Sweet16 Debugger Protocol Version 1

The Sweet16 Debugger Protocol is a JSON-based protocol which uses WebSockets to transmit information between Sweet16 and a debugger. This lets any modern Web browser serve as a debugger for Sweet16.

Enabling the debugger in Sweet16 starts up a WebSocket server. The debugger (implemented using HTML and JavaScript) then controls Sweet16 by sending commands in JSON format to Sweet16, which sends system status information, details about the code being executed, and so forth back to the debugger also using JSON packets.

Note: Nowhere close to all of this is implemented yet! Sorry!

Commands

These are the commands sent by the debugger to Sweet16. Even commands that say they don't return a result may in theory result in sending back an `error` message, if something goes wrong.

getEmulatorInfo	getInstructions	readMemory
getRegisters	setRegisters (one field per register, only include those you want to change)	getEmulatorStatus (paused? speed? etc)
setEmulatorStatus		sendEvent
setMemory	restart	addBreakpoint
clearBreakpoint	getBreakpoints	step

getStack	setStatusBits (individual control over status register bits)	clearMemory
trace	halt	clearAllBreakpoints

Standard fields

Every debugger command has the following fields.

Field name	Type	Description
command	string	The name of the command to execute.
order	number	A numeric value used to track the order in which commands are sent; each command should have a value 1 greater than the last command sent. The first command sent during a session should be 1. Replies to specific commands will include that command's order number.

getEmulatorInfo

This command returns information about the emulator. An `emulatorInfo` message is sent in reply.

clearAllBreakpoints

Removes all breakpoints. No reply.

getInstructions

Returns one or more instructions. An `instructions` message is sent in reply.

Field name	Type	Description
address	number	The address at which to start fetching instructions. If 0, the current program bank and program counter are used.
count	number	The number of instructions to disassemble.

readMemory

Returns one or more bytes read from the emulator's memory. A `memory` message is sent in reply. **NOT IMPLEMENTED.**

Field name	Type	Description
address	number	The address to start reading from.

Field name	Type	Description
count	number	The number of bytes to retrieve.

getRegisters
Returns the values of all of the 65816 registers. A `registers` message is sent with the results.

setRegisters
Sets the values of one or more registers. No reply is sent to this message. You may specify one or all of these fields; you don't have to provide any you don't wish to specify a new value for.

Only PBR and PC are implemented here.

Field name	Type	Description
A	number	The accumlator's value.
X	number	The X register's value.
Y	number	The Y register's value.
PC	number	Program Counter value.
DBR	number	Value for the Data Bank Register.
PSR	number	Value for the status register.
PBR	number	Value for the Program Bank Register (K).
SP	number	Value for the Stack Pointer.
DP	number	Value for the Direct Register.

getEmulatorStatus
Returns information about the status of the emulator. An `emulatorStatus` message is sent with the results.

setEmulatorStatus
Sets the emulator's current state. No reply is sent. You may specify any combination of these fields.

Field name	Type	Description
paused	boolean	Whether or not the emulator should be paused.
breakpointsEnabled	boolean	Whether or not breakpoints are enabled.

sendEvent
Sends an event to the emulated computer. No reply is sent.

Field name	Type	Description
type	string	The type of event to send. • keydown • keyup • mousedown • mouseup • mousemove
keyADB	number	For key events, the ADB keycode. You may send either this **or** keyASCII, but not both.
keyAscii	number	For key events, the ASCII character of the pressed key. You may send either this **or** keyADB, but not both.
x	number	The mouse X coordinate at event time (optional for key events).
y	number	The mouse Y coordinate at event time (optional for key events).
commandKey	Boolean	true if the Command key is down
optionKey	Boolean	true if the Option key is down
shiftKey	Boolean	true if the shift key is down
controlKey	Boolean	true if the control key is down
capsLock	Boolean	true if the caps lock is down
keypad	Boolean	true if the key pressed is on the keypad (only for key events)
rightClick	Boolean	true if the click is a right-click

setMemory

Sets the value of one or more bytes of the emulated computer's memory. **NOT IMPLEMENTED.**

Field name	Type	Description
address	number	The address to start writing to.
bytes	array of number	An array of bytes to write into the GS memory.

restart

Restarts the emulated computer. No reply is sent. **NOT IMPLEMENTED.**

addBreakpoint
Adds a new breakpoint. No reply is sent.

