PACMAN103 – Data Structure Generator Encodings

Author: Simon Davidson

Version: 2.0

1 Command List

Command Byte	Name	Notes	Implemented In DSG Lib?	Implemented In Spec Exec?
0x00	BREAK	Halts spec execution with an error	Y	Y
0x01	NOP	No operation. Can be used as filler	Y	Y
0x02	RESERVE	Reserves a block of memory ready for filling	Y	Y
0x03	FREE	Releases previously reserved memory.		
0x05	DECLARE_RNG	Declares a new random number generator	Y	Y
0x06	DECLARE_RANDOM_DIST	Declares a new random distribution	Y	Y
0x07	GET_RANDOM_NUMBER	Returns a random number drawn from the given distribution	Y	Y
0x10	START_STRUCT	Begins declaration of new structure	Y	Y
0x11	STRUCT_ELEM	Declare single element in a structure	Y	Y
0x12	END_STRUCT	Ends declaration of new structure	Y	Y
0x1A	START_PACKSPEC	Begins definition of a Packing Specification		
0x1B	PACK_PARAM	Writes one bit field inside a single parameter from a bit field of a source parameter		
0x1C	END_PACKSPEC	Ends definition of a Packing Specification		
0x20	START_CONSTRUCTOR	Begins definition of a function to write data structures to memory	Y	Y
0x25	END_CONSTRUCTOR	Ends definition of the write function	Y	Y

0x40	CONSTRUCT	Invokes a constructor to build a data structure	Y	Y
0x41	WRITE	Perform a simple write or block write operation	Y	Y
0x43	WRITE_STRUCT	Performs a write from a predefined structure	Y	Y
0x44	BLOCK_COPY	Copies a block of data from one area to another		
0x50	SWITCH_FOCUS	Swap between different reserved memory regions to work on several at the same time	Y	Y
0x51	LOOP	Set-up a loop	Y	Y
0x52	BREAK_LOOP	Early exit from a loop	Y	Y
0x53	END_LOOP	End of loop	Y	Y
0x55	IF	Perform a condition and execute the following instructions only if the condition is True	Y	Y
0x56	ELSE	Else clause for associated IF statement	Y	Y
0x57	END_IF	Close block of instructions begun with the IF instruction	Y	Y
0x60	MV	Place a value in a register, from an immediate or another register	Y	Y
0x63	GET_WR_PTR	Copy current write address to a register	Y	Y
0x64	SET_WR_PTR	Move the write pointer to a new location, either relative to the start of this reserved memory area or relative to the current write pointer	Y	Y
0x65	ALIGN_WR_PTR	Moves the write pointer so that it points to the next block with a given address granularity	Y	Y
0x67	ARITH_OP	Perform arithmetic operation with operand 2 coming from a register	Y	Y
0x68	LOGIC_OP	Perform logical operation with operand 2 coming from a register	Y	Y
0x6A	REFORMAT	Reformats a value in an internal register		
0x70	COPY_STRUCT	Create an identical copy	Y	Y

		of a structure		
0x71	COPY_PARAM	Copy a parameter from one structure to another	Y	Y
0x72	WRITE_PARAM	Modify a single parameter in a structure using an immediate value or register-held value		
0x73	WRITE_PARAM_ COMPONENT	Modify a single parameter in a structure	Y	Y
0x80	PRINT_VAL	Output the value of a register to the screen	Y	Y
0x81	PRINT_TXT	Print a text string to the screen		
0x82	PRINT_STRUCT	Prints the current state of one structure to the screen	Y	Y
0xFF	END_SPEC	Cleanly ends the parsing of the Data Spec	Y	Y

Table 1- Data Spec Commands (opcodes)

1.1 Command Structure

In the version 2.0 command structure, each command is one or more 32-bit words. The format is as follows:

Bits 29:28	Bits 27:20	Bits 18:16	Bits:15:12	Bits 11:8	Bits 7:4	Bits 3:0
Command length	Command code (see table 1)	Field usage bits	DEST_REG	SRC1_REG	SRC2_REG	Unused

The command length field is used to indicate how many 32-bit words are required for this command:

0b00 = 1 32-bit command word
0b01 = 2 words (command word + single data word)
0b10 = 3 words (command word + two data words)
0b11 = 4 words (command word + three data words)

Field usage bits indicate which of the three register fields are actually used by the command. These are one-hot encoded, and are ORed together:

0b100 = Destination register
0b010 = Source 1 register
0b001 = source 2 register

The command code is an 8-bit field that follows table 1. The three register fields encoded a 4-bit register number.

Note that when a particular register field is not used, the bits can be re-used for some command-specific purpose (see the descriptions of the individual commands for details).

