This document was kindly scanned by :
Gordon Beckmann - Technical Director
gbeckmann@elitesoftware.co.uk
Elite Software Company
Thank to him for his permission to share it.

WILDCARD PLUS TECHNICAL NOTES

This documentation is copyright material and unauthorised reproduction in any form is strictly prohibited.

The information provided here is designed for machine code programmers who fully understand the operation of both the Apple // range of computers and the 65Ø2 microprocessor. It is useful for those people who wish to know more about the operation of Wildcard Plus and who are particularly interested in writing their own machine code utilities for this card. Supplied with this documentation should be an assembler listing of the Wildcard Plus ROM and a DOS 3.3 floppy disc containing files to aid in the production of utilities.

Copyright 1984 : Elite Software Company

WILDCARD PLUS - HARDWARE DESCRIPTION

Wildcard Plus contains its own 6502 microprocessor, and acts as a co-resident processor card. When the button is pressed on the Wildcard Plus it pulls the DMA line on the Apple low, thus stopping the Apple's 6502. The microprocessor is halted during the entire operation of Wildcard Plus, until the Resume command is selected.

The memory map of Wildcard Plus is as follows.

$0000 - $07FF Wildcard plus RAM includes zero page, stack. While the Wildcard is running (i.e. Apple is paused) $0400 - $07FF contains the Apple text screen image.

$0800 - $08FF A store anywhere in this block sets the 256 byte page available to WINDOW.

$0900 - $09FF WINDOW. This is a window into the Apple. It allows load and store access into a 256 byte area of the Apple address map (all 64K can be used). The high byte of the address is set by $800 (as above). Therefore to read a byte from $C000 in the Apple from the Wildcard Plus,
LDA #$C0 STA $0800 LDA $0900

Other hardware locations are not described as they should only be used via the subroutines in the Wildcard Plus ROM for reasons connected with the timing of the card.

$1000 - $1FFF Wildcard Plus ROM. See the separate information on subroutines available. It is intended that the entry points to these subroutines will not change.

An additional 2K of RAM is available for utilities. This is in the Apple address space, not the Wildcard Plus memory map. Normally it is transparent to the Apple, i.e. it appears to all software as if no card is in the slot. The RAM can be enabled using the SLOT.ONN and SLOT.OFF subroutines. When enabled the bottom 256 bytes of the RAM appears in the slot area $CN00 - $CNFF. Once the RAM has been enabled the whole 2K can be mapped into the 2K common area ($C800 - $CFFF) by reading $C080+N0 in the Apple address map, and the RAM can be removed from this area by reading $C081+N0. N is the slot number. The hardware of Wildcard Plus does not stop a memory conflict with any other card in the common area. It is up to utility software to switch out any other interface card. The files provided handle all this automatically as described in later sections.

Note that this mapping means that the RAM in areas $CN00 - $CNFF and $C800 - $C8FF is the same.

See the separate note on loading and using utility software.

RAM LOCATIONS

A Study of the listing of the Wildcard Plus ROM will reveal that many locations used in the Wildcard memory map are well described by their name. A list of some of the more useful locations follows.

WC.RAM	$00,$01	Power on checksum. Do not use.
WC.ARROW	$02	Can be used by utilities.
WC.LC	$04	Contains zero if no 16K card.
WC.80COL	$05	Not zero if //e 80 col card installed.
WC.64K	$06	Not zero if card is extended.
WC.A	$20	Apple 6502 accumulator value at time of Wildcard Plus being activated. Other registers follow.
WC.TEMP.ZP	$3B	Can be used by utilities, but may be used by any subroutine in the ROM at will.
WC.PTR	$50,51	General purpose pointer. May be used by ROM subroutines. Check the listing.
UTILITY	$55	Contains slot number of Wildcard Plus only if a utility is installed, otherwise zero. If a utility sets this to zero it effectively destroys itself.
	$100-$1FF	Stack. Exit code resides here if using the supplied utility files. Utilities may call subroutines with impunity, but otherwise do not use page 1.
	$0200-$07FF	This RAM area is completely unused by the ROM subroutines (except the disc access ones, but if you want to use these please try and work out the listing), and can be used by utilities for code, buffers etc.

