Behavior Research Methods, Instruments, & Computers
2004, 36 (2), 291-303

Self-validating presentation and response timing in cognitive paradigms: How and why?

RICHARD R. PLANT, NICK HAMMOND, and GARRY TURNER
University of York, York, England

With the proliferation of commercial experiment generators and custom software within cognitive psychology and the behavioral sciences, many have assumed that issues regarding millisecond timing accuracy have been largely solved. However, through empirical investigation of a variety of paradigms, we have discovered numerous sources of timing error. These can range from poor scripting practices, to incorrect timing specifications, to hardware variability. Building upon earlier research, we have developed a commercial device and associated software that enables researchers to benchmark most computer-based paradigms in situ and without modification. This gives them the opportunity to correct timing errors where practicable, increase replicability, and reduce variability by altering onset times for stimuli, by replacing inaccurate hardware, or by post hoc statistical manipulation should the source of error be constant. We outline the features of the device and accompanying software suite, stress the importance of such independent validation, and highlight typical areas that can be subject to error.

Accurate timing is as important to psychologists as it has always been. It is commonplace to use complex paradigms to search for ever smaller effects, which demands precise stimulus presentation in multiple modalities and accurate response measurement. Priming studies are a typical example. Some authors argue that faster hardware, multitasking operating systems, and bloated application software has actually made accurate timing more difficult, not less (e.g., Myors, 1999). The opposing view is that, with certain caveats, multitasking operating systems can be used (e.g., MacInnes & Taylor, 2001; Forster & Forster, 2003). The argument seems to center on the level of control that the researcher can have over the hardware itself. This apparent paradox can be illustrated with a retrospective anecdotal quote from a researcher within our own department, reminiscing about his paper "Using a VDU for Tachistoscope CRT Displays"—specifically, the Cifer 2600 series VDU (Monk, 1981):

> Back in the old days I wrote my experiment in a low level language, I knew when the image would appear on screen as I could check the vertical sync flag on the display board; I could put characters straight in there—I had complete control. I even knew the phosphor delay!

We thank Jeff P. Hamm, Joseph MacInnes, Armand De Clercq, and Jonathan Vaughan for their helpful comments on an earlier draft of the manuscript. We also thank the anonymous researchers within our own department who were brave enough to let us test their equipment and paradigms during development. The Black Box Toolkit has a Web site, http://www.blackboxtoolkit.com/. It should be noted that the lead author now has a commercial interest in the specific equipment described in this article. Correspondence concerning this article should be addressed to R. R. Plant, Department of Psychology, University of York, Heslington Road, York, YO10 5DD, England (e-mail: r.plant@psych.york.ac.uk).

With an identical paradigm constructed in any modern experiment generator running on the latest hardware and operating system, would a researcher be as certain? Experiment generators have gone a long way to improve the lot for psychologists, and most claim millisecond precision, if not accuracy.

Cheap PCs[1] are everywhere, and unfortunately, commodity status can be their undoing. On closer examination, one would be hard-pressed to find a single identical component, software configuration, keyboard, or mouse across all the computers in a laboratory. Unlike the older generation of Apple IIs, Commodore PETs, or Cifer VDUs, current PCs are not a closed standard where each one is identical to the last. Variability in hardware components, operating systems, and software configuration cannot help but affect timing characteristics.

Does this matter? Won't an experiment generator iron out any problems? Yes, it does matter, and no, the experiment generator will not. Even the best experiment generator can know very little about the hardware with which it interacts. For example, reaction time errors within a simple paradigm using a mouse as a response device are subject to the timing variability inherent within the mouse hardware itself (Plant, Hammond, & Whitehouse, 2003). If one uses two different mice, one may obtain two statistically different sets of results even on the same PC. In this example, the largest component of timing error is accounted for by the mouse electronics and would affect any experiment generator or in-house software to an equal extent.

Problems become compounded where synchronization between two or more stimulus types is required or responses are made using uncalibrated devices. This is before one considers any driver interactions, hardware

conflicts, and background tasks that may need to be running at the same time. The result can be decreased statistical power or, worse, a systematic conditional bias invalidating apparent experimental effects. In short, there can be many sources of timing variability across all platforms. PC, Mac, and Linux systems are all affected, since they share a common hardware lineage. Such variability cannot be controlled unless researchers know where it lies, regardless of whether a recognized experiment generator is used or the researcher has written his or her own software.

Many researchers have expressed concern and have outlined various methods to help ensure timing accuracy. For example, McKinney, MacCormac, and Welsh-Bohmer (1999) have gone to elaborate lengths, using both hardware and software, to help ensure accurate timing when attempting to use a PC for tachistoscopic-style research paradigms. In common with the present authors' approach, later researchers, such as De Clercq, Crombez, Buysse, and Roeyers (2003), have used a second PC to test the timing accuracy of a first by use of inexpensive hardware that can monitor for keypresses and visual presentations. Developers such as Hamm (2001) have produced generalizable object-oriented modules that can be incorporated into researchers' own code to help ensure timing accuracy, whereas other researchers have concentrated on making Windows- or Macintosh-based experiment generators as accurate as possible (e.g., Forster & Forster, 2003, with the Windows-based DMDX; Bates & D'Oliveiro, 2003, with Macs). Finney (2001) has also outlined the real-time data-gathering capabilities of newer operating systems, such as Linux. Other authors have focused on the accuracy of alternative response devices, such as the joystick port (Shimizu, 2002).

One thing is clear. Since the development of the tachistoscope by Wundt in about 1875, psychologists have been concerned about timing accuracy. Many authors have suggested a variety of more or less complex methods for checking timing accuracy, for devices that register responses accurately, and for software techniques for improving temporal resolution. However, to date, there has been no simple, noninvasive, one-stop solution that can be applied to all platforms and the majority of paradigms in use today.

Fortunately, there may be a solution for those wishing to check their own timing accuracy quickly and easily. Following the ethos of electronic engineering, where equipment is calibrated yearly, we advocate self-validation of both the researchers' own hardware and the paradigms themselves through external chronometry. Building upon our earlier work (Plant, Hammond, & Whitehouse, 2002), we have developed a commercial device capable of acting as a *virtual human*. The device itself is connected to a Windows-based PC, which provides control and stores collected data. Software on the controlling PC enables the device to make responses to a variety of stimulus materials as might a human—the difference being that the equipment records any interaction with submillisecond accuracy and generates responses with known characteristics. Stimulus presentation timings can then be compared with those intended, and responses generated can be checked against those recorded by a given paradigm. It is useful to outline the interfaces offered by the toolkit hardware and then discuss the software suite before giving concrete examples of its use.

Benchmarking Paradigms Using the Black Box Toolkit

At the core of the Black Box Toolkit, or BBTK, is an eight-line digital input/output (IO) capability offering both powered and nonpowered interfaces to standard external peripherals. To maintain as wide a compatibility level as possible, the toolkit is connected to a host[2] Windows NT/2000/XP PC by means of a standard EPP or bidirectional parallel port (IEEE 1284). Using a typical PC, such as a 1.3-GHz Athlon, sampling rates of 48 KHz are readily achievable, meaning that events can be monitored and subsequently generated with submillisecond accuracy. By utilizing the parallel port, this means that standard PCs can be used without the need to resort to expensive timing and digital IO cards or use of complex bench equipment, such as digital oscilloscopes. Although the timing host must run on a PC, the toolkit's external sensor array can be used to benchmark paradigms running on any platform (see Figure 1).

