SANE Tool Set ERS
Apple Numerics
Version 0.80 – 24 June, 1986

Modification History

0.00	06 Nov 85	JWT	Original.
0.01	11 Nov 85	JWT	Reflect change in tool set call mechanism. Specify initialization functions.
0.02	12 Dec 85	CRL	Parameter passing in SANE zero page, conformance with 11/21/85 Tool Locator ERS
0.80	24 Jun 86	CRL	Add example, register return info, and haltvector info, remove direct page parameter passing scheme; add timing diagram and update use of direct page.

Project Identification

Related Documents

Apple Numerics Manual, Don Reed
Tool Locator ERS, Steve Glass

Product Abstract

This tool set will provide the Cortland assembly-language programmer with the Standard Apple Numeric Environment (SANE). SANE is scrupulously conforming extended-precision IEEE Standard (754) arithmetic, with elementary functions. SANE is designed for sufficient, convenient numeric support for most applications. It includes

- IEEE types single (32-bit), double (64-bit), and extended (80-bit)
- a 64-bit type for exact fixed-point computations, as in accounting
- basic floating-point operations (+ – * / √ rem)
- comparisions
- binary-decimal and float-integer conversions
- scanning and formatting for ASCII numeric strings
- logs, trigs, and exponentials
- compound and annuity functions for financial computations
- a random number generator
- functions for management of the floating-point environment
- other functions required or recommended by the IEEE Standard

The functionality of the SANE tool set matches that of the Macintosh SANE packages, as well as that of the 6502 assembly-language SANE software from which it is derived.

Use of the SANE tool set follows the general scheme for accessing Cortland tools.

Use Environment

Hardware Environment

The SANE tool set is part of Cortland system software and requires only the basic Cortland hardware.

Software Environment

You can program with the SANE tool set using any assembler that generates code for the Cortland. The equate file SANE.equs and the macro file SANE.macros facilitate use with the Cortland Programmer's Workshop assembler.

Functional Specifications

General Overview

The SANE tool set comprises the usual tool-set housekeeping functions, and functions (FP816, Elems816, and DecStr816) that serve as entry points for the major pieces of SANE code: each call to FP816, Elems816, or DecStr816 passes an opword parameter specifying the operation to be performed. For example, the opword 0206 (hex) passed to FP816 indicates *divide by a value of type single*.

FP816 contains
- basic arithmetic operations
- comparisons
- conversions
- environment control
- IEEE auxiliary operations.

Elems816 contains
- elementary functions
- financial functions
- random number generator.

DecStr816 contains
- numeric scanners and formatter.

User Scenarios

An application that uses the SANE tool set must first call the memory manager to reserve a 256-byte *SANE direct page*, and pass the 2-byte address of this page as an argument to the initialization function for SANE. The following code will effect application initialization:

```
PUSHWORD <zero bank address>
_SANEStartup
```

The following code for a binary operation illustrates a typical invocation of a SANE tool set function:

```
PUSHLONG  <4-byte source operand address>
PUSHLONG  <4-byte destination operand address>
Fxxxx
```

The name of each macro is the same for 6502 SANE, 68000 SANE, and 65816 SANE. For the Cortland, the Fxxx macro expands to

```
PUSHWORD <OpWord>
LDX       #SANEtsNum + FuncNum*256
JSL       $E10000
```

The expansion of the macro differs from its 6502 and 68000 counterparts.

Some SANE tool set operations require different numbers of arguments, some pass 16-bit integer arguments by value, and some return results in the X, Y, and status registers.

