Int. J. Human-Computer Studies (2000) **52,** 1031–1069
doi:10.1006/ijhc.1999.0341
Available online at http://www.idealibrary.com on IDEAL®

Calculators are needlessly bad

HAROLD THIMBLEBY
Middlesex University, Bounds Green Road, London, N11 2NQ, UK.
email: harold@mdx.ac.uk

(Received 25 June 1999 and accepted 26 July 1999)

In the two decades hand-held calculators have been readily available, there has been ample time to develop a usable design and to educate the consumer public into choosing quality devices. This article reviews a representative calculator that is "state of the art" and shows it has an execrable design. The design is shown to be confusing and essentially non-mathematical. Substantial evidence is presented that illustrates the *inadequate documentation*, *bad implementation*, *feature interaction*, and *feature incoherence*. These problems are shown to be typical of calculators generally. Despite the domain (arithmetic) being well defined, the design problems are profound, widespread, confusing—and needless. Worrying questions are begged: about design quality control, about consumer behaviour, and about the role of education—both at school level (training children to acquiesce to bad design) and at university level (training professionals to design unusable products). The article concludes with recommendations.

"The problem of efficient and uniform notations is perhaps the most serious one facing the mathematical public." Florian Cajori (1993)

"[...] contrivances adapted to peculiar purposes [...] and what is worse than all, a profusion of notations (when we regard the whole science) which threaten, if not duly corrected, to multiply our difficulties instead of promoting our progress." Charles Babbage, quoted in Cajori (1993).

KEYWORDS: consumer product user interfaces; feature interaction; feature incoherence; calculator and calculator user interfaces.

1. Introduction

My daughter is 11 years old and her school recommends that she buys a particular make and model of hand-held calculator. This calculator baffles me, even though I have three degrees, one in physics and two in computing. My difficulties are not because I am incompetent in my use of the calculator; my difficulties are unusual because I have noticed them and spelt them out. They are everyone's problems (Thimbleby 1999). Given that a primary school recommends the calculator to its children, and its manufacturers intend it to be better than the already most popular calculator of its class (Appendix C), I would have thought that no basically numerate adult should have any trouble understanding anything written here! This is not so; therefore there is a problem.

Something odd is going on in the world: that a primary school recommends something so complex, and that someone with a numerate Ph.D. cannot understand it.

1071-5819/00/061031 + 39 $35.00/0

Nobody seems to notice or care that there might be a problem in all this. That critical and analytic thought is not recruited by consumers to critique complicated gadgets suggests manufacturers should take special care over their product design. On the contrary, it is possible that manufacturers do not want consumers to understand their products. Electronic calculators have been around for many years. There is evidence that manufacturers do develop their products (Section 3); thus each year, the manufacturers could have made further improvement in the products, using standard methods such as iterative design (Nielsen, 1993)—had they been engaged in it. There are no valid excuses for manufacturers, like cost, ignorance or novelty.

Calculators have been widely studied, by computer scientists (Aho, Sethi & Ullman, 1985; Thimbleby, 1996*a*) psychologists (Halasz & Moran, 1983; Young, 1981, 1983) and designers. Standard design tools, such as compiler–compilers have been widely available since the 1970s (Johnson, 1975); their use would have *automatically* picked up some of the numerous design problems discussed here. The calculator this papers critiques is made by a leading manufacturer, and it is hardly exceptional in its problems (Section 3). Why are these things so badly designed? One wonders just how long the manufacturers themselves spent evaluating the design or, more pointedly, just *what* they evaluated it for. Given that these products are sold to children, and especially given that they are sold to do calculations—often a serious activity, for financial work, for medical work, for teaching children what mathematics is about—there ought to be an obligation to design them with care and to make them easier to understand. This article indicates numerous ways in which the particular calculator and calculators in general can be improved.

There is a temptation to blame usability problems on users, rather than on design (Thimbleby, 1993). Doing studies of users might tend to focus attention on to users' behaviour rather than on to design issues. Blaming users happens particularly when errors have significant consequences, as in safety critical systems (Leveson, 1995)—in hindsight, users could have done something else, so users are to blame for the errors—or they should have bought a different calculator to start with.

We already know that users try to understand systems and that documentation plays a major part in the success of users (Carroll, 1998). Lack of design insight, a symptom of which is inadequate explanation, is one of the main reasons for poor design. The lack of documentation then exacerbates the users' problems: the device is badly designed, badly documented, and therefore the bad design is even harder to cope with. Knowledge of the difficulties of users does not in itself provide an explanation that would help make the calculator better. Moreover, users' experience alone is not a guide for better design (Christensen, 1997), because better design requires a deeper understanding.

What of the user's conceptual models? A few thousand years of the development of the notations of mathematics have developed a representation for mathematical models. Calculators *can be* designed that are directly consistent with such models (Thimbleby, 1996*a*, 1997). (How cognitive approximations to mathematical models are compromised by broken calculators is an interesting area of study but is beyond the scope of this article.)

The calculator's domain, mathematics, is well defined, its notations are well defined, and its technologies, microelectronics and computer science, are well established. All the technical problems (such as portability, parsing, numerical techniques and reliability) are solved. One infers that opportunities to design well have not been exploited by the manufacturers. We take the evidence from the calculator's inadequate documentation as a symptom of a serious mismatch between the potential and what actually happens.

Style of this article: This article describes problems with a "simple" gadget. I have specified particular models so readers can double check facts. The calculators are as I describe them. It would not be possible to make substantial claims without grounding this article in specific products. However, there may be a danger that this article is taken as an attack on Casio, the manufacturer of the calculator, or on a specific product. Section 3 shows that the problems are generic: they apply to more calculators and to more manufacturers. In any case, a school recommends the specific Casio calculator: in some sense, then, there is external evidence that this calculator is "best", and therefore it is appropriate to critique this particular model. Casio themselves appear to think highly of the model (Appendix C).

This article is analytic, not empirical. Experiments are a good way of falsifying conjectures or of determining statistical parameters; they are not, however, a good way of evaluating nonsense—an example given by Deutsch (1997) is that no experiment is required to establish that eating grass does not cure the common cold because we have a good explanation of the cold. We have a good explanation for calculators, but nobody has been using it.

Notation: The Casio *fx-83W* is complex, so there is a balance between explaining it clearly against being faithful to how it actually is. The manufactuer's *User's Guide* adopts the convention that when a key has several meanings (e.g. see Figure 1) the *User's Guide* shows the meaning determined by the context. Thus, to store the current result in the memory *M* actually requires the keys [STO] [M+] to be pressed, but as M is one of the several labels of the key [M+] it is clearer to write [STO] [M], even though no *key* [M] exists.

There is one exception to this convention used in this article (but not in the *User's Guide*) the key [SHIFT] is not shown. For example, the percent key is represented here by [%] but in fact is keyed by pressing two keys, [SHIFT] [=]. This one exception means we do not discuss, for instance, that [SHIFT] [hyp] [sin] and [hyp] [SHIFT] [sin] have the

FIGURE 1. Schematic of the [M+] key. The key is black with M + in white. The labels M– and CL are in yellow; the M (top right) in red; and the DT and lines underneath are in blue. The SD mode ("standard deviation") uses blue labels, and the shifted functions are yellow: the colour coding of the CL function is meant to indicate that in SD mode, a shifted [M+] press obtains the function CL, not M–.

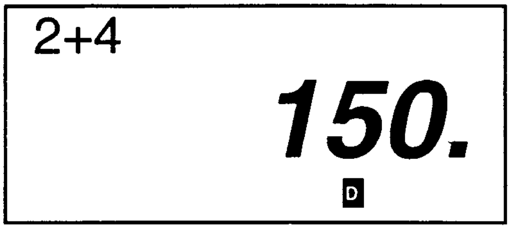

FIGURE 2. Schematic of the *fx-83W* calculator liquid crystal display. The "2 + 4" (top left) is displayed in a dot matrix font, and the "150." is displayed in a 7-segment font. The small D is a custom symbol, indicating the calculator is in degrees mode. While a sum is entered, it is displayed in the top line and the answer line is blank. Here, the sum 2 + 4 has been entered, and the answer is shown as 150 (it would have been 6 if [=] had been pressed, but [%] was pressed instead—note that it is not possible to tell what causes the result shown).

same effect, because both meanings would be represented here as [hyp] [$\sin^{-1}$].† We will not be concerned with the shift key as such further in this article, because it only causes minor confusion that could be easily avoided if desired (by relabelling keys)—an example taken from the calculator's *User's Guide* is: [SHIFT] [B] by which it means pressing [SHIFT] [8], whereas [ALPHA] [B] means pressing [ALPHA] [°′″] (note that two separate keys are labelled B).

This article's convention suppresses certain design details, for instance that in some modes some keys have different meanings. A reader of this article will not know which keys have which meanings (a consideration that is relevant for some design decisions, such as key labelling). However, this article is not a manual for the calculator but a critique of its design. The criticisms stand, even though a reader may not know, without reference to the calculator's manual, which physical key is referred to.

2. The Casio *fx-83W*

Consumers select a calculator by its image; if people want a different design then they can select a different model. We are therefore justified in taking the design as given, and seeing how badly it works out, taking it on its own terms.

The *fx-83W* has 51 button (including the "P" button on the rear of the case), at least 11 modes and 159 functions including hyperbolic sines and regression. The *User's Guide* is 54 pages long.‡

The number and range of functions itself is not a problem. An English dictionary may have hundreds of thousands of words that will never be "used" in the lifetime of its owner. The problem with the calculator is that, unlike a dictionary, its functions are badly organized and interact confusingly with each other. Its spurious complexity is unnecessary.

† Because $\sin^{-1}$ is the shifted meaning, this label is beside the key not on the key top as our convention might be thought to suggest. Incidentally, the calculator's choice of notation is undesirable: arcsin (or asin) is preferred, since $\sin^{-1} x$ is too easily confused with $(\sin x)^{-1}$ (Gullberg, 1997)—especially as $\sin^2 x$ is standard notation meaning $(\sin x)^2$.

‡ Had the *fx-83W* used a conventional alphabetic keyboard (26 letters), 10 digits and the usual operators ($+, -, \times, \div, (,), =, \sqrt{}$, etc.), then all of its functionality could have been achieved very simply, with fewer buttons and only one mode.

2.1. OVERVIEW OF *FX-83W*'S DESIGN PROBLEMS

The design of the *fx-83W* appears to be *ad hoc* and there is no clear and easy way to summarize the design problems. This section gives a brief list of high-level design problems; the Appendices continue with further substantiation of this list.

1. *Inadequate documentation*: The *User's Guide*, Quick Reference, key legends, and diagnostics all provide information on how the calculator works and can be used. In all areas the documentation is weak. Bad documentation obviously makes the calculator harder to understand, since the user has to work out how the calculator actually works. Bad documentation is a symptom of the manufacturers themselves being unable to understand the calculator; it is a symptom that the technical authors do not understand the calculator.

   For specific examples of inadequate documentation, see A.1.
2. *Feature interaction*: Various features seem all right individually but interact in strange ways. Feature interaction makes the calculator harder to understand: features cannot be understood in isolation, but must be understood in all possible contexts. Feature interaction makes some tasks impossible to perform. Examples include the interaction between implicit multiplication and the "speed up" recall of memory values; this and further examples of feature interaction are discussed more fully sections 2.3 and A.2.

   The calculator has *useful* feature interactions that are not documented. When the [=] key is pressed the last calculation is repeated, and combined with using the [Ans] button it is possible to do interesting things. For example, [Ans] [+] [1] [=] makes a simple counter: each press of just [=] adds one to the running count. The technique can be used to explore chaos theory, iterative convergence to numerical problems, and even to do cryptography.
3. *Feature incoherence*: Various features seem all right individually but duplicate or overlap each other in strange ways. Examples include the two forms of [(−)] and [−], that almost mean the same thing. The features individually may be reasonable, but the user's problem is that a coherent task may have to be split between different features in arbitrary ways. Feature incoherence makes some tasks difficult to perform. Examples of feature incoherence are provided in Appendix A.3.
4. *Bad implementation*: What is implemented is not general. The rules for [%] do not seem systematic (Section 3 and Appendix 5): it is not a general operation, and it never appears in the calculator display, so a calculation with it cannot be fully edited. A single decimal point is sometimes treated as zero, sometimes a syntax error—but the user has to find out for themselves when, because the implementation is *ad hoc*. Bad implementation makes the calculator harder to understand; explanations do not generalise.
5. *Poor usability engineering*: The calculator's bad design, despite many years of experience, supports the view that the manufacturer benefits from no usability engineering process. Nielsen (1993) has a list of eleven usability heuristics, and the calculator breaks every one, apart from having clearly marked exits (which it achieves with the [AC] key—and even that has problems in Section A.2.4). The calculator provides no undo, so users' errors are exacerbated.

Ironically for a calculator being aimed at the educational sector (Appendix C) all features—simple and advanced—are equally accessible. Even disregarding the feature interaction, presenting all features *at once* is known to be counter-productive (Carroll & Carrithers, 1984). A teacher cannot help children focus on particular sorts of tasks or methods; nor can a user temporarily restrict the features of the calculator to make their tasks less likely to be subject to interference from other features. The *fx-83W* does not have a basic mode; no mode is available that restricts the calculator to elementary operations. Even to make the calculator do normal calculations, displayed normally in degree mode, seven keys must be pressed: [AC] [MODE] [1] [MODE] [4] [MODE] [9] (switching the calculator off and on does not change its mode; there is no short-cut for setting this normal mode). The calculator provides no prompts for the essential [1], [4] or [9] keys here, and there is no error checking—missing a digit out, for instance, means that the next [MODE] key press would get quietly ignored.

