Author Manuscript

Author Manuscript

Author Manuscript

Author Manuscript

HHS Public Access

Author manuscript

*Anesth Analg*. Author manuscript; available in PMC 2021 December 01.

Published in final edited form as:

*Anesth Analg*. 2020 December ; 131(6): 1923–1933. doi:10.1213/ANE.0000000000005220.

# A Forensic Disassembly of the BIS Monitor

**Christopher W Connor, MD, PhD (Connor) [Assistant Professor]**

Department of Anesthesiology, Perioperative and Pain Medicine, Brigham and Women's Hospital, Harvard Medical School, Boston, MA, USA, Research Associate Professor, Departments of Physiology & Biophysics and Biomedical Engineering, Boston University, Boston, MA, USA

## Abstract

**Background—**The BIS Monitor has been available for clinical use for more than twenty years and has had an immense impact on academic activity in Anesthesiology, with more than three thousand articles referencing the bispectral index. Despite attempts to infer its algorithms by external observation, its operation has nevertheless remained undescribed, in contrast to the algorithms of other less commercially successful monitors of electroencephalogram (EEG) activity under anesthesia. With the expiration of certain key patents, the time is therefore ripe to examine the operation of the monitor on its own terms through careful dismantling, followed by extraction and examination of its internal software.

**Methods—**An A-2000 BIS Monitor (gunmetal blue case, amber monochrome display) was purchased on the secondary market. After identifying the major data processing and storage components, a set of free or inexpensive tools was used to retrieve and disassemble the monitor's onboard software. The software executes primarily on an ARMv7 microprocessor (Sharp/NXP LH77790B) and a digital signal processor (Texas Instruments TMS320C32). The device software can be retrieved directly from the monitor's hardware by using debugging interfaces that have remained in place from its original development.

**Results—**Critical numerical parameters such as the Spectral Edge Frequency (SEF), Total Power and BIS values were retraced from external delivery at the device's serial port back to the point of their calculation in the extracted software. In doing so, the locations of the critical algorithms were determined. To demonstrate the validity of the technique, the algorithms for SEF and Total Power were disassembled, comprehensively annotated and compared to their theoretically ideal behaviors. A bug was identified in the device's implementation of the SEF algorithm, which can be provoked by a perfectly isoelectric EEG.

**Corresponding Author:** Dr. Christopher W. Connor, Department of Anesthesiology, Perioperative and Pain Medicine, Brigham and Women's Hospital, 75 Francis Street, CWN L1, Boston, MA 02115, cconnor@bwh.harvard.edu.

Author Contributions:

Christopher W Connor: This author conceived and performed all the work described in this study, and wrote the manuscript.

**Paper Type:** Original Laboratory Research Report

Disclosures:

Dr. Connor is a consultant for Teleflex, LLC on airway equipment design. This activity is unrelated to the material in this manuscript.

Conflicts of Interest:

None

Prior Presentations

None.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

**Conclusions—**This article demonstrates that the electronic design of the A-2000 BIS Monitor does not pose any insuperable obstacles to retrieving its device software in hexadecimal machine code form directly from the motherboard. This software can be reverse engineered through disassembly and decompilation to reveal the methods by which the BIS Monitor implements its algorithms, which ultimately must form the definitive statement of its function. Without further revealing any algorithms that might be considered trade secrets, the manufacturer of the BIS Monitor should be encouraged to release the device software in its original format in order to place BIS-related academic literature on a firm theoretical foundation and to promote further academic development of EEG-monitoring algorithms.

## Introduction

Almost twenty-five years ago, the Federal Drug Administration approved the marketing of the BIS Monitor with the indication "to monitor the state of the brain by data acquisition of EEG signals in the intensive care unit, operating room and for clinical research."[1–3] Although never a standard for anesthesia monitoring, the device was accepted enthusiastically into practice. It has become the *de facto* monitor for depth-of-anesthesia, supplanting comparable devices like the Masimo SedLine or the Datex-Ohmeda Entropy Module,[4] despite controversies such as its utility in preventing intraoperative awareness,[5,6] the acausal nature of the "triple-low",[7,8] and even its use in an execution by lethal injection.[9]

The BIS Monitor is unusual in comparison to other commonly used anesthesia monitors in that its underlying function remains secret. An electrocardiogram (ECG) measures electrical potentials across the chest and displays them for interpretation; an automated blood pressure cuff measures fluctuations in arterial pulsations and computes systolic, mean (MAP) and diastolic pressures; an end-tidal gas analyzer measures infrared absorption and computes gas concentrations using known spectra.[10] A BIS Monitor measures two frontal EEG leads and computes… well, what, exactly? The device is a *nostrum* for the monitoring of anesthesia. Such secrecy should invite our disapproval and demands our most stringent professional scrutiny. The proprietary and costly patient electrodes, for example, are not inherently superior to simple ECG electrodes.[11]

The most common approach to understanding the BIS has been to match external clinical observation and measurements to the index value. Both Gu[12] and Lee[13] recently applied machine learning to produce facsimiles of the BIS index. This approach is, however, fundamentally unsatisfactory; it layers the inscrutability of neural networks[14] upon the opacity of the device's algorithm. Bruhn[15] reported a linear relationship between burst suppression ratio (SR) and BIS indices below 50, such that BIS = 50 – (SR/2). This simple relationship is belied by the counter-example that physiological sleep produces BIS index values in the mid-20s without burst suppression.[16,17] The healthy, sleeping child shown in Figure 1 has a BIS index of 26, a minimum alveolar fraction of zero, and MAP ≤ 75 mmHg almost certainly, thus meeting the technical definition of "triple-low".

Plainly, rather than using indirect techniques to understand the BIS Monitor, it is time to accept the monitor as the definitive statement of its own function, and then carefully reverse-

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

engineer it. This article therefore presents an initial approach to forensic disassembly of the BIS hardware and software.

## Methods

When reverse-engineering a device, it is usually preferable to choose the earliest version that is broadly available. As a device becomes more commercially successful, its manufacturing process is usually optimized through custom components which are harder to analyze. An A-2000 BIS Monitor was purchased via the secondary market, for $120 (Figure 2A).