Example

This example illustrates the use of the numeric scanner and formatter. The procedure accepts as argument an ASCII string representing a number of degrees and returns the trigonometric sine of its argument as a numeric ASCII string. Both input and output are Pascal strings: the zeroth byte gives the length, the first byte contains the first character in the string. The caller of the procedure pushes the address of the input string and does a JSR to SINE. The procedure overwrites the input string with the result, whose length may be as large as 80, and clears the stack. SINE uses FP816, ELEMS816, and DecStr816 which are the three principal functions in the SANE toolset.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Somewhere early in the program, initialize SANE.                           ;
;     Call the memory manager to reserve 256 bytes of zero bank for use      ;
;     as SANE direct page. #SANEdirectpg is the address of this memory.      ;
                                                                             ;
             PUSHWORD      #SANEdirectpg                                     ;
             -SANEStartup                                                    ;
                                                                             ;
; Near the end of the program, shut down SANE.                               ;
                                                                             ;
             _SANEShutdown                                                   ;
                                                                             ;
; And call the memory manager to release the memory reserved for the        ;
;     SANE direct page (often by releasing ALL reserved memory).             ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; procedure SINE ( var s : DecStr )

; Symbols
;     s :         i/o string
;     index :     16-bit integer index
;     theDec :    decimal record
```

```
;     vp :       boolean for validprefix
;     theForm :  decform record
;     x :        extended temporary
;     const :    extended constant = pi/180
;
SINE          ENTRY
              PLA
              STA          return       ; save return address

              PLA
              STA          sAdr         ; address of s -> sAdr
              PLA
              STA          sAdr+2

              LDA          #1
              STA          index        ; 1 -> index

              PUSHLONG     sAdr
              PUSHLONG     #index
              PUSHLONG     #theDec
              PUSHLONG     #vp
              FPSTR2DEC                 ; s -> theDec

              PUSHLONG     #theDec
              PUSHLONG     #x
              FDEC2X                    ; theDec -> x

              PUSHLONG     #const
              PUSHLONG     #x
              FMULX                     ; convert to radians:  x*const -> x

              PUSHLONG     #x
              FSINX                     ; sin(x) -> x

              PUSHLONG     #theForm
              PUSHLONG     #x
              PUSHLONG     #theDec
              FX2DEC                    ; x -> theDec

              PUSHLONG     #theForm
              PUSHLONG     #theDec
              PUSHLONG     sAdr
              FDEC2STR                  ; theDec -> s

              LDA          return
              PHA
              RTS

index         ds           2
vp            ds           2
const         dc           h'AE C8 E9 94 12 35 FA 8E F9 3F'  ; const = pi/180
x             ds           10
theForm       dc           i'1,10'      ; first style, then digits
theDec        ds           33           ; sgn, exp, length, ascii = (2+2+1+28)
return        ds           2
sAdr          ds           4
```

Performance Characteristics and Limitations

The application must preserve bytes 24-29 (decimal) of the SANE zero page between calls to SANE. These six bytes hold the floating point environment and halt vector. The remainder of the SANE zero page is scratch space used only during SANE execution: it is not preserved across calls to SANE and is available to the application. Warning: future implementations of SANE on Cortland may NOT store the floating point environment and halt vector on zero page. Applications may therefore forfeit upward compatibility if they access these variables directly. Always access these variables only through calls to SANE.

Except for the _SANEVersion call, the SANE tool set removes all arguments from the stack and returns no results on the stack. Temporary stack growth during engine calls does not exceed 40 bytes (50 for elementary function calls).

The SANE tool set conforms to the general tool set rules for management of the CPU registers, modes, and busy flag.

Typical timings based on a few sample values:

0.5 - 1.2 ms	add, subtract
1.0 - 5.0 ms	multiply
1.9 - 5.2 ms	divide
0.7 - 1.4 ms	scanner
0.8 - 1.0 ms	formatter
2.4 - 5.8 ms	extended-to-decimal
0.8 - 3.5 ms	decimal-to-extended
50 - 100 ms	trigonometric, exponential, logarithmic

Applicable Standards

The SANE tool set conforms to IEEE standard 754 for binary floating-point arithmetic and to the proposed IEEE standard 854, which is a radix- and word-length-independent standard for floating-point arithmetic. The SANE tool set fully supports the Standard Apple Numeric Environment.

Interface

Housekeeping Functions

The boot initialization function _SANEBootInit, and the reset function _SANEReset do nothing.

The application initialization function _SANEStartup takes a single 2-byte input parameter: the address of the application-allocated SANE direct page. Initialization clears the SANE environment word, installing the default settings of round-to-nearest, round-to-extended-precision, all exceptions clear, and all halts disabled. It also sets the halt vector to zero. As with 6502 SANE, halts remain inoperative until the application both enables halts and also gives the halt vector a nonzero value.

The application shutdown function _SANEShutdown does nothing. It is the responsibility of the application to call the memory manager to release SANE direct page.

_SANEVersion returns version information as a two-byte value. Space for this value must be reserved on the stack before invoking the _SANEversion macro. After the call, top-of-stack will contain the two-byte version number.

SANE Functions

The (non-housekeeping) SANE functions and their interfaces are documented in Part II (The 6502 Assembly-Language SANE Engine) of the *Apple Numerics Manual*. The SANE tool set deviates from this documentation in these ways:

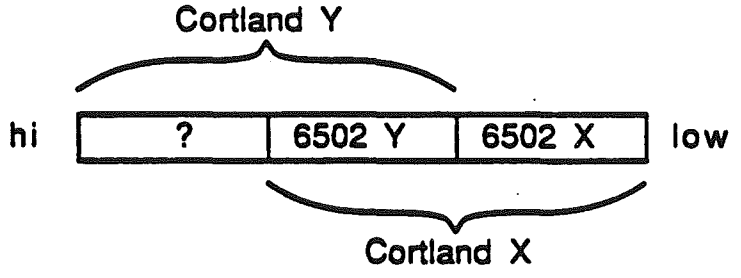
- Access is via the tool set dispatcher and not by JSR. Thus, invocations end with