6. *Poor quality control*: Many problems come down to unnecessary complexity and details that are misleading or just plain wrong. Figure 3 illustrates the complicated and varied rules for displaying fractions. The Quick Reference card with the calculator gives an example of calculating $\cdot 5\frac{2}{3} + 3\frac{3}{4}$, but the $\cdot$ symbol, which looks like a decimal point, is irrelevant—the key presses given for this example are [5] [a b/c] . . . The answer is `9⌟5 ⌟12.` (both in the Quick Reference and when actually performed on the calculator): the trailing decimal point is potentially misleading, since (as Figure 3 and Section A.2.2 make clear) decimals cannot be combined with fractions successfully. The users of this calculator who *need* fractions are not going to be helped by such sloppiness.
7. *Hyperbolic presentation*: The calculator includes "Casio's VPAM logic" (Appendix C), and elsewhere called "super VPAM" (Section A.1.1) but it should be no more

Typical format		Achieved by	Brief comment
`1.23`	decimals fixed	[MODE] [7] [2]	Permanent mode
`1.23`	significant figures fixed	[MODE] [8] [3]	Permanent mode
`1°2°3.`	sexagesimal	[←]	Temporary
`1⌟2⌟3.`	fraction	(see caption)	Pressing [a b/c]
`5⌟3.`	vulgar fraction	(see caption)	Pressing [d/c]
`1.23⁰³`	engineering	[ENG]	Temporary
`5.⁻⁰³`	NORM 1	[MODE] [9]	Permanent mode
`0.005`	NORM 2	[MODE] [9] (same as NORM 1)	Permanent mode

FIGURE 3. Summary of different approaches to display formats. Notice the occasionally superfluous decimal points. The rules for fractions are complex: to be displayable as a fraction, a calculation must apparently—the manual does not specify any rules—contain uses of [a b/c] (*not* [d/c]), integers, and no real operators (such as square root) applied to a fraction, even if the result is rational. For example [1] [⌟] [2] [÷] [3] results in a display `1⌟6.`, but using [1] [+] [·] [5] obtains `1.5`, which, although numerically equal to both 3/2 and 1 + 1/2, *cannot* be displayed as `3⌟2` or `1⌟1⌟2.`

than doing arithmetic conventionally and hardly need a proprietary or special technology.

The manufacturers further claim, "equations can now be entered as they are written" (Appendix C). This is misleading, for example, -2^3 has to be entered as [(−)] [2] [x^y] [3] [=], and it gets displayed as −2 x^y 3, or fractions (which are written using lines or slashes, as in $\frac{1}{2}$) are entered using [a b/c], and get displayed like 1⌟2..

Although the *User's Guide* gives $56 \times (-12) \div (-2.5)$ as a worked example, which *could* be entered as it is written, the *Guide* gives 56 [×] [(−)] 12 [÷] [(−)] space keys = as the key presses, which is hardly as it is written!

2.2. DENOTATIONAL SEMANTICS

The bulk of this article discusses the design of the *fx-83W* in a naturalistic way. It may be, then, that the design difficulties we supposedly identify are artifacts of the inadequate way of discussing the design. ("If you don't understand mathematics, no wonder you don't understand a calculator!") Indeed, the discussion of the mathematical aspects of the calculator's design should be appropriately mathematical. This section therefore shows, to the contrary, that a rigorous mathematical exploration of the design merely exposes further problems.

The denotational semantics approach splits the definition of a notation into its syntax, which defines its phrase structure, and its semantics, which defines its meaning (Allison, 1986; Tennent, 1981). As shown in Figure 4, the syntax of arithmetic expressions is easily defined in BNF (an explanation of BNF and a more substantial syntax for arithmetic expressions is given in Backus *et al.* (1976)). It is assumed that any phrase that cannot be generated by the syntax is a syntax error: a calculator would display "syntax error" and possibly point to the first point of departure of the keys entered and correct syntax.

The semantics of a notation are defined by mapping each syntactic form into a corresponding mathematical expression. The mapping is written using "emphatic brackets", so $[\![\cdot]\!]$ maps a syntactic form into its value. Since the semantics of expressions are mostly just the usual meaning of arithmetic, the semantic equations used in this article look trivial.

In Figure 4, there is a production defining expressions recursively as expressions followed by + followed by terms: this is written in BNF as $\langle expression \rangle ::= \langle expression \rangle + \langle term \rangle$. This form of expression has a meaning, written $[\![e + t]\!]$, where e is an

$$
\begin{aligned}
\langle expression \rangle &::= \langle expression \rangle + \langle term \rangle \,|\, \langle expression \rangle - \langle term \rangle \,|\, \langle term \rangle; \\
\langle term \rangle &::= \langle term \rangle \times \langle factor \rangle \,|\, \langle term \rangle \div \langle factor \rangle \,|\, -\langle factor \rangle \,|\, +\langle factor \rangle; \\
\langle factor \rangle &::= (\langle expression \rangle) \,|\, \langle numeral \rangle; \\
\langle numeral \rangle &::= \langle numeral \rangle \langle digit \rangle \,|\, \langle digit \rangle; \\
\langle digit \rangle &::= 0|1|2|3|4|5|6|7|8|9
\end{aligned}
$$

FIGURE 4. A syntax for simple integer arithmetic.

$$n \in \langle numeral \rangle$$
$$d \in \langle digit \rangle$$
$$[\![nd]\!] = 10 \times [\![n]\!] + [\![d]\!];$$
$$[\![0]\!] = 0;\ [\![1]\!] = 1;\ [\![2]\!] = 2;\ [\![3]\!] = 3;\ [\![4]\!] = 4;$$
$$[\![5]\!] = 5;\ [\![6]\!] = 6;\ [\![7]\!] = 7;\ [\![8]\!] = 8;\ [\![9]\!] = 9$$

FIGURE 5. A semantics for ⟨*numeral*⟩, decimal numerals. For example, the numeral 07 has a value of 7, since $[\![07]\!] = 10 \times [\![0]\!] + [\![7]\!] = 7$.

expression and t is a term.† The meaning is, of course, the value that is the sum of what e means, and of what t means. Thus the semantic rule is: $[\![e + t]\!] = [\![e]\!] + [\![t]\!]$. Here it looks like the $+$ is used twice: but on the left (inside the $[\![\cdot]\!]$ brackets) $+$ is merely a symbol; on the right, $+$ is the conventional mathematical operator that adds numbers. Notice that this rule defines a meaning in terms of the meanings of *smaller* phrases. In turn, the meaning of these phrases will be defined in terms of still smaller phrases—until we reach the meanings of digits. Figure 5 shows how simple the meanings of numerals and digits are.

A desirable property of a notation is that it is referentially transparent, that the meaning of expressions do not depend on how they are referred to. However, in general, the semantic equations for a phrase t could depend on what syntactic category t is. If so, several semantic functions $[\![\cdot]\!]_A$, $[\![\cdot]\!]_B \ldots$ would be required. In the arithmetic example (Figure 6), a third semantic equation could be used $[\![t]\!]_{\langle expression \rangle} = [\![t]\!]_{\langle term \rangle}$. Since the meaning of t is the same, this equation serves to make the referential transparency explicit. Because arithmetic is indeed referentially transparent, the meanings of a phrase t, well formed both as a term and as an expression, are the same in both contexts.

It is straight-forward to provide the complete denotational semantics of calculations, building on the outline provided here. To complete the denotational semantics, we would also add conditions that the length and complexity of an expression is limited—the *fx-83W* can only handle expressions up to 127 symbols—and that there should be no numerical overflow. Finally, semantics may be provided to define the precision of the arithmetic. Interestingly, these conditions are merely "meanings" and can be handled in exactly the same fashion as the arithmetic semantics. For example, the length requirement semantics would be a collection of equations like $\mathscr{L}[\![e + t]\!] = 1 + \mathscr{L}[\![e]\!] + \mathscr{L}[\![t]\!]$, which says that the length of a phrase $e + t$ is the length of e plus the length of t plus one (for the length of the $+$ sign itself). If the top-level production is ⟨*calculation*⟩ ::= ⟨*expression*⟩ =, then the meaning of this is that the overall calculation is not too long, namely, $\text{Valid}[\![e =]\!] = \mathscr{L}[\![e]\!] < 127 \rightarrow \text{VALID} : \text{TOO-LONG}$.

Having reviewed denotational semantics of conventional arithmetic, and shown that we can handle numeric value, length limitations and numerical limitations, we now move on to showing that the denotational semantics of the *fx-83W* is very obscure. In particular, it loses the elegance and simplicity seen above in the semantics of everyday arithmetic, where the semantic equations are so trivial they are hardly needed. In arithmetic, this simplicity means that the meaning of $+$ (as a symbol) is "the same" as the meaning of $+$ (as an arithmetic operator); it means that the meaning of 234 (as

†One might write $[\![e\ \boxed{+}\ t]\!]$ more specifically, but doing so begs, at least for the *fx-83W*, very complex questions on the (unfortunate) arithmetic side-effects of editing keys: see Section 2.3. Since editing is *in principle* (though not for this calculator!) separate from the mathematical meaning, this section simply takes the keys somehow, and in a way we do not detail, as generating the expressions whose meaning we wish to define.

$$
\begin{aligned}
&e \in \langle expression \rangle \\
&\quad t \in \langle term \rangle \\
&\qquad [\![e + t]\!] = [\![e]\!] + [\![t]\!]; \\
&\qquad [\![e - t]\!] = [\![e]\!] - [\![t]\!]
\end{aligned}
$$

FIGURE 6. A partial semantics for $\langle expression \rangle$ (conventional arithmetic).

a numeral) is "the same" as the meaning of 234 (as a number). In everyday usage, then, we do not need to, and in fact do not make the distinctions. Unfortunately, as we now show, the *fx-83W* forces peculiar distinctions on us, and makes our everyday competence with arithmetic breakdown.

For brevity, we shall only consider a small part of the *fx-83W*. First, a binary operator at the start of a calculation has a different meaning from one elsewhere. To account for this, $\langle calculation \rangle$ has to be defined in terms of $\langle expression \rangle$ and $\langle binary\text{-}operator \rangle \langle expression \rangle$. For simplicity, we will only consider $+$ as an example binary operator, and the relevant productions are then as follows:

$$
\begin{aligned}
\langle calculation \rangle &::= \langle expression \rangle = | + \langle expression \rangle = \\
&\quad | \langle expression \rangle \% | \langle expression \rangle \% + ; \\
\langle expression \rangle &::= \langle expression \rangle + \langle term \rangle | + \langle factor \rangle \\
&\quad | \cdots
\end{aligned}
$$

This is an abstract syntax: it does not define a unique phrase structure for a sequence of symbols. For example, $+2 =$ might be phrased as $+ \langle expression \rangle =$ or as $\langle expression \rangle =$, with the $\langle expression \rangle$ in this case being $+ \langle factor \rangle$. This ambiguity may confuse users, depending on what syntax they understand the calculator as using, but it does not "confuse" the calculator, since (in this case) it reliably takes the first phrasing. Thus, we use an abstract syntax merely to nominate syntactic categories, rather than to define an unambiguous phrase structure. Where an abstract syntax is used, some other means, apart from the productions, will be required to disambiguate phrase structure (alternatively, one may introduce subsidiary productions, as in Figure 4, which has varieties of expression (term, factor), to make the appropriate distinctions). For example, the *User's Guide* itself could be considered to define an abstract syntax (though instead of productions it gives concrete examples), which it disambiguates with a separate table of operator priorities. Curiously, the *User's Guide* does not disambiguate the example considered above, because it fails to consider $+$ as anything other than an infix operator—it does not explain how to phrase $+ 4 =$, or expressions like $- + 2!$.

Now an expression, possibly containing $+$, followed by % has a different meaning than one followed by $=$, thus the meaning of $+$ depends on the context. The calculator is not referentially transparent, so we need to introduce more than one semantic function. Let $\mathscr{E}[\![\cdot]\!]$ be the conventional meaning of a calculation. We start with the top-level semantic equations:

$$
\begin{aligned}
&e \in \langle expression \rangle \\
&\qquad \mathscr{E}[\![e =]\!] = \mathscr{E}[\![e]\!]; \\
&\qquad \mathscr{E}[\![+ e =]\!] = a + \mathscr{E}[\![e]\!];
\end{aligned}
$$

$$\mathscr{E}[\![\times e =]\!] = a \times \mathscr{E}[\![e]\!];$$

$$\mathscr{E}[\![\div e =]\!] = a \div \mathscr{E}[\![e]\!];$$

$$\vdots \quad \text{(other binary operator equations omitted)}$$

where a is the result of the last calculation

The remaining equations for $\mathscr{E}$ are defined, much as in Figure 6, directly, with the meanings of expressions being defined in terms of the meanings of their constituents, right down to digits. With the exception of the top-level equations (above) which really define the meaning of [=] for the calculator—and with the exception of certain idiosyncratic operators (e.g. ⌟, %, °′″)—all the other equations are trivial: for all operators (represented by $\oplus$, as respectively a binary, postfix or prefix operator) they are of the form

$$\mathscr{E}[\![a \oplus b]\!] = \mathscr{E}[\![a]\!] \oplus \mathscr{E}[\![b]\!],$$

$$\mathscr{E}[\![a \oplus]\!] = \mathscr{E}[\![a]\!] \oplus$$

$$\text{or } \mathscr{E}[\![\oplus b]\!] = \oplus \mathscr{E}[\![b]\!]$$

For example, if we consider ! (factorial), then, by the second line above, the meaning of an expression a ending in ! is $\mathscr{E}[\![a!]\!]$, and that is defined as $\mathscr{E}[\![a]\!]!$ namely the factorial of the value of a.