1.2 Table of Data Types

Currently, a 5-bit encoding is used for all data types. The encoding is as follows:

Text Label	Meaning	Encoding (5-bit)
uint8	8-bit unsigned	0x00
uint16	16-bit unsigned	0x01
uint32	32-bit unsigned	0x02
uint64	64-bit unsigned	0x03
int8	8-bit signed	0x04
int16	16-bit signed	0x05
int32	32-bit signed	0x06
int64	64-bit signed	0x07
u88	Unsigned 8.8 fixed point	0x08
u1616	Unsigned 16.16 fixed point	0x09
u3232	Unsigned 32.32 fixed point	0x0A
s87	Signed 8.7 fixed point	0x0B
s1615	Signed 16.15 fixed point	0x0C
s3231	Signed 32.31 fixed point	0x0D
u08	Unsigned 0.8 fixed point	0x10
u016	Unsigned 0.16 fixed point	0x11
u032	Unsigned 0.32 fixed point	0x12
u064	Unsigned 0.64 fixed point	0x13
s07	Signed 0.7 fixed point	0x14
s015	Signed 0.15 fixed point	0x15
s031	Signed 0.31 fixed point	0x16
s063	Signed 0.63 fixed point	0x17

Table 2 - Valid data types (5-bit encoding)

2 Detailed Command Structure

2.1 Command code 0x00: BREAK

Causes execution of the spec to halt. This will prevent a spec from containing to execute if it accidentally finds itself in areas of zeroed memory.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00	Single command word – no data
27:20	CMD_CODE	0x00	Command to halt execution of the Data Spec
18:16	FIELD_USE	0b000	No registers used

2.2 Command code 0x01: NOP

Performs no action. Can be used as a filler code.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00	Single command word – no data
27:20	CMD_CODE	0x01	Command to perform no operation
18:16	FIELD_USE	0b000	No registers used

2.3 Command Code 0x02: RESERVE (memory space)

This command reserves memory in SDRAM that will be subsequently used to hold data structures for the application. Reserved data is always in multiples of one word (32-bit) and so a request at a lower granularity will lead to a rounding up the next 32-byte boundary.

Reserved data is not modified, this must be done separately.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b01	Single command word + 32 bit data word
27:20	CMD_CODE	0x20	Reserve memory command
18:16	FIELD_USE	0b000	No registers required
7	LEAVE_UNFILLED	0b0 or 0b1	If set, space is reserved but not written to.
4:0	SLOT	Any up to max (32)	Entry number in the reserved memory table, used to refer to this region during filling.

The second word is the size of the reserved memory region in bytes, rounded up to the next whole word (32-bit chunk).

2.4 Command Code 0x03: FREE (memory space)

Not yet implemented. May not be required.

2.5 Command Code 0x05: DECLARE_RNG

This command declares a new source of random numbers. This allows an implementation to have several sources (e.g. Mersenne twister) and select between them for a given random distribution.

Up to sixteen different RNGs can be defined (this should be sufficient for any network). The source field of the RNG allows different underlying generators to be declared. By default we use source =0.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b01	Single command word + 32 bit data word (seed)
27:20	CMD_CODE	0x05	Indicates that we are declaring a new Random Number Generator (RNG)
18:16	FIELD_USE	0b000	No registers required
15:12	RNG_ID	Any 4-bit (0→15)	The slot ID for this RNG source. This is used in the declaration of Random Distributions to indicate the source RNG from which numbers will be drawn.
11:8	RNG_SOURCE	Any 4-bit (0→15)	Indicates the source of the random numbers. 0x0 by default. This allows for different source types in future.

The following 32-bit word is a 32-bit unsigned (u32) value, giving the starting seed for the RNG.

2.6 Command Code 0x06: DECLARE_RANDOM_DIST

This command creates a random distribution that is used to generate random parameter values. Each such distribution requires three pieces of information:

1. The Specific distribution (uniform, Gaussian, exponential, etc.)
2. The range of valid values (max and min)
3. The previously-defined Random Number Generator (RNG) that will be used to supply the basic random numbers

This information is passed to the command using a parameter list with the following format:

```
{
        uint32 distType;        // 0 = uniform, others = RESERVED
        s1615 param1;           // For uniform distribution, the min value of the random number
        s1615 param2;           // For uniform distribution, the max value of the random number
        uint32 rngId            // The index (identifier) for the previously defined RNG
} randDistParams
```

This parameter list is passed to the command as follows:

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00	Single command word
27:20	CMD_CODE	0x06	Indicates that we are declaring a new Random Distribution
18:16	FIELD_USE	0b000	No registers required
13:8	DIST_ID	Any 6-bit (0→63)	The slot ID for this random distribution. Up to 64 distributions are allowed.
5:0	PARAM_LIST	Any 6-bit (0→63)	List of parameters defining the random distribution (see above)

2.7 Command Code 0x07: GET_RANDOM_NUMBER

This function returns a random value from a previously defined random distribution (defined using command code 0x06). Values are 32-bit and are encoded as s1615 values. (In future, we might support different return formats for random numbers, using spare bits in the command word to encode this).