The host Microsoft Windows system runs the toolkit's software suite and is connected physically to the toolkit via a standard parallel port. The remote system and paradigm that is being benchmarked is interfaced to the host via the various detection and generation modules offered. Figure 2 shows an opto-detector being used to monitor and time any visual presentations while an active switch closure lead is soldered, or clipped, to the left mouse button (not shown) of the remote PC. This lead feeds simulated mouse button events into the remote mouse under precise control. These appear to the paradigm running on the remote system as if they had been made by a human.

Currently, the eight-line BBTK offers the digital interfaces shown in Table 1.

Detection Interfaces

Opto-detectors. Up to four opto-detectors, each with an adjustable sensitivity threshold, can be used to monitor for screen events that occur on a remote system.

BBTK digital microphones. The BBTK offers support for up to two custom-built high-speed sampling digital microphones with adjustable trigger thresholds.

Passive switch closure detection on remote response devices. The BBTK can monitor for up to two passive switch closures on the remote system. Then when a participant presses a key on the response device, the BBTK will also register the properties of the response alongside the paradigm under test.

BBTK four-button response pad. The BBTK response pad functions as a standard four-button response

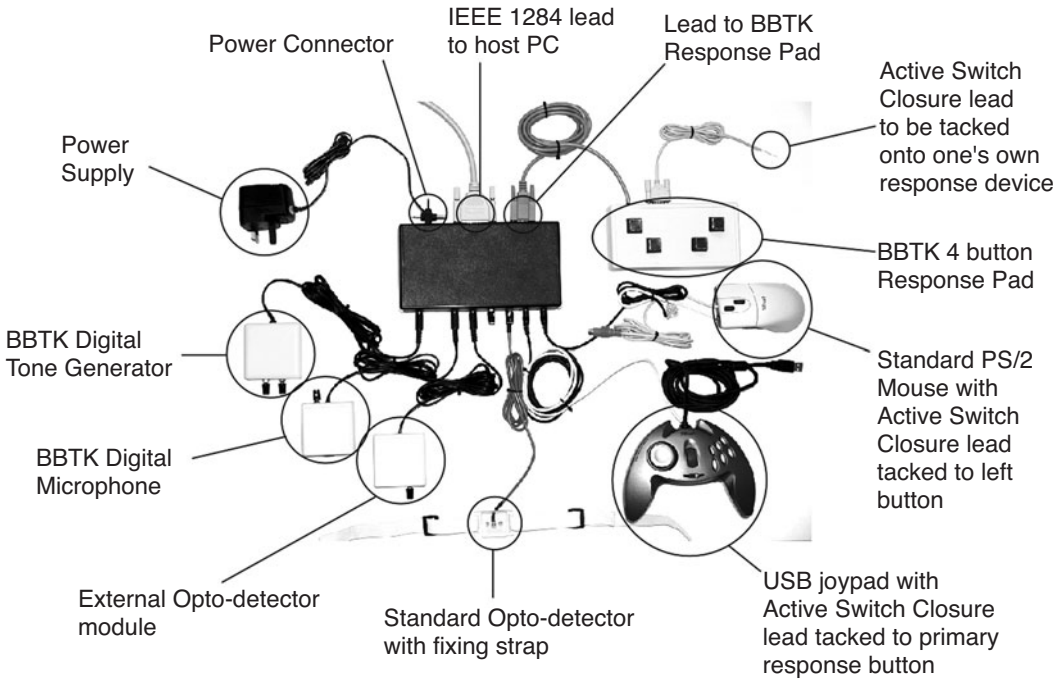


Figure 1. Detection and generation interfaces usable with the Black Box Toolkit (BBTK, center). Ordinarily, not all of the sensor and generating devices would be used at the same time.

pad. A nine-way interface links it to the rear of the BBTK. When BBTK software runs with the pad, it allows for all four buttons to be monitored and independently timed.

The toolkit can monitor for a visual and auditory stimulus on the remote system at the same time that it is detecting responses. The onset, duration, and offset for each buttonpress in relation to other monitored events are recorded with submillisecond accuracy. These can then be analyzed later to determine the exact time between any event on each of the eight lines.

Using the active response flying leads, the four BBTK response pad buttons can be soldered, or clipped, to those of the remote system's standard response device. This means that when a response pad button is pressed, the remote device is also triggered. In this scenario, the BBTK monitors any stimulus presentations and also records response characteristics. This may allow the researcher to make use of stimulus materials that typically adversely affect timing on the remote system, such as video playback.

Generation Interfaces

BBTK tone generators. The BBTK allows for the use of two tone generator modules constructed with piezoelectric speakers that have known timing characteristics

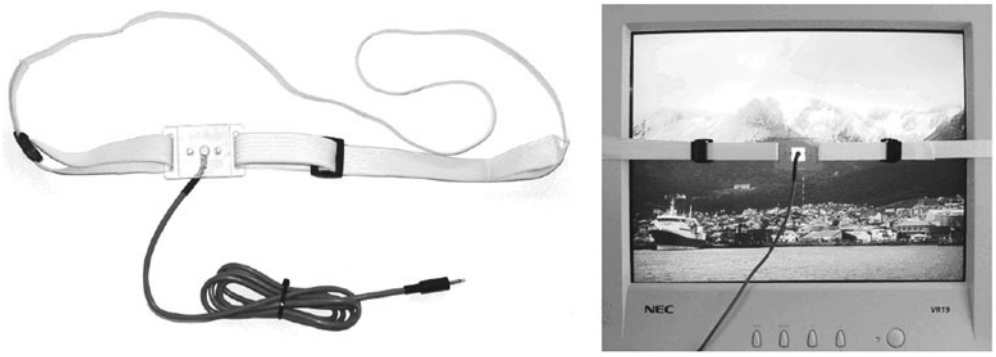


Figure 2. A standard Black Box Toolkit photodiode opto-detector shown left and attached to a standard CRT.

Table 1
Detection and Generation Interfaces Offered by the Black Box Toolkit (BBTK)

Detection Interfaces	Generation Interfaces	Dual-Mode Interfaces
Opto-detector (×2/×4): Image stimulus detection	BBTK Digital Tone Generator (×2): Triggering voice keys	BBTK Response Pad: A four-button pad enables use of the toolkit to independently time responses. In addition, each of the four buttons can be used to simultaneously trigger an external response device by means of four active switch closure leads.
BBTK Digital Microphones (×2): Audio stimulus detection	Active switch closure (×2): Feed controlled response events into response devices	
Passive switch closure (×2): Detect remote button down activity or sync pulse from external equipment such as an fMRI scanner		

Note—Some interfaces may be used in different roles as required.

to trigger voice keys as if a human had made a vocal response. It is also possible to make use of these powered-out lines for other purposes, such as robotic servo control.

BBTK active switch closure lines. The BBTK offers up to two active switch closure lines. These enable generation of key- or button-down events like those made by a human interacting with a remote system. For example, a researcher could solder, or clip, one wire to the left button of a mouse and the other to the right. He or she is then free to activate either button of the mouse to respond to events on the remote system. Three typical response devices with active switch closure leads attached are shown in Figure 3.

For a breakdown of the timing specifications of the toolkit and supporting peripherals, see the Appendix. As of May 2004, the commercial version of the toolkit is priced at approximately $1,000 U.S.

Typical Self-Validation Scenarios

A typical benchmarking scenario would see an opto-detector attached onto the screen of a system running an experimental paradigm. A microphone could then be placed next to a speaker, and one switch closure lead could be attached to the button of a mouse or response box. Using the BBTK software, this would allow for the following key measures to be taken: visual and auditory stimulus onset, offset, and duration; interstimulus interval; stimulus onset asynchrony (SOA) between stimuli types; and response onset, offset, and duration.