For each operator there is exactly one semantic equation. (Some symbols, like +, may have two equations since they can be used as prefix and binary operators.) We do not consider it here, but if we defined the entire semantics of the calculator, we would need to introduce further semantic functions† associated with each mode (the implication is that, as mode dependency makes the denotational semantics worse and more complex, mode dependency is a bad design idea).

The function $\mathscr{P}$ defines the semantics of expressions involving percentage. $\mathscr{P}$ is, however, completely different from $\mathscr{E}$. Unlike $\mathscr{E}$, which is general, $\mathscr{P}$ is only defined for a few special cases. It is defined at the level of direct calculations on numbers, rather than on expressions (Figure 7). Although the manufacturer defines most operators (such as +) in terms of numbers, because of referential transparency it is implicit that when the *User's Guide* say "43" that an expression (perhaps in brackets) might be substituted. This generality is not assured for %.

The equations in Figure 7 look like they could be defined in terms of $\mathscr{E}$ acting on *exactly the same* expression, rather than just the components of the expression. For example, we might expect that

$$\mathscr{P}[\![a + b\%]\!] \simeq \mathscr{E}[\![a + b]\!] \times \frac{100}{\mathscr{E}[\![b]\!]},$$

$$\mathscr{P}[\![a - b\%]\!] \simeq \mathscr{E}[\![a - b]\!] \times \frac{100}{\mathscr{E}[\![b]\!]},$$

† Alternatively, additional mode parameters could be used.

$$\langle calculation \rangle ::= \langle expression \rangle = | + \langle expression \rangle = \\ | \langle numeral \rangle + \langle numeral \rangle \% \,|\, \langle numeral \rangle - \langle numeral \rangle \% \\ | \langle numeral \rangle \times \langle numeral \rangle \% \,|\, \langle numeral \rangle \div \langle numeral \rangle \% \\ | \langle numeral \rangle \times \langle numeral \rangle \% + | \langle numeral \rangle \times \langle numeral \rangle \% - ;$$

$$a, b \in \langle numeral \rangle$$

$$\mathscr{P}[\![a + b\%]\!] = 100 \frac{\mathscr{E}[\![a]\!] + \mathscr{E}[\![b]\!]}{\mathscr{E}[\![b]\!]};$$

$$\mathscr{P}[\![a - b\%]\!] = 100 \frac{\mathscr{E}[\![a]\!] - \mathscr{E}[\![b]\!]}{\mathscr{E}[\![b]\!]};$$

$$\mathscr{P}[\![a \times b\%]\!] = \frac{\mathscr{E}[\![a]\!]\mathscr{E}[\![b]\!]}{100};$$

$$\mathscr{P}[\![a \div b\%]\!] = 100 \frac{\mathscr{E}[\![a]\!]}{\mathscr{E}[\![b]\!]};$$

$$\mathscr{P}[\![a \times b\% +]\!] = \mathscr{E}[\![a]\!] + \frac{\mathscr{E}[\![a]\!]\mathscr{E}[\![b]\!]}{100};$$

$$\mathscr{P}[\![a \times b\% -]\!] = \mathscr{E}[\![a]\!] - \frac{\mathscr{E}[\![a]\!]\mathscr{E}[\![b]\!]}{100}$$

FIGURE 7. Manufacturer-defined syntax and semantics for percent.

$$\mathscr{P}[\![a \times b\%]\!] \simeq \mathscr{E}[\![a \times b]\!] \times \frac{1}{100},$$

$$\mathscr{P}[\![a \div b\%]\!] \simeq \mathscr{E}[\![a \div b]\!] \times 100.$$

This superficial elegance is misleading. In ordinary arithmetic, $a + b + c = (a + b) + c = a + (b + c)$ because $+$ is associative, so we rightly expect that $\mathscr{E}[\![a + b + c]\!] = \mathscr{E}[\![(a + b) + c]\!] = \mathscr{E}[\![a + (b + c)]\!]$; or because $+$ is commutative, we rightly expect $\mathscr{E}[\![a + b]\!] = \mathscr{E}[\![b + a]\!]$.† But $+$ in a percentage equation is neither associative nor commutative. What this means is that the meaning of something as *simple* as $2 + 3 + 4\%$ is going to be a mystery—is it the same as $2 + 7\%$ or the same as $5 + 4\%$?—and even being familiar with the *User's Guide* won't justify any confidence. In other words, calculations involving percentages cannot be understood as ordinary, conventional calculations—operators take on new, restricted, meanings. The *User's Guide* does not define percentage expressions generally—indeed the whole approach to per cent (as an action like $=$) forbids any general semantics. As the examples show, we are not even in a position to work out what trivial percentage expressions mean—in contrast, we are not surprised that the *User's Guide* does not define the meaning of $a + b + c$ because it can *correctly* be understood as $(a + b) + c$. Furthermore, since $\mathscr{P}[\![a + b\%]\!] = 200 + \mathscr{P}[\![a - b\%]\!]$, in this context, %, $+$ and $-$ do not have their usual relation in any case (Figure 8).

† For floating point numbers, these equations are only an approximation because of potential significant figures losses.

$$\langle calculation \rangle ::= \ldots$$
$$| \langle numeral \rangle \lrcorner \langle numeral \rangle \% \,|\, \langle numeral \rangle\, x^y \langle numeral \rangle \%$$
$$| \langle numeral \rangle \sqrt[x]{y} \langle numeral \rangle \%;$$
$$a, b \in \langle numeral \rangle$$
$$\mathscr{P}[\![a\%]\!] = \mathscr{E}[\![a]\!];$$
$$\mathscr{P}[\![a \lrcorner b\%]\!] = \mathscr{E}[\![a \lrcorner b]\!];$$
$$\mathscr{P}[\![a\ x^y\ b\%]\!] = \mathscr{E}[\![a]\!]^{\mathscr{E}[\![b]\!]/100};$$
$$\mathscr{P}[\![a \sqrt[x]{y}\ b\%]\!] = \sqrt[\mathscr{E}[\![a]\!]]{\mathscr{E}[\![b]\!]/100}$$

FIGURE 8. Experimentally determined additional syntax and semantics for percent. Thus, with the exception of $\lrcorner$, $+$, $-$, a binary operation followed by % seems to divide the right operand by 100. Since % is a postfix operator, and the *User's Guide* defines the precedence of postfix operators as high, we might interpret % as, exceptions notwithstanding, merely dividing its closest operand by 100.

We know from experiments with the *fx-83W* that the permitted syntax for per cent *is* more general: the syntax appears to be $\langle expression \rangle\%$, but we do not know what the semantics are because they are not defined. Consider implicit multiplication: a simple experiment shows that in the cases where $\mathscr{E}[\![ab]\!] = \mathscr{E}[\![a]\!] \times \mathscr{E}[\![b]\!]$, we find that $\mathscr{P}[\![a \times b\%]\!] = 100 \times \mathscr{P}[\![a \times b\%]\!]$; in other words, implicit multiplication and percent do not work consistently.

Although we considered the denotational semantics of a *very* simplified *fx-83W* calculator, it still has numerous semantic equations needed for just $+$ as compared with the two that are needed for the conventional *and much more general* arithmetic syntax of Figure 4. As well as being more numerous, the right-hand sides of the semantic equations, unlike in conventional arithmetic, bare very little correspondence with the left-hand sides. In other words, the calculator forces the user to consider its symbols (such as $+$) to be quite other than the corresponding arithmetic operators. It creates a needless confusion.

In short, using per cent in any context other than trivial expressions involving only $+$, $-$, $\times$, $\div$ is undefined and hard to understand. Even in these cases, the meaning has to be learnt specially for the calculator. By treating [%] differently, not displaying it in an expression, treating it in many arbitrary ways, leaving many contexts of its use undefined, the user is discouraged from thinking of it as a mathematical operator.

There are two superficially plausible counter-arguments that may be raised in defence of the calculator's percentage design:

— "Users demand a per cent feature". It is more likely that users demand per cent if they are *just* asked whether they want it—it is just another potentially useful feature. But one imagines if users were asked whether they knew what they were asking for, or were asked which specific sort of percent they wanted, then the survey results would be very different. Postman (1992) makes similar critical comments on market surveys more generally.
— "Percent is done this way on all calculators". Section 3, below, defeats this position.

Appendix B discusses percentage in more detail.

2.3. FEATURE INTERACTION

In principle, the denotational semantics of everyday arithmetic notation could have been the *fx-83W* calculator's model: if the user pressed a sequence of keys *s*, then the two line display would show, on the top line the symbols *s*, and on the bottom line the value of *s*, namely $[\![s]\!]$. Indeed, for very simple calculations, this is how the *fx-83W* does work, with the proviso that the final = of *s* is not shown (and subject to certain numerical constraints, such as nine significant figures for the displayed answer).

The *fx-83W* however provides numerous features that interact, and hence break this clear approach. First, as described in the previous section, % has no obvious meaning when used in non-trivial calculations. Such confusion is not limited to %, but applies to some other operators, discussed at length in the appendices (e.g. fractions, implicit multiplications, sexagesimal). Second, if a calculation uses per cent or memory store, the display does not show the calculation. Third, if the user edits the calculation, the meaning changes.

The *fx-83W* provides several keys for editing a calculation. As the user presses (most) keys, they appear in the top line of the display. The user can delete keys, pressing [DEL] to delete one at a time, or [AC] to delete everything. Two keys move a cursor (the symbol "under the cursor" alternates with an underscore symbol) left or right in the entered expression so corrections can be made elsewhere. Normally, pressing a key replaces the symbol where the cursor is in the top line of the display. The calculator can be put in insert mode, by pressing a shifted delete key: the calculator stays in insert mode until the cursor is explicitly moved or a new calculation started.

In principle, a calculator with editing features such as these could be described as a straight-forward generalization of the simple model suggested above: (i) the user presses keys that create and edit the top line of the display; (ii) the top line of the display shows the edited calculation *s*; (iii) when the calculation is completed the bottom line displays its mathematical value, namely $[\![s]\!]$.

Unfortunately, on the *fx-83W*, there is feature interaction between the editor and the calculator.

Pressing [AC] [2] [=] [+] [1] [=] calculates 2 then adds 1 to it, finally displaying 3: that is, the [+] is taken to add the second calculation to the first. However, pressing [AC] [2] [=] [5] [DEL] [+] [1] [=], which has an accidental [5] that is immediately deleted, finally displays 1, not the expected 3. Thus the meaning of [+] depends, not on its position in a calculation, but when it was pressed. To be recognised as meaning "add to last answer" [+] must be pressed first; putting it first in a calculation by means of editing has a different meaning.

Thus editing interacts with the meaning of calculations. The meaning of the "first" symbol in a calculation is not whether it is the first key on the edited calculation, but whether it is the first key pressed.

Moreover, as Figure 7 makes clear, even this restriction is not sufficient to understand the first symbol—since the previous calculation may have used [%] in a certain way, in which case the calculator takes [+] to immediately add *part* of the previous calculation. This is another of the many percent feature interactions.

Further examples of feature interaction are raised in Section A.2.

3. Not just the *fx-83W*

There is a danger that this article is taken as a review of a specific calculator and as a critique of Casio. This section looks at calculator design more generally. Together with Appendices D and E it shows: similar looking calculators from the same manufacturer are different; elementary calculators are difficult to use; calculators from other manufacturers are broadly similar in their design failures; and that more advanced calculators (which exploit presumably better technology) do not overcome design problems. One concludes the industry has widespread design problems, and that technological limitations are not the reason.

(1) *Similar-looking calculators from the same manufacturer are different*. The UK Mathematics Coordination Group's revision guide (Parsons, 1978) says in its section on calculator buttons, "These instructions are mostly for nice simple Casio calculators". They give nine suggestions (pp. 9–11) and the instructions in seven of them do not work with the Casio *fx-83W*. For example, to calculate $64^{2/3}$ it is suggested that [6] [4] [x^y] [2] [⌟] [3] [=] is pressed. On Casio's *fx-115s* this gives 16 (as the book expects),† but on the *fx-83W* it gives `1365⌟1⌟3` (i.e., $(64^2)/3$ rather than $64^{(2/3)}$).

Casio's own MC100 says that 10 + 20%, keyed as [1] [0] [+] [2] [0] [%], is 10.204081, while the very similar-looking Casio SL300LC says 10 + 20%, keyed identically, is `12`. The *fx-83W* gives `150` for the same sum keyed identically.

(2) *Simple calculators are also difficult to use*. Many simple calculators (such as the Casio MC100) have a memory, and buttons: [MRC] (to recall the memory; to clear it if pressed twice); [M+] (to add to it); and [M−] (to subtract from it). There are no other memory keys. Given that the designers evidently think a memory useful, here is a problem: if you have just calculated 124.8624, which the calculator currently displays, and you wish to store it in the memory, what do you do? (Don't press [M+] because that adds to the memory; don't press [MRC] because that loses the number you want to remember.) There is a solution taking five key presses (plus one more to restore the display to its initial condition)—but it does not work if the memory already has a number in it (like 99999999) that could cause overflow.

