read/written parameter in the Data field

Start Device Configuration			1	0x52	0?????????				This instruction triggers the configuration of a specified card using the configuration data stored in the RAM on the CC.	Echo the instruction
				0x53						
				0x54						
				0x55						
				0x56						
				0x57						
				0x58						
				0x59						
				0x5A						
				0x5B						
				0x5C						
				0x5D						
				0x5E						
				0x5F						
Power Card Specific										
Start Voltage Supply	Stop Voltage Supply			0x60						
				0x61						
				0x62						
				0x63						
				0x64						
				0x65						
				0x66						
				0x67						
				0x68						
				0x69						
				0x6A						
				0x6B						
				0x6C						
				0x6D						
				0x6E						
				0x6F						
	Boxes highlighted in yellow are used to associate parameters that are required for one of a pairing of commands that use the same command code. For most commands, parameters give the command new meaning if included.									
	Boxes highlighted in grey indicate that these fields will be bit-stuffed with a 'null' value, 0xff									