```
LDX     #SANEtsNum + FuncNum*256
JSL     $E10000
```

  instead of

```
JSR     <xx6502>
```

- All address parameters are four bytes instead of two.
- The low bytes of the X and Y-registers return information as documented in Part II of the *Apple Numerics Manual*. The high byte of X duplicates the contents of the low byte of Y. The high byte of Y is undefined. The diagram below shows the relationship between the return information in 6502 registers and in 65816 registers.



- The Remainder call uses the N-bit and both bits seven and fifteen of the X–register to return the sign of the quotient. The low 7 bits of X contain the absolute value of the quotient.
- The halt vector is the address of a halt handling routine. It is stored as a four byte address. SetHaltVector expects the 4-byte halt vector to be passed by value on the stack. GetHaltVector returns a 3-byte halt vector in the X and Y registers. X contains the low two bytes (first and second) of the halt vector, and Y contains the second and third bytes; the second byte of the halt vector occurs in both the X and Y registers. The diagram above may help visualize how X and Y return the halt vector.

- The halt mechanism is changed. When a halt occurs, the input parameters and SANE opcode are located in SANE zero page as shown in the diagram below. For one and two argument calls, C holds the address of DST. For two argument calls, B holds the address of SRC. For binary-to-decimal conversion, A holds the address of the decimal record, B holds the address of the binary value, and C holds the address of the decform record. It is important to remember that halts occur only on calls to FP816. Elems816 stimulates halts only through a procexit call to FP816. DecStr816 makes no calls to FP816 and therefore never stimulates halts.

  When a halt occurs, 65816 SANE executes JSL HaltVector. When the halt handler receives control, the A-register contains 'pending exceptions'. The halt handler can continue execution as if no halt had occurred by executing RTL. After return to the FP816 code, the A-register, not 'pending exceptions', is used to set the final floating point exceptions. Executing LDA PendingExceptions at the end of the halt handler will ensure that the exceptions from the current call are handled correctly. 'Pending exceptions' is a sum of the five exception constants, represented as an integer from 0 to 31. Unpredictable results will occur if the A-register contains a value out of this range on exit from the halt handler.