(3) *Other manufacturer's calculators are similar*. The Sharp EL-546L is comparable to the Casio *fx-83W*: it has a two-line display, it is the same size, the same style grey plastic case and cover. Some of the Sharp calculator's design problems are listed in Appendix D: they are very similar to the Casio calculator's design problems (see also Thimbleby (1996*a*, *c*, 1997), which also review Canon, Hewlett-Packard, Texas Instruments and other examples).

(4) *More advanced calculators have similar design problems*. It is possible that the problems of the *fx-83W* arise from technological limitations. As an excuse for bad design this is implausible, as we now discuss.

† Unless the calculator does symbolic arithmetic, the answer cannot be exactly 16 for this sum, but if the answer is displayed to the precision of the calculation, the answer will *appear* exact. Most calculators calculate to a higher precision than their display.

The Casio *fx-570W* is an up-market and more sophisticated calculator, and therefore freed of the technical limitations that may have beset the *fx-83W*. Interestingly, it is sold in a box over 1 cm thicker (3.2 cm compared to 2 cm) than the *fx-83W* even though it is the same size: so packed (with air) to *feel* like a more substantial calculator! It is clearly a more sophisticated calculator: the *fx-570W* is broadly similar to the *fx-83W*, with many more features—359 functions (compared to 159), including complex numbers, numerical integration, calculations in binary and other bases, metric conversion, scientific constants, formula memories, … and so on. It has a faster processor; it seems to calculate 69! immediately, whereas the *fx-83W* noticeable takes over a second. Yet it has similar design problems. The technical developments, more advanced features, and so forth, have not overcome the flawed approach to design.

The *fx-570W* improves on the *fx-83W*'s limited approach to conversion from decimals to fractions. Thus, if the *fx-570W* is showing 2.5 *however it has been calculated* then pressing [a b/c] makes it display `2⌟1⌟5`, converting the number displayed to fractional form. The *fx-83W* cannot do this—in general, the key [a b/c] does nothing under these circumstances. Another change is that initial binary operators now work slightly differently: when a calculation is started with, say, [+] the *fx-83W* would display the last answer numerically, followed by +; the *fx-570W* instead displays `Ans +`. The advantage of this is that the top line of the display is less cluttered with long numbers (possibly in the fraction notation, etc)—the bottom line still shows the last result in full.†

That such a sophisticated calculator can be manufactured shows that technology *per se* is not the barrier to good design. Yet the *fx-570W* (where it is comparable) retains most of the same design problems of the simpler *fx-83W*, and it has numerous additional design problems (see Appendix E).

Overall, the *fx-570W* appears like the *fx-83W* but with the addition of more *independent* calculators—such as complex mode, engineering mode, integration, formula memory. Each mode has limitations, but the modes cannot be used together: simple examples are that a binary number cannot be entered in normal mode, nor can a number using μ as a multiplier be used in statistics mode, and so on. That features clearly accidentally overlap, as in the dual case of `F` to mean the Faraday constant or the memory F—so a display of `F-F` is ambiguous—further convey the extent of the feature interaction problem: independent design damages usability.

4. Knowing, seeing, understanding and mathematics

The casio *fx-83W* has a wide range of features, and an expert in the calculator (for example, a well-trained school child) would know what each function does. But would they understand it? Parts of this article have been hard to understand (and they have taken some experimentation on my part before I was confident writing them). One of the problems is that articulating poor features of a design is not easy, so the necessity of understanding is often overlooked.

† When the *fx-580W* is switched on, it shows 0. Pressing a binary operator does not automatically insert `Ans`—however, if [0] [=] is pressed first, the display will be the same, and `Ans` *does* get inserted. This is an initialization bug.

It is possible to demonstrate the calculator, indeed any artifact, and to impress by its power and sophistication (Thimbleby, 1996*b*). Thus, in a shop, one can *see* that the calculator can do impressive things—some artifacts have demonstration buttons to make them easier to demonstrate (presumably because shop assistants do not know how to work them otherwise). However, regardless of how persuasive a demonstration is, seeing is not understanding.

The calculator appears to have been designed as an arbitrary collection of features that have no straight-forward explanation; the features are hard to relate to mathematics as conventionally understood. Fractions, percentages, negation and other operations are idiosyncratic. It is certainly possible to learn to know how to use the calculator for simple operations, such as one might encounter in school exercises, but this is different from understanding it. In my view, the *fx-83W* shows so many signs of *ad hoc* and ill-thought out design that in would be futile to attempt any systematic understanding of it. That even the manufacturer's own documentation writers failed to explain the calculator satisfactorily (see the Appendices), I believe, substantiates my view that the calculator is a mess.

Throughout this article (including the appendices) are suggestions and critiques that could be used to lead to a simpler design that would achieve the same calculational power. Such a calculator would have a shorter complete and correct manual (the *fx-83W* has a long, incomplete and incorrect manual).

If such a properly designed calculator is viewed as an artifact in the world, and its manual as a theory, then developing for a shorter manual would be an application of Occam's Razor—not in the choice of explanatory theory, but in the design. Occam's Razor has a pedigree in the philosophy of science, but here we are concerned with design and with human understanding of artifacts. It is conjectured that the job of the brain is to construct compact codifications of knowledge (Ballard, 1997); if so, and unsurprisingly, an artifact designed to have simpler documentation would be easier to understand.

In the real world, we form theories to explain and understand what reality is. Most of reality is the subject of study for scientists; but a growing area of study for everyone is made up of artificial objects manufactured by others. These artefacts present challenges to be understood [26, chapter 9]. In contrast to natural objects, artefacts are designed by their makers to be easier or, as may be, harder for people to understand.

In the real world, we write $1 + 2/3$ or perhaps $1\frac{2}{3}$ (which is the style the *User's Guide* itself uses). On the *fx-83W*, we press [1] [a b/c] [2] [a b/c] [3], it is displayed as `1⌟2⌟3.`, and it does not always behave like a fraction (e.g., Section A.2.1). Such variation between mathematical tradition and the calculator's behaviour is gratuitous, and also serves to make the documentation longer, less reliable, and harder to understand than it need have been. The manufacturers are oblivious to this: they say "equations can now be entered as they are written" (see Appendix C).

Mathematics has had a special place in helping form our explanations of reality. Artifacts that claim to represent mathematics are therefore of particular interest. In this article, we examined a state-of-the-art calculator as one such mathematical artifact. The calculator is sophisticated, and was designed and built by skilled people. What is also clear is that calculators, as presently designed, are not easy to understand. They exhibit no clear, uniform design structure and do not admit clear ways for the user to understand them, neither explicitly (in provided documentation) nor implicitly (in ways one infers they are designed). Further, there is evidence that trying to form a rational understanding would be pointless: the designs are manifestly *ad hoc*. The manuals are inadequate; inadequate explanation is no explanation.

Drawing on the analogy of a "design science of artifacts" (Simon, 1996; Thimbleby, 1990)—the approach to calculator design is reminiscent of Feynman's discussion of cargo cult science (Feynman, 1992). The cargo cult imitated the outward forms that made aeroplanes land: they made runaways, lit fires and waited for aeroplanes to land. They did everything right, but it did not work. Feynman defined cargo cult science as following all the apparent precepts and forms of scientific investigaton, but missing something essential. What does cargo cult science miss? "[...] to try to give *all* of the information to help others to judge the value of your contribution [...]" This requires scientific integrity of a high order, as Feynman explains. In calculators, it appears we have gadgets that mimic the outward form of what "real" calculators would be like, yet they do not work. Where they fail is that neither they nor their manufacturers explain them; they are not designed to be explained, they are not designed to be explainable. They do not provide the information necessary to judge them.

5. Recommendations

At root, design problems stem from poor technical control of the design. Documentation appears to be an after-thought. Features are added independently, and without any over-arching theory or coherent policy. Does it matter? The confusion for users may or may not be significant, but there are reasons why it does matter: first, the *User's Guide* is confused, in itself suggesting that the design process is bad; second, a better design would be easier to manufacture and ensure quality control (it would also be easier to write a correct manual for); finally, a more uniform approach would provide usability benefits.

5.1. TECHNICAL RECOMMENDATIONS

Priority technical recommendations are:

1. Adequate computer science should be used to implement artifacts. (The problems of feature interaction are completely avoidable, almost mechanically if proper specification procedures are followed.) Standard tools (such as compiler–compilers) should be used (Aho *et al*., 1985; Johnson, 1975). (Compiler–compilers typically work in a way that is directly compatible with a specification based on denotational semantics). As Marvin Minsky put it in his 1969 ACM Turning Award Lecture (Minsky, 1987). "The computer scientist thus has a responsibility to education".
2. The lexical, syntactic and semantic aspects of the design should be done properly:
   (a) *Lexical*: The user presses keys; keys and key sequences have to be mapped onto the vocabularly of functions the calculator provides. If there are as many functions as keys this is trivial. Typically there are more functions than keys, and key sequences have to be used to denote functions. Most calculators use a "shift" approach: each shift key introduces another meaning for almost all other keys (e.g., the `OFF` key may not have a shifted meaning, and `SHIFT` keys are rarely shifted). Sometimes (as in the *fx-83W*) the lexical shifts for keys are

confused for semantic modes, resulting in feature interaction, as well as problems that may arise when the user thinks the calculator is in one mode when it is in another. (See footnote in Section 2 for an alternative lexical approach: using a standard mathematical notation, spelling-out functions.) Thus, *the lexical structure of calculators must be designed properly*.

(b) *Syntactic*: Once the user has entered functions (typically single key presses), their structure depends on syntax. Thus in $2+3\times 4$, multiplication takes precedence over addition, and the structure is $2+(3\times 4)$. The ambiguous example of calculating $63^{(2/3)}$ in Section 3 by [6] [4] [x^y] [2] [⌟] [3] [=] proves that the syntax of calculators should be defined. Few user manuals describe syntax, instead defining functions in isolation, again suggestive that manufacturers do not use a syntactically well-defined design. Thus, *the syntactic structure of calculators must be designed properly*.

(c) *Semantic*: Once syntax has established a structure, semantics gives it meaning. For example, on the *fx570-W*, [√] [−] [1] [=] (i.e., trying to calculate $\sqrt{-1}$) is accepted as syntactically correct by the calculator (rather than being trapped as a syntax error because an expression to subtract the 1 from is missing): but the calculator attaches no meaning to it, even though in conventional mathematical semantics, $i=\sqrt{-1}$. Yet the calculator does have a key for i and it 'knows' that $i^2=-1$, since [i] [x^2] [=] works correctly. Thus the semantics are inconsistent internally, and inconsistent with convention. Thus, *the semantic structure of calculators must be designed properly*.

3. Using appropriate design tools would enable full documentation to be generated, at least partly automatically by appropriate tools.

These are all high-level recommendations. More detailed ones—like how to avoid feature interaction—are pointless, since such symptoms of bad design would be avoided by using a more mature design process, as recommended above. Nevertheless, the following few points summarise some of the major oversights:

1. *Understand the domain, and relate to it*. Calculators are supposed to do mathematics, which (at least at the level considered here) is easily understood by anyone able to build a calculator.
2. *Understand users, and relate to them*. There is a whole discipline of developing user-centred systems, and designers should be familiar with it (e.g. Nielsen, 1993). An important way to relate to users is to explain the design in their terms—writing users manuals is therefore an important part of design. Manuals should not be written at the "end" of a design process, they are an active part of design and thinking through the design in a way that relates more closely to user.
3. *Detect and handle errors appropriately*. "Every use of a function should be clearly defined or its use detected as an error" is a design precept that can in principle only apply to certain well-behaved functions. However, in this sense, all functions on the calculator are mathematically well-defined.† This design rule has not been applied to %.

†For example log -1 is detected as an error. Some inverse functions like arcsin are defined specially, but there is still no problem.

4. *Avoid feature interaction.* In general, detecting feature interaction is a hard problem, but the calculator does not have many features, and there are standard ways of specifying what it should do (e.g. denotational semantics) that avoid feature interaction. However the calculator looks like it was intended to be a collection of separate and quasi-independent features. Feature interaction and its converse, feature incoherence, are discussed at length in Sections A.2 and A.3.
5. *Fit to task.* Identify relevant features of the task domain (here, mathematics) and enforce these in the design. As an example of this design heuristic in use: since mathematics is declarative (i.e., $2 + 3 = 5$ is true always, not just when [=] is pressed), why doesn't the calculator's display change so that it is *always* true? Clearly the designers have this idea in mind sometimes: if the display is edited, the then-obsolete answer disappears—since it would be incorrect; on the other hand it is *never* correct when [%] has been used. In general, having a *continuously* true display would make the calculator much easier to understand. Why have exceptions to truth? In fact, when the display is made *invariantly* correct, rather than only correct when [=] is pressed, new forms of interaction become possible (Thimbleby, 1996*a*, 1997), and (as shown in those references) a calculator becomes more flexible and much easier to use.

Having produced proper documentation, the designers would themselves be in a position to understand the design. Since automatic procedures (e.g. compiler–compilers) can check and regenerate new designs rapidly, it would be possible to undertake iterative design on a realistic time scale.

Once designers know what they are doing, standard usability engineering practice can be applied. Unfortunately, it will not be comfortable for manufacturers to realize that they currently design badly, nor will they want customers to know this. It will be difficult to change the entrenched value network (Christensen, 1997) of current design practice.

5.2. NON-TECHNICAL RECOMMENDATIONS

The design of calculators is not just a technical problem for which there are technical solutions. Manufacturers wish to make a profit and stay in business; usability or good design are not in themselves goals. It is appropriate, then, to provide a few wider recommendations.

- *School recommendations.* Schools have to make the best of poor resourcing, and they have to balance between preparing their students for the world as it is, and educating them to be able to change the world for the better. Modern technology must be questioned, and children must be taught to have a critical attitude. However, to teach this will be to go against almost everything modern western culture stands for. Fortunately, calculators themselves are at the centre of educational controversies—the debate should now distinguish clearly between calculators as aids and calculators as obfuscated objects of desire.
- *Consumer recommendations.* Consumers buy calculators for a variety of reasons, from school recommendations, and status through to being better able to solve problems. With the exception of status derived from owning a complex gadget, all uses would be enhanced by appropriateness, usability and clarity.

There are three reasons for a school recommending the *fx-83W*: the recommended model has a two-line display that shows the calculation and answer; all calculators are complex; and all calculators are different. The first reason does not make the *fx-83W* a unique choice (cf. Appendix D). The other two reasons, however, ensure it is easier for teachers if they require a particular calculator for their classes and examinations. Evidently marketing considerations encourage obscurity, complexity and variety. Therefore, consumers should demand usability metrics, so that they can tell easily which products are more or less usable.

This article has repeatedly made the case for good user manuals. Consumers should not buy products without good manuals, and they should not buy from stores that do not have the manuals available for inspection.

- *Professional recommendations.* Almost all of the problems identified in this article could have been avoided by making appropriate use of computer science. The reason why the calculator is bad is because it is relatively easy to make things work (hence they have lots of features), whereas far greater professional skills are required to make things work well (with, for example, no unfortunate feature interactions). Professional, properly qualified engineers should take a greater role in design; conversely, manufacturers should employ more professionally qualified engineers to work in design.
- *University recommendations.* There is a field of research, human–computer interaction (HCI), which has been established for at least a decade. The purpose of the field is to improve the design of systems to make them better, generally easier to use and more appropriate for the tasks for which they are designed. Students graduate with skills in HCI. Somehow these graduates are not ending up in the right places with the influence to put what they have learnt into effective use, or perhaps they are not learning the right skills to improve design practice. The goal has to be to find ways of teaching design so that people with the knowledge and ideas can see how apply their ideas to reality and see how to make a real impact.
- *Research recommendations.* This article has made a variety of plausible but untested claims about the usability of calculators. Although there are many minor claims, such as the significance of particular feature interactions on usability, the central untested claim is that a calculator should behave in a mathematical way if it is to be easy or reliable to use to perform mathematical tasks. This claim was taken as self-evident, but it should be tested (for example, an alternative design (Thimbleby, 1990, 1996*a*, 1997), referenced elsewhere in this article, has not been empirically compared with conventional calculators). If corroborated, then empirical weight is added to the abstract arguments used here: empirical evidence would have more persuasive impact on commercial considerations, such as questions of design strategies for market penetration (and even reduced litigation). If refuted, the results would raise interesting doubts about numerous assumptions in user interface design practice: a system that did not implement its task domain consistently performing better than one that did would undermine the concept of task fit and the relevance or scope of task analysis. Corroboration and refutation are extremes; a more likely result is useful insight into design trade-offs for calculators.

6. Conclusions

Calculators have been manufactured for many years, the correct way to parse arithmetic is widely know, and user-interface design guidelines are widely known. There is no excuse to impose such badly designed products on anyone, let alone school children.

The variation in design between calculators may be caused by marketing. Despite the conventions of mathematical notation, competition between manufacturers encourages slight variations in feature provision. Variation for a single manufacturer, such as Casio, may be caused by market positioning (for example, calculators for finance, for students, for children). However, the variation in just one manufacturer's features—as evidenced by Casio's three methods of percentage (Section 3)—seems hard to explain, unless one is extremely cynical. It is plausible that variation is deliberate so that schools and others are effectively forced to standardise on particular brands and models. Thus the manufacturer who has the greatest market penetration gets to sell more calculators, even if none of them are much good. This would make marketing sense if it is easier to achieve market dominance than product quality.

Even so it is very hard for us to understand how the errors in the *User's Guide* and the widespread feature interactions arise or have been allowed to remain over a period of years. These are unambiguous symptoms of bad quality control and bad design—or we must conclude usability and appropriateness for arithmetic are *not* a concern of the manufacturers. If the calculator was a new product, with no manufacturing experience behind it then maybe we would excuse its design. But it is an established product, with all the relevant technical and human factors know-how well known, and—like all other calculators—it is positioned in the market *as* a device for helping with mathematics. This it fails to do.

It is ironic that the terrible and varied design of products forces schools to recommend a particular one, and this, by benefiting the market leader in the field, serves to further conceal the problems. Perhaps many parents are now taking the school's apparent endorsement of one manufacturer's products as a recognition of quality? "The product sells and schools recommend it; so it must be all right". Next the consumers—on finding the product is obfuscated and mysterious—blame themselves for the problems it has. For, surely, if it is recommended for children it must be easy to use? Assuming it is easy to use, yet knowing we have problems, it would be easy, but mistaken, to conclude that we are incompetent; or, more comfortably, we would change our usage of the calculator to avoid exposure of 'our' problems. For example, surprisingly few adults use the percent key on any calculator. Thus, consumers do not recognise bad design. There is then no market pressure to improve products, which is sad when improvement would be so easy if only it was demanded.

But in such a well-defined area, why should the market have to demand something that could so easily have been done properly?

All this has been going on for so long that it would be hard to accept any excuse that the apparently so successful but misleading marketing is accidental. So where are the professionals with the integrity to start making gadgets easier to use? Where are the manufacturers who are prepared to make the world a better place?

Jemima Thimbleby very generously lent her Casio *fx-83W* calculator to help check this paper. The following have very helpfully commented on versions of this paper: Ann Blandford, Tim Bell, George Buchanan, Tony Durham, Michael Harrison, Matt Jones, Prue Thimbleby, Peter Thomas, Richard Young.

An earlier version of this paper was sent to Casio, and their reply is included in Appendix C. I am grateful to the editor and typesetters for the excellent work put into publishing this paper.

References

AHO, A. V., SETHI, R. & ULLMAN, J. D. (1985). *Compilers: Principles, Techniques and Tools*. Reading, MA: Addison-Wesley.

ALLISON, L. (1986). *A Practical Introduction to Denotational Semantics*. Cambridge Computer Science Texts, Vol. 23. Cambridge: Cambridge University Press.

BACKUS, J. W., BAUER, F. L., GREEN, J., KATZ, C. MCCARTHY, J., NAUR, P., PERLIS, A. J., RUTISHAUSER, H., SAMELSON, K., VAUQUOIS, B., WEGSTEIN, J. H., VAN WIJNGAARDEN, A. & WOODGER, M. (1996). Modified report on the algorithmic language ALGOL 60. *Computer Journal*, **19,** 364–379.

BALLARD, D. H. (1997). *An Introduction to Natural Computation*. Cambridge, MA: MIT Press.

CAJORI, F. (1993). *A History of Mathematical Notations*. New York: Dover Publications Inc. (originally published by the Open Court Publishing Company, 1928 & 1929).

CASIO (undated). *fx-350W, fx-270W, fx-83W, fx-82W User's Guide*, E SA9802-A HA 310494-9, Casio Computer Co., Ltd., Tokyo.

CARROLL, J. M. (1998). *Minimalism: Beyond the Nurnberg Funnel*. Cambridge, MA: MIT Press.

CARROLL, J. M. & CARRITHERS, C. (1984). Training wheels in a user interface. *Communications of the ACM*, **27,** 800–806.

CHRISTENSEN, C. M. (1997). *The Innovator's Dilemma*. Harvard Business School Press.

DEUTSCH, D. (1997). *The Fabric of Reality*. Harmondsworth: Penguin.

DOERFLER, R. W. (1992). *Dead Reckoning: Calculating without Instruments*. Honston: Gulf Publishing.

FEYNMAN, R. P. (1992) *Surely You're Joking Mr. Feynman*. Vintage.

GULLBERG, J. (1997). *Mathematics from the Birth of Numbers*. W. W. Norton & Co.

HALASZ, F. G. & MORAN, T. P. (1983). Mental models and problems solving in using a calculator. *Human Factors in Computing Systems, Proceedings CHI'83*, pp. 212–216. New York: ACM Press.

HARDY, G. H. & WRIGHT, E. M. (1979). *An Introduction to the Theory of Numbers* (5th edn). Oxford: Oxford Scientific Publications.

JOHNSON, S. C. (1975). *Yacc—yet another compiler compiler*. Computing Science Technical Report 32, AT&T Bell Laboratories, NJ.

LEVESON, N. G. (1995). *Safeware: System Safety and Computers*. Reading MA: Addison-Wesley.

VAN DER MEIJ, H. & CARROLL, J. M. (1998). Principles and Heuristics for designing minimalist instruction. in J. M. CARROLL, Ed. *Minimalism: Beyond the Nurnberg Funnel*. Cambridge, MA: MIT Press.

MINSKY, M. (1987). Form and content in computer science *ACM Turing Award Lectures*, pp. 219–242. New York: ACM.

NIELSEN, J. (1993). *Usability Engineering*, New York: Academic Press.

PARSONS, R. (1998). *Revision Guide for G.C.S.E. Mathematics*, (3rd edn.). The Mathematics Coordination Group.

POSTMAN, N. (1992).*Technopoly*. Vintage.

SIMON, H. A. (1996). *The Sciences of the Artificial*. (3rd edn), Cambridge, MA: MIT Press.

TENNENT, R. D. (1981). *Principles of Programming Languages*, Englewood Cliffs, NJ: Prentice-Hall.

THIMBLEBY, H. W. (1983). Guidelines for 'Manipulative' Editing. *Behaviour and Information Technology*, **2,** 127–161.

THIMBLEBY, H. W. (1990). *User Interface Design*. Reading MA: Addison-Wesley.

THIMBLEBY, H. W. (1993). Computer literacy and usability standards? In C. D. EVANS, B. L. MEEK & R. S. WALKER, general Eds. *User Needs in Information Technology Standards*, pp. 223–230. London: Butterworth-Heinemann.

THIMBLEBY, H. W. (1996). A new calculator and why it is necessary. *Computer Journal*, **38,** 418–433.

THIMBLEBY, H. W. (1996) Internet, discourse and interaction potential. In L. K. YOUNG, L. HERMAN, Y. K. LEUNG & J. MOYES, *First Asia Pacific Conference on Human Computer Interaction*, 3–18.

THIMBLEBY, H. W. (1996). Review of Texas Instruments T1-92. *Times Higher Education Supplement Multimedia*, 25, pvi, June 14.

THIMBLEBY, H. W. (1997). A true calculator. *Engineering Science and Education Journal*, **6,** 128–136.

THIMBLEBY, H. W. (1999). Smart to be simple. *The Times Higher Education Supplement*, 15, February 12.

YOUNG, R. M. (1981). The machine inside the machine: user's models of pocket calculators. *International Journal of Man-Machine Studies*, **15,** 51–85.

YOUNG, R. M. (1983). Surrogates and mappings: two kinds of conceptual models for interactive devices. In D. GENTNER & A. L. STEVENS, Eds. *Mental Models*, pp. 35–52. Hillsdale, NJ: Lawrence Erlbaum Associates.

Appendix A: Continued critique of the *fx-83W*

Note. Features of the *fx-83W* are classified for convenience under headings like "inadequate documentation" and "feature interaction". There is some overlap between the headings, but each feature is only discussed under one heading. This appendix does not duplicate substantive points raised in the body of the article.

A.1. INADEQUATE DOCUMENTATION

There is a wide range of help material that can be provided with a device. The *fx-83W* calculator includes: the disposable box; the *User's Guide* (Casio, undated), the Quick Reference sheet on the hard cover; text on the calculator itself; and plus display diagnostics. Although minimalism—an approach to documentation—includes "be brief; don't spell out everything" as a principle (van der Meij & Carroll, 1998), the appearance of the manufacturer's documentation suggests that it was not *trying* to be minimal.

A.1.1. Disposable box

The calculator is sold in a cardboard box, that proclaims the *fx-83W-w* has 159 functions, 2-line big display, S-V.P.A.M. (which it defines as "super visually perfect algebraic method") There is a two-thirds size photograph of the calculator. Including the photograph, SVPAM appears 6 times (and once in French). *Nothing* elsewhere in the documentation explains this term.

A.1.2. User's guide

Given there is no other reference material, the *User's Guide* should be complete and accurate. For example, the internal rounding function is achieved by pressing the `Rnd` button—but what does it do? There is an obscure, unexplained example of its use in the *User's Guide*, but no explanation of what it does.

The *User's Guide* explains that there are two ways to clear the memory, *M*. One is to store zero in it (by doing `0` `STO` `M`) and the other is to do `Mcl`. In fact, `Mcl` clears all memories—so anyone doing what the *User's Guide* recommends would lose all the values in the other six memories.

The *User's Guide* is wrong when it says [0] [STO] [M] will clear the memory M. For example if [4] had just been pressed, following the instructions could leave M containing 40 (or anything else). In fact the correct way to clear the memory is to press [AC] [0] [STO] [M]. (Pressing [AC] [STO] [M] does nothing, and doesn't make anything appear on the display.)

The key [Mcl] itself is not defined in the *User's Guide*. It works in a quirky way (perhaps because it is recognised as a "dangerous" function). It can only be used if it is the entire calculation, as in [AC] [Mcl] [=]. Any other use is a syntax error.