### Initial Physical Dismantling and Inspection

The A-2000 BIS Monitor contains three large printed circuit boards (PCBs), with a mixture of surface-mount and through-hole construction typical of the late 1990s. The PCBs, respectively, control the monochrome display (Electroluminescent Display), handle power supply and external interfacing (Power Supply PCB), and perform computational work (Main PCB). The front of the A-2000 can be detached (Figure 2B) to access the computing components on the Main PCB. Debugging equipment can be connected while allowing the device to be closed up again for experimental operation. Examination of the Main PCB reveals a number of identifiable components and access points for debugging. The most immediately important are (Figure 2C, numbered circles):

1. The LH77790B microprocessor, based on the standard ARMv7 chipset. Although this processor would itself be considered obsolete, ARM-based microprocessors are now used in almost all mobile phones and represent a well-documented architecture. This processor supports on-chip debugging, and includes a module called ICEBreaker[18] which allows for monitoring of program execution and memory access.
2. The Texas Instruments TMS320C32 Digital Signal Processor (DSP), is used for performing intensive floating-point calculations required to process EEG data in real-time. Although the TMS320C32 is now obsolete, complete documentation remains available from the manufacturer.[19]
3. A 40-pin TSOP (Thin Small Outline Package) 16Mbit (2MB) Flash memory, containing the device software. For comparison, Figure 2D shows a dismantled USB drive from the early 2000s, containing a 48-pin TSOP with 512Mbit (64MB) of storage.
4. The Joint Test Action Group (JTAG) connector, upper-left corner. JTAG is a debugging protocol that allows an external device to, amongst other tasks, pause the operation of processors and query and alter their memory. JTAG is explicitly supported by the LH77790B microprocessor.

### Selection of Reverse-Engineering Tools

**JTAG Debugger—**Electrical inspection of the Main PCB with a continuity probe shows (Figure 3A) that a JTAG Interface connects to both the LH77790B Microprocessor and the Xilinx Field Programmable Gate Array (FPGA). JTAG is often used during the development

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

of complex electronic devices, since it allows the designers to halt the device, and download and query its current state. As these features also facilitate reverse-engineering, designers will often attempt to disguise this interface or render it electrically inoperative to end-users.[20] In the A-2000 BIS Monitor, the TDO line (Test Data Out) is disconnected in the device's default configuration. To reactivate the JTAG interface, one or more jumpers must be placed onto the J99 connector (Figures 3B and 3C). Appropriate placement of jumpers reconnects the JTAG interface pins on the ARM microprocessor and, optionally, the Xilinx FPGA. An external computer can be connected via a JTAG bridge. The J-Link (SEGGER Microcontroller GmbH, Germany) is an acceptable choice.

**Serial Data Converter—**When reverse-engineering a device, it is very helpful to have a known data-point or quantity that can be retraced internally. The A-2000 has a serial port to connect it to external hardware, e.g. charting systems. This data stream contains numerical values such as burst suppression ratio, spectral edge frequency, and BIS values themselves. Retracing these will be the primary approach to software disassembly. The Sabrent SBT-USC6K (Sabrent, Los Angeles, CA) is a compatible and inexpensive serial bus adapter.

**Disassembler and Decompiler—**The device software will be retrieved in the form of raw hexadecimal code. Hexadecimal uses the numerical digits 0–9 and additionally the letters a-f to represent values 10–15. Hexadecimal is written prefixed with 0x; consequently, 0×100 represents the decimal value 256. On a 32-bit microprocessor, 8 hexadecimal digits represent an *opcode*, which is one discrete processor operation. Raw machine code is dependent on the particular processor on which the code executes, and bears little resemblance to the source code written by the device programmer in a higher-level language. The original source code would feature informative variable names, identified function calls and additional commentary describing the operation of the software. When the software is compiled, these helpful but redundant semantics are discarded.

For reverse engineering, the code is first disassembled from *opcodes* back into *assembly language*, which is human-readable though the overall intent of the code can be very difficult to discern. A sophisticated disassembler, given a valid *entry point*, can proceed from that point to not only disassemble but also track which parts of the code interact with which other parts, tracing the flow of execution. A decompiler takes assembly language and attempts to reconstruct higher level abstractions such as loops, function calls and *if…then…else* constructs to more intuitively describe its behavior, usually represented in C.[21] Ghidra is an open-source disassembler/decompiler that was recently declassified and released by the United States National Security Agency (Fort Meade, MD) in 2019, after its existence was disclosed in 2017 (https://ghidra-sre.org).

**DSP Disassembler and Instruction Simulator—**A different disassembler is needed to read DSP instructions. An Instruction Simulator allows the execution of those instructions to be examined in software. Instruction simulation is a less complete form of device emulation that is sufficient to test algorithms. The Unisim TMS320C3X disassembler and simulator is suitable, and freely available from the French Atomic Energy Commission (https://www.unisim-vp.org).

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

**Return of the Device to Baseline State—**Unplugging the described debugging hardware and closing the device case returns the BIS Monitor to its baseline state without trace.

## Results

### ARM Initialization

The LH77790B stores a table in memory at 0xFFFFA000 that defines its memory allocation. Using the JTAG interface to halt the ARM processor while the monitor is running, it becomes clear that the active software is held in working memory beginning at 0×800000 on the Dynamic Random Access Memory (RAM) (Figure 2C). Using the ARM ICEBreaker module, a watchpoint can be set to halt the processor when memory at 0×800000 is first written after turning on the device. Examining the program code that causes this first memory-write (Table 1A) reveals a function which copies the majority of the Flash memory (mapped to 0×200000) to the working memory. This active software can therefore be captured by downloading an image of the ARM processor's memory between locations 0×800000 and 0×8DFFFF through the JTAG interface.

Examination of the memory image reveals readable text. The text includes material like the prompts that the monitor displays onscreen in all languages. Copyright messages are present, including at memory location for 0×870598 for "Copyright (c) 1993–1999 ATI - Nucleus PLUS", identifying that the BIS runs upon the Nucleus Real Time Operating System (RTOS). Nucleus is unusual in that it does not require a filesystem to function. Knowing the underlying operating system makes the software easier to interpret when it makes calls to identifiable functions. The text also includes the terms "powerup.io" and "biseng.io", representing the filenames of embedded modules of DSP code.

### Checksums and Code Integrity

The A-2000 monitor periodically calculates a checksum against the contents of its own memory. Checksums are calculated beginning at 0×83EF18, in which the program loops over itself and trivially sums the opcodes into microprocessor register r5. At 0×83EF40, the expected checksum result is loaded from 0×90E4B8 into register r0. Such simple mathematical operations do not provide cryptographic security, but only protect the code against corrupting itself accidentally. This is a sensible precaution in a medical device - the most prudent course of action is then to alert and halt (Figure 4A). However, it is helpful to make deliberate changes to the code in order to probe its operation, which requires suspending this checksum. At 0×83EF44, the processor performs the instruction cmp r5,r0 to compare these two values (opcode 00 00 55 E1). This test fails when the calculated checksum does not match the expected value. However, by rewriting the opcode to 00 00 50 E1, the instruction becomes cmp r0,r0 which compares the expected value to itself. The code integrity check then always passes, regardless of modifications. This, however, does not constitute a security vulnerability. The A-2000 has no Ethernet port, and has no capability for interaction with a hospital network. These changes can only be implemented after the device is powered up in the partially dismantled state shown in Figure 2B. These changes are only temporary, and are lost whenever the device is rebooted.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