Other areas and locations may be safe to use, or they may not. Check in the listing any subroutines that you call, and if it works, it's right.

WILDCARD PLUS - ROM SUBROUTINES

MENU.WILDCARD $157A

This is the main options menu of the Wildcard Plus.

CLEAR $17FE

Clear all of text screen 1 except the top two lines. This preserves the WILDCARD PLUS header. The copy of the text screen made in the Wildcard Plus RAM is not altered; thus the interrupted program still resumes correctly. This routine is used to clear the screen for Wildcard Plus display.

PAGE.CØ $17ØØ

Point WINDOW ($Ø9ØØ) at page $CØ in the Apple memory map. This is useful to access the keyboard, soft switches etc..

BASCALC $1885

If the accumulator contains the number of a valid line on the text screen, this routine sets WINDOW to point to the high part of the address and stores in WC.TEMP.ZP ($3B) the low part of the address of the start of the line. The low part is also returned in the accumulator. Thus to plot an A at the fourth position in line 8,
LDA #$Ø8 JSR BASCALC CLC ADC #$Ø4 TAY LDA #'A' STA WINDOW,Y

GET.KEY $189E

Get a key from the keyboard in the accumulator. Letters are forced to be upper case. The key is returned with the top bit set.

FORCE.UC $18A9

If the accumulator contains a lower case letter make it upper case.

SLOT.OFF $1ADB

Disable the utility RAM area. Use this routine; do not access the hardware switches directly. After calling this routine the 2K utility RAM area can not be accessed by the Apple.

SLOT.ONN $1BE4

Enable the utility RAM area. Use this routine; do not access the hardware switches directly. After calling this routine 256 bytes of the utility RAM is mapped into slot space, with the option of accessing all 2K in the C8ØØ common area.

CREATING UTILITY SOFTWARE

There are three separate problems in creating utility software for Wildcard Plus. They are: loading the software into the Wildcard, starting the software running and ensuring that it runs correctly when it has started. These will be discussed individually.

LOADING THE UTILITY SOFTWARE

Utility code is loaded into the Wildcard Plus from the Wildcard menu by selecting the U option. This checks the contents of a zero page location in the Wildcard map called UTILITY ($55) and, if that location contains zero takes the code in page 3 ($300 - $3FF) in the Apple memory, transfers it to page 3 in the Wildcard and executes it. It is the task of this code to perform the move of the utility into the Wildcard utility RAM. It must also set UTILITY to contain the slot number of the Wildcard so that next time the U option is selected the Wildcard does not attempt to load more utility software.

Fortunately all the hard work is done for you. Provided with this package is a disc containing various files. One of these is a binary file called UL. This can be appended to any utility and performs all the loading functions. It occupies none of the 2K utility space; after the utility is loaded this loading code disappears. How to merge it with your utilities is described later.

RUNNING A UTILITY

We recommend that a utility is run by transferring the code from the utility RAM into the main Wildcard Plus RAM, where it can be executed in Wildcard space. This is not the only way to run a utility. If you decide to adopt a different approach good luck!

If you choose the recommended method the disc supplied contains more routines to aid you. They are all source files supplied in S-C Macro Assembler format.

The only file you should ever assemble is A.U. It is the main controlling file, and will load up other source files (including the utilities you write) and merge them. The other files mentioned below are all used by A.U.

U.CONSTANTS contains all the locations used in the Wildcard memory space, and all locations used by the other routines supplied. Because this file is included you do not need to worry about declaring any ROM subroutines or Wildcard Plus RAM locations you use.

U.CN00 is a file which will occupy 256 bytes of the utility space. It is responsible for loading the rest of the utility code into the Wildcard and it preserves the Wildcard's copy of the Apple text screen by storing it where the utility code used to be.

U.EXIT contains the code which replaces the screen in the Wildcard RAM and returns to the Wildcard menu. It also occupies 256 bytes and is included automatically by A.U. The correct way to exit a utility is described in the next section.

RUNNING UTILITY SOFTWARE

Now for the fun bit. It's time to acually produce a working utility. Your utility code resides in normal Wildcard plus RAM in the area $Ø2ØØ - $Ø7FF. To exit a utility all you need do is JMP EXIT.UