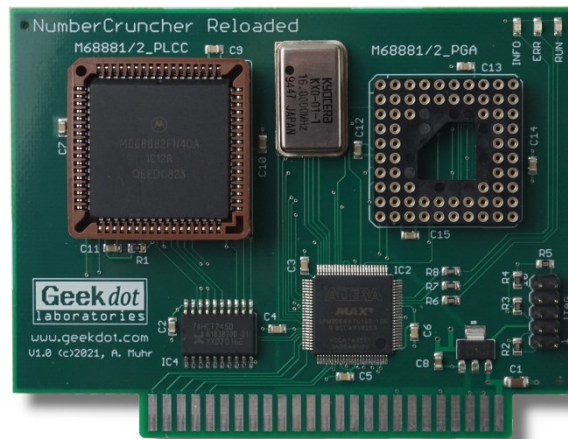




# NumberCruncher Reloaded User's Manual



*Compatible with Apple IIe  
and IIgs.*

Created with ♥ at the



# The GeekDot Laboratories

This manual and the software described in it are copyrighted, with all rights reserved. Under the copyright laws, this manual or the software may be copied, in whole or part, with written consent of **The GeekDot Laboratories**. The same proprietary and copyright notices must be affixed to any permitted copies as were affixed to the original. This exception does not allow copies to be made to be sold to another person. Under the law, copying includes translating into another language or format.

You may use the software on any computer owned by you, and if you need extra copies just do so.

**The GeekDot Laboratories** logo is a trademark of GeekDot Laboratories registered nowhere we can think of.

Honestly...

Do not use this manual or the **NumberCruncher Reloaded** card for any mission-critical applications, or for any purpose in which a bug or failure could cause you a financial or material loss. This product was designed to enhance your Apple II computing experience but may contain design flaws that could inhibit its proper operation or result in a loss of the data recorded on the storage devices attached to it. When using this product, you assume all risks associated with operation or data loss.

Legalese Version:

Axel Muhr, aka **The GeekDot Laboratories**, is a private person and hobbyist. He makes no warranties either expressed or implied with respect to this manual or with respect to the software or firmware described in this manual, its quality, performance, or fitness for any particular purpose. All software and firmware is sold or licensed —as is. Any risk of incidental or consequential damages resulting from the use of the information in this manual or the software / firmware / hardware described herein, shall be assumed by the user or buyer or licensee. In no event will **The GeekDot Laboratories** be liable for direct, indirect, incidental or consequential damages resulting from any defect in the software / firmware / hardware described in this manual.

**The GeekDot Laboratories** reserves the right to make changes and improvements to the product described in this manual at any time and without notice.

### Last **words of warning**:

You should avoid electrostatic discharge to the **NumberCruncher Reloaded** card. Like all electronics devices, static shock can destroy or shorten the life span of the **NumberCruncher Reloaded** card. Avoid touching the **NumberCruncher Reloaded** card after you have walked across the room, especially over carpet, and especially in dry weather.

You should safely discharge yourself before you handle **the NumberCruncher Reloaded** card. This can be done by momentarily touching a grounded piece of metal.

In all cases, please exercise common sense and observe all electrical warnings provided by the manufacturers of the equipment you are using.

Parts within this Guide were taken from the original FPE manual. This was not done due to laziness, but because some things are so undeniably good, you just make them worse if you try to work around it. We're all standing on the shoulders of giants...

### **Finally, ...**

I'd like to deeply thank my loved ones Gesa and Zoe for being forgiving about the time I've ~~wasted~~ spent, developing this thing.

*Vagöldsgott*, Mike Brüstle, without whom I would still stare into my VHDL like a pig into a clockwork.

*Kiitos paljon* goes out to Petri Kukko who did all the "Hausfrauentests" for me.

*Merci beaucoup* Antoine Vignau for adding some "Brutality Deluxe".

*Thanks* a metric ton to Richard Edwards for proofreading "zhe manual" through a native-speakers eye.

With this off my heart, let's proceed for the fun part.

<b>Preface</b> .....	6
Before You Start .....	6
About the NumberCruncher Reloaded.....	7
About this guide .....	9
Some visual cues .....	9
<b>Chapter 1</b> .....	<b>10</b>
Preparing the Card.....	10
Installing the NumberCruncher Reloaded.....	11
Software.....	12
Apple IIgs.....	13
Apple II, II+ and IIe .....	13
AppleWorks™ .....	14
<b>Chapter 2</b> .....	<b>15</b>
SANE Background .....	15
Interfacing to SANE .....	15
About the MC68881 and SANE.....	16
<b>Chapter 3</b> .....	<b>19</b>
Access the NumberCruncher Reloaded.....	19
How the NumberCruncher Reloaded transfers Data .....	21
MEMREG and REGMEM Operations.....	21
REGREG Operations.....	23
<b>Chapter 4</b> .....	<b>25</b>
Programming.....	25
Assembly.....	25
C Language .....	28

Applesoft.....	28
Programming Hints .....	30
<b>Appendix A .....</b>	<b>32</b>
The 68881/2.....	32
Data formats .....	33
Construction of an MC68881/68882 Command .....	34
Constructing a command step-by-step .....	35

## Before You Start

**N**ow you have the ultimate math power at your hands. With the NumberCruncher Reloaded installed in your Apple II computer, you can take advantage of the dedicated floating point calculation power and run many applications on your Apple IIgs unchanged but much faster. With little programming Apple IIe applications can be accelerated, too. This guide tells you how.

## About the NumberCruncher Reloaded

The **NumberCruncher Reloaded** is a peripheral card that features a math co-processor, often also called a Floating Point Unit (FPU) which is specialized on, well, floating point calculations. Doing so, it is much, *much* faster than any 6502 or 65816 CPU ever will be.

The **NumberCruncher Reloaded** will not automatically speed-up your programs as CPU accelerators like the Transwarp GS or ZIP CHIP would do. Programs will have to be either specifically written to use the **NumberCruncher Reloaded** or use a floating-point library like the SANE interface which then needs to be patched to itself use the **NumberCruncher Reloaded** for calculations instead of the main CPU.

You will find all the necessary patches, tools and demos in the provided archive.

Like many great inventions, The **NumberCruncher Reloaded** stands on the shoulders of giants. The 'Reloaded' hints towards its predecessors:

In the beginning, 1988, there was the *Floating Point Engine* (FPE) created by Innovative Systems ('iS' for short). While it was a great idea, it wasn't the most stable design – but it laid the foundation especially and most importantly for the software we're still using today.

Due to the FPE's issues there was quite some displeasure in the usership and in 1990 a German company called Alternative Systems announced the *Number Cruncher*, a 'correction' to the original design – here's their newsgroup announcement:

*"The FPE is suffering from a major problem, namely the coproc is crashing internally and has to be reset in software. This happens in a non-deterministic way, and software written for that engineering junk must be adapted to that.*

*The Number Cruncher is compatible with the FPE but is actually what the FPE was supposed to be - a math coproc that works. It performs very well."*

Over the years the FPE as well as the NC faded in unobtainium. Because they were cool, and we love processors of all kinds it was time to reload the Number Cruncher.

To learn more about its history, visit its very own page at: [www.geekdot.com/number-cruncher](http://www.geekdot.com/number-cruncher)

Like its predecessors the **NumberCruncher Reloaded** provides the most efficient floating point math capability for all members of the Apple II™ family.

And like the FPE and the original NC, the **NumberCruncher Reloaded** supports the Motorola MC68881 floating point processor but was improved in many aspects to make it much more usable in the 21<sup>st</sup> century:

- it also supports the enhanced and the easier to find MC68882
- FPU's can be used either in pin-grid-array or PLCC package thanks to the two sockets provided. Again, the latter being much more common these days
- Increased stability by using low-power SMD parts and a 4-layer PCB with dedicated supply layers
- Speed optimized FPU protocol handling
- 2 more LEDs, which we consider very important.
- Upgradable firmware (ALTERA JTAG programmer required)



**About this guide** This guide contains all the information you need to use the **NumberCruncher Reloaded** with your Apple II™. Here's what you'll find in this guide:

- Chapter 1, “Preparing the Card,” tells you how to install your NumberCruncher Reloaded into your Apple II™ and what you need to set up.
- Chapter 2, “The Software” explains you some details about the software delivered with this card.
- Chapter 3, “Access the NumberCruncher Reloaded” contains a brief overview of the different ways to put your card to use.
- Chapter 4, “Programming” gives you some examples of how to use the card with Assembler, C and even Applesoft Basic
- Appendix A, “The 68881/2” will give you a brief overview of the MC6888x FPU and its internals.

**Some visual cues** This Manual uses some special text elements to help guide you. Use them as visual cues as you read:

❖ **By the way:** Text set off in this manner presents sidelights or interesting pieces of information. ❖

△ **Important:** Text set off in this manner presents important information. △

▲ **Warning:** Warnings like this alert you to situations in which you might damage your equipment or lose data if you don't follow the instructions carefully. ▲

Special terms appear in *italics* when they are introduced; these terms are defined in the glossary at the back of the guide.

## Preparing the Card

Being optimized for maximum convenience there is not much for you to prepare. There are no jumpers to set and no software to configure it.

Depending on the version you have ordered your NumberCruncher Reloaded might look slightly different. It might have just one or both sockets soldered and already features a 68881 or 68882 FPU.

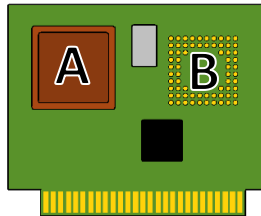


Fig. 1

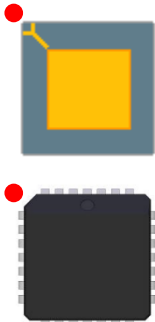


Fig. 2

If you purchased your NumberCruncher Reloaded without any FPU, please install one now. Like all integrated circuits FPUs are prone to static charge. Ground yourself before handling the FPU. Holding the card with its slot-connector facing down (Fig. 1), insert the FPU in either the left PLCC (A) socket or the right PGA (B) socket with pin-1 in the upper left corner (Red dot in Fig. 2).

△ **Important:** All you must make sure is that there's just one FPU installed. Populating both sockets will result in unpredictable behavior of your card and Apple computer. △

## Installing the NumberCruncher Reloaded

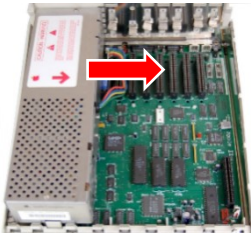


Fig. 3

- 1) Turn off your Apple II computer. Carefully remove the case cover from your computer as described in the owner's manual supplied by Apple.
- 2) Face the computer as you normally would if using it (keyboard toward you. Refer to Figure 3.).
- 3) Ground yourself by touching the top of the metal cover of the power supply on the left-hand side of the computer.
- 4) Remove the NumberCruncher Reloaded from the box and the anti-static plastic wrapping. Undertake a final check to ensure that there is an FPU (68881/2) installed in one of the two sockets. Just *one*.
- 5) Install your NumberCruncher Reloaded in any slot numbered between 1 and 7. Ensure that the component side of the board (the side with the lettering) faces to your right in the direction away from the power supply. Refer to the red arrow in Fig.2.  
Don't try to use the memory expansion slot in the IIGS - it is not a peripheral slot.
- 6) Replace the case cover, plug the computer power cord into the power outlet, apply power, and boot your computer as normal.
- 7) The computer should boot normally and is now ready for software installation.