### Retracing of Numerical Results in the Serial Output

Table 1B shows a representation of a packet of serial data captured from monitor with no patient connector attached. The strategy is to retrace a numerical result from its presentation back to its generation, like following a river back to its source. Many of the values are non-zero: the spectral edge frequency SEF07 (in fact, a 95% spectral edge frequency) settles to an unusual value of 30.0 Hz when there is no patient connector attached. Such idiosyncratic values are easier to trace in device memory. Table 1B includes a representation of the time, and the text of the device memory image includes a formatting string “%m/%d/%Y %H:%M:%S” at memory location 0×828728. This syntax is used by operating systems to define how a date should formatted: %m, %d and %Y adopt the values of the month, date and year respectively. Identifying the part of the software that uses this formatting string identifies the code that prepares internal numerical values for serial transmission. This routine, at 0×82A96C, includes the formatting strings “%8.1f|%8.1f|%4.4x|%8.1f|” and “%8.1f|%8.1f|%8.1f|%8u|%8.8lx|”, representing the correct formatting signatures for the numerical results in Table 1B, where %f indicates a floating-point value, %x a hexadecimal integer and %u an unsigned integer. These formatting strings are passed along with parameter blocks at 0×82ACB0 and 0×82AD44 respectively, thus identifying the parameter blocks in memory that hold these values. By placing a BIS probe on an awake subject, capturing serial values, and scanning device memory, the concordance shown in Table 1C can be observed. Hence, the BIS monitor stores its reported values in memory not as floating point but rather as scaled integers. Further reverse tracing can be performed by setting a series of watchpoints and breakpoints so that the processor halts when these parameters are accessed or written. Such retracing ultimately reveals that the ARM processor obtains these values by reading from an out-of-memory address at 0×1000000, and further investigation reveals that this address represents the memory-mapped serial output of the DSP. Calculation of the BIS parameters therefore occurs on the DSP, rather than on the ARM processor itself, and attention must be turned to the DSP code modules powerup.io and biseng.io that are embedded in the Flash memory image.

### DSP Initialization

Shortly after powering up, the BIS monitor runs through a sequence of self-tests before beginning normal operation. At the moment shown in Figure 4B, the ARM processor is being used to initialize the DSP. The process of booting the DSP from a host processor is complicated, though completely described in the relevant technical documentation.[19] Essentially, packets of DSP code are transferred, potentially out-of-order, from the host processor. These packets are then stored into the DSP memory so that the complete DSP program is reconstructed and started from the correct entry point. Appendix A contains a module written in Python that can be run on a desktop computer to recreate this process: given the ARM memory image, the initialized DSP memory image is reconstructed and saved as a binary file for further disassembly and analysis. The A-2000 uses a two-stage boot process. It first resets the DSP, and uploads the program powerup.io. This program is very short, comprising about only 1.5 kB, so it is relatively straightforward to disassemble and read in its entirety. The program tests the basic functions of the DSP, then replies back to the ARM processor with the hexadecimal number 0×1234 to declare success. Although those are the only functions of powerup.io, it is useful to reverse-engineer because it

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

delineates the basic communications between the DSP and the ARM processor. Next, the ARM processor completely resets the DSP again and uploads the program biseng.io, which is substantially longer at around 156 kB. On successful startup of biseng.io, the DSP signals back with the hex value 0×5678, allowing device initialization to complete.

## Analysis of the DSP BIS Engine (biseng.io)

The BIS Engine is loaded into the DSP memory from 0×8F6400 to 0×8FFEDA, and contains the plaintext "(c) 1995. Aspect Medical Systems, Inc.", fixing the year in which the code was created. The TMS320C32 DSP has many instructions to allow it to manipulate floating-point values natively, but many critical basic arithmetical operations that are required to manipulate EEG data are lacking. The biseng software therefore implements approximations to these functions, and these can be readily identified in the disassembly from their known implementations. For example, division and square-root operations are implemented using Goldschmidt's algorithms[22] at 0×8FDA8B and 0×8FDB20 respectively. Cosine and sine are implemented using the standard Maclaurin[23] expansions at 0×8FDBB7 and 0×8FDCA3. These sine and cosine functions are used to pre-compute trigonometric lookup tables of various lengths at 0×8F9AD6, which are then later used to calculate Fourier Transforms at 0×8FD9DA. Other identifiable elementary functions include summing over a range in memory to approximate integrating under a curve, logarithms, and simple statistics. By examining how calls to these elemental mathematical functions are organized, a picture begins to emerge of how the DSP calculates those parameters that are ultimately made available at the serial port of the BIS monitor. In this regard, one of the most useful identifiable function is at 0×8F7E97: it converts floating-point values into scaled integers of precisely those types shown in Table 1C. This conversion routine is called from only one other place in the BIS Engine code, a function at 0×8F7ECF which reads and tabulates a small number of variables. The memory access patterns of this tabulating function therefore indicate where the important processing algorithms are, since the memory addresses from which this tabulation function reads must be the memory addresses to which those critical processing algorithms ultimately wrote. Tracing backwards from this point therefore completes the analysis of following back from externally-accessible serial port data to the point of origin within the logic of the device. The function that calculates the BIS values can be found at memory locations 0×8F9172 through 0×8F94A2. Let us consider, though, the algorithm that calculates the total power (TOTPOW) and spectral edge frequency (SEF), which begins at 0×8FA739, as it is more informative to compare known algorithms to their implementation and thus demonstrate the correctness of this disassembly. The raw machine code for this function is shown in Table 2, accompanied by its disassembly into TMS320C32 assembly language and associated commentary on its operation. At the beginning of this fragment of code, R4 holds the location of an array in memory which contains an EEG power spectral density (PSD) distribution with bin sizes of 0.5 Hz from 0 Hz up to 30 Hz. The code implements a discrete version of an integration under the PSD curve to obtain the signal power:

$$\text{Signal Power} = \int S_{xx}(f)\, df \qquad \text{(Eq 1.1)}$$

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

where $S_{xx}$ is the power spectral density, and is here evaluated only between 0.5 Hz and 30 Hz, thus omitting any DC component. The total power (TOTPOW) is expressed in decibels, and is calculated as:

$$\text{TOTPOW} = 10 \cdot \log_{10}(\text{Signal Power}) + 40 \quad \text{(Eq 1.2)}$$

As per Rampil,[2] "SEF95 is the frequency below which 95% of the power in the spectrum resides." The subroutine FindSpectralEdge (Table 2) progressively sums underneath the PSD curve until a frequency is reached at which the cumulative amount *strictly exceeds* 0.95 of the signal power. In function form, this subroutine defines SEF95 as the lowest frequency that satisfies this inequality, evaluating at discrete intervals of $f$ of 0.5 Hz:

$$\int_0^{\text{SEF95}} S_{xx}(f)\, df > 0.95 \times \text{Signal Power} \quad \text{(Eq 1.3)}$$

This value is stored for the tabulating function to later retrieve and present as the parameter SEF.