A variable response schedule could be associated with a stimulus event on any line. This would enable a response to be made at a known interval after a chosen stimulus appeared. Comparing the response onset generated by the BBTK with that recorded by the paradigm gives the response error and a measure of its variability. In short, this gives a calibrated measure of the performance of the paradigm and the equipment it is running on. Alternatively, the responses of a human participant could be monitored if a line was used to detect when a button was pressed on a remote response device. So rather than use a virtual human, a real set of trials with a human participant could be monitored with submillisecond precision. Again, the same timing information would be readily available, enabling comparison of visual and auditory presentation against responses made by a participant.

Revealing the hitherto unknown accuracy of the timing measures might enable either a redesign or a post hoc statistical correction. For example, if a sound stimulus should have been synchronized with a visual presentation but was late by 40 msec, the onset of the sound could be advanced

Figure 3. Typical response devices with active switch closure flying leads soldered to their primary and/or secondary buttons so that they can be controlled from the Black Box Toolkit.

by 40 msec, or the image could be displayed 40 msec later. By allowing noninvasive calibration of almost any paradigm, researchers have the opportunity for markedly improving their methodology. Even with judicious use of a four-channel signal generator and a four-channel digital oscilloscope, achieving this level of control is difficult, if not impossible, even for an experienced electronic engineer. Ease of use and interpretation are an important factor for which the software suite that accompanies the toolkit has been specifically designed.

The Software Suite

At the heart of the software suite lies an application that enables the toolkit to act as a virtual human by detecting a stimulus on any of the external sensors and then generating a response that is fed into a remote system. The deceptively simple application *Digital Stimulus Capture and Response* (DSCAR) enables the researcher to define patterns of activity and responses that can be made when those patterns are detected. In Figure 4, the lower spreadsheet defines the pattern of activity across the various sensors that will trigger a simulated response. The upper sheet allows for definition of a bank of reusable responses that will be generated in response to a stimulus pattern match.

In Figure 4, a single possible response has been defined in the upper sheet. For example, this could trigger an active switch closure line wired to the "Yes" key of a remote systems response device. The duration, in milliseconds, for which the line will be active is entered (100 msec in this case). Responses can also be made on other lines in order to generate tones or control TTL devices. Banks of up to 32,000 reusable responses can be defined.

Fifteen stimulus patterns that will trigger a response are shown in the lower sheet. With such notation, four blocks of four active lines can be monitored at any one time. Each block can contain a pattern of activity on up to four lines, which must be matched in order to trigger the associated response. In effect, the lines within a block are joined by a logical *and*. If required, another three blocks of four lines can be added. In effect, these additional line blocks are joined by the logical operator *or*. Once the stimulus pattern match has been defined and associated with one of the possible responses, the delay after detecting the stimulus pattern and generating the response is entered. For example, if we take Event ID 1 from the lower sheet, when an image is detected as being displayed on the screen of the remote system, the active switch closure line will be closed for exactly 100 msec,

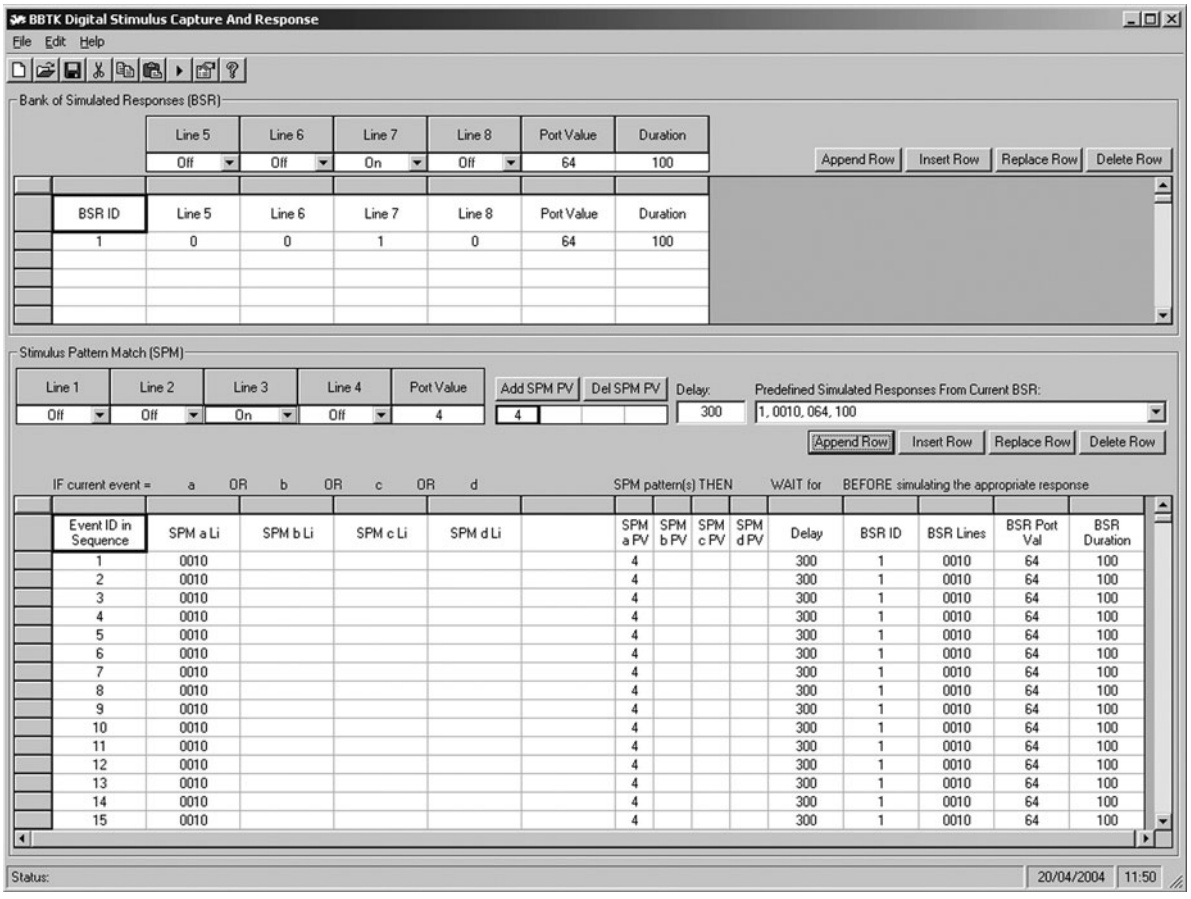


Figure 4. The Digital Stimulus Capture and Response application showing a sequence of 15 stimulus events to respond to.