You may have to enable the slot in which the Number Cruncher Reloaded is installed. Follow the instructions in your user's manual to use the control panel to select "Your Card" for the appropriate slot.

No enabling is required for Apple IIe computers because the slot I/O is normally active.

**Software** One reason to revive the FPE/NC was the already existing software base. So, for example in contrast to the *T2A2 Transputer interface card* (another splendid product from The GeekDot Laboratories) you are not required to program your own software to take advantage of the Number Cruncher Reloaded.

△ Your card came with a floppy which contains a basic set of programs discussed in this manual. You will find all these (and more) on the NumberCruncher Reloaded homepage, too:

<http://www.geekdot.com/numbercruncher-reloaded/> △

The easiest and most transparent way to have your software accelerated is to pick software which use the Standard Apple Numerics Environment™ (SANE).

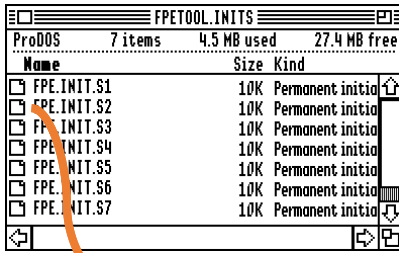
The SANE defines a series of calls which provide numeric operations in accordance with IEEE Standard 754 Binary Floating-Point Arithmetic. This environment provides very accurate numerics. Unfortunately, SANE operations can be very slow. The NumberCruncher Reloaded provides the numeric operations, but at a much faster rate.

Because SANE is standard with the Apple II™ computer, Innovative Systems provided a numerics software package which replaces most of the routines in the SANE toolset. This software uses the same calling sequences, processes the commands in 80-bit precision, and generally provides the same results as those described in the "Apple Numerics Manual".

Finally, there's some software natively using an FPU card. These programs are the fastest and make your Apple II™ feel like a rocket (at least compared to one without an FPU).

## Apple IIgs

To use the numerics software on an Apple IIgs, you must have copied the FPE.INIT from the archive/disk to the SYSTEM folder on the system disk.



Name	Size	Kind
FPE.INIT.S1	10K	Permanent initia
FPE.INIT.S2	10K	Permanent initia
FPE.INIT.S3	10K	Permanent initia
FPE.INIT.S4	10K	Permanent initia
FPE.INIT.S5	10K	Permanent initia
FPE.INIT.S6	10K	Permanent initia
FPE.INIT.S7	10K	Permanent initia



In the provided archive you will find a initialization file, coded specifically for each slot. These slot dependent files provide a small speed improvement over files which automatically locate the **NumberCruncher Reloaded** slot, because the code uses direct addressing of the FPE slots rather than using indirect indexed addressing. Thus, if the **NumberCruncher Reloaded** is in slot 2, you must have the file "FPE.INIT.S2" in your "/SYSTEM/SYSTEM.SETUP" directory on your startup disk.

- ▲ **Warning:** Use of any FPE.INIT file not corresponding to the slot number containing the NCR will crash your system. ▲

Optimized code for accessing the **NumberCruncher Reloaded** from a higher-level language will be included in the software package you purchase (such as ORCA/C) and requires no installation on your part.

## Apple II, II+ and IIe

The replacement for the SANE interface in the Apple II, II+ or IIe is customized (to a specific absolute memory address) and is included on the FPETOOLS distribution disk in the "/FPETOOLS/FPE.6502/TOOLSET" directory. This toolset uses the following calls:

```
jsr $2100      to call the fp6502 routines
jsr $2104      to call the ELEMS6502 routines
```

This toolset loads into locations beginning at \$(00)2100 and has a length of less than \$1000 bytes. The toolset has a filetype of BIN.

## AppleWorks™ AppleWorks™ Classic

The replacement for the AppleWorks Classic calls to the 8-bit SANE software is included on the FPETOOLS distribution disk. Boot that disk and it will automatically locate the NumberCruncher Reloaded, report its slot number and displays a menu.

If you have AppleWorks Classic, select option 2 to install a patch which provides the capability for AppleWorks to use the NumberCruncher Reloaded when doing math.

Because AppleWorks can be in a subdirectory, please provide the volume name and the subdirectory in response to the prompt from the initialization program. For example, if AppleWorks is installed in directory "/AppleWorks" on volume "/hard1", please enter "/hard1/AppleWorks" when prompted. Also, if your Startup disk and your Program disk are the same, enter the same information after both prompts.

## AppleWorks™ GS

Support for AppleWorks GS is automatically provided as this package uses the GS/OS SANE tool set calls.

## SANE Background

**Interfacing to SANE** The Standard Apple Numerics Environment™ (SANE) defines a series of calls which provide numeric operations in accordance with IEEE Standard 754 Binary Floating-Point Arithmetic. SANE also provides several utility functions which include conversions of data from an ASCII representation to binary floating point and back again. This environment provides very accurate numerics. Unfortunately, SANE operations can be very slow. The Number Cruncher Reloaded provides the numeric operations, but at a much faster rate.

Because SANE is standard with the Apple II computer, Innovative Systems provides numerics software package which replaces most of the routines in the SANE toolset. The software uses the same calling sequences, processes the commands in 80-bit precision, and generally provides the same results as those described in the "Apple Numerics Manual. One difference is that the transcendentals returned are slightly less accurate (76 bits or more of accuracy versus 80 bits from SANE); however, this change in accuracy should not adversely affect the performance of your software (see "Apple Numerics Manual. Second Edition". Chapters 28. and Chapter 10 of this manual for the details). Another difference is that the NumberCruncher Reloaded does not process COMP type variables. COMP calls will work with the FPE toolset (except at the speed of the Apple II since the calls use the standard SANE code). Because the FPE toolset is a hybrid of calls to the NumberCruncher Reloaded and to the standard SANE toolset code, use of the FPE toolset is automatic and transparent to most existing software.

## About the MC68881 and SANE

The information in this chapter is excerpted from the "Apple Numerics Manual, Second Edition" chapters 27, 28, and 29. While all the information in the SANE manual may pertain to the operation of the MC68881 in the Macintosh II, the data here pertains only to the operation of the NumberCruncher Reloaded when called by the FPE toolset.

### Functions the same on both MC68881 and FPE software and SANE

The MC68881 and the FPE toolset return identical results for the following operations:

- Addition
- Subtraction
- Multiplication
- Division
- square root
- remainder
- round-to-integral value
- conversions between floating point formats
- negate
- absolute value

### Functions similar

For transcendental operations, the NumberCruncher Reloaded gets results slightly less accurate than those returned by SANE; for some operations, the FPE gets different results for cases involving zero, Infinities, and NaNs.

The NumberCruncher Reloaded returns slightly less accurate results than those returned by SANE in the following cases:

- binary scale (FPE truncates scale factors to 14 bits)
- base-e logarithm
- base-2 logarithm
- base-e logarithm of  $1 + x$
- base-e exponential
- base-2 exponential



- base-e exponential minus 1
- sine, cosine, tangent, arctangent
- integer exponentiation
- general exponentiation
- base-2 logarithm of  $1 + x$
- base-2 exponentiation minus 1
- compound interest
- annuity factor

The NumberCruncher Reloaded returns results with the same accuracy but behaves differently for zero, denormalized numbers, Infinities, and NaNs:

- round-to-integer (when out-of-range the NC-R preserves the sign)
- truncate-to-integer (when out-of-range the NC-R preserves the sign)
- binary logarithm (same results except for 0 and Infinity).

All remaining operations available from SANE can be assumed to be as accurate and operate in the same manner for calls to the FPE toolset.

### Accuracy of the MC68881's elementary functions

For the elementary functions, both the SANE and the FPE (MC68881) packages have errors in the least significant bits of the fraction part of the extended format results, but the SANE package errors rarely exceed the last bit, whereas the FPE errors can extend to as many as five bits. Hence, for individual elementary functions, both packages return results nearly identical when rounded to single or double precision. For complicated expressions involving elementary functions, the NumberCruncher Reloaded is more likely to return an error in double precision results than the SANE packages are.

## Controlling the environment

The FPE toolset converts the standard SANE environmental control calls to those needed by the MC68881.

## Halts and Traps

The FPE toolset handles halts in the same way that the SANE package does.

Traps are not supported.

## Access the NumberCruncher Reloaded

How does the System know which slots contains the NumberCruncher Reloaded or its predecessors?

1. If you have an Apple IIGS and you have loaded the FPETool.INIT file corresponding to the slot containing the FPE, all calls to SANE will automatically go to the NumberCruncher Reloaded.

2. If you write your own code to directly access the NumberCruncher Reloaded, you must use the correct address for the slot locations: that is,

$\$c080 + 16 * slot\_number$  (e.g.,  $\$c090$  for slot 1)

Refer to Chapter 3 for information on how to determine the NumberCruncher Reloaded slot number without hard coding the slot number into your source code.

Direct access means that software writes information directly to or reads data directly from the Number Cruncher Reloaded coprocessor interface registers. These interface registers reside in the 6 locations reserved for the slot in which the card resides. These 16 locations are designated as Read-only, Write-only, or Read/Write, depending upon their purpose. In using direct access, the software does not need to "pass through" unnecessary general-purpose code.

Direct access is the most efficient method of communicating with the FPE. It eliminates overhead; this is not to say it is always the best method of interfacing, however. For Direct refer to Chapter 3 containing additional information necessary to do direct accessing of the NumberCruncher Reloaded.

The Motorola MC6888x communicates with the host processor (6502, 65C02, or 65816) by way of Coprocessor Interface Registers (CIR). These registers are used for control of, transferring operands to, and returning status from the MC6888x. The Apple II technical manuals and the Motorola "MC68881/68882 Floating-Point Coprocessor User's Manual" contain valuable information on accessing the registers and details which explain the uses for the CIRs. The NumberCruncher Reloaded allows access to all the CIRs that are required to implement all MC6888x instructions. The only CIRs not accessible are those intended for use with the 68020/68030 microprocessors, and which do not impact performance with the 6502/65816. The registers implemented in the NumberCruncher Reloaded and their base addresses are given in this table:

Base Address			
Register	Location	Width	Type
Response	\$C0k0	16	R
Control	\$C0k2	16	W
Save	\$C0k4	16	R
Restore	\$C0k6	16	R/W
Command	\$C0k8	16	W
Condition	\$C0kA	16	W
Operand	\$C0kC	32	R/W

1. All transfers are byte swapped from normal 6502/65816 storage; that is, the MSB of the data is contained in the lowest memory address.
2. k is the number of the slot containing the NC-R + 8
3. Word transfers (16 bits) to the Operand register use addresses \$C0kC and \$C0kD. Multiple word transfers (32, 64, 80, and 96 bits) use all four locations (\$C0kC-\$C0kF). Note that for 80-bit transfers, the first data transfer requires that \$C0kE and \$C0kF receive \$0 values and that bits 65 to 80 are transferred to \$C0kC and \$C0kD.
4. All locations are in the I/O page (\$00 or \$E1) of 65816 RAM space.
5. All locations are in page \$C0 of the 6502 RAM space.

## How the NumberCruncher Reloaded transfers Data

The NumberCruncher Reloaded fully supports Motorola's MC68881 coprocessor dialog. The dialog consists of a rigidly structured combination of commands and response primitives. The commands tell the MC68881 what to do, and the primitives indicate actions that are required, including: transfer data, wait for synchronization, wait for completion of operation, and handle error conditions. Failure to follow the coprocessor protocol can result in destruction of your code during program execution.

The NumberCruncher Reloaded allows three types of operations: Memory-to-Register (MEMREG), Register-to-Memory (REGMEM), and Register-to-Register (REGREG). MEMREG and REGMEM operations may be done at any precision. REGREG operations are always done in extended precision.

## MEMREG and REGMEM Operations

MEMREG and REGMEM operations move data from Apple memory to a MC68881 floating point, control, or status register, and from a MC68881 floating point, control, or status register to Apple memory (refer to the flow chart in Figure 3). These operations are often called move-in or move-out operations, respectively. They require that the software

1. Write a command word (16 bits) to the Command register (\$C0k8)
2. Check the word in the Response register (\$C0k0) for a Null Come-Again (CA) (any value other than \$8900)
3. Transfer the operand byte(s) to or from the Operand register (\$C0kC)
4. Check the word in the Response register (\$C0k0) for a Null Release (i.e., the most significant bit - CA bit is equal to 0)

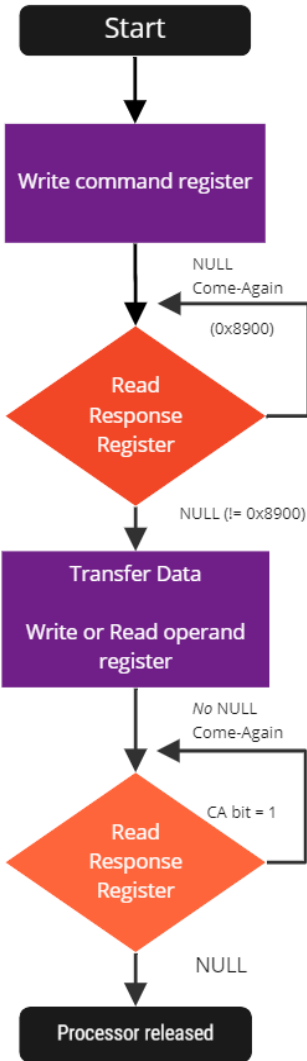


Fig.3

The \$8000 and \$8900 signify that the values are written the way the MC6888x expects to write them; however, the 6502/65816 must read and write all data in byte reversed order (\$0089 in this case). The reason for the byte reversal is that the 6502 and the 65816 write the low byte of the accumulator to the low byte of memory or to a peripheral slot – they are little-endian CPUs. This is opposite to the requirements of the MC68xxx series, which are called big-endian CPUs. Hence, the 6888x expects or reports the most significant byte (MSB) as the low address byte. You must transpose the byte order of all data (including 80-bit data) to satisfy the MC6888x. Remember this because it applies to every command or operand transfer to, and operand and response transfer from the NumberCruncher Reloaded.

You might even be wondering why we check for a value of \$8900. The answer is adaption. If the MC6888x was being used with an MC68020 microprocessor, the value read from the Response register would indicate the number of bytes to be transferred. In FPE applications, the MC6888x does the same, but the 6502/65816 cannot easily make sense of this value. So, to improve processing time, it was noted that \$8900 is the only response primitive that requires the 6502/65816 to wait before transferring data. Any other value from the Response register of the is FPE implementation indicates that the 6502/65816 may transfer an operand.

▲ **Warning:** Do not try to test for a non-\$8900 value as this will confuse the MC6888x and destroy any data in the NumberCruncher Reloaded. ▲

## REGREG Operations

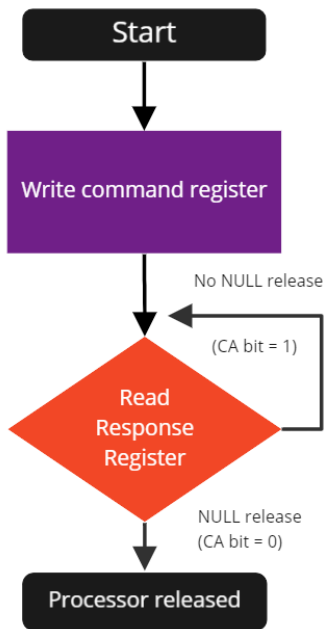


Fig.4

REGREG operations are used for operations that do not require operand data from memory to register transfers (refer to Figure 4). Examples include adding two registers (both registers having a data value), taking the sine of a value in a register or even transferring a constant value from the ROM internal to the MC6888x to a register. The sequence of operations is:

1. Write the command to the Command register (\$C0k8)
2. Check the Response register for a Null Release (\$C0k0)

Since there are no external operands, a REGREG operation does not require that the software test for a Null Response in the Response register as the MEMREG and REGMEM operations do. Once it has written the command to the Command register (with correct byte order), the software need only test the Response Register for the Null Release.

▲ **Warning:** Again, don't try to test for a non-\$8900 value as this will confuse the MC6888x and destroy any data in the FPE. ▲

Here are some Example code segments for checking the status from the NumberCruncher Reloaded. Let's start with a 65c816 version:

```

loop1    ldy #response
         lda [<mc68881],y    assumes the location containing
                             the base address of the NR-C is
                             in the direct page
         cmp #$0089         check for Null Come-Again
         beq loop1
         ldy #response
loop2    lda [<mc68881],y    check for Null Release
         and #$0080
         bne loop2

```

The same again, now for your Apple II 6502 processor:

```

loop1    ldy #response
         lda (mc68881),y    check for Null Come-Again
                             (location containing the base
                             address of the FPE is somewhere
                             in memory, location designated
                             by mc68881)
         tax
         iny
         lda (mc68881),y
         bne loop2
         cpx #$89
         beq loop1
loop2    ldy #response
         lda (mc68881),y    check for Null Release
         iny
         lda (mc68881),y    always must read upper byte
         asl a
         bcs loop2

```

For greater detail about construction of an MC6888x command, register- and operation-values, constants in ROM please refer to Appendix A, giving you more in-depth information.



# Programming

There are many ways to have your programs talk to the NumberCruncher Reloaded and thus have its floating-point performance enormously sped up.