This value is written into a register.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00	Single command word
27:20	CMD_CODE	0x07	Indicates that we are executing a get_random_number command
18:16	FIELD_USE	0b100	Destination register specified
15:12	DEST_REG	Any 4-bit (0→15)	The register into which the new random value is written
5:0	DIST_ID	Any 6-bit (0→63)	The slot ID for this random distribution. Up to 64 distributions are allowed.

2.8 Command Code 0x10: START_STRUCT

Begins the definition of data structure (equivalent to a Class definition in C++) that will be instantiated one or more times to create a more complex data structure in memory.

As a minimum, a valid Struct block consists of a START_STRUCT command, one or more ELEM commands and a END_STRUCT command.

Structure definitions CANNOT be embedded. To define hierarchical (nested) data structures, the leaf nodes should be defined first, so that they can be referenced by branch nodes in the data tree (see ELEM for details).

The START_STRUCT command has the following syntax:

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00	Single command word
27:20	CMD_CODE	0x10	Indicates that we are defining a new structure
18:16	FIELD_USE	0b000	No registers are required
4:0	STUCT_ID	Any 5-bit (0→31)	Handle by which this structure will be referenced. Also, its slot number in struct table

2.9 Command Code 0x11: STRUCT_ELEM

This command declares one constituent of a structure.

There can be one or more ELEM commands between the opening START_STRUCT and the closing END_STRUCT commands. The format of one command is as follows:

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00 or 0b01	Single command word (+ optional 32-bit data word)
27:20	CMD_CODE	0x11	Indicates that we are defining a new element for the currently open structure definition
18:16	FIELD_USE	0b000	No registers specified
12:8	ELEM_ID	5-bit type field	Slot number within this struct
4:0	ELEM_TYPE	5-bit type field	Identifies the data type (see Table 2)

If an optional data word is specified, the value is supplied in the next 32-bit word. If not, zero is assumed.

2.10 Command Code 0x12: END_STRUCT

This command closes the definition of the current structure. This structure is added to the table of structures, in the slot specified in the original START_STRUCT command.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00	Single command word
27:20	CMD_CODE	0x12	Indicates end of structure definition
18:16	FIELD_USE	0b000	No registers specified

2.11 Command Code 0x1A: START_PACKSPEC

This command begins the declaration of a packing specification (packspec), which is used to assemble parameters that actually contain several bit fields. The pack spec is called using a WRITE_PACK_PARAM command (code 0x73), which supplies the IDs of up to two source structures and a single pack spec. These source structures are referred to as src0 and src1 and can each contain up to 32 parameter values. The pack spec will create a single output parameter by combining pieces of parameters from these source lists. The output parameter can be of any size and it is left to the programmer to ensure that the destination of this parameter is matched in size to the pack spec's intended output.

A single pack spec is delimited by START_PACKSPEC and END_PACKSPEC commands. Inside, it may contain any normal operation except a new definition: nesting of declarations is not permitted. Bit fields of the parameter are specified using the PACK_PARAM command. Any unassigned bit fields are assumed to be zero.

The format of the START_PACKSPEC command is as follows:

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00	Single command word
27:20	CMD_CODE	0x1A	Command to define a packing specification (packspec)
18:16	FIELD_USE	0b000	No registers specified
14:10	PACKSPEC_ID	Any 5-bit number (up to 32 constructors allowed)	Identifier for the packspec

2.12 Command Code 0x1B: PACK_PARAM

This command assigns the bit pattern of the output parameter. The command has two command words. The first specifies the bit range to be changed in the output parameter. The second provides the source of the new bit pattern.