Some functions are not illustrated (e.g., [ALPHA] [3] gets the value of *n*. but there is no *n* shown on the calculator as a reminder). The [Rnd] button is next to the [Rnd#] button, which generates random numbers, but [Rnd] has nothing to do with random numbers. There is, however, ample space to spell `Rnd` in full, as `Round`, so that there would be no confusion. Indeed, the *User's Guide* does not specify the distribution of random numbers (presumably it is rectangularly distributed over [0, 1), and presumably to three digits precision).

The "sexagesimal" button is mentioned in the *User's Guide* in a summary of functions, but nowhere is there an explanation of it. (It looks like [°′′′], meaning degrees, minutes and seconds, though that wouldn't be obvious to anyone who didn't know already—particularly since the symbols are "in line" rather than raised as they would be in normal use.) One of the two left arrow functions is explained as "Decimal ↔ sexagesimal" in a table, but in fact the button does not work both ways: it only converts decimal to sexagesimal, not both ways as the double arrow suggests.

The difference between the [(−)] and the [−] keys is not explained in the *User's Guide*.

The *User's Guide* lists six key meanings that are not shown on the key pad. For example, [ALPHA] [3] gives the number of samples in statistics mode. It displays as `n` when it is entered, but the keyboard does not show it. Given that these six meanings are only used in two modes of the calculator, one could anticipate that the user would find them harder to remember—so all the more reason why they should have been engraved on the keyboard (or the calculator designed quite differently to avoid the problem).

The *User's Guide* does explain that NORM is to reset the FIX and SCI modes, without changing the degree/radian/grade mode. Setting the calculator to NORM mode does not get it to (what a user might call) normal mode; it may still be in LR or S.D. but set to use degrees. In fact, the user may have to set it to NORM mode *twice* to have the desired effect, as there are two NORM modes.

For a mathematical sophisticate, the two NORM modes are essentially the same—they affect the display formatting under certain circumstances (see Figure 3). But for a school child an unanticipated answer in exponential notation would make the calculator's results hard to interpret correctly.

S.D. means "standard deviation", but in this mode the calculator does more general statistics—why isn't the mode called STAT? COMP probably means "compute" mode, but the *User's Guide* doesn't say. SCI (or `Sci` as it is displayed), the *User's Guide* says, means "significant figures", even though it is not an abbreviation for it (but, confusingly, it is an abbreviaton for "scientific").

A.1.3. Quick reference

The calculator has a protective cover that has a "Quick Reference" on its inside. This isn't as helpful as it could be. For example, the front of the calculator tells us that there are nine modes, 1: COMP, 2: S.D., 3: LR, … to 9: NORM, and the cover's Quick Reference hardly tells us any more by saying that MODE 1 is COMP, MODE 2 is S.D., MODE 3 is LR, etc. We would be still in the dark about what they are, especially S.D. and LR, without the *User's Guide*.

The Quick Reference cover provides very scanty information. It at least provides the `ALPHA` functions that the keyboard does not show, but some of it is wasted in "advertising" that would be familiar to a user—e.g., an example of calculating $\sin 30 + \cos 60$ using Perfect Algebraic Method (*sic*—not "super visually perfect algebraic method").

The Quick Reference cover does tell us that FIX and SCI require a following number, which the calculator itself does not say either. Neither of them tell us—as the *User's Guide* does—that there are two NORM modes, entered alternately. Indeed, the *User's Guide* calls them NORM 1 and NORM 2, and it warns "There is no indication on the display of which format is currently in effect, but you can determine the setting by performing the following calculation". This does not sound very obvious.

A.1.4. The display

The display, despite being big enough to give clear diagnostics, only provides three abbreviated messages. For example, the calculator can display the text "`Ma ERROR`" if you do `1` `÷` `0` `=` (obviously it means maths error); but what does "`Stk ERROR`" mean? `(stuck error?)` Why aren't these messages written out in full? They only take up a fraction of the screen space, so that wasn't the reason.

A.1.5. Key labels

The keyboard could be made clearer. For example, the keyboard has two yellow left arrows on it. Neither are described in the *User's Guide*; one of them divides the mantissa of so-called engineering notation by 1000, the other converts to sexagesimal (temporarily). There are two keys labelled `A`, and two labelled `B`, and the `A`'s and `B`'s mean different things; there is one key with five different meanings, but 18 with only one. (Most have two meanings.) This is a very uneven allocation—and as reported elsewhere in this article, some of the overloading of keys does cause problems. These problems would have been avoided by a more sensible function-to-key allocation.

Some keys have many functions, and some have only one. Since some modes make some functions (silently!) inaccessible, a rationalisation of function allocation is called for. The calculator has a wide range of modes, and it does not distinguish clearly between orthogonal modes and basic modes. For example, statistics and linear regression are mutually exclusive modes, but in either the calculator can be in degrees or radians, scientific or normal display, etc. Entering COMP mode seems to clear the calculator, whereas NORM can be entered repeatedly without clearing it, and NORM changes the display mode, alternating between what for some results are indistinguishable modes, but are what the *User's Guide* calls NORM 1 and NORM 2 modes.

Key	Labelled meaning SHIFT + key	Hidden meaning ALPHA + key	Labelled meaning in terms of hidden meaning
1	$\bar{x}$	$\sum x^2$	$\frac{\sum x}{n}$
2	$x\sigma_n$	$\sum x$	$\sqrt{\frac{\sum x^2 - \frac{1}{n}(\sum x)^2}{n}}$
3	$x\sigma_{n-1}$	n	$\sqrt{\frac{\sum x^2 - \frac{1}{n}(\sum x)^2}{n-1}}$
4	$\bar{y}$	$\sum y^2$	$\frac{\sum y}{n}$
5	$y\sigma_n$	$\sum y$	$\sqrt{\frac{\sum y^2 - \frac{1}{n}(\sum y)^2}{n}}$
6	$y\sigma_{n-1}$	xy	$\sqrt{\frac{\sum y^2 - \frac{1}{n}(\sum y)^2}{n-1}}$

Example: the mean of x (labelled $\bar{x}$) can be obtained by ALPHA 2 / ALPHA 3 =.

FIGURE A.1. How the labelled meanings of keys could be obtained from their hidden meanings using standard formulae, but the opposite is not the case. The implication is that if only one or the other set of functions can be labeled, then swapping the labels would be an improvement.

The key ; (in SD and LR modes) repeats the previous entry; thus 3 ; 2 DT is the same as 3 DT 3 DT . Why isn't the ; symbol something more mnemonic, like "Repeated" (or REPT)?

ALPHA 1 ... 6 have misleading key labels. The key 1 is labelled $\bar{x}$ (in the blue/yellow SD style for a shifted meaning) but it means $\sum x^2$ if ALPHA is pressed—a meaning which is not shown on the keyboard. The ALPHA meanings of the other five keys 2 to 6 (see Figure A1) are not labelled either. Since the meanings are sophisticated, and arbitrarily allocated to keys, the user would be unlikely to remember them. Although space around the keys is not a problem (compare with the M+ shown in Figure 1), it is arguable that if some meanings are not to be shown, the calculator has it the wrong way around: the meanings of the keys that are shown are easily derived from the hidden meanings. If it was done that way around, at least a user with an elementary knowledge of statistics could use the calculator's facilities (see Figure A.1); the way that it is done, there is no easy way to find out what keys mean without access to the *User's Guide*.

The side-effects of Rec(and Pol(on memories E and F are not indicated on the relevant keys. We discuss these functions further below.

A.1.6. Inconsistency between different documentation

The display and the keypad are two features, which should be consistent. However symbols are shown differently on the display and on the keypad. The key (−) is displayed as −, and a value shown in sexagesimal does not display the final ° symbol that is required to enter it as part of a calculation (and the key for sexagesimal

`1°2°0.5°`	(which is entered by [1] [°'''] [2] [°'''] [0] [·] [5] [°''']) correctly represents 1 degree, 2 minutes, and half a second
`1⌟2`	(which is entered by [1] [a b/c] [2]) correctly represents one half
`1°2°1⌟2°`	is a syntax error
`1°2°(1⌟2)°`	represents 1 degree, 2 min, all multiplied by one-half.

FIGURE A.2. Sexagesimal notation and fractions feature interaction.

uses primes, when the display only shows degrees). When a number is displayed in sexagesimal notation, it is displayed in a notation that the user cannot enter: e.g., 1.5 is displayed as `1°30°0`, but to enter this value it would have to be entered as [1] [°'''] [3] [0] [°'''] [0] [°'''] (with an extra degree symbol).

A.2. FEATURE INTERACTION

Features may sound good in isolation, but they may combine in surprising ways. The *fx-83W*'s fractions, implicit multiplication, unary minus, and sexagesimal notations can combine in peculiar ways that are open to misinterpretation by users.

A.2.1. Sexagesimal notation and fractions

The sexagesimal operator allows the input of degrees, minutes and seconds. The fraction operator allows the input of fractions. However, it is not possible to enter a number of degrees, minutes, seconds and fractions of seconds, as this results in a syntax error. And if you use brackets to try to avoid the syntax error, the calculator does a multiplication, and the supposed fractional seconds becomes a factor of the answer (Figure A2).

One help the calculator could have provided would have been to use the standard symbols for degrees, minutes and seconds. (The display resolution is good enough for this.) The last example above would at least have been displayed as `1°2′(1⌟2)°` instead of the expected `1°2′(1⌟2)″`, so the user might have been less surprised.

A.2.2. Fractions and decimals

Fractions and decimals should combine appropriately. For example, [1] [+] [1] [a b/c] [2] [+] [0] [·] [2] [5] gives `1.75`, whereas [1] [+] [1] [a b/c] [2] [+] [1] [a b/c] [4] gives `1⌟3⌟4`, which means 1 + 3/4. Thus exact fractions represented as decimals unnecessarily make the calculator change from fraction display mode to decimal mode (Figure 3). There is no need for a "fraction" mode at all, as there are standard algorithms for converting decimals to fractions (Doerfler, 1993, Hardy & Wright, 1979).

There is no reason to have a fraction notation, especially not an unconventional one using the ⌟ symbol. `1⌟3⌟4` is numerically the same as 1 + 3/4 and it can be entered in this way just as easily—and without having to learn anything new. Given that the display is a flexible 5×7 dot matrix display, fractions should have used a standard notation.

A.2.3. Implicit multiplication and memories

Implicit multiplication should work uniformly. But it doesn't. For example, πA (which is entered as [π] [RCL] [A]) and $A\pi$ (which is entered as [RCL] [A] [π]) give different results. You cannot multiply AB, but you can do $2A$ (but not $A2$). These problems arise because there is a feature interaction between implicit multiplication and a special-case "speed up" on recalling memories.

When [RCL] [A] is entered at the start of a calculation, the calculator immediately displays its value. (Almost as if a final [=] was added automatically only when [RCL] is the first thing in a calculation.) If a memory is used as the first value of a calculation it then only plays a part of the calculation if the next operator is a binary operator or a postfix operator—then the calculator's normal approach of combining with the previous result "consumes" the memory's value correctly. Thus [RCL] [A] [!] works out the factorial of A, because [RCL] [A] recalls the value of A and displays it, then the [!] starts a new calculation, but one that uses the previously displayed value. The end result is the same as [RCL] [A] [!] should have been. But the catch is that a calculation like A sin 30, which looks like it should be done by [RCL] [A] [sin] [3] [0] fails because the calculator does not use the previous value for an implicit multiplication. You'd have to remember to enter it as [RCL] [A] [×] [sin] [3] [0] to get the right result, or perhaps, tortuously, as [(] [RCL] [A] [)] [sin] [3] [0], to avoid the [RCL] being the first step of the calculation.

A.2.4. Modes and function overloading

In SD mode, the [M+] button takes on the meaning DT. Although [STO] [M] and [RCL] [M] still work correctly (which also use the [M+] button), the function [M−] ceases to work, instead becoming [CL]. It is understandable that the [=] key continues to mean [=] and that SD mode requires an additional key for data entry [DT], but the *User's Guide* does not warn that [M−] and [M+] won't work, and the calculator itself gives no warning when an attempt is made to use them. So [M+], for example, will do nothing—which is exactly what it normally looks as though it does, so the user will be surprised at the end of a sequence of add-to-M operations that M is still zero. Ironically [M+] is exactly the sort of function that someone might want to use when doing statistics (e.g., to calculate running totals)—and it has been made unavailable.

The clear key, [AC] has four meanings. It switches the calculator on; it clears the current calculation (completely, and with no undo possible; the [DEL] key should be used for deleting recent key presses); it clears memories (when shifted in all but SD mode); it clears the statistics memories (when shifted in SD mode). Thus, in SD mode it is not possible to clear normal memories.

A.2.5. Modes and replay

The calculator's "replay" function consists of two keys; if a calculation has been evaluated, then pressing one of these keys brings the calculation back so it can be edited and re-evaluated. Suppose a calculation is entered accidentally in S.D. mode (e.g. an attempt is made to use an operation that S.D. does not support). If the user now changes mode to NORM, the supposedly replayable calculation is lost. In other words, a *known* case, easily anticipated by the designers, where replay would be useful is not supported.