## Discussion

Earlier, it was noted that the spectral edge frequency variable settles unusually to 30.0 Hz when there is no patient connector attached: this occurs when the EEG spectral density is zero throughout (i.e. perfectly isoelectric) for a prolonged time. The reason for this idiosyncratic result is clear in Table 2 at instruction 0×8FA72C. This subroutine cumulatively sums through the PSD to find the requested target fraction of the total power. The subroutine exits either when it obtains a sum that is strictly greater than the target value, or when it reaches the upper end of the PSD such that no further iterations are possible. The latter exit must occur when the PSD is zero throughout. The subroutine progressively sums zeros until it reaches the end of the supplied PSD, as it can never strictly exceed its target value, thus it then returns 30Hz as the requested spectral edge frequency. This is, unmistakably, a bug in the software code causing an idiosyncratic result. Firstly, the code would have returned 30 Hz as the spectral edge for any requested percentile, such as the median at 50%. Secondly, the correct value cannot be 30 Hz. The spectral edge frequency of an empty PSD should either be considered to be an undefined quantity or, more pragmatically, 0 Hz. This subroutine only produces 30 Hz as its answer because it has no further data to consider, and consequently this becomes the spectral edge frequency value that is sent out to charting systems via the device's serial port. The remedy is straightforward: the BGT (branch-greater-than) instruction should have been replaced with the BGE (branch-greater-than-or-equal-to) instruction, requiring a change in only one byte of the opcode 6A 09 00 02 to 6A 0A 00 02. The mistake is simple, but it highlights the difficulties inherent in implementing even the most trivial and well-understood algorithms. As Table 2 also indicates, it is an intellectually effortful task to convert a stream of machine code back into a clear expression of its overall intent. Although access to an automated decompiler is certainly helpful, the process is as much psychological as it is technical, since

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

it involves reconstructing the intentions and thought processes of the original programmer from a sequence of hexadecimal values.

Considering the overall structure of the BIS software, it is apparent that the ARM processor handles device initialization, and interface tasks such as user interaction, graphical display and communication with external devices via the serial port. The DSP processor and its BIS Engine code (biseng.io) are responsible for the calculating the critical BIS parameters. One could, in principle, use a software emulator of the TMS320C32 chip to execute the binary image in its entirety without much further examination. Such emulation software already exists within the free and open-source arcade machine emulator, MAME. The TMS320C32 DSP is an integral component of the Midway Zeus family of arcade machines, the most famous example of which is the game Mortal Kombat 4. Therefore, if it were desired to calculate BIS parameters from existing EEG data for research purposes, one could in principle execute the BIS Engine DSP code through direct emulation on a desktop computer or any other suitably powerful device, collecting and feeding in raw values appropriately in software, and collating the results from the emulated device's memory.

Of course, the primary attraction of disassembling the BIS Monitor internal software is to comprehend its workings. Since the BIS Engine was written in 1995, and the lifetime of a US patent is 20 years, one can infer that any pertinent patents should have expired or at least be rapidly approaching the end of their lifetime.[24,25] The device may contain trade secrets, but a trade secret is in itself no defense against independent discovery by an unrelated, unencumbered third party. Reverse engineering of a device sold on the open market is a legally recognized form of such discovery.[26] However, the purpose of this article is most certainly not to impugn the work of, or indeed the memory of, the original developers of the device. A search of PubMed for "bispectral index" reveals over 3000 articles published between 1993 and today. If the device had not had such an immense academic and clinical impact, there would be no merit in attempting to reverse engineer it. Meanwhile, the questions of what constitutes unconsciousness, and how it should be practically measured, remain absolutely central to the specialty of Anesthesiology. This raises two further troubling thoughts. Firstly, given that this body of literature is fundamentally predicated on a *nostrum*: how can we be certain that this scientific work is foundationally sound when we cannot know the assumptions on which it is based? Secondly, given that a specialty makes progress by being able to stand on the shoulders of its giants: how can one of the most intriguing developments of the last quarter-century be permitted to remain secret? The first point is simply one of necessary intellectual rigor. The second point seems particularly dolorous when the decline in academic research in Anesthesiology is lamented at annual conferences and in journals.[27,28]

The BIS Engine algorithms do not appear to have changed to any significant degree in years.[29] As this article demonstrates, there are no insuperable barriers to recovering the device's algorithms directly from the A-2000 motherboard using only publicly available and inexpensive tools. Would now not be a good time for the manufacturer to disclose the original source code for the BIS Monitor, for the good of the specialty? Much could be gained in terms of clinical and basic science research. Even if the algorithms were completely disclosed, the market position of the BIS Monitor should remain impregnable,

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

given the large and mature installed userbase and the regulatory barriers to entry for any new competitor.

## Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

## Acknowledgments

The author wishes to thank:

Dr. Jim Philip (BWH) and Dr. Aurora Burds (MIT) for providing feedback on this article during its writing.

Dr. Andrew Huang (bunnie:studios) for discussions on the technical capabilities of field-programmable gate arrays and flash storage.

Max Connor for pediatric patient modeling, Figure 1.

Financial Support:

NIH R01 GM121457

Departmental support

## Glossary

| | |
|---|---|
| **DSP** | Digital Signal Processor |
| **ECG** | Electrocardiogram |
| **EEG** | Electroencephalogram |
| **FPGA** | Field Programmable Gate Array |
| **JTAG** | Joint Test Action Group |
| **MAP** | Mean Arterial Pressure |
| **MPSD** | Modular Port Scan Device |
| **PCB** | Printed Circuit Board |
| **PSD** | Power Spectral Density |
| **RAM** | Random Access Memory |
| **RTOS** | Real Time Operating System |
| **SEF** | Spectral Edge Frequency |
| **SR** | Suppression Ratio |
| **TCK** | Test Clock |
| **TDI** | Test Data In |
| **TDO** | Test Data Out |

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

| | |
|---|---|
| **TMS** | Test Mode Select |
| **TSOP** | Thin Small Outline Package |