The first command word has the following format:

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b01	Two command words
27:20	CMD_CODE	0x1B	Command to assign a bit field in the packed parameter
18:16	FIELD_USE	0b000	No registers specified
15:8	DEST_MSB	0->255	Most significant bit of the bit field to be modified
7:0	DEST_LSB	0->255	Least significant bit of the bit field to be modified

The second command word has the following format:

Bit Range	Field Name	Value	Notes
29:28	ZEROES	0b00	Zeroes
27:20	SRC_START_BIT	0->255	LSB of the start of the bit field as it appears in the source parameter
18:16	FIELD_USE	0b000 or 0b010	Optional use of SRC1 to provide a register value instead of a struct parameter
11:8	SRC_REG	0->15	Optional register value used to provide source bit pattern, only if FIELD_USE bit 1 is set.
or			
15:10	SRC_PARAM	0->31	ID of the parameter to provide the source value, used if FIELD_USE bit 1 is clear
9	SRC_STRUCT	0 = src0 structure, 1 = src1 structure	Choose between two supplied structures for source values, used if FIELD_USE bit 1 is clear

2.13 Command Code 0x1C: END_PACKSPEC

This command terminates the declaration of a single packing specification.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00	Single command word
27:20	CMD_CODE	0x1C	Command to close the definition of a pack spec.
18:16	FIELD_USE	0b000	No registers specified

2.14 Command Code 0x20: START_CONSTRUCTOR

Starts the declaration of a CONSTRUCTOR, which is a sequence of instructions that generates a data structure. This constructor can embed other constructors to permit hierarchical data structures to be created.

A constructor is assigned a slot number that is referenced when it is invoked (using a CONSTRUCT command). The invocation will also typically include a parameters list to customise the generation of the data.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00	Command word
27:20	CMD_CODE	0x60	Command to define a CONSTRUCTOR
18:16	FIELD_USE	0b000	No registers specified
15:11	CONSTRUCTOR_ID	Any 5-bit number (up to 32 constructors allowed)	Identifier for the constructor
10:8	ARG_COUNT	0->5	Indicates number of structures passed into the constructor as arguments.
4:0	READ_ONLY	5-bit bit-mask	For each argument, indicates if it is passed as read-only to the constructor.

The constructor ID is stored in a table so that it can be called later using a CONSTRUCTOR command. Up to five arguments (in the form of previously defined structures) may be passed into the constructor. There is no type-checking so the constructor will assume that a correctly formatted structure has been provided.

Instructions inside the constructor can refer only to structures that have been passed in explicitly. The numbering scheme for these structures is then given by their ordering in the call (CONSTRUCT instruction).

Structures flagged as read-only in the START_CONSTRUCTOR declaration can be changed inside the constructor, but their new values are not passed back when the constructor terminates. Conversely, changed values in structures not marked as read-only are passed back from the constructor.

2.15 Command Code 0x25:END_CONSTRUCTOR

This command closes the definition of a constructor block. Constructor definitions cannot be nested.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00	Single command word
27:20	CMD_CODE	0x25	Close definition of constructor

2.16 Command code 0x40: CONSTRUCT

This command invokes a constructor to build a data structure beginning at the current write pointer in the currently open memory region. Constructors may call other constructors.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00 or 0b01	Command word + optional parameters
27:20	CMD_CODE	0x70	Indicates that we are constructing a data block
18:16	FIELD_USE	0b000	No registers specified
12:8	CONSTRUCTOR_ID	Any up to max (0→31)	Indicates which previously defined constructor to use.

If parameters are required, the second word has the following format, consisting of a list of structure IDs:

Bit Range	Field Name	Value	Notes
28:24	ARG4	Any valid struct_ID, 0->15	
22:18	ARG3	Any valid struct_ID, 0->15	
16:12	ARG2	Any valid struct_ID, 0->15	
10:6	ARG1	Any valid struct_ID, 0->15	
4:0	ARG0	Any valid struct_ID, 0->15	

. Inside the constructor, any reference to a structure will use copies of these structures, indexed using the arg number (0->4) given here. Whether or not the values in each of these structures can be modified inside the constructor is defined in the READ_ONLY bit mask in the constructor definition.

2.17 Command code 0x41: WRITE

Writes one or more data values (either immediate values or the contents of a register) to the currently open memory region, beginning at an address given by the current write pointer for that region. It is intended for filling blocks of up to 256 data words.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	Any of 0b00, 0b01, 0b10	Single command word + zero, one or two 32-bit data words
27:20	CMD_CODE	0x41	Indicates that we are executing a write immediate command
18:16	FIELD_USE	0b000 or 0b010 or	Either additional data word(s) or

		0b011	src1_reg provides data. Src2 (if used) provides the number of repeats
13:12	DATA_LEN	0b00 = 8-bit 0b01 = 16-bit 0b10 = 32-bit 0b11 = 64-bit	Length of the data item to write (the exact format of the data is ignored by this command)
11:8	SRC1_REG	Any (0->15)	Register providing the value (if FIELD_USE == 0b010, indicating register use)
7:4 **or**	SRC2_REG	Any	Num repeats
7:0	NUM_COPIES	0->255	How many copies of the data item to be written

A register value always has priority over an immediate. The number of repeats is limited to an immediate value of 255. A register can contain any 32-bit number.