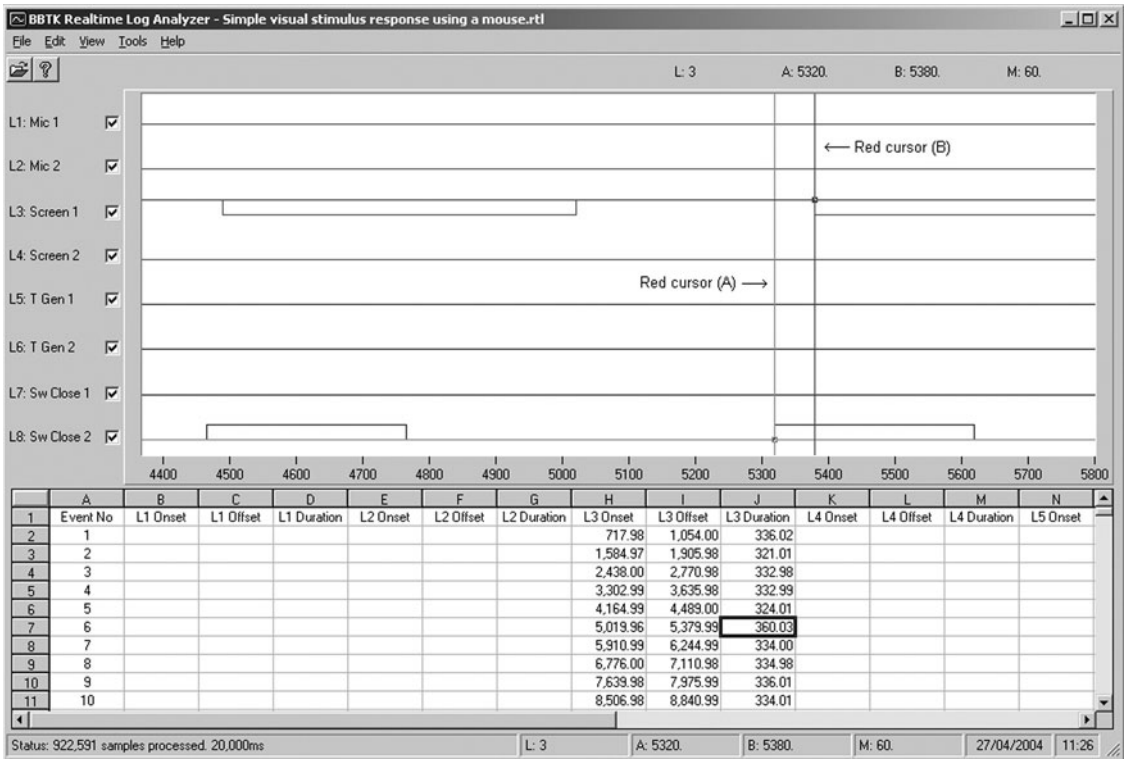
triggering the response device on the remote system as if a human had made the response. The response generated will occur exactly 300 msec after the leading edge of the visual stimulus has been detected. This could occur as the beam on a CRT passes the opto-detector, or it could occur when the sound wave reaches a BBTK digital microphone. In human terms, this would simulate that a human had seen the onset of an image, taken 300 msec to react, pressed the mouse button, and then held it down for 100 msec before releasing it.

The researcher needs to ensure that the paradigm will trigger each event in the sequence by displaying a stimulus pattern that will trigger a response. Each pattern definition in the sequence will be worked through until the list is complete or a timeout value has been reached. There can be up to 32,000 possible stimulus events, with each having unique timing, stimulus patterns, and range of possible responses.

DSCAR, in common with all the BBTK software, runs with real-time scheduling priority on the Windows-based host PC so that it takes as much of the processor time as possible, with very little or no interruption from other processes or applications. This means that, once running, the host's keyboard and mouse become inactive until the sequence is finished or a set timeout value reached. Data collected via DSCAR and all other software modules is streamed in real time to a predefined log file on the hard drive of the host PC. The sampling rate obtainable is dependent on the processor speed of the host PC. Typically, a 1.3-GHz Athlon can sample at around 48 samples per millisecond (48 KHz). Once the sequence is complete, the BBTK data analyzer (DA) can be used to examine the state of each of the eight lines at any time during the run. Theoretically, there is no limit on how long data can be captured, other than the size of the host hard drive.

The DA is analogous to an eight-channel digital oscilloscope display that can be used to replay any captured real-time log file. Four channels show inputs, and four show outputs. In the case of the BBTK, an opto-detector and powered microphones (passive switch closure or TTL detection) are used for input and tone generators, and active switch closure or powered TTL signals are used for outputs. A screen shot of a typical real-time log trace is shown in Figure 5.

The eight-track graph clearly shows the state of each line at any stage in the benchmarking run. This can be zoomed down to submillisecond granularity and panned left or right as required. Two moveable cursors are available, which enable one to measure the time between any two points in time on any line. In the example shown, Cursor A (green) has been positioned at the offset of a screen event on the opto-detector, and Cursor B (red) has been positioned at the point of the onset of a simulated switch closure that was fed into the mouse. Within the status bar, we can see that the offset for A is 5,320 msec and the onset for B is 5,380 msec, giving a time measure (*M*) of 60 msec. If we were interested in how accurately



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Event No	L1 Onset	L1 Offset	L1 Duration	L2 Onset	L2 Offset	L2 Duration	L3 Onset	L3 Offset	L3 Duration	L4 Onset	L4 Offset	L4 Duration	L5 Onset
2	1							717.98	1,054.00	336.02				
3	2							1,584.97	1,905.98	321.01				
4	3							2,438.00	2,770.98	332.98				
5	4							3,302.99	3,635.98	332.99				
6	5							4,164.99	4,489.00	324.01				
7	6							5,019.96	5,379.99	360.03				
8	7							5,910.99	6,244.99	334.00				
9	8							6,776.00	7,110.98	334.98				
10	9							7,639.98	7,975.99	336.01				
11	10							8,506.98	8,840.99	334.01				

Figure 5. A typical data analyzer trace and analysis sheet.

response time has been recorded by the paradigm under test, we would examine the response time it recorded and compare it against the expected 300 msec generated by the BBTK. Using a cursor-based approach means that measuring the onset, offset, duration, and SOA of any event is straightforward. In this example, the paradigm on the remote computer should have terminated the image display as soon as a response was detected. However, event six in Figure 6 shows that the actual image was displayed for 60 msec longer than it should have been. In addition, if we examine the reaction times recorded by the remote paradigm, we can determine whether there is any response time error. In this case, the mean reaction time is 352.27 msec, rather than the simulated value of 300 msec, with a standard deviation of 6.93 msec and a variance of 48.07 msec. The actual sequence of reaction times recorded by the paradigm is 351, 346, 354, 352, 346, 375, 355, 352, 349, 354, 350, 347, 350, 349, and 354 msec.

To aid the researcher still further, the DA automatically computes the onset, offset, and duration for each detected or simulated event and displays the results in a familiar spreadsheet view. The DA allows export of results and plots as Microsoft Excel, HTML, or BMP format file. Full clipboard functionality is also available. It is hoped that even the most demanding researcher would find the flexibility and power on offer welcome.

Other software modules within the suite offer digital stimulus capture (DSC) and event generation (EG). DSC offers functionality similar to an eight-channel digital oscilloscope. It records within a real-time log file anything that happens on any of the eight lines. The DA can then be used to check the resulting sequence of events. A typical use for DSC might be one in which two powered microphones are placed next to left and right speakers, two opto-detectors are placed on screen, and two passive switch closure lines are tacked to a "Yes/No" button of a standard response device. This would enable independent time stamping of all stimulus displays and responses made by a human during a set of trials. These data could then be used in comparison with what was recorded by the paradigm under test. Alternatively it could be used as the sole method for recording time-stamped data for both presentation and response.

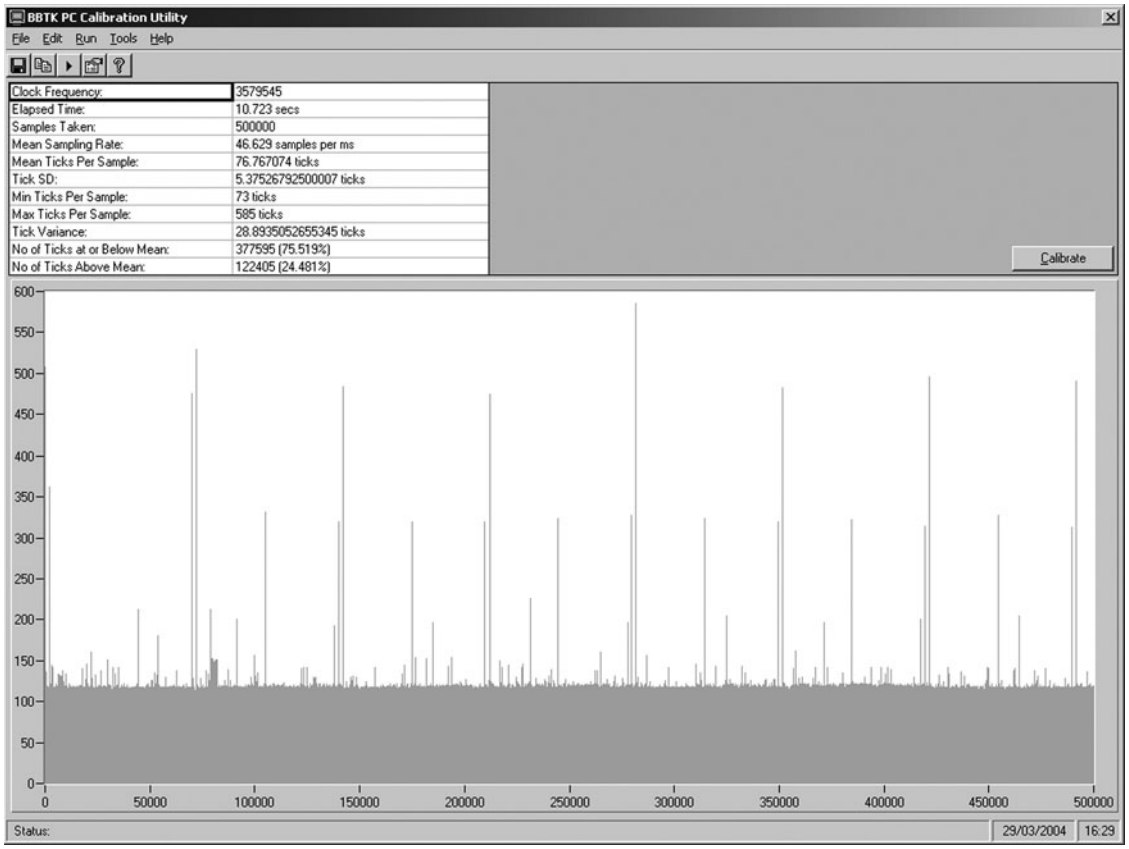


Figure 6. The calibration utility displays the number of machine cycles per sample after testing a typical PC (ordinate: machine cycles per sample; on this Athlon 1.3-GHz PC, the high-resolution performance counter runs at 3579545 Hz; therefore, each cycle takes 279.365 nsec). Note that 500,000 samples are shown. The mean number of machine cycles per sample is 77, but a very large number of sampling intervals are longer (120 machine cycles per sample), and a few are very much longer.

EG, on the other hand, offers functionality similar to a four-channel signal generator. The major difference is that EG offers constantly variable onsets, offsets, and durations on any of the four lines, as opposed to a bench generator, which typically offers fixed pulse trains. Two switch closures or two power outputs can be used. A typical use might be the generation of tones at regular intervals to test the performance of a voice key.

The final module of the software suite is the calibration module. This enables the toolkit to ascertain whether the host PC is suitable for high-speed data collection with the BBTK. It checks the sampling rate and variance obtainable from the PC used to host the BBTK hardware, determines the frequency of task scheduler interruptions, and varies the base scheduling priority between normal, high, and real time. By default, 500,000 samples are taken as rapidly as possible. Using the Windows API called "QueryPerformanceCounter," we are able to monitor the number of clock ticks that each sample takes. The lower the number of clock ticks per sample, the higher the resolution achievable on a given host. The number of samples per millisecond is also calculated (KHz). The plot shows the number of ticks (*y*-axis) against the sample number (*x*-axis). Any peaks indicate where another unknown process "stole" clock ticks from the software. The frequency of these peaks should be as low as possible. If they are frequent, they may affect the sampling consistency, and the researcher is so informed.

If the PC passes all tests and can achieve a high enough sampling rate consistently, it can be used for benchmarking with the BBTK. Recommendations for acceptable sampling rates are discussed in the BBTK software manual. The results of a typical calibration run are shown in Figure 6, and in Table 2 common sampling rates achievable from a range of PC processor speeds are shown. Note that in Table 2, there is a wide variation in performance, with only the PCs over 1 GHz achieving usable rates.

Examples of Advanced Use of the Toolkit

The BBTK can be used for very advanced timing verification and for paradigms that are known to raise problems with standard experiment generators. For example, it can be used to time video presentation on a remote PC. For example, visual markers (e.g., small white and black blocks) could be placed in the video stream to aid presentation timing via the opto-detectors. Response timing could be collected by using the BBTK response pad, rather than relying on an experiment generator's own response timing.

Other examples include using two toolkits to simulate and verify the timing of a paradigm that interoperates with an fMRI scanner. In order to test the presentation and response timing of the remote PC, BBTK No. 1 would generate a 1-msec-wide pulse every 1,993 msec, exactly matching a typical fMRI scanner. This pulse would be fed into the remote PC and also into BBTK No. 2. BBTK No. 2 would run DSC in Response Pad mode. It would detect images appearing on the remote PC as would a human in the real scanner. It would also detect any audio stimulus materials. The researcher would mimic a participant in the scanner and respond to either an image or the audio presented by the remote PC's paradigm. Responses would be made using the BBTK response pad. The active switch closure lead from the pad would close the response device on the remote PC at the same time.

By using a setup like this, we can check all aspects of the remote PC's performance in terms of sync pulse registration, visual display onset, duration and offset, auditory synchronization, response time, interstimulus interval, and so forth. This is done by comparing what is recorded using the second toolkit against what is recorded by the paradigm on the remote PC.

Using the Toolkit with other software. Since the BBTK uses a standard parallel port, this provides the opportunity to make use of all the toolkit interfaces on offer with other software so long as it has full access to the parallel port. For example, the four-button response pad could be used as a standard response device, or the two powered digital microphones could be used as voice keys. It is also possible to make use of the opto-detectors or passive switch closure features for response registration. Numerous pieces of commercial and noncommercial software can make use of the parallel port from within experiments, such as E-Prime, ERTS, Inquisit, or Presentation.

Three Case Studies Demonstrating Typical Sources of Timing Variance

Although researchers may desire millisecond accuracy, many may be unaware of all the factors that can influence both presentation and response timing. Without this awareness, replication can be difficult, and at worst

Table 2
Typical Sampling Rates for Common Processors When Sampling in Real-Time Priority Mode

	Celeron 433 MHz, 2000 SP4, 128-MB RAM	PIII 500 MHz, 2000 SP4, 256-MB RAM	Athlon 1 GHz, XP SP1, 1-GB RAM	Athlon 1.3 GHz, XP SP1, 1-GB RAM
Elapsed time (seconds)	50.24	48.40	13.85	10.72
Mean sampling rate (KHz)	9.95	10.33	36.10	46.63
Mean ticks per sample	359.70	346.49	99.15	76.77
Variance of ticks per sample	3,053,012.58	3,787.72	55.19	28.89

Note—The two Athlon CPUs are the same system with varied clock speed.

conditional biases can creep in. Here, we outline three sources of timing error that we believe to be commonplace. The toolkit we have developed is able to detect such errors with relative ease. However, in these real-world examples, a standard signal generator and digital oscilloscope were used together with the ETSL benchmarking kit outlined in Plant et al. (2002).

Case Study 1: Using TFTs and data projectors for visual presentation. TFT panels are rapidly replacing standard CRTs in all areas of computing. On the surface, they offer much the same functionality but are based on fundamentally different technologies, which can have an inherent side effect of introducing presentation delays. So rather than have to worry about refresh rates, researchers should now be contending with slower display onsets in timing-critical studies. For example, slow warm-up and image onset times can mean that a visual stimulus will no longer be synchronized with an auditory one, as before. Since data projectors are based on fundamentally the same technology, the issue of response time applies here, too.

In order to illustrate some of the issues involved, we constructed a simple visual presentation paradigm, using E-Prime Version 1.1 (Psychology Software Tools, Inc.).[3] Each trial consisted of a parallel port high signal (using in-line code), which lasted for 100 msec, followed by an 800 × 600 bitmap displayed for 100 msec and, finally, a blank black screen displayed for 100 msec. A digital oscilloscope, together with a fast-response photodiode (BPX65) and flying lead from the PC's parallel port, was used to "calibrate" the E-Prime paradigm. A 19-in. NEC CRT was driven at 100 Hz (10-msec screen redraw) with the photodiode attached mid-screen. Testing with the scope showed the 100-msec parallel port signal to be constant at 100 msec, as was expected. However, it was found that each bitmap, which should have appeared on the refresh after the parallel port signal, was late by a consistent 30 msec. Its duration was accurate at 100 msec. This effectively extended the image onset to 130 msec relative to the parallel port signal. To compensate for this, we reduced the wait after the end of the image display to 70 msec. For faster presentations, images can be loaded into memory, using the canvas object together with in-line code. Here, this was not a concern, since consistency was the main goal. Once calibration had been completed, this gave a consistent interstimulus interval of 300 msec, which matched our conceptual understanding of the paradigm.

Once calibration had been completed, we had a solid base by which to assess various display technologies and interface types (digital and analog). With as many factors as possible held constant, differences in image display onset, duration, and offset can be attributed to differences in display technology and interface type. A summary of the display timings obtained is shown in Table 3.

On cursory examination, the performances of the majority of the display devices look similar. However, on examination of the data recorded by the digital scope, the illumination curves detected via the photodiode are strikingly varied. Exactly when the image is seen by a human participant is dependent on the illumination curve, together with the panel's inherent response time. A typical display curve for a TFT is shown in Figure 7. The majority of the TFT panels match their specifications in terms of quoted response time (B+C), or typically around 30 msec. However, due to the way some manufacturers calculate response time, those quoted may be faster than the panel actually operates in the real world. It is also worth bearing in mind that many panels are locked in the 60–75 Hz refresh rate range, with most actually specifying 60 Hz as the recommended rate. This is the rate that software will actually use to calculate display intervals, regardless of panel speed.

The response time of TFT panels can lead to ghosting effects when games are played or DVD movies are watched. Given the ability to calibrate displays, one could in theory start an image 20 msec earlier (B) with the panel shown in Figure 7 and make the duration 130 msec, to achieve a 100-msec presentation (this takes account of the fall time, C). In the case of E-Prime, it would also be necessary to start the whole process 30 msec earlier, to take account of the image load time (A). However, synchronizing displays with other stimuli may still be unreliable unless their timing characteristics were also al-

Table 3
Sample Timings (in Milliseconds) From a CRT, a Range of TFT Panels, and a Data Projector

Display Device	Resolution	Interface	Windows Reported Refresh Rate (Hz)	Time From End of LPT Signal to Start of Display (A)	Time for Display to Warm Up (B)	Time for Display to Cool Down (C)	Total Display Time (D)
NEC 19-in. CRT	800 × 600	D-SUB	100	20–30	NA	NA	100
NEC 19-in. CRT	800 × 600	D-SUB	75	20–30	NA	NA	100
Viglen panel	800 × 600 (native)	D-SUB	75	30	20	10	120
LG	800 × 600 (scaled)	D-SUB	75	40–50	10	20	130
LG	1,280 × 1,024 (native)	D-SUB	75	40–50	10	20	130
LG	1,280 × 1,024 (native)	DVI	60	40–50	10	20	130
LG	800 × 600 (scaled)	DVI	75	40–50	10	20	130
Sony Vaio laptop TFT	800 × 600 (scaled)	Internal	75	40	15	5	120
Sanyo data projector	800 × 600 (scaled)	D-SUB	100	40	100	20	120

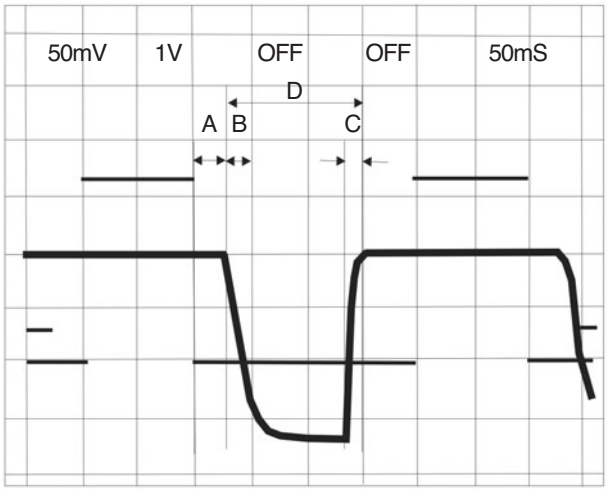


Figure 7. Example TFT panel timings. Thick line: display intensity (downward deflection is brighter). (A) Time from parallel output off state to start of monitor display (30 msec). (B) Time for display warm-up (20 msec). (C) Time for display cool-down (10 msec). (D) Total display time (120 msec).

tered. It was comforting to note that scaling the image on panels had no effect on display timing over their native resolution. Altering the refresh rate and interface used between digital (DVI) and analog (D-SUB) again made no difference to display timing.[4]

The biggest surprise was the timing characteristics of the Sanyo Data Projector,[5] shown in Figure 8, with a rise time of 100 msec and a fall time of 20 msec. For example, synchronizing an auditory stimulus with this display device could mean having to start the image display some 100 msec earlier. It is hard to know at exactly which point during the long 100-msec rise a human would perceive the image, as compared with a standard CRT. This would suggest that data projectors should not be used for

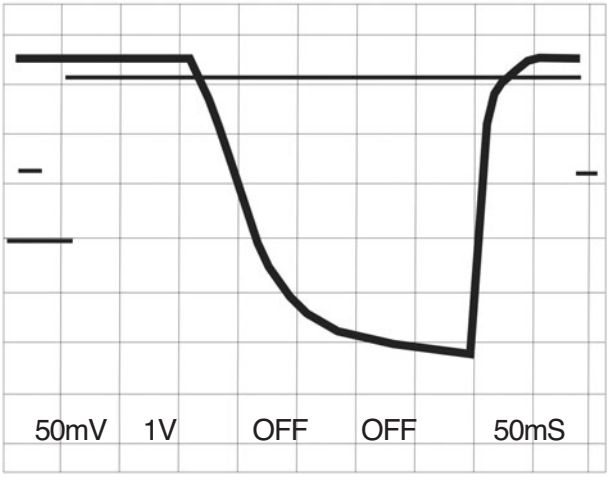


Figure 8. The display curve for a Sanyo data projector. Thick line: display intensity (downward deflection is brighter).

presenting visual stimulus materials where timing is critical without careful consideration.

Undoubtedly a CRT offers the best performance for work involving high-speed presentation. With TFT technology becoming more common, the concern is that CRT and TFT technologies, with their inherent response times, are not equivalent. Certainly, for frame-by-frame presentation, CRTs remain the only option. Our data also highlight the danger in simply swapping display devices within an existing paradigm and assuming presentation will remain constant.

Case Study 2: Using standard peripherals for response registration. Research by such authors as Segalowitz and Graves (1990), Beringer (1992), and more recently, Chambers and Brown (2003) and Plant et al. (2003) tells us that if we use a mouse or other standard peripheral as a response device, we are open to the variability inherent in that device's electronics over and above that within the PC and software in use. Anecdotal evidence would suggest that newer optical, infrared, and wireless mice are slower and less consistent than older examples.

In Plant et al.'s (2003) article, we compared seven mice, a typical keyboard, and a PST response box, using a simple visual stimulus response paradigm constructed in E-Prime. Each response device was first tested when detached from a PC, using a signal generator and an oscilloscope to find the inherent delay present in the electronics and microcode. The benchmarking rig outlined in Plant et al. (2002) was used together with E-Prime to detect and respond to a visual stimulus by generating a response after 300 msec. This original rig, although considerably more complex, is functionally identical to the BBTK.

When we examined the response times recorded by E-Prime, there was an additional contribution almost equaling that predicted by independent hardware testing. The response box contributed the least with the mice and keyboard, ranging from a mere 8 msec to around 80 msec in terms of absolute error, with some devices displaying a variability of over 20 msec. Figure 9 shows a plot of the variability in reaction times recorded by E-Prime when subtracted from the 300-msec button-down event fed into each device.

If one used the longest latency mouse (top line, M = 66.30 msec, SD = 6.48 msec) and the shortest latency mouse (bottom line, M = 10.15 msec, SD = 0.53 msec) in two between-subjects experimental conditions, there would be an apparently statistically significant difference. For example, this could happen if one condition was tested in the lab and the other on site, using two lots of equipment. Counterbalancing might not help much if the mouse difference added to within-group variability.

The key issue here is that there are no industry timing standards for mice, soundcards, or just about any peripheral one can think of. Even with supposed standards, there can be a huge variation in latency. So long as devices work and are electrically safe, they can be sold both to the general public and to scientists. Such inherent hardware delays affect every platform equally. Without self-

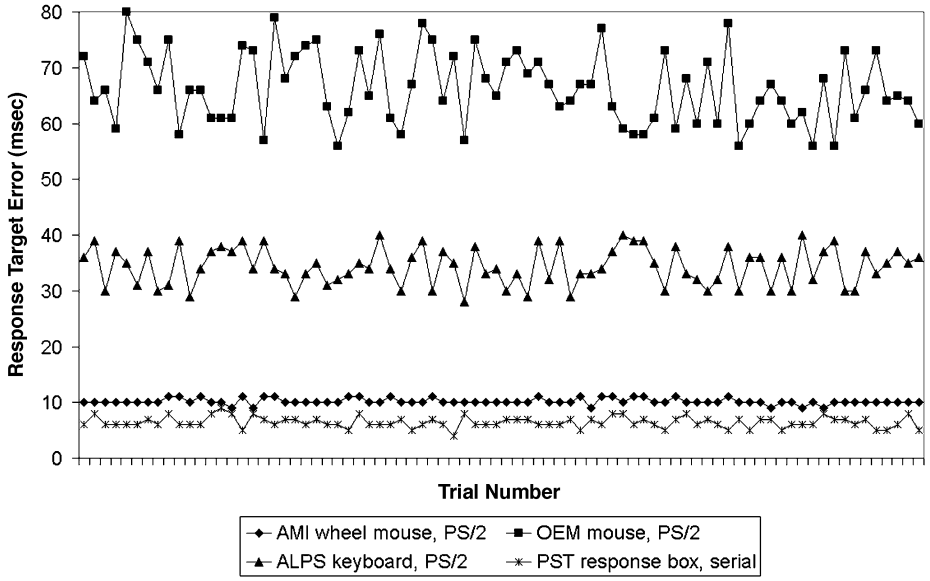


Figure 9. A plot of variability in response times recorded by E-Prime.

validation, one has no way of knowing the delays inherent in one's own hardware. Whether it be response devices or the start-up latency of a soundcard, such variance can make replicating one's own paradigms difficult.

Case Study 3: Cross-modal priming using PsyScope and an external button box on a Mac. Again, the 2002 rig was used to benchmark an active researcher's paradigm within our own department. That paradigm was running on a Mac Performa 630 (OS 7.1) with PsyScope, using an external CMU button box for response and experimental timing. The paradigm tested participants' ability to respond to one of three stars displayed to the left, middle, or right of the screen. On some trials, audio was present in either the left or the right channel, and in different conditions, the participants were instructed to respond either to the star or to a tone. Responses were made using the left–right button on the CMU button box. The tone and star were presented together on some trials and not on others, were presented concurrently, or were presented after a delay of 50 or 100 msec. The duration of the stars on the screen should have been 200 msec, as should the duration of the audio if present on a given trial. Our equipment was set up to act as a virtual human, as is shown in Figure 10. It could detect all visual presentations, detect audio events on either the left or the right channel, and make responses into the button box. Responses were fed into the CMU button box exactly 300 msec after a star on the screen was detected. This was around average for a human participant in the study. We also reran the paradigm to see whether switching from the Mac's internal timers to those in the CMU button box made any significant difference in timing.

The equivalent of the BBTK's DSCAR application software was set up to make responses to 100 trials. Once it was running, a response was automatically generated as scheduled each time a star appeared. By analyzing the data, we were able to check the timing of every aspect of the paradigm. This encompassed all visual and auditory display timings. By comparing reaction times recorded by PsyScope against the 300 msec that was simulated by the rig, we were able to calculate response time error figures. A summary of the general findings is outlined in Table 4.

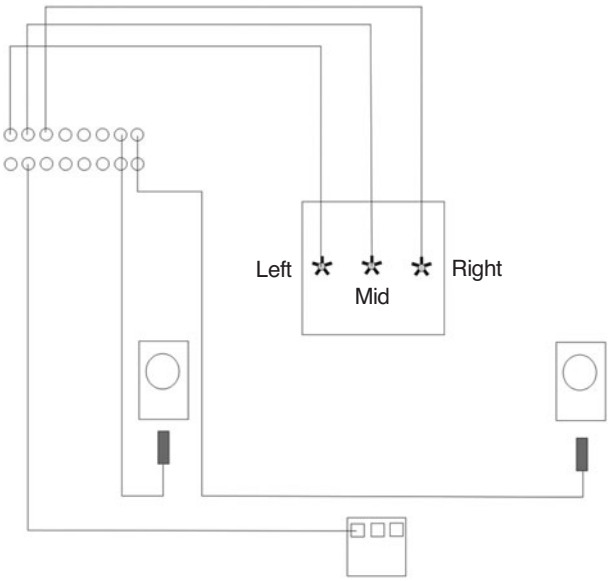


Figure 10. An overview of the cross-modal priming study and position of the opto-detectors over each of the star display locations, with digital microphones in front of the speakers, and active switch closure leads tacked to the back of the primary response button.