Let's start with the lowest level, Assembly language.

**Assembly** In the previous Chapters we already saw some very low-level assembly coding. To make things easier the NumberCruncher Reloaded comes with macro library files developed by iS back in the days. These files are compatible with the APW, ORCNM, LISA816, and MERLIN assemblers. M16.FPE, in conjunction with the E16.EQU file, (for LISA816 use only M16.68881) contains macros for use with the 65816 microprocessor in the Apple IIgs. M8.FPE contains the macros for the 6502-based Apple computers. These macros are assembler specific and are contained in folders labeled for the appropriate assembler.

The macros define the command for each operation desired. You just need to supply the operation wanted, the address of the correctly formatted data, and the register(s) to use. These macros will load or retrieve the results of the operation. The general format of the macros is as follows:

## APW/ORCA/LISA816 Assembly

Memory-to-Register:

```
MEMREGv OPERATION_CODE, DESTINATION_FP_REG, DATA_ADDRESS
```

where v = precision of operation (X, D, S, L, W)

Register-to-Memory:

```
REGMEMv OPERATION_CODE, SOURCE_FP_REG, DATA_ADDRESS
```

where v = precision of operation (X, D, S, L, W)

Register-to-Register:

```
REGREG OPERATION_CODE, SOURCE_FP_REG, DESTINATION_FP_REG
```

## MERLIN Assembly

Memory-to-Register:

```
MEMREG PREC;OPERATION_CODE;DESTINATION_FP_REG;DATA_ADDRESS
```

where PREC(ission) = X, D, S, L, W

Register-to-Memory:

```
REGMEM PREC;OPERATION_CODE;SOURCE_FP_REG;DATA_ADDRESS
```

where PREC(ission) = X, D, S, L, W

Register-to-Register:

```
REGREG OPERATIOCODE, SOURCE_FP_REG, DESINATION_FP_REG
```

The source code below is an example of macro usage and shows the form for code which uses the NumberCruncher Reloaded.

```
*****
* SAMPLE TASK FOR ADDING TWO EXTENDED PRECISION
* NUMBERS, SHOWING THE USE OF MACROS
*****
      MLOAD 2/AINCLUDE/M16.UTILITY
      MLOAD M16.FPE
TEST   START
      COPY M16.FPE
MC68881 EQU $00          ; DIRECT PAGE LOCATION OF FPE
      CLC                ; BASE REGISTER
      PHK
      PLB
      STZ $00            ; ZERO DIRECT PAGE DATA
      STZ $02
      PUSHLONG LOCATION_OF FPE+2 ; PUT NC-R ADDRESS ON
      PLA                ; STACK AND STORE FPE ADDRESS IN DIRECT
      STA $00            ; PAGE
      PLA
      STA $02
A1     MEMREGX FMOVE,FP1,EXT_1 ; DATA IN REG1
A2     MEMREGX FADD,FP1,EXT_2  ; ADD 2nd VALUE
A3     MEMREGX FMOVE,FP1,ANS_1 ; GET RESULT

      RTL

LOCATION_OF_FPE DC H'C0C0 0000' ; ASSUME SLOT 4
; FLOATING POINT EXTENDED DATA AREA
EXT_1 DC H'0000 0000 0000 8000 3FFF ; Value= 1.0
EXT_2 DC H'0000 0000 0000 8000 3FFF
; RESULT SHOULD BE '0000 0000 0000 8000 4000' OR 2.0
ANS_1 DS 10
END
```

**C Language** The de-facto standard C-compiler ORCA/C already comes with FPE support, i.e. the NumberCruncher Reloaded is supported out of the box. You can either just go ahead and simply use the build-in floating-point library which in turn uses SANE . If the SANE patch INIT is installed, your C code will be accelerated automatically.

The even faster alternative is to replace the SANE-based library with one talking to the NumberCruncher Reloaded directly. To achieve this, you have to replace the “SysFloat” library located in your ORCA/C “Library” folder with that provided in the folder called “FPE”.

Additionally, you have to add these two lines to your code:

```
#pragma float 1 1
#define __FPE__ 4 /* FPE card's slot number */
```

Please refer to the ORCA/C manual for more information.

- ❖ Check <https://github.com/byteworksinc/ORCA-C> to make sure you have the latest patches installed. ❖

**Applesoft** Yes, you can put the NumberCruncher Reloaded to use in Applesoft BASIC, too!

In 1989 Glen Brendon wrote a cool so-called ProCMD module which sets up an interface between Applesoft programs. The downside is, that it uses some 65816/65802 specific commands.

- ▲ **Warning:** The interface requires an IIGs, or a IIE with a 65802 processor. The program does *not* check whether it is running in such an environment and will crash horribly if you try to use it on a machine with only a 6502 or 65C02. ▲

To use this module put the FPE file in memory by an instruction "-FPE" from Applesoft immediate mode or a PRINT CHR\$(4) "-FPE" from a program.

These things are speeded up: SIN, COS, TAN, ATN,  $X^Y$ , LOG, EXP, SQR, RND, multiplication and division. Here are some timing examples:

<b>Function</b>	<b>Applesoft</b>	<b>NC-R</b>	<b>factor</b>
SIN	102	13,9	7.34
COS	103	14.0	7.36
TAN	187	14.0	13.36
SQR	172	13.8	12.46
^	219	23.0	9.52
* and /	51.4	18.4	3.52
RND	24	12.5	2.60

On the provided disk/archive you'll find this module in the "Applesoft" folder as well as being used by the Mandelbrot generator in the "Fractal" folder.

## Programming Hints

1. If the NumberCruncher Reloaded returns \$0D1D in the response register, then the attempted operation was invalid (aka “protocol violation”).