The following 32-bit word is the data value to be written. If the DATA_LEN field is 16-bits, the upper 16-bits are ignored. Similarly, if the DATA_LEN field is 8-bit, the upper 24-bits are ignored. If the data value is 64-bit, the two following 32-bit words are used, with the first being the lower 32-bits and the second being the upper 32-bits of the 64-bit value.

2.18 Command code 0x43: WRITE_STRUCT

Writes one or more previously assigned structures to the currently open memory region, beginning at an address given by the current write pointer for that region. The block size for multiple copies is taken as the size of the source parameter list.

Bit Range	**Field Name**	**Value**	**Notes**
29:28	LENGTH	0b00	Single command word
27:20	CMD_CODE	0x43	Indicates that we are executing a write structure command
18:16	FIELD_USE	0b000 or 0b010	Zero or one register specified (bits 11:8)
11:8 **or**	REG_COPIES	0→15	Register specifies how many copies of the data item to be written (if FIELD_USE is 0b010)
11:8	IMM_COPIES	0->15	Immediate indicating how many copies to be written (if FIELD_USE is 0b000)
3:0	SRC_STRUCT	0→15	Immediate to specify the structure to use.

2.19 Command code 0x44: BLOCK_COPY

This command performs a memory copy from one region to another. For flexibility, the source, target and block size must all be held in registers. Any memory region (ITCM, DTCM or SDRAM) can be the source or target of the copy.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00	Single command word
27:20	CMD_CODE	0x79	Indicates that we are executing a memory copy operation
18:16	FIELD_USE	0b111	Destination plus two sources specified
15:12	REG_DEST	0→15	Register specifies the target address for the start of the block copy
11:8 or	REG_SIZE	0→15	Register specifies how many bytes are being transferred (if FIELD_USE bit 1 is set)
11:8	IMM_SIZE	0->15	Immediate value of how many bytes to transfer (if FIELD_USE is clear)
7:4	REG_SRC	0→15	Register specifies the source address for the start of block copy

2.20 Command code 0x50: SWITCH_FOCUS

Change the focus of future writes to the specified memory region. Each region retains its own write pointer, so further writes continue where the last write to that region occurred.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00	Single command word
27:20	CMD_CODE	0x70	Indicates that we are executing a switch_focus command
18:16	FIELD_USE	0b010 or 0b000	Can optionally place the new region number in a register (src1 field) or an immediate
11:8 or	SRC1_REG	0->15	Register holding memory region ID
11:8	MEM_SLOT_ID	Any up to max (0→15)	Indicates which memory region to switch to, used when FIELD_USE is 0b000.

2.21 Command code 0x51: LOOP

Begins a loop. Requires a start counter, end counter and increment, all given in registers. Loop will exit when the counter is greater than the end value.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00 (may be longer, see text)	Single command word + plus immediates for values not passed in registers
27:20	CMD_CODE	0x51	Indicates a START_LOOP command.
18:16	FIELD_USE	Any 3-bit value.	Any set bit indicates use of register. Any clear bit indicates that value is given in extra 32-bit word.
15:12	START_REG	Any register (0→15)	Start value from this register (used if FIELD_USE bit 2 is set, if not second 32-bit word is used)

11:8	END_REG	Any register (0→15)	End value from this register (used if FIELD_USE bit 1 is set, if not then the following 32-bit word is used)
7:4	INC_REG	Any register (0→15)	Increment value from this register (used if FIELD_USE bit 0 is set, if not then the following 32-bit word is used)
3:0	COUNT_REG	Any register (0→15)	This register is used as a counter for the loop index

There can be zero, one or two trailing 32-bit words, to provide immediate values for any of the three required parameters that are not specified in registers. Note that there are three immediate values and no destination register (the START_REG value takes the place of a destination register, in fact).

2.22 Command code 0x52: BREAK_LOOP

Causes the program to jump to the end of the current loop.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00	Single command word
27:20	CMD_CODE	0x52	Indicates a BREAK_LOOP command.
18:16	FIELD_USE	0b000	No registers specified

2.23 Command code 0x53: END_LOOP

Signals the end of the current loop, triggering the increment of the loop counter register, a check on the exit condition and (if the condition is still true) a return to the instruction following the associated START_LOOP instruction.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00	Single command word
27:20	CMD_CODE	0x53	Indicates an END_LOOP command.
18:16	FIELD_USE	0b000	No registers specified

2.24 Command Code 0x55: IF (condition check)

This command loads performs a check on a register (or registers), changing the flow of execution depending on the result. This is the classic IF-THEN-ELSE construct, with the instructions following the If executed if the condition is TRUE and the instructions following any ELSE instruction executed if the condition is FALSE. If statements can be nested.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00 or 0b01	Single command word with optional immediate data value
27:20	CMD_CODE	0x55	Indicates a condition check instruction (IF)
18:16	FIELD_USE	0b010 or 0b011	One or two registers specified
11:8	SRC1_REG	0->15	Source 1 register (if FIELD_USE bit 1 is set)
7:4	SRC2_REG	0->15	Source 2 register (if FIELD_USE bit 1 is set)

3:0	CONDITION	0->15 (see table below)	Specifies the condition to check for.