Table 4
Summary of PsyScope Timing When Benchmarked in Situ, Running a Cross-Modal Priming Study

Measure	Expected (msec)	Observed (msec)	
		M	*SD*
Star duration	200	196.46	0.12
Tone duration	200	210.66	2.07
Tone synchrony to star presentation	0	6.38	0.09
	50	56.88	0.30
	100	106.62	0.30
Reaction time			
Internal timers, no tone	300	304.58	0.58
CMU timers, no tone	300	303.10	0.47
Internal timers, tone present 0 msec	300	313.81	0.60
Internal timers, tone present 50 msec	300	321.30	0.61
Internal timers, tone present 100 msec	300	321.78	0.50
CMU timers, tone present 0 msec	300	311.47	0.56
CMU timers, tone present 50 msec	300	320.10	0.55
CMU timers, tone present 100 msec	300	320.12	0.69

As can be seen, display duration was remarkably accurate at just under 200 msec. Tone duration was later by nearly 11 msec. The tone-to-star synchrony, where present, was good at around 7 msec over the expected target time. Reaction time with no audio was good at around +3 msec for the CMU timers and +5 msec when internal timers were used. However, when a tone was present, response registration was slowed, in each condition with internal timers, by 14 msec for the concurrent condition (0 msec) and around 22 msec for 50- and 100-msec lags. This was slightly better than that for the CMU timers, at around 11 and 20 msec, respectively. This highlights a worrying conditional bias where audio is involved. This error had previously gone unaccounted and unnoticed by the researcher in question. With this knowledge, the researcher was able to make a post hoc statistical correction, since the actual variation was small, albeit with a large absolute difference. The researcher had previously checked his presentation timings with an oscilloscope, which accounts for why they were so good. At the time, however, he did not have the facilities to check stimulus–response timing. This case study highlights the benefits of using a virtual human to check timings and also the effectiveness of self-validation.

Conclusions

Our hope for the toolkit is that researchers will take the issue of presentation, synchronization, and response timing far more seriously than they currently do. We appreciate that this is difficult to accomplish without the correct training and specialized equipment. Here, we both offer a suitable technical solution and outline a methodology that makes use of an infallible, submillisecond accurate virtual human that can tirelessly complete hundreds of trials.

In common with other scientific endeavors, we would like to see researchers noting that they have actively checked timing within published academic papers. We feel that within some fields, the instability of research effects may be at least in part attributable to equipment timing error—perhaps, for example, in the priming literature.

The three case studies detailing timing errors in the field help illustrate the pressing need for caution. These apply regardless of platform and whether one uses an experiment generator or one's own custom-written software. Without checking, human error in defining paradigms within experiment generators or code is also an ever-present issue despite the best intentions of the researcher.

We foresee self-validation of timing accuracy as the only way forward for today's complex computer-centric studies. Although a set of component benchmarks (see http://www.psychology.ltsn.ac.uk/etsl/) has been tested against E-Prime, ERTS, and Superlab in 2000 and 2001, using the rig outlined in Plant et al. (2002), the tests did not sample the full variety of hardware and systems that other researchers may be using, and so the results cannot be generalized beyond the hardware actually tested. Although the benchmarks may be sound and may be negotiated and agreed on as fair with the three companies involved, the quality of data will be best served by calibration benchmarks conducted with the specific systems used in one's own laboratory.

REFERENCES

Bates, T. C., & D'Oliveiro, L. (2003). PsyScript: A Macintosh application for scripting experiments. *Behavior Research Methods, Instruments, & Computers*, **35**, 565-576.

Beringer, J. (1992). Timing accuracy of mouse response registration on the IBM microcomputer family. *Behavior Research Methods, Instruments, & Computers*, **24**, 486-490.

Chambers, C. D., & Brown M. (2003). Timing accuracy under Microsoft Windows revealed through external chronometry. *Behavior Research Methods, Instruments, & Computers*, **35**, 96-108.

De Clercq, A., Crombez, G., Buysse, A., & Roeyers, H. (2003). A simple and sensitive method to measure timing accuracy. *Behavior Research Methods, Instruments, & Computers*, **35**, 109-115.

Finney, S. A. (2001). Real-time data collection in Linux: A case study. *Behavior Research Methods, Instruments, & Computers*, **33**, 167-173.

Forster, K. I., & Forster, J. C. (2003). DMDX: A Windows display program with millisecond accuracy. *Behavior Research Methods, Instruments, & Computers*, **35**, 116-124.

Hamm, J. P. (2001). Object-oriented millisecond timers for the PC. *Behavior Research Methods, Instruments, & Computers*, **33**, 532-539.

MacInnes, W. J., & Taylor, T. L. (2001). Millisecond timing on PCs and Macs. *Behavior Research Methods, Instruments, & Computers*, **33**, 174-178.

McKinney, C. J., MacCormac, E. R., & Welsh-Bohmer, K. A. (1999). Hardware and software for tachistoscopy: How to make accurate measurements on any PC utilizing the Microsoft Windows operating system. *Behavior Research Methods, Instruments, & Computers*, **31**, 129-136.

Monk, A. F. (1981). Using a VDU for tachistoscopic CRT displays. *Current Psychological Reviews,* **1**, 357-361.

Myors, B. (1999). Timing accuracy of PC programs running under DOS and Windows. *Behavior Research Methods, Instruments, & Computers*, **31**, 322-328.

Plant, R. R., Hammond, N., & Whitehouse, T. (2002). Toward an experimental timing standards lab: Benchmarking precision in the real world. *Behavior Research Methods, Instruments, & Computers*, **34**, 218-226.

Plant, R. R., Hammond, N., & Whitehouse, T. (2003). How choice of mouse may affect response timing in psychological studies. *Behavior Research Methods, Instruments, & Computers*, **35**, 276-284.

Psychology Software Tools, Inc. (n.d.). *E-Prime 1.0*. See http://www.pstnet.com/e-prime/, for details on E-Prime for Windows. Surface mail: 2050 Ardmore Boulevard, Suite 200, Pittsburgh, PA 15221-4610.

Segalowitz, S. J., & Graves, R. E. (1990). Suitability of the IBM XT, AT, and PS/2 keyboard, mouse, and game port as response devices in reaction time paradigms. *Behavior Research Methods, Instruments, & Computers*, **22**, 283-289.

Shimizu, H. (2002). Measuring keyboard response delays by comparing keyboard and joystick inputs. *Behavior Research Methods, Instruments, & Computers*, **34**, 250-256.

NOTES

1. In this context, *PC* refers to the term *personal computer* and covers IBM-compatibles, Mac, Linux boxes, and so on.

2. The term *host* is used to refer to the Microsoft Windows-based PC that controls the toolkit. The term *remote* is used to refer to the second machine running the paradigm that is being benchmarked.

3. Although E-Prime was used, the same results would apply to other experiment generators, since the timing delays are inherent within the display hardware itself.

4. Our Connect3D/ATI Radeon 7000 64Mb graphics card had both DVI and D-SUB outputs.

5. A new model as of June 2003.

APPENDIX
Timing Specifications

The timing tests were carried out using a digital oscilloscope and our own specialized toolkit microphones and tone generators.

Table A1
Black Box Toolkit Timing Specifications

Black box sampling rate across all 8 lines (typical sampling rate on a 1.3-GHz Athlon)	48 KHz
Powered input/output timed from parallel port to input/output pin of peripheral or switch closure	<100 nsec
Microphone timed from output of tone generator to parallel port (Op-amp amplified Electret microphone uses microcontroller to detect peak and cancel false triggering)	<50 μsec
Opto-detector input timed from diode to parallel port	<100 nsec
2.5-mm switch closure timed from parallel port to contact	35 μsec
Tone generator timed from parallel port to piezo sounder pin (Piezo sounder with pitch and amplitude control via 2 potentiometers)	50–625 μsec

(Manuscript received December 4, 2003;
revision accepted for publication April 29, 2004.)