The only way to recover, short of powering off the system, is to call SANEReset from the toolbox or to use the following code:

```
lda #0
sta FPE_restore      ; (base register + 6)
lda FPE_restore
```

Note that this is a 16-bit operation. If you are using a 6502/65C02-based system, you must do two 8-bit writes and two 8-bit reads.

- ❖ In contrast to its predecessors the NumberCruncher Reloaded has an error-LED (RED) which lights up in case of a protocol violation. The error-LED will only switch off again after a reset or power-cycle of your Apple II ❖

2. When using the FPE Toolset from Pascal, C, or Basic, save intermediate results in the extended format. Use of other formats forces the compiler to convert your data values to and from extended, operations which will increase the execution time of your programs.

3. Whenever possible, store intermediate results in the NumberCruncher Reloaded (i.e. FPU registers). Register-to-register operations can provide more than 10 times the performance of memory-to-register operations.

4. The NumberCruncher Reloaded contains five (5) ID bytes of which the first 4 are conform to the Apple standard. These bytes and their locations are:

Location	Value
\$(00)Cx05	\$38
\$(00)Cx07	\$18
\$(00)Cx0B	\$01
\$(00)Cx0C	\$AF
\$(00)Cx0D	\$Fz - Revision of firmware

where x = the slot number.

The 5<sup>th</sup> byte is a NumberCruncher Reloaded specialty and returns its firmware revision. Its “high-nibble” is always set and can be used to differ a NC-R from the FPE or original NC. The lower nibble represents the firmware revision starting with “1” – thus the first release will return \$F1.

- ❖ Before reading the data in these locations, slotROM must be enabled by writing a value to \$(00)C00B. Once done, the slot ROM must be disabled by writing a value to \$C00A.

Note that all accesses to the values should be done with the computer in 8-bit (short) index or accumulator mode. ❖

# The 68881/2

The Motorola MC68881 or 68882 used on the NumberCruncher Reloaded is the same floating point processor used in a 68020 or 68030 Apple Macintosh and many other 68k computers.

Although you may need not be concerned with specific capabilities of the NumberCruncher Reloaded, software and programmers have access to:

- Eight general purpose, 80-bit floating-point data registers.
- Forty-six instructions, including 35 arithmetic operations.
- Full ANSI-IEEE 754-1985 floating point standard.
- Enhanced functions, including a complete set of trigonometric and transcendental functions.
- Seven data formats: byte, word, and long word integers; single, double and extended precision real numbers; and packed binary coded decimal string real numbers.
- Twenty-two constants like pi, e, and powers of 10.
- Concurrent instruction execution with the Apple II.

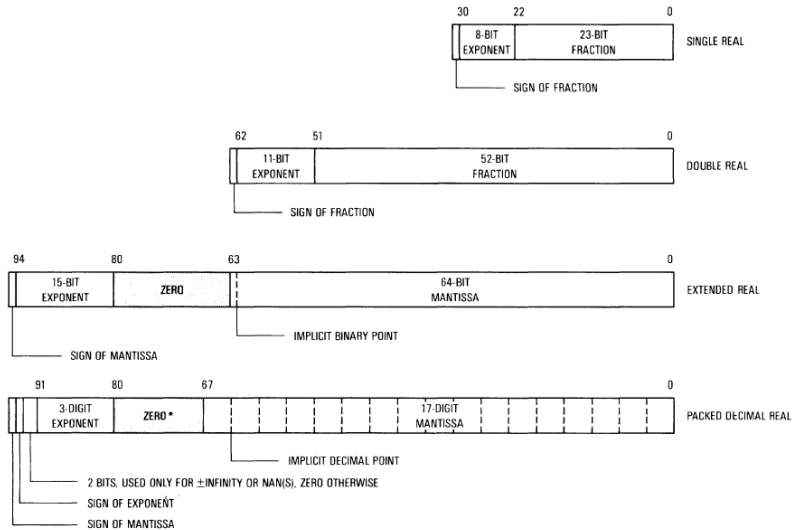
We cannot cover the complete MC6888x functionality, registers, commands and everything beyond in this little manual.

Luckily Motorola published a very detailed manual which is also available as PDF file at the NumberCruncher Reloaded webpage (or just google for it).

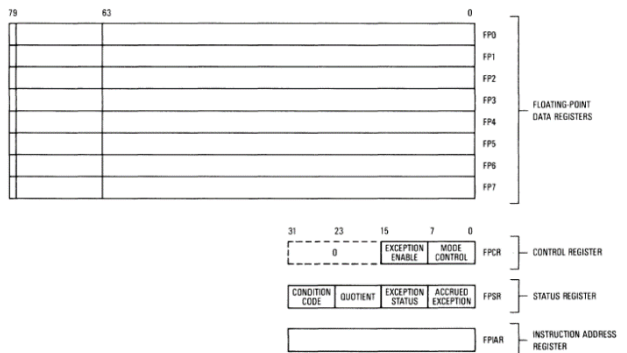
So these pages are intended to give you a brief overview what's going on behind the scenes.



**Data formats** For signed integers the 6888x FPUs have the same formats as the 68k CPU. That is: 8-bit byte, 16bit word and 32bit long. For real data 4 formats are supported: Single-, Double- and Extended-Real as well as Packed Decimal Real.



The 6888x' have eight 80bit FP data registers as well as 3 special registers all of which are extensively described in the MC68881 Manual. It is recommended to read that description thoroughly.



**Construction of an  
MC68881/68882  
Command**

Each command written directly to the floating point coprocessor Command register requires 16 bits of information. The format for the command (as seen by the MC6888x) is:

MSB														LSB	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	R/M	0	S	S	S	D	D	D	0	0	C	C	C	C	C

Where:

[R/M] Field - Specifies the source operand address mode.

0 - The operation is register to register.

1 - The operation is memory to register or register to memory

[SSS] (Source Specifier Field) - Specifies the source register or data format.