Field name	Type	Description
address	number	The address at which to create the breakpoint
type	string	The type of breakpoint to create. One of: • `break` • `read` • `write` • `readwrite` • `conditional` `break` is a standard unconditional breakpoint that pauses the emulator before executing code at the specified address. `read`, `write`, and `readwrite` are watchpoints; the emulator pauses whenever the specified address is accessed in the specified manner. `conditional` is a conditional breakpoint.
name	string	An optional name for the breakpoint.
condition	?	TBD; I doubt anything but `break` will be implemented anytime soon.

clearBreakpoint
Removes a breakpoint. No reply is sent.

Breakpoints may be identified by either `address` or `name`.

Currently, only `address` is supported.

Field name	Type	Description
address	number	The address of the breakpoint to remove. Either this or name (but not both) must be specified.
name	string	The name of the breakpoint to remove. Either this or address (but not both) must be specified.

getBreakpoints
Returns a list of all current breakpoints. The result is sent as a `breakpoints` message. **NOT IMPLEMENTED.**

step
Tells the emulator to execute one step of instruction. No reply is sent.

Currently only "in", "skip", and "stop" are supported. All others are accepted but will have unpredictable results (they set the status appropriately, but that status is ignored, so the results will likely not be good).

Field name	Type	Description
type	string	The type of step to perform. One of: • `in` • `over` • `out` • `stop` • `skip` `in` executes one instruction, going into a subroutine if the instruction is a `JSR` or `JSL`. `over` executes one instruction, or, if the instruction is a `JSR` or `JSL`, executes the entire subroutine, returning control to the debugger when PC and PBR indicate the next instruction following the current one. `out` executes all instructions until the current routine exits by calling either `RTS` or `RTL`. `stop` ends stepping mode and resumes running the CPU normally. `skip` skips over the current instruction to the next one.

getStack

Returns the contents of the stack. The result is sent as a `stack` message.

Field name	Type	Description
`numBytes`	number	The number of bytes to return. Since the stack doesn't have a bottom, this is used to indicate how far down you want to go.

setStatusBits

Sets the value of one or more of the processor status bits; any one or more of the fields may be specified. No reply is sent. **NOT IMPLEMENTED.**

Field name	Type	Description
c	Boolean	The new value for the carry flag.

Field name	Type	Description
n	Boolean	
z	Boolean	
d	Boolean	
m		
x		
i		
b		
e		

clearMemory NOT IMPLEMENTED.

Clears the specified memory. No reply is sent.

Field name	Type	Description
address	number	The address to start erasing at.
count	number	The number of bytes to erase.
value	number	The value (0-255) to write into each byte in the specified range.

trace

Starts or stops trace mode. A `traceStatus` message is sent in reply. **NOT IMPLEMENTED.**

While in trace mode, the emulator runs normally but automatically sends out processor status and disassembly information to the debugger as it executes each instruction; one disassembly message is sent for each instruction, as is one register instruction.

Field name	Type	Description
enable	Boolean	true to start tracing, false to stop.

halt

Stops the CPU and enters single-step mode. No reply is sent. **NOT IMPLEMENTED.**

Messages

These are messages sent by Sweet16 to the debugger. Some are sent solely in response to specific commands, some are sent as-needed to provide updated information, and some can be sent in either case.

emulatorInfo	console	registers
memory (one byte or a range of bytes)	emulatorStatus	break
instructions	breakpoints	statusBits
error	step	stack

Standard fields

Every message has the following fields.

Field name	Type	Description
message	string	The name of the message.
inReplyTo	number	The order number of the command to which this message is a reply, or 0 if this is not a reply to a command request.
cycle	number	The CPU cycle number at which the message was sent.
timestamp	number	The time at which the output occurred, in milliseconds since January 1, 1970 at midnight UTC.