Currently, only 32-bit, signed comparisons can be made. The supported 4-bit conditions are as follows:

Bit Value ([3:0])	Name	Notes
0000	IS_EQUAL	
0001	IS_NOT_EQUAL	
0010	LESS_THAN_OR_EQUAL	Currently a signed 32-bit op
0011	LESS_THAN	Currently a signed 32-bit op
0100	GREATER_THAN_OR_EQUAL	Currently a signed 32-bit op
0101	GREATER_THAN	Currently a signed 32-bit op
0110	IS_ZERO	Source 2 not used
0111	IS_NON_ZERO	Source 2 not used
1xxx	RESERVED	May include unsigned variants in future

2.25 Command Code 0x60: MV (immediate or register to register)

This command loads a value into a register, either from an immediate value or another register.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00 or 0b01	Single command word with optional immediate data value
27:20	CMD_CODE	0x60	Indicates a register-to-register or immediate-to-register move command
18:16	FIELD_USE	0b110 or 0b100	One or two registers specified
15:12	DEST_REG_ID	0->15	Destination register
11:8	SRC_REG_ID	0->15	Source register (if FIELD_USE bit 1 is set)

If the FIELD_USE value is 0b100 then the value to load is given in a separate 32-bit word.

2.26 Command Code 0x63: GET_WR_PTR

Loads a register with the current value of the write pointer (byte aligned). This is command is useful to allow a value to be written to a table in one region that is the pointer in another. Note that the write pointer value is relative to the base address for its region.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00	Single command word
27:20	CMD_CODE	0x63	Indicates a get-write-pointer command
18:16	FIELD_USE	0b100	Destination register specified
15:12	DEST_REG	0->15	Target internal register

2.27 Command Code 0x64: SET_WR_PTR

Sets the current write pointer to the value given in the register or immediate value. This value is assumed to be a 32-bit unsigned number (‘uint32’).

This is command is useful to allow a value to be written to a table in one region that is the pointer in another. Note that the write pointer value is relative to the base address for its region.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00 or 0b01	Single command word + optional 32-bit immediate
27:20	CMD_CODE	0x64	Indicates a set-write-pointer command
18:16	FIELD_USE	0b010 or 0b000	Either single source register or none
11:8	SRC1_REG	0->15	Source register if FIELD_USE bit 1 is set
0	REL_ADDR	1 = Relative address 0 = Absolute address	

If a register is not specified, the following 32-bit word is used to provide the new pointer value.

2.28 Command Code 0x65: ALIGN_WR_PTR

Writes a block of zeroes to pad out the current region from the current write pointer to a boundary defined by the source register or immediate, whose value is used as a power of 2. For example, if the boundary register value is 10, writes are performed until the current write pointer is a multiple of 1,024 (= 2^{10}).

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00	Single command word
27:20	CMD_CODE	0x65	Indicates an align-write-pointer command
18:16	FIELD_USE	0b?00 or 0b?10	Optional dest register to return new pointer, plus optional register with block size (in bits)
15:12	DEST_REG	0->15	Register to return the new write pointer (only valid if FIELD_USE bit 2 is set)
11:8 or	BLOCK_SZ_REG	0->15	Register supplying the size of the block boundary (if FIELD_USE bit 1 is set)
4:0	BLOC_SZ_IMM	0->31	Immediate value of block boundary, in bits (used if FIELD_USE bit 1 is clear)

2.29 Command Code 0x67: ARITH_OP

Performs one of a number of arithmetic operations, returning the result in a register. The sources can be either registers or immediates. Currently, operands are always taken as 32-bit values, though the operation can be selected as signed or unsigned.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00 or 0b01 or 0b10	Single command word + one or two optional source parameters
27:20	CMD_CODE	0x68	Indicates an arithmetic operation command
19	SIGNED	0 = unsigned, 1 = signed	Selects whether the operation will be unsigned or signed
18:16	FIELD_USE	0b1xy where x=0 or 1, y=0 or 1	Dest always a register. Sources can be register or immediate.
15:12	DEST_REG	0->15	Register id for result
11:8	SRC1_REG	0->15	Register ID for source 1, used if FIELD_USE bit 1 is set
7:4	SRC2_REG	0->15	Register Id for source 2, used if FIELD_USE bit 0 is set
3:0	OP_SELECT	0b0 = ADD 0b1 = SUB 0b2 = MUL Others reserved	Selects the operation to perform, 32-bit signed quantities are assumed

Since source 1 and source 2 can each be either a register number or an immediate, the length of the command can be extended to two or three words to add these extra parameters. When both are provided, source 1 is given first.