Pol(x, y) gives the radius, and assigns to memories:
E the radius, $\sqrt{x^2 + y^2}$
F the angle, $\tan^{-1}(y/x)$

Rec(r, θ) gives the x coordinate, and assigns to memories:
E the x coordinate, $r \cos \theta$
F the y coordinate, $r \sin \theta$

Function	Shortcut cost	Using no shortcuts
[Pol(] x [,] y [)]	result in 3; E, F, in 6	result in 4; E, F in 7
$\sqrt{x^2 + y^2}$	6	7
$\tan^{-1}(y/x)$	5	6
[Rec(] r [,] θ [)]	result in 4; E, F in 7	result in 5; E, F in 8
$r \cos \theta$, $r \sin \theta$	2	2

FIGURE A.3. Meanings and costs of polar/rectangular functions and equivalents. The costs include pressing the [SHIFT] key as required. The shortcut cost assumes [=] is used to provide closing brackets; the full cost includes pressing closing brackets as well (as would be required in the middle of a larger calculation). The costs of entering the parameters are not counted.

A.2.6. Degrees, radians grades

Although there is a key to enter degrees, the calculator can be in any of three modes: degrees, radians or grades. It is possible to enter a number apparently in degrees when the calculator is in radian mode; the results are bizarre: the calculator can display $\sin 30^\circ$ complete with the degree sign as having the answer $-0.988\ldots$ when in fact $\sin 30^\circ$ should be $+0.5$. The feature interaction is that the degree symbol only makes it look like a degree, but the calculator has worked out the sine of 30 radians.

If there were 100 grades in a full circle, they might be useful for pie charts, but unfortunately there are 400. They are close to degrees (360 in a circle) and lead to confusion. The *Shorter Oxford English Dictionary* says that grades are degrees, 90 per right angle, and were last used in 1593. It isn't obvious that a school calculator benefits from having them—except that many other calculators have them, so it may help for market position in the minds of those customers who like high feature counts … if so, we could suggest better features!

A.2.7. Modes and overloading

In LR mode (linear regression), an equation of the form $y = A + Bx$ is fitted to data. In LR mode, the calculator can be used to determine the A and B constants (and regression coefficient, and so on). The feature interaction is that this A and B are different from the "usual" memories A to F (and M). If the calculator is going to use A and B for the equation, why not use the regular memories A and B (for other functions side-effect the memories, so that is not the problem), or if there is a need to keep them separate, why not call the function $y = ax + b$ (or, better, $y = \alpha x + \beta$) so there is at least some notational difference?

A.2.8. Negate, minus and implicit multiplication

The calculator provides both negate [(−)] and minus [−] keys, even though by convention the real world uses the same signs for both operations. The calculator does not provide a change sign key [±] though it does provide a reciprocal key (which does for multiplication what change sign does for addition).

When a calculation starts with a binary operator (such as multiply) the previous result is retrieved. So typing [×] [3] [=] multiples the last result by three (it is a syntax error if the calculator has been cleared and there is no 'previous result'). It seems this is the only occasion where [(−)] and [−] are distinguished: a calculation starting with [−] and not following an [AC] will subtract from the last result, but starting it with [(−)] will not. Given that the calculator has a function [Ans] (which is evidently considered so important it has no other meaning in any mode) which would do this unambiguously and without exception, why have the unnecessary and broken "operate on last result" feature?

Unary minus [(−)] and minus [−] could be combined, removing an unnecessary function and a confusing button, as well as some spurious error messages. Already, the calculator copes with [−] in unary minus contexts. For example, [4] [−] [−] [3] and [4] [−] [(−)] [3] give the same results, as do [4] [×] [(−)] [5] and [4] [×] [−] [5], and [−] can be used (it seems) anywhere [(−)] can be. However, [4] [(−)] [−] [3] is a "Syn ERROR." My recommendation is to get rid of the [(−)] key. I cannot think of any purpose it serves that isn't better done by [−].

A.2.9. Interaction with memories

Although rectangular coordinates conventionally use variables (x, y), and polar coordinates conventionally use variables (r, θ), the calculator's coordinate conversion functions assign results to E and F. This risks losing some important numbers the user has saved in these memories. The interaction isn't necessary since there are keys to recover *other* special values, such as averages (when in statistics mode).

A.2.10. Shortcuts and key overloading

"Easier to use" can mean reducing key-press counts, but it should not do so at the expense of increasing memory load. For example, the key [M+] is an abbreviation for [=] [+] [RCL] [M] [STO] [M] and superficially seems easier to use. But the [M+] memory key has five different meanings, and therefore the "shortcut" interacts with other features. In particular, adding to the memory and subtracting from it differ in whether the [SHIFT] key has been pressed—it is therefore easy to make a mistake, and not know because the memory isn't shown. Removing the short cut keys would also make the M memory consistent with the other six memories which don't have equivalents to the [M+] and [M−] keys.

A.2.11. Proximity of unrelated keys

A special case of feature interaction is the likelihood that a user will invoke inappropriate features as a result of the design. Poor keyboard layout means that features interact because they are close to each other.

The [DEL] is adjacent to [AC], so an attempt to correct a calculation may lead to its complete loss. The functions [DEL] and [INS] are the *same* key, yet do completely different jobs.

A.2.12. Co-ordinate conversions and memory corruption

The keys `Pol(` and `Rec(` convert between rectangular and polar co-ordinates. For example, to convert the rectangular co-ordinates (1, 1) to polar, one would press `Pol(` `1` `.` `1` `)` `=` (possibly omitting the closing `)` since `=` provides necessary closing brackets). The result would be the radius, and the memories *E* and *F* would be modified, with *E* the radius and *F* the angle. There is therefore a feature interaction between co-ordinate conversion and memories.

To use co-ordinate conversion functions, the user has to know that *E* and *F* are affected (so that no stored values are lost by mistake) and must know which is which. The order of the parameters of `Pol(` and `Rec(` must also be known, though this follows convention so is easy to remember. The functions are unusual in being the only ones to provide their own opening brackets—perhaps this helps the user remember that a comma `,` is also required, again which no other functions require.

It is not obvious how to combine more than one use of `Rec(` and `Pol(` in a calculation, since the final values of *E* and *F* will not be defined: there is a feature interaction between co-ordinate conversions. For example, one might wish to know the *x* and *y* co-ordinates of adding two vectors together: `Rec(` … `)` `+` `Rec(` … `)` would give the *x* co-ordinate, but the *y* co-ordinate would not be known. To find that, the user would have to do the first `Rec(`, press `=` to evaluate it, then use `RCL` `F` `STO` `A` `RCL` `E` `+` `Rec(` … to get the *x* coordinate, then use `RCL` `A` `+` `RCL` `F` `=` to get the *y* co-ordinate. This is very complicated compared to the first-principles approach: $r_1 \cos\theta_1 + r_2 \cos\theta_2$ is the *x* co-ordinate, and $r_1 \sin\theta_1 + r_2 \sin\theta_2$ is the *y* co-ordinate. Both of these calculations can be entered directly, as in `2` `sin` `4` `5` `+` … The equations are the standard equations, and there is no *ad hoc* manipulation of memories *A*, *E*, *F* to anticipate or figure out.

According to the *User's Guide*, operations of the same priority are performed from right to left. Other operations are performed from left to right. The *User's Guide* gives an example of unary operations, but this example cannot clarify what the *User's Guide* means. It *seems* that $\texttt{Rec}(1,1) + 1 + \texttt{Pol}(1,1)$ should be calculated as $\texttt{Rec}(1,1) + (1 + \texttt{Pol}(1,1))$, yet the value left in the *F* register is 45 degrees, *whichever* form is entered, even $(\texttt{Rec}(1,1) + 1) + \texttt{Pol}(1,1)$. Experiments indicate that the right-hand expression is executed last in all cases, regardless of the *User's Guide's* rules.

In other words, providing "packaged" functions like the co-ordinate conversions has made doing anything other than the most trivial operations—*if* you can remember how to do those—far more complicated than not having the functions at all. Figure 11 shows, moreover, that working out three out of four of the intended values with these keys is harder in terms of number of key presses required than using the first-principles equations directly.

Finally, co-ordinate conversion has a feature interaction with the angle mode of the calculator. So there is one more argument for removing `Pol(` and `Rec(` functions: to remove three hidden dependencies on the calculator's angle (degree, radian, grade) mode.

A.3. FEATURE INCOHERENCE

Feature interaction arises when two or more features fail to work together; in contrast, feature incoherence is when two or more features are unrelated when they could have been rationalised.

A.3.1. Dual meanings for the fraction operator

The key [a b/c] has two uses: it is an operator to construct fractions and it is a display operator (Figure 3). Suppose the display shows `3⌟2`. Pressing [a b/c] under one interpretation should create a fraction starting with `3⌟2⌟`—in the usual way of operating on the last result (like + would); in the other case it should change to display either `1⌟1⌟2` or or `1.5`. Confusingly, it does both, depending on the mode.

A.3.2. Unary and binary minus

There is no "change sign button" with the meaning commonly given on calculators (e.g., a key [±] that changes the sign of the displayed number). Instead, there are both subtract, [−], and unary minus buttons, [(−)]. The unary minus button is unnecessary as everything it can do can be done by [−], which, moreover, is conventional mathematical notation.

A.3.3. Two ways to use memories

There are two ways to obtain the value in a named memory. You can use [RCL] [A] to get *A*'s value, or use [ALPHA] [A]. At the beginning of a calculation, [RCL] [A] immediately displays the value of *A*. Here, it is equivalent to [ALPHA] [A] [=]. This is feature incoherence, but in a calculation involving implicit multiplication, like *AB* (i.e., $A \times B$), the immediate action of [RCL] inhibits the implicit multiplication working—and this is feature interaction.

When [ALPHA] [A] [=] is entered, the calculator shows `A` on the top line, and the value of *A* on the bottom line. However, if [RCL] [A] is used, the calculator shows `A` = on the top line. This is the only context in which = can appear on the top line.

A.3.4. Powers of ten and [EXP]

There is a button [EXP] whose purpose is to multiply a preceding number by a power of ten. This is very similar to the function [10ˣ] whose purpose is to multiply a preceding expression by a power of 10. Given implicit multiplication, anywhere [EXP] can be used, [10ˣ] can be correctly substituted. It is also clearer, since the display symbol for [10ˣ] is conventional and displayed exactly, rather than the less well-known symbol `E` which is used to represent [EXP] in the display (and, confusingly, is not consistent with the key label—`EXP`).

Parsons (1998) discusses this key: "It would be a lot more helpful if the calculator manufacturers labelled it [×10ⁿ] because that's what you should call it as you press it [...] For example to enter 6×10^3 you must only press [6] [EXP] [3] and NOT, as a lot of people do: [6] [×] [1] [0] [EXP] [3]". Parson's advice would be unnecessary if, simply, the [EXP] key was removed—then the [10ˣ], already provided, would be used and would work as people expect.

A.3.5. Insert and delete modes

Editing could be much simplified, thereby saving at least one function and removing a mode. The display shows the calculation as it is entered, and a cursor can be moved left and right, and keys deleted or replaced. The calculator allows expressions to be edited:

this is a good thing, but it does it in a way that requires additional modes. Editing overloads the DEL key (with a function that is not used to delete!). It has been known for a long time that both modes are unnecessary (Thimbleby, 1983); modes just introduce opportunities for mode errors—exacerbated on the *fx-83W* since the mode change operation is a moded use of the delete key.

The delete key works in *two* ways. At the right hand end of a calculation, DEL deletes characters, working left; inside a calculation, DEL deletes characters, working right. More precisely, when inside a calculation, the character at the cursor position is deleted—and characters to the right move left; when at the end of a calculation, the cursor is over a blank, beyond the right of the calculation, and moves left before the deletion occurs.

In conventional word processors: there is only an insert mode and deletion always deletes the character to the left of the cursor. This avoids modes, and makes deletion work consistently.

A.3.6. Functions work differently in the display

The two-line display is often misleading. For example, the way the % key works ensures it will display a "wrong" result (such as `4` × `5` on the top line, and `0.2`, not `20`, on the bottom line). The MODE key does not show in the display *even though* it takes one or two parameters—the parameters do not show, and thus there is no prompt for the user that any are required. To be consistent (e.g., with Rec( ), pressing MODE should have displayed `Mode(`.

A.3.7. Changing modes and invalidating the display

The calculator's modes can be changed, and these do not cause the calculation to be redisplayed in the new mode. So if the calculator is in degrees and arcsine of 1 is calculated, the calculator will show `90` and `D`. If we change to grades, the calculator shows it as having the value `90 G`, which is wrong (it should be `100 G`).

A.3.8. Linear regression and statistics

Statistics and linear regression are both complicated modes, with their own idiosyncratic conventions. For example, the M+ key changes its meaning in these modes, and the ALPHA key has new invisible meanings (Section A.1.5 and Figure A.1). Since the calculator has a two line display, why is the data for statistics or regression not shown on the display? A user would then have been able to edit the data easily; instead, each data item disappears and cannot be edited. It can, however, be deleted by pressing CL (i.e., shifted M+ , unrelated to the DEL or AC keys that are normally used for this purpose). Using the two line display's editing capability would have meant no new feature for editing was required.

Statistics mode is indicated by a little SD icon appearing in the display. Since one of the other modes is "significant digits", this is confusing. (Significant digits itself is abbreviated `Sci` in the display, which in turn might be confused for scientific notation …)

A full discussion of these modes, other than a lengthy detailing of their obscurities, would not make any new design points.

Appendix B. Percentage

The *fx-83W*'s idea of percentage is confused (see also Section 3, etc).