## References


1. U.S. Food and Drug Administration: 510(k) Premarket Notification K963644: A-1000 EEG Monitor and A-1050 EEG Monitor. Division of Cardiovascular, Respiratory and Neurological Devices Washington, DC, 1996
2. Rampil IJ: A primer for EEG signal processing in anesthesia. Anesthesiology 1998; 89: 980–1002 [PubMed: 9778016]
3. Sigl JC, Chamoun NG: An introduction to bispectral analysis for the electroencephalogram. J Clin Monit 1994; 10: 392–404 [PubMed: 7836975]
4. Viertio-Oja H, Maja V, Sarkela M, Talja P, Tenkanen N, Tolvanen-Laakso H, Paloheimo M, Vakkuri A, Yli-Hankala A, Merilainen P: Description of the Entropy algorithm as applied in the Datex-Ohmeda S/5 Entropy Module. Acta Anaesthesiol Scand 2004; 48: 154–61 [PubMed: 14995936]
5. Avidan MS, Zhang L, Burnside BA, Finkel KJ, Searleman AC, Selvidge JA, Saager L, Turner MS, Rao S, Bottros M, Hantler C, Jacobsohn E, Evers AS: Anesthesia awareness and the bispectral index. N Engl J Med 2008; 358: 1097–108 [PubMed: 18337600]
6. Myles PS, Leslie K, McNeil J, Forbes A, Chan MT: Bispectral index monitoring to prevent awareness during anaesthesia: the B-Aware randomised controlled trial. Lancet 2004; 363: 1757–63 [PubMed: 15172773]
7. Monk TG, Saini V, Weldon BC, Sigl JC: Anesthetic management and one-year mortality after noncardiac surgery. Anesth Analg 2005; 100: 4–10 [PubMed: 15616043]
8. Sessler DI, Turan A, Stapelfeldt WH, Mascha EJ, Yang D, Farag E, Cywinski J, Vlah C, Kopyeva T, Keebler AL, Perilla M, Ramachandran M, Drahuschak S, Kaple K, Kurz A: Triple-low Alerts Do Not Reduce Mortality: A Real-time Randomized Trial. Anesthesiology 2019; 130: 72–82 [PubMed: 30312182]
9. Lang J: Awakening, The Atlantic. Washington, DC, January/February 2013
10. Connor CW: Commonly Used Monitoring Techniques, in Clinical Anesthesia. Edited by Barash PG, Cullen BF, Stoelting RK, Cahalan MK, Stock MC, Ortega R. Philadelphia, PA, Wolters Kluwer, 2013
11. Hemmerling TM, Harvey P: Electrocardiographic electrodes provide the same results as expensive special sensors in the routine monitoring of anesthetic depth. Anesth Analg 2002; 94: 369–71 [PubMed: 11812701]
12. Gu Y, Liang Z, Hagihira S: Use of Multiple EEG Features and Artificial Neural Network to Monitor the Depth of Anesthesia. Sensors 2019; 19: 2499
13. Lee HC, Ryu HG, Park Y, Yoon SB, Yang SM, Oh HW, Jung CW: Data Driven Investigation of Bispectral Index Algorithm. Sci Rep 2019; 9: 13769 [PubMed: 31551487]
14. Connor CW: Artificial Intelligence and Machine Learning in Anesthesiology. Anesthesiology 2019; 131: 1346–1359 [PubMed: 30973516]
15. Bruhn J, Bouillon TW, Shafer SL: Bispectral index (BIS) and burst suppression: revealing a part of the BIS algorithm. J Clin Monit Comput 2000; 16: 593–6 [PubMed: 12580235]
16. Sleigh JW, Andrzejowski J, Steyn-Ross A, Steyn-Ross M: The bispectral index: a measure of depth of sleep? Anesth Analg 1999; 88: 659–61 [PubMed: 10072023]
17. Benissa MR, Khirani S, Hartley S, Adala A, Ramirez A, Fernandez-Bolanos M, Quera-Salva MA, Fauroux B: Utility of the bispectral index for assessing natural physiological sleep stages in children and young adults. J Clin Monit Comput 2016; 30: 957–963 [PubMed: 26515742]
18. Advanced RISC Machines (ARM): The ARM7DI ICEBreaker Module, in ARM7DI Data Sheet. ARM Developer Documentation, Cambridge, England, 1994 [https://developer.arm.com/documentation, accessed August 5th, 2020.]

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

19. Texas Instruments: TMS320C3x General-Purpose Applications. Texas Instruments Design Support Technical Documents, Dallas, TX, 1998 [https://www.ti.com/technical-documents/techdoc, accessed August 5th, 2020.]
20. Huang A: Hacking the Xbox: an introduction to reverse engineering. San Francisco, CA No Starch Press, 2003
21. Kernighan BW, Ritchie DM: The C programming language, 2nd edition. Englewood Cliffs, NJ, Prentice Hall, 1988
22. Goldschmidt RE: Applications of division by convergence. Thesis, Department of Electrical Engineering, Massachusetts Institute of Technology, 1964 [https://dspace.mit.edu/handle/1721.1/11113, accessed July 14th 2020.]
23. Stroud KA, Booth DJ: Engineering Mathematics, 8th Edition. London, England, Red Globe Press (Springer Nature), 2020
24. Chamoun NG, Sigl JC, Smith CP: Cerebral biopotential analysis system and method. US Patent 5458117, Aspect Medical Systems, Inc, 1995
25. Cordero RM, Devlin PH, Chamoun NG, Shambroom JR, Merrick EB, Melo J: Smart electrophysiological sensor system with automatic authentication and validation and an interface for a smart electrophysiological sensor system. US Patent 6298255, Aspect Medical Systems, Inc, 2001
26. Samuelson P, Scotchmer S: The Law and Economics of Reverse Engineering. The Yale Law Journal 2002; 111: 1575–1663
27. Chandrakantan A, Adler AC, Stayer S, Roth S: National Institutes of Health-Funded Anesthesiology Research and Anesthesiology Physician-Scientists: Trends, Promises, and Concerns. Anesth Analg 2019; 129: 1761–1766 [PubMed: 31743198]
28. Gelman S: Anesthesiologist scientist: endangered species. Anesthesiology 2006; 105: 624–5
29. U.S. Food and Drug Administration: 510(k) Premarket Notification K072286: BIS EEG Vista Monitor System and BISx. Division of General, Restorative and Neurological Devices Washington, DC, 2007

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

**Key Points (Question, Findings, and Meaning, each in one sentence)**

- After more than twenty years in clinical use, can the proprietary algorithms in the BIS monitor now be determined by a direct, forensic disassembly of the device hardware?
- The A-2000 BIS Monitor software executes on an ARM-based microprocessor and a digital signal processor: the program code for these devices can be extracted directly from the device motherboard and disassembled using free and inexpensive tools that are readily-available.
- With the expiration of the underlying patents, the device manufacturer should release the algorithms and their source code implementation to the anesthesia research community to promote further development of EEG monitoring algorithms and to place the current body of BIS-related research on a firm theoretical foundation.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript


**Figure 1:**

A healthy 12-year old child in natural, physiologic, unanesthetized sleep records a decline in BIS into the mid-20s. No burst suppression is seen. The low BIS scores seen during physiological sleep are remarkable, and the suppression ratio of zero calls into question the posited linear relationship between burst suppression and BIS scores below 50.