2.30 Command Code 0x68: LOGIC_OP

Performs one of a number of logical operations, returning the result in a register. The sources can be either registers or immediates. Currently operands are always taken as 32-bit values.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00 or 0b01 or 0b10	Single command word + one or two optional source parameters
27:20	CMD_CODE	0x68	Indicates a logical operation command
18:16	FIELD_USE	0b1xy where x=0 or 1, y=0 or 1	Dest always a register. Sources can be register or immediate
15:12	DEST_REG	0->15	Register id for result
11:8	SRC1_REG	0->15	Register ID for source 1, used if FIELD_USE bit 1 is set
7:4	SRC2_REG	0->15	Register Id for source 2, used if FIELD_USE bit 0 is set
3:0	OP_SELECT	0b0= LSL src1 by src2 0x1 = LSR src1 by src2 0x2 = src1 OR src2 0x3 = src1 AND src2 0x4 = src1 XOR src2 0x5 = NOT src1	Selects the operation to perform

Since source 1 and source 2 can each be either a register number or an immediate, the length of the command can be extended to two or three words to add these extra parameters. When both are provided, source 1 is given first.

2.31 Command code 0x6A: REFORMAT

Xxx TODO xxx

Do this when we have clarity on what formatting options are required.

2.32 Command Code 0x70: COPY_STRUCT

This command creates a new copy of a pre-existing structure. The user provides the Id for the existing and new structures (this can overwrite a pre-existing structure).

The source and target structure IDs can be passed either as immediate or given in registers.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00 or 0b01 or 0b10	Single command word + one or two optional source parameters
27:20	CMD_CODE	0x70	Indicates a command to copy one structure into another
18:16	FIELD_USE	0b000 (could set bit 1 or bit 2)	No registers are required, but could be used to provide the structure number for source or destination.
15:12 or	DEST_STRUCT_REG	0->15	Optional register to specify the destination struct ID (used if FIELD_USE bit 2 is set)
15:12	DEST_STRUCT_ID	0->15	Optional register to specify the destination struct ID (used if FIELD_USE bit 2 is clear)
11:8 or	SRC_STRUCT_REG	0->15	Optional register used to specify the source struct ID (used if FIELD_USE bit 1 is set)
11:8	SRC_STRUCT_ID	0->15	Optional register used to specify the source struct ID (used if FIELD_USE bit 1 is clear)

2.33 Command Code 0x71: COPY_PARAM

This command copies the value of one parameter directly from one structure to another without having to go via an intermediate register first. This is a time saving command when creating new structures that are similar (but not identical) to existing ones. If the lengths of the two parameters are not the same an error will occur. Due to the number of parameters this takes, the syntax is more restrictive, requiring the structure IDs and parameter IDs for both the source and target structures to be immediate values. This command is unusual in that it has two compulsory command words, the second being used to specify the source and destination element indices.

The command word has the following syntax:

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b01	Two command words
27:20	CMD_CODE	0x71	Indicates a command to copy a parameter value from one structure into another
18:16	FIELD_USE	0b000	No registers are permitted
15:12	DEST_STRUCT_ID	0->15	Specifies the destination structure ID
11:8	SRC_STRUCT_ID	0->15	Specifies the source structure ID

The second command word has the following syntax:

Bit Range	Field Name	Value	Notes
15:8	DEST_PARAM_ID	0->255	Specifies the parameter to be written in the destination structure
7:0	SRC_PARAM_ID	0->255	Specifies the parameter to be copied from the source structure

2.34 Command Code 0x72: WRITE_PARAM

Assigns a value to one parameter of a given parameter list.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00 or 0b01	Single command word + optional parameter value
27:20	CMD_CODE	0x72	Indicates assignment to a parameter
18:16	FIELD_USE	0b000 or 0b010	Optional register for source 1
15:12	STRUCT_ID	0->15	ID of the structure in which the parameter is to be changed
11:8	VAL_REG	0->15	Register holding the new value (used if FIELD_USE bit 1 is set)
7:0	PARAM_ID	0→255	Element index within the structure

If FIELD_USE is 0b000, then the length field is 0b01 and a following 32-bit word is used to provide the value.

Note that the value is cut to fit the size of the parameter – so a 16-bit parameter that is assigned a 32-bit data word will proceed by chopping off the upper 16-bits of the data word. No sign extension occurs.