When speaking aloud, we might say "12% of 50," but the calculator only accepts such a calculation in the reverse order, as in [5] [0] [×] [1] [2] [%] . In English saying "50 of 12%" (saying "of" for ×, which usually works) is quite different—that is, order *does* matter for percent, so the reversal for the calculator is a significant alteration to conventional use. If the English equivalent of the calculator's notation is meaningless, the design of the calculator appears hard to justify. It is possible that the designers chose [%] to evaluate calculations (like [=] does) rather than to be an ordinary operator (like [×]): if so, then necessarily [%] must come last, and hence the confusion. Percentage is 'obviously' useful, and if it is designed to be used arbitrarily, then various features and short-cuts will grow up around it; this appears to be what happens. For anyone who uses the calculator regularly, percent may make sense out of habit, but for anyone who has prior expectations (e.g., from everyday usage), the calculator appears arbitrary.

Arguably, the percent key should be defined consistently, and it should be shown in the two-line display. For example, if we calculate [1] [+] [8] [=] , the display shows `1 + 8` on one line, and `9`, on the other. If we calculate [1] [+] [8] [%] , we still get `1 + 8` on one line, but `112.5` on the other. The 112.5 means (1 + 8)/8 as a percentage, but why doesn't the display show the [%] key press as well?

Perhaps that percent is badly thought-out provides a reason for *not* showing the `%` symbol in the display: it might be too confusing! Consider that [1] [×] [8] [%] calculates `0.08.` Then [1] [×] [1] [×] [8] [%] should probably be 1 + 0.08, or 1.08. It isn't; it is `112.5`. Or [(] [1] [+] [8] [%] is `9.`—it is neither right nor reported as an error. Or [2] [x^y] [3] [%] is `1.021012126` (i.e., $2^{(3/100)}$, which is hardly a 'percentage'). There is not an obvious relation between these sums. Suggesting that the calculation 2^3% is contrived is to miss the point—is testing, say, $(\sin 2)^3$ contrived? If anything is defined, it should mean something, and the rules for knowing what it means should be clear; and if it is not defined it should be detected as an error. (The calculator detects 2.3! as a `Ma ERROR`, so it can do this for other operators.)

Whatever [%] is supposed to mean, it is not very clear. This argument was justified by trying to work out the semantics of a calculator. Now, I argue (independently of any calculator) that what percent means is itself not very clear, and therefore it should not be provided on calculators because of the potential confusion. This is an iconoclastic view. However, just because something—in this case, percentage—is popular does not imply a "corresponding" feature need be provided on a calculator. A less contentious example is that most numbers most people deal with are currency, pounds or dollars say, but there is no need for a calculator to have a [£] key on it. Pounds happen to work like numbers, as do apples and oranges. To put a pound or apple button on a calculator confuses the task of a calculator with that of a cash machine in a grocery store.

Percent means four different things:

— $f \rightarrow p = 100 \times f$. A fraction f multiplied by 100 is a percent p. Interestingly, the *fx83-W* does not provide this meaning for [%] unless the fraction is entered using [÷] (though, surprisingly, using *the* fraction operator [⌟] does not work with percent—see Figure 8). Indeed, multiplying by a hundred is probably far too simple a concept alone to require of a calculator button, so a [%] key typically does some or all of the following more sophisticated operations:

— $a, b \rightarrow p = 100 \times (a - b)/b$. An increase $a - b$ can be expressed as a percent p. (Sometimes called profit.)
— $b, p \rightarrow a = b + pb/100$. Increasing b by a percent p gives a.
— $a, p \rightarrow b = 100 \times a/(100 + p)$. A value a represents a percent p increase over b.

Calling all these percent is like calling $2 - 3$ addition. Compare the last two uses of percent. Adding 5% to a number then subtracting 5% from it does not leave you with the number you started with—it leaves you with 99.75% of it. In other words, any calculator that tries to relate adding and subtracting percents, particularly by using the conventional operators + and −, is going to be confusing.

Elminating percent from calculators is perhaps too much to hope for. Usability has to be balanced against marketability. Indeed percent is a flexible and attractive concept. If calculators are to be simple and easy to use, then they should either provide a range of separately-identified percentage features (discount, margin, *etc*—and these should be meanings associated with the % function, rather than combined with special case of uses of +, ÷, as on the *fx-83W*), or they should not have a percent key at all.

Appendix C: An advert for the *fx-83W*, and Casio's comments on this paper

Complete text from the full page advert in the *Times Educational Supplement*, May 21, 1999, page 31:

> The *fx83-WA* twinline
>
> A new era in educational calculators
>
> When Casio decided to replace the much loved FX82 with a new model it goes without saying that we did our homework first. After all, the FX82 is the country's most popular scientific for GCSE, so its replacement had to be very special indeed.
>
> The new Casio FX83WA takes everything students relied on from the FX82 and has added everything they've ever needed to create the definitive educational aid. For example, the FX83WA includes Casio's VPAM logic so equations can now be entered as they are written. We've also added a Twinline display so students can keep their maths on track by seeing every step they've taken, and by editing and replaying their calculations, they can ensure that they not only get the right answers but can see how they achieved them. What's more, with the FX83WA Twinline, maths teachers can see every entry and so help each student to understand every calculation more clearly. The new Casio FX83WA Twinline. The next generation in educational calculators.
>
> For more details of the FX83WA call your usual Casio dealer or any approved Casio supplier.
>
> CASIO® **SCIENTIFIC** CALCULATORS

The advert was illustrated by a full photograph (at an unremarked 119% full size) and an example of the twin line display. The *fx-83WA* is a version of the *fx-83W* that seems the same.

Casio were invited to respond to this paper; their comments are as follows:

Your comments have been received and as we continually strive to improve our products, they will be passed onto our research and development team in Japan.

It should be noted that the FX83WA is a worldwide product and is used with children from the ages of 11, right up to adults at university level. Therefore the calculator has functions that not every single individual uses but which are available if the need arises. The product is also a best-seller in many countries, has received praise from many leading educationalists and is considered the benchmark for GCSE mathematics.

Many of its features such as the twinline display, replay and edit features and 9 memories have assisted in the understanding of mathematics by younger students and they have also helped students use a calculator more effectively in the classroom.

Appendix D: The Sharp EL-546

The Sharp EL-546 is a calculator very similar in appearance and functionality to the Casio *fx-83W*. This appendix lists a few sample design issues of the EL-546, to show that its design problems are comparable to the Casio calculator's. See Section 3 in the main body of the article.

— The display is used in two steps. As a calculation is entered, it is displayed in the top line, but as numbers are entered, they are shown on the bottom line. The top line, where the calculation will be, is hard to follow. The Casio *fx-83W* instead always has all calculations going straight to the top line.
— The display is too low resolution for what it tries to do. The function [10ˣ] (i.e., raise to a power of ten) is displayed as `10`. Thus 4×10^3 using implicit multiplication looks like `4103`. Resolution is not the only problem the display has; for example, multiplication (whose key is the standard cross symbol, $\times$) is displayed as an asterisk.
— The calculator has a complex number mode. But it does little other than trivial vector calculations: for example, asking for the square root of -1 gives a display of "`Error 2`." Yet the calculator can correctly calculate $i^2 = -1$ by [i] [x^2] [=] and get -1.
— The keys are overloaded with many meanings, yet there are two sets of labels `A`, `B`, `C`, `D` (*and* a set `a`, `b`, `c` *and* a single key with label (`x`, `y`) ...). The key functions use yellow, silver and green on the case, and the key tops use white, green and black. And there are black legends on yellow backgrounds ... it is a very confusing visual design.
— In complex mode, it is not possible to use fractions, because the same key has different meanings in the two modes.
— (This list is not exhaustive.)

Appendix E: The Casio *fx-570W*

The Casio *fx-570W* is very similar to the *fx-83W*, but with many more features. The purpose of this appendix is to show that more advanced technology has not been taken advantage of to solve design problems: this appendix lists sample design issues with the *fx-570W* additional to the ones it shares with the *fx-83W*. See §3 in the main body of the article.

- Like the Sharp EL-546L (see Appendix D) the *fx-570W*'s complex number functions are partially implemented; for example, it cannot find square roots of negative numbers. It cannot do the classic $e^{i\pi}$ (which is equal to -1) and all the obviously useful complex operations; conversely, [Abs] *only* works in complex mode—despite being both useful and mathematically well-defined for real numbers.
- The *fx-570W* corrupts memories D, E, F, X and Y in complex mode; moreover going out of complex mode and back again, the imaginary components of A, B, C are set to zero.
- When a complex number is calculated, the result is shown without any hint whether the imaginary part is non-zero—the user must remember to press a shifted key [Re↔Im] to reveal it.
- It is possible to store a formula. The manual gives one example "to save [a] formula, recall it, and then use it" by pressing [Y] [=] [X] [x^2] [+] [3] [X] [−] [1] [2] [CALC], apparently storing a formula in Y. Since the $=$ does not occur at the end of the calculation, a different key to [=] (but also labelled $=$ and keyed as [ALPHA] [CALC]) must be used. Having done this, the calculator shows X? and a number for X can be entered, followed by [=] (the *usual* [=]). Now, if, say, [RCL] [X] is pressed, the formula disappears and seems to have been lost and cannot be recovered. Although the *User's Guide* refers to the variable Y in the example, it seems to serve no purpose; the Quick Reference Guide (a plastic sheet kept in the calculator's cover) gives a similar example *without* the Y (or any other named formula memory)—perhaps confirming that the *User's Guide* is mistaken.
- In "base-*n*" mode numbers can be entered in binary, octal, decimal and hexadecimal. The key [x⁻¹] takes on an exclusive meaning [LOGIC] which can be pressed three times to reveal a choice of four operators, d h b o. (Pressing it fewer than three times gives operator choices.) The user presses [1]–[4] to select one, and the calculator shows the corresponding letter as a prefix operator. A number can then be entered in this base, for example it is possible to add a number in decimal and a number in binary and show the result in octal (if that is the base mode). The display might show: d42 + b11 and 55o as the answer. Note that the display shows the base of the answer after the number, whereas the numbers in various bases are specified with a base letter before the number. Why is the base specifier only available in base-*n* mode? Why is the base specifier chosen in such a tedious way, when six keys do absolutely nothing in this mode (and hence, the six logical operators that are accessed by pressing [LOGIC] once or twice could have been accessed directly, and the base operator accessed quite simply?
- In "base-*n*" mode, the keys that are normally [ALPHA] or [RCL]-shifted to get memories A, B, C etc, become keys to enter hexadecimal digits without any shifting (they even mean this in binary mode, when they have no meaning mathematically). Thus

the normal functions of these keys are lost. Losing `sin` may not be a problem, but unary minus `(−)` has also gone. Either this is a problem, or it indicates that the manufacturers do not think the unconventional minus is needed in any case.

— The binary, octal and hexadecimal bases would make the calculator appeal to users who wish to do calculations to check computer arithmetic. There are obvious functions that are missing: it is not possible to specify word length (the *fx-570W* has different, but fixed, word lengths in each base); there is no choice but to represent numbers in 2s-complement; there is no logarithm or exponentiation to base 2; it only handles integers (and not, for example, fixed point numbers).

— The calculator provides forty constants. Pressing `CONST` in the middle of a calculation displays CONST _ _ and expects the user to press two digits. Pressing `CONST` `2` `2` *immediately* displays F (in an indistinguishable form to having entered `RCL` `F`), which represents the Faraday constant (the charge of a mole of electrons). Constant 23 is the charge of an electron; this is displayed as e, *exactly* the same as the e of exponentiation (so you could get ee-ee displayed, and have the value 1). There is no way to search for a constant the user knows the calculator has, but whose two digit code is not known. (Note that many other choices the calculator provides, such as the choice of number base operator, are provided as "multiple choice" where the calculator displays the choices, so—if the calculator had been designed consistently—the user would not have needed to know the codes.) Although the constants are based on international standards, the *User's Guide* does not tell us their units.

— Although many keys are overloaded with multiple labels, the numerical integration key `∫dx` is prominent and has no other labels—thus causing other keys to be unnecessarily awkward. To do an integration $\int_a^b f(x)\,dx$, `∫dx` is pressed, then the function of X, then `,` a `,` b `,` n `)` `=`. The value n is used in the method of integration (the *User's Guide* says it is "the number of partitions (equivalent to $N = 2^n$) for integration using Simpson's rule." Although other functions (like `MODE`, `LOGIC` and `DRG▷`) have menu choices, integration does nothing to remind the user of its conventions—such as requiring the variable X, or even that it requires n (it appears to have a default value, so the user is not warned if it is omitted). When the calculator is integrating, the display goes blank—nothing is displayed at all—and the calculator appears identical to being off.

— Since X is the bound variable of integration, double integrals are not very useful (for example, integrating to find the volume of a solid generally requires two bound variables). Nevertheless, $\int_0^1 \int_0^1 1\,dx\ dx$ is the volume of a $1 \times 1 \times 1$ cube. Entering it in the appropriate way is treated as a syntax error.

— The key `DRG▷` provides a postfix operator representing degree, radian or grade units, which is converted according to the angle mode of the calculator. (So 200^g when in radian mode has value π, but would have value 200 in grade mode.) Unfortunately the display symbol is identical to the representation of `°′″`, so ambiguous displays are possible—and none of them are syntax errors, even though meaningless.

— The calculator has an "engineering mode" which enables shifted meanings for the digit keys, so that multipliers like μ, n, k, or M can be pressed. In normal mode, these shifted meanings, indeed any shifted meanings, are inaccessible—in other words, there is no obvious need for the mode because, without any ambiguity, the shifted meanings could have been made available.

— (This list is not exhaustive.)

Paper accepted for publication by Editor, Dr B. Gaines