Author Manuscript
Author Manuscript
Author Manuscript
Author Manuscript



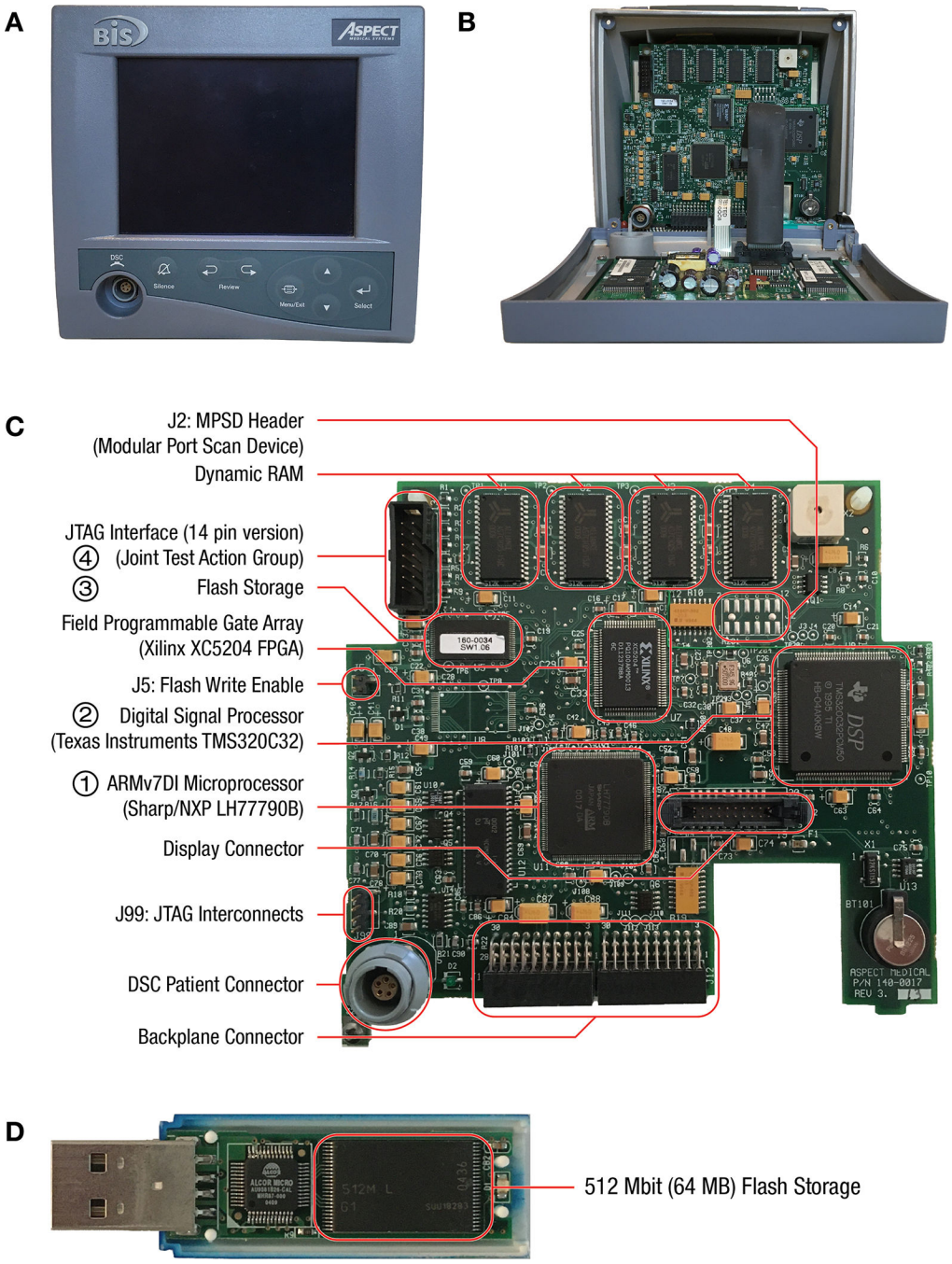


**Figure 2:**

(A) Front view of the Aspect Medical Systems A-2000 BIS Monitor.

(B) Detaching the display and front bezel reveals the Main PCB (Printed Circuit Board) within the body of the device enclosure behind. Dismantling the case in this way allows the critical computing components to be accessed easily with debug and test tools, while the Power Supply PCB and Interconnect PCB remain relatively protected against accidental damage.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

(C) The Main PCB removed from the BIS Monitor, with critical components and interconnects labeled as referenced in the Methods.
(D) For comparison, a partially dismantled flash drive of comparable vintage. Data is stored on the highlighted TSOP chip, similar to the TSOP chip that holds the BIS software on the Main PCB.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

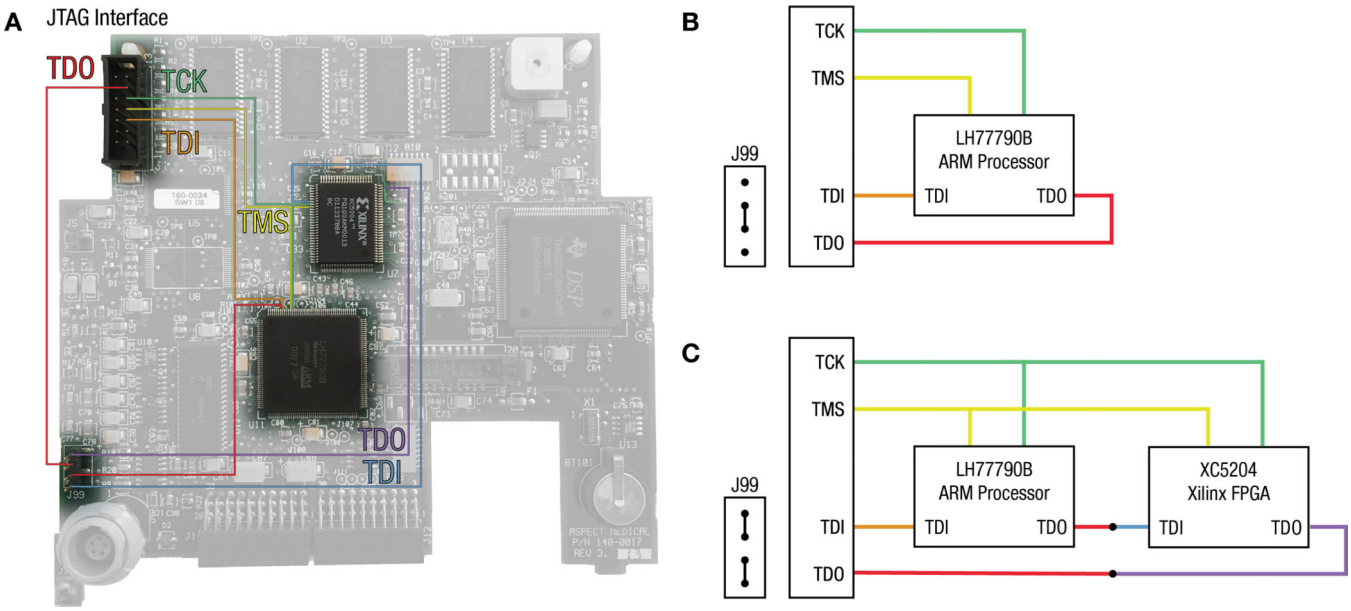


**Figure 3:**

(A) Functional layout of the JTAG Interface, involving the LH77790B ARM Microprocessor, the Xilinx FPGA, and the circuit board jumper J99. The major JTAG control lines are TDI (Test Data In), TDO (Test Data Out), TCK (Test Clock) and TMS (Test Mode Select).