If R/M = 0, specifies the source floating point data register, FPM.

If R/M = 1, specifies the source data format:

000 L Long Word Integer (32-bits)

001 S Single Precision Real (32-bits)

010 X Extended Precision Real (96-bits)<sup>1</sup>

011 P Packed Decimal Real (96-bits)<sup>2</sup>

100 W Word Integer (16-bits)

101 0 Double Precision Real (64-bits)

110 B Byte Integer (8-bits)

[DDD] (Destination Register) - Specifies the destination floating point register, FPN.

[CCCCC] (Execution Command) - Specifies the operation to perform.

**NOTE**

1. Only 80 bits contain valid data. but 96 bits must be transferred.
2. Only 84 bits contain valid data. but 96 bits must be transferred.
3. See "MC68881/MC68882 Floating-Point Coprocessor User's Manual", pages 3-1,3-2, and 3-7 for format information)
4. All operations which input data to the FPE transfer information from the source (SSS or memory) to the destination register (DOD). This means that the source value is moved (e.g. added) to the destination register.
5. All register-to-register operations move data from the source register to the destination register (e.g. the source register is added to the contents of the destination register).

## Constructing a command step-by-step

The already mentioned library files E16.FPE and E8.FPE contain macros and definitions for the R/M and Source Specifier fields, the Destination Register field and the Execution Command field. While these macros will generate the most used commands for you, it might be interesting to know how to define a command e.g. to add an extended real number to register #1:

**First**, get the Memory-to-Register Extended Precision value from this Command Primitives definitions table:

### Register-to-Memory Movement

Single Precision	\$6400
Long Integer	\$6000
Word Integer	\$7000
Byte Integer	\$7800
Double Precision	\$7400
Extended Precision	\$6800
Packed BCD	\$6C00

### Memory-to-Register Movement

Single Precision	\$4400
Long Integer	\$4000
Word Integer	\$5000
Byte Integer	\$5800
Double Precision	\$5400
Extended Precision	\$4800 ←
Packed BCD	\$4C00

### Register -to-Register Movement

Extended Precision (only)	\$0000
---------------------------	--------

### Constant in ROM-to-Register Movement (see Table 6.14)

Extended Precision (only)	\$5C00
---------------------------	--------

### Memory-to-Control. Status or Instruction Register

Long Integer (only)                      \$0000

### Control. Status or Instruction Register-to-Memory

Long Integer (only)                      \$2000

**Second**, get the value for Floating Point Register 1 from the table below and put this value into the Destination register field (DDD, bits 7-9). The command word should now be \$4880.

Floating Point Register 0	%000
Floating Point Register 1	%001 ←
Floating Point Register 2	%010
Floating Point Register 3	%011
Floating Point Register 4	%100
Floating Point Register 5	%101
Floating Point Register 6	%110
Floating Point Register 7	%111
Control Register	\$9000
Status Register	\$8800
Instruction Address	\$8400

**Third**, put the value for the desired command into bits 0-4. From the definition file (see the following table for Operation Values), FADD equals \$22. The final command word should now be \$48A2.

△ **Important:** Remember that the word is in “reverse order” (known as big-endian) as seen from the Apple II computer. So you have to swap the data bytes. The value for the command in 6502 little-endian form is, therefore, \$A248. △

Similarly, a register 1 (SSS value) to register 2 (DDD value) add would have a final command value of "%0010001000000101" or \$2205 in Apple II little endian notation.

The 16-bit binary values for commands are given in non-Apple memory order

FMOVE	\$00	Move
FINT	\$01	Integer Part
FSINH	\$02	Hyperbolic sine
FSQRT	\$04	Square Root
FLOGNPI	\$06	$\text{LOGe}(1+X)$
FETOXM1	\$08	$((e^{**X})-1)$
FTANH	\$09	Hyperbolic tangent
FATAN	\$0A	Arctangent
FASIN	\$0C	Arcsine
FATANH	\$0D	Hyperbolic arctangent
FSIN	\$0E	Sine
FTAN	\$0F	Tangent
FETOX	\$10	$e^{**X}$
FIWOTOX	\$11	$2^{**X}$
FTENTOX	\$12	$10^{**X}$
FLOGN	\$14	Natural log
FLOG10	\$15	Log base 10
FLOG2	\$16	Binary log
FABS	\$18	Absolute Value
FCOSH	\$19	Hyperbolic cosine
FNEG	\$1A	Negate
FACOS	\$1C	Arccosine
FCOS	\$1D	Cosine
FGETEXP	\$1E	Get exponent
FGETMAN	\$1F	Get mantissa
FDIV	\$20	Divide
FMOD	\$21	Modulo Remainder
FADD	\$22	Add

FMUL	\$23	Multiply
FSGLDIV	\$24	Single precision divide
FREM	\$25	IEEE Remainder
FSCALE	\$26	Scale exponent
FSGLMUL	\$27	Single precision multiply
FSUB	\$28	Subtract
FCMP	\$38	Compare SSS with DOD
FTST	\$3A	Test
FSINCOS	\$30	Simultaneous sine and cosine *

\*) FSINCOS requires three registers, one source and two destinations, and is a register-to-register operation only. The form for this command is:

%00SSDDDD0000ddd + operation

ddd = destination 2 register (cosine value)

DDD = destination 1 register (sine value)

SSS = source register

You have reached the end of this manual.

While it was fun to read, it's static and will not change.

To get recent, ever-changing updates and more background information visit

[www.geekdot.com/numbercruncher-reloaded/](http://www.geekdot.com/numbercruncher-reloaded/)

to learn more about the **NumberCruncher Reloaded**, download software and all those things which this poor manual made from wood pulp can't do.

And again, **thank you** for your purchase!

Manual version 1.0a for NumberCruncher Reloaded v.1.0

© 2021 Axel Muhr, The Geekdot Laboratory.

Created with  at the