2.35 Command Code 0x73: WRITE_PARAM_COMPONENT

This command assigns a value to a field within one parameter of a structure. It requires two command words. The first specifies the source of the data, the second specifies the destination. The first command word has this format:

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b01	Two command words
27:20	CMD_CODE	0x73	Indicates assignment to a parameter
18:16	FIELD_USE	0b1xx = data value in a register 0bx1x = register holds source struct ID 0bxx1 = register holds source param ID	Different sources of the value: either register or structure parameter. In the latter case, the choice of a particular structure or parameter can be given in a register.
15:12	DATA_REG	0->15	Register holding data value to be used
11:8 or	SRC_STRUCT_ID	0->15	Id of a parameter structure that will provide the data value
11:8	SRC_STRUCT_REG	0->15	Register index for structure that holds the data value (used if FIELD_USE is 0bx1x)
7:0 or	SRC_PARAM_ID	0->255	Index of the parameter within the source structure that provides the data value
7:4 or	SRC_PARAM_REG	0->255	Register that is holding the index of the parameter within the source structure that provides the data value (used if FIELD_USE is 0bxx1

If a data register is provided as a source, any information about source structures is ignored.

The second command word gives the Id of the destination structure to be written to, the chosen parameter within that structure and the bit range (LSB and number of bits) to which the value should be written.

Bit Range	Field Name	Value	Notes
31:26	INSERTION_LEN	0->32	Number of bits to inserts from the data word. Overspills are dropped without error
24:20	INSERTION_LSB	0->31	Bit position in target where LSB of new data is to be placed
18:16	FIELD_USE	0b000	No registers specified
11:8	DEST_STRUCT_ID	0->15	ID of the target structure
7:0	DEST_PARAM_ID	0->15	ID of the target parameter

A block of bits from the source data is copies to the bit range in the destination. If the number of inserted bits overruns the end of the word, the extra bits are dropped without flagging an error.

2.36 Command Code 0x80: PRINT_VAL

Displays a value (either from a register or immediate) to the screen. Used for debugging purposes.

The format of the data is given in the 5-bit FORMAT field, using table 2 for the enumeration. For data words of 64-bit, two 32-bit immediates would be required, with the first being the upper 32-bits of the final value.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00 or 0b01 or 0b10	Singe command word or with data value (32-bit or 64-bit)
27:20	CMD_CODE	0x80	Indicates the print register command
18:16	FIELD_USE	0b000 or 0b010	Zero or one registers specified
11:8	SRC1_REG	0->15	The register whose value is to be printed
4:0	FORMAT	0->31	Format of the data to be printed (see table 2)

2.37 Command Code 0x81: PRINT_TXT

Prints a series of 8-bit ASCII characters to the screen, given in immediate data. The immediates follow the command word, with the upper 8-bits representing one symbol, the next 8-bits the second, etc. The NUM_CHARS field in the command word indicates how many characters are valid. The encoded value is one less than the true value.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b01, 0b10 or 0b11	Command word plus 1->3 data words
27:20	CMD_CODE	0x73	Indicates that we're printing text
18:16	FIELD_USE	0b000	No registers specified
3:0	NUM_CHARS	0->11	Number of characters to print minus one (from 1 to 12 characters)

2.38 Command Code 0x82: PRINT_STRUCT

This is typically a debug aid. It prints the contents of one structure to the screen/set to tubotron.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b01, 0b10 or 0b11	Command word plus 1->3 data words
27:20	CMD_CODE	0x73	Indicates that we're printing the contents of a structure
18:16	FIELD_USE	0b000 or 0b010	Zero or one registers specified
11:8	SRC1_REG	0->15	The structure whose content is to be printed
3:0	STRUCT_ID	0->15	Immediate value of ID of the structure to be printed (used if no register specified)

The print format shows the Structure index, then one line per entry in the structure, with fields representing the entries index, size (in bytes) and current value.

This command could be extended in the future to allow greater control over how the individual elements are displayed.

2.39 Command Code 0xFF: END_SPEC

This command signals the end of the specification process. At this point all data structures required by the application should be in place. The Spec Executor releases any memory reserved for itself and shuts down. In the current host-based Spec Executor model control passes back to the output_generator and thence back to the controller.

Bit Range	Field Name	Value	Notes
29:28	LENGTH	0b00	Single command word
27:20	CMD_CODE	0xFF	Indicates that we are recording the current write pointer for later access.
18:16	FIELD_USE	0b000	No registers specified
23:00	RESERVED	0x000000	

3 Required Supported formats

S87	S1615	S3231
U88	U1616	U3232