(B) Debugging must be enabled by placing a jumper on J99. Connecting the middle two pins of J99 completes the continuity of the TDO line from the ARM Microprocessor to the JTAG Interface port, allowing external debugging to begin.

(C) JTAG allows multiple devices to be queried in sequence from one interface bus. Connecting the pins of J99 in pairs, as shown, additionally loops in the JTAG circuitry of the Xilinx FPGA by daisy-chaining the TDI/TDO communication path.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

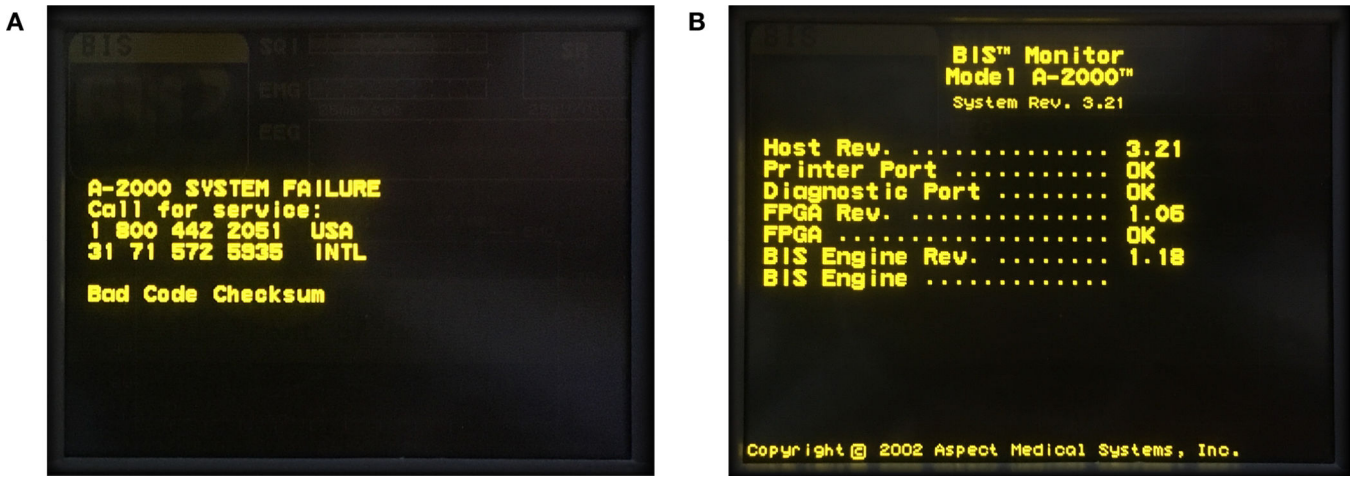


**Figure 4:**

Screen displays on the A-2000 BIS Monitor at key moments.

(A) The ARM internal software periodically calculates a validation checksum against itself to assure its own integrity. If this test fails, the device displays an alert and halts as shown, entering an infinite loop that can only be reset by turning the device off and on again.

(B) The moment during startup at which the device uses its main ARM processor to test the function of the TMS320C32 DSP chip and then initialize it with the software that calculates BIS parameters.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript

**Table 1:**

**Disassembly and analysis for device initialization, and representation of sensor data at the serial port and in internal memory.**

(A) A simple routine in ARM machine code that is used to copy program data from the memory-mapped location of the Flash storage to Dynamic RAM for execution. The columns show, respectively, the raw hexadecimal opcodes as extracted from the BIS monitor, disassembly into ARMv7 assembly language, and a decompilation of the assembly language into comparable C.

(B) Representation of a standard serial data packet captured from the serial port when no sensor array is connected to the patient. This dataset is useful as its contents are highly predictable and reproducible, but nevertheless non-zero in many regards.

Author Manuscript Author Manuscript Author Manuscript Author Manuscript



(C) Data from the serial port captured from an awake volunteer, along with a concordance showing hexadecimal data storage in memory at the same time. It is notable that many important parameters are stored in the format of scaled integers.

**A**

| Address | Machine Code | ARM7 Disassembly | Decompilation into C with Commentary |
|---|---|---|---|
| 09D056C | 22 08 A0 E3 | mov r0,#0x220000 | // Memory location 0x9d0598 holds value 0x800000. |
| 09D0570 | 24 10 9F E5 | ldr r1,[PTR_9d0598] = 00800000 | // Copying from flash storage to dynamic RAM. |
| 09D0574 | 03 26 A0 E3 | mov r2,#0x300000 | uint32 *source = 0x220000; |
| 09D0578 | 04 30 90 E4 | ldr r3,[r0],#0x4 | uint32 *dest = *ptr_9d0598; |
| 09D057C | 04 30 81 E4 | str r3,[r1],#0x4 | do { |
| 09D0580 | 02 00 50 E1 | cmp r0,r2 | *dest++ = *source++ |
| 09D0584 | FB FF FF 1A | bne 009d0578 | } while (source != 0x300000); |

**B**

| TIME | | | DSC | PIC | Filters | Alarm | Lo-Limit | Hi-Limit | Silence |
|---|---|---|---|---|---|---|---|---|---|
| 05/07/2020 19:05:35 | | | 10 | 0 | On | None | Off | Off | No |
| | SR11 | SEF07 | BISBIT00 | BIS | TOTPOW07 | EMGLOW01 | SQI09 | IMPEDNCE | ARTF2 |
| Ch. 1 | 0.0 | 30.0 | 0000 | 0.0 | 0.0 | 59.1 | 0.0 | 76 | 02000188 |
| Ch. 2 | 0.0 | 5.2 | 0000 | 0.0 | 0.0 | 59.1 | 0.0 | 50 | 02002000 |
| Ch. 12 | 0.0 | 30.0 | 0000 | 0.0 | 0.0 | 59.1 | 0.0 | 0 | 02000188 |