The cycle number is sent even if the command is not related to the CPU in any way (including in the sweet16Version message). The value will be 0 if the emulator hasn't been started.

emulatorInfo

Provides information about the emulator.

Field name	Type	Description
name	string	The emulator's name.
version	string	The version number of the emulator, such as "3.0b1".
copyright	string	The copyright string for this version of Sweet16.
protocolVersion	number	The debugger protocol version number supported by the emulator.
date	string	The date on which the emulator was compiled.
time	string	The time at which the emulator was compiled.

console

Outputs data to the debugger console. Only one of the fields may be specified.

Field name	Type	Description
text	string	A text string to output to the console, in UTF-8 encoding.
html	string	HTML to output to the console. **NOT IMPLEMENTED.**

registers

The values of all Apple IIgs registers.

Field name	Type	Description
A	number	The accumlator's value.
X	number	The X register's value.
Y	number	The Y register's value.
PC	number	Program Counter value.
DBR	number	Value for the Data Bank Register.
PSR	number	Value for the status register.
PBR	number	Value for the Program Bank Register (K).
SP	number	Value for the Stack Pointer.
DP	number	Value for the Direct Register.

memory

The values of a range of Apple IIgs memory. **NOT IMPLEMENTED.**

Field name	Type	Description
address	number	The address of the first byte in the block.
count	number	The number of bytes (this is the size of the `bytes` array).
bytes	array of number	The values of each of the bytes in the block.

stack

This response from the `getStack` request contains information describing the contents of the stack. Since there's no way to know how deep the stack can actually be, this describes the number of bytes worth of stack requested by the `getStack` request.

Field name	Type	Description
count	number	The number of stack entries returned.
items	array of object	An array of objects describing each entry in the stack. In the future, these may be able to represent the size of the values pushed onto the stack, but for now they will always be bytes, so `size` will always be 1 for now. • `address` - The address (as a 16-bit value; the bank is always $00) of the value • `value` - The numeric value stored on the stack • `size` - The size of the value in bytes (1-4) Eventually information will also be added to support recording what instruction was used to create the entry, etc.

break

Tells the debugger that a hard break (either a BRK or an ORCA breakpoint) has occurred.

Field name	Type	Description
address	number	The address of the first byte of the BRK or COP instruction that caused the hard break.
name	string	The ORCA function name, if any. **NOT IMPLEMENTED.**

emulatorStatus

The emulator's current status. Generally sent in response to a `getEmulatorStatus` request, but also sent when certain states change, such as when the emulator is paused or unpaused.

Field name	Type	Description
paused	boolean	Whether or not the emulator is paused.
breakpointsEnabled	boolean	Whether or not breakpoints are enabled.

instructions

Contains one or more instructions as requested by a prior `getInstructions` request, or as indicated by the current tracing mode.

Also sent each time a step occurs to update the debugger about what's going on.

Field name	Type	Description
count	number	The number of instructions in the list.
list	object	A list of instructions. This is an array, each element of which contains the following fields: • address: The address of the instruction (number) • instruction: The value of the instruction register (number) • disassembly: A string that disassembles the instruction • numBytes: The number of bytes the instruction uses
type	string	Indicates the type of instruction list this is: • `list` - An instruction listing, typically in response to a `getInstructions` request. • `step` - A single instruction sent by the CPU, indicating the instruction that will be executed next

breakpoints

A list of breakpoints; sent in response to a `getBreakpoints` request. **NOT IMPLEMENTED.**

Field name	Type	Description

statusBits

The values of the status bits, broken out from the processor status register. **NOT IMPLEMENTED.**

Field name	Type	Description
c		
v		
n		
z		
d		
m		
x		

Field name	Type	Description
e		
i		
b		

error

Describes an error that has occurred. These may be sent in response to any command. **NOT IMPLEMENTED.**

Field name	Type	Description
type	string	The type of error: “command” if it’s due to a syntax or other error in a command sent to the debug server, or “emulation” if it’s an emulation related error.
message	string	The error message.