**C**

| Captured Serial Port Values For An Awake Subject | | | | | | | |
|---|---|---|---|---|---|---|---|
| | SR11 | SEF07 | BISBIT00 | BIS | TOTPOW07 | EMGLOW01 | SQI09 |
| Ch. 1 | 0.0 | 15.9 | 40c5 | 97.7 | 52.0 | 48.9 | 30.0 |
| Ch. 2 | 0.0 | 0.3 | 40c5 | 97.4 | 52.0 | 48.9 | 84.0 |
| Ch. 12 | 0.0 | 15.9 | 40c5 | 97.7 | 52.0 | 48.9 | 30.0 |

| BIS A-2000 In-Memory Parameter Block | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Address | SR11 (x10) | SEF07 (x100) | BISBIT00 | BIS (x10) | BIS (x10) | BIS (x10) | TOTPOW07 (x100) | EMGLOW01 (x100) | SQI (x10) | Array Padding | | |
| 0x99490C | 0x0000 = 0 | 0x0639 = 1593 | 0x40c5 | 0x03ce = 974 | 0x03d1 = 977 | 0x03d1 = 977 | 0x144e = 5198 | 0x1318 = 4888 | 0x0000 | 0x0000 | 0x0000 | 0x0000 |
| 0x994924 | 0x0000 = 0 | 0x001a = 26 | 0x40c5 | 0x03ce = 974 | 0x03d1 = 977 | 0x03d = 977 | 0x144e = 5198 | 0x1318 = 4888 | 0x0348 = 840 | 0x0000 | 0x0000 | 0x0000 |
| 0x99493C | 0x0000 = 0 | 0x0639 = 1593 | 0x40c5 | 0x03ce = 974 | 0x03d1 = 977 | 0x03d1 = 977 | 0x144e = 5198 | 0x1318 = 4888 | 0x012c = 300 | 0x0000 | 0x0000 | 0x0000 |

Author Manuscript Author Manuscript Author Manuscript Author Manuscript



**Table 2:**

Disassembly and commentary for the TMS320C32 DSP BIS Engine routines that calculate total signal power in decibels (TOTPOW) and 95$^{th}$ percentile spectral edge frequency in Hz (SEF). A small bug, shown in boldface, is present at 8FA72C, which causes nonsensical spectral edge frequencies to be returned if there is no signal power present (i.e. a perfectly isoelectric EEG).

| BIS Engine Code for TOTPOW (Signal Power, dB) and SEF (Spectral Edge Frequency, Hz) | | | |
|---|---|---|---|
| **Address** | **Machine Code** | **TMS320C32 Disassembly** | **Commentary** |
| 8FA739 | 50 63 00 3C | LDIU #0x3C,R3 | R4 holds the power spectral density (PSD) curve, with |
| 8FA73A | 50 62 00 01 | LDIU #1,R2 | bin resolution of 0.5 Hz. Integrate under this |
| 8FA73B | 50 0A 00 04 | LDIU R4,AR2 | whole power distribution with limits: |
| 8FA73C | 62 8F A6 63 | CALL SumOverRange | R2 = 0.5*1 = 0.5 Hz to R3 = 0.5 * 0x3C = 30 Hz |
| 8FA73D | 40 06 00 00 | LDFU R0,R6 | Which obtains the total signal power. |
| … | … | … | … |
| 8FA741 | 40 02 00 06 | LDFU R6,R2 | |
| 8FA742 | 62 8F DC 66 | CALL Log10OfR2ToR0 | Calculate the log (base10) of the total signal power. |
| 8FA743 | 50 68 02 39 | LDIU #0x239,AR0 | Calculate the correct storage location for this result for the |
| 8FA744 | 50 11 00 0C | LDIU AR4,IR0 | tabulation function later. |
| 8FA745 | 0A 60 32 00 | MPYF #10.0,R0 | Convert signal power into decibels. |
| 8FA746 | 01 E0 52 00 | ADDF #40.0,R0 | Power in dB = 10*$\log_{10}$(totalPower) + 40 |
| 8FA747 | 14 40 40 00 | STF R0,*+AR0(IR0) | Store this TOTPOW result for the table renderer. |
| … | … | … | … |
| 8FA74D | 50 0A 00 04 | LDIU R4,AR2 | Load AR2 with the location of the PSD curve, in R4. |
| 8FA74E | 40 02 00 06 | LDFU R6,R2 | Load R2 with the total power calculated above, from R6. |
| 8FA74F | 40 23 F2 6C | LDFU 0x8FF26C [= #0.95],R3 | Memory location 8FF26C holds the value 0.95 in floating point. |
| 8FA750 | 62 8F A7 25 | CALL FindSpectralEdge | Finds the 95th percentile of the distribution (subroutine below). |
| 8FA751 | 50 68 02 42 | LDIU #0x242,AR0 | Calculate the correct storage location for this result for the |
| 8FA752 | 50 11 00 0C | LDIU AR4,IR0 | tabulation function later. |
| 8FA753 | 14 40 40 00 | STF R0,*+AR0(IR0) | Store this SEF result in R0 for the table renderer. |
| FindSpectralEdge: A subroutine to find the frequency below which a specified fraction of the power spectral density occurs. | | | |
| 8FA725 | 24 82 02 03 | MPYF3 R3,R2,R2 | Scale the signal power in R2 by 0.95; sets the target value. |
| 8FA726 | 50 09 00 0A | LDIU AR2,AR1 | Put the starting address of the PSD into register AR1. |
| 8FA727 | 50 61 00 00 | LDIU #0,R1 | R1 is the current index into the PSD array, initialized at 0. |
| 8FA728 | 40 60 80 00 | LDFU #0.0,R0 | R0 is the running total of the PSD up to and including index R1. |
| 8FA729 | 50 68 00 3B | LDIU #0x3B,AR0 | Set the maximum number of loops, here 0 to 0x3B = 59. |
| 8FA72A | 01 C0 21 01 | ADDF *AR1++(1),R0 | Sum the power in the bin at memory location AR1 into R0. |
| 8FA72B | 23 00 00 02 | CMPF3 R2,R0 | Compare the running total of the PSD to the target in R2. |
| 8FA72C | **6A 09 00 02** | **BGT 8FA72F** | **Is the running total greater than the target value? If so, done.** |
| 8FA72D | 02 61 00 01 | ADDI #1,R1 | Not done, so add 1 to the index value. |
| 8FA72E | 6E 00 FF FB | DBU AR0,0x8FA72A | Loop back, unless the maximum number of loops is exhausted. |
| 8FA72F | 05 80 00 01 | FLOAT R1,R0 | Done, convert the integer index value in R1 into floating point. |
| 8FA730 | 0A 20 F2 6B | MPYF 0x8FF26B [= #0.5],R0 | Each bin is 0.5Hz, so scale the index value by 0.5. |
| 8FA731 | 78 80 00 00 | RETSU | Return from this subroutine to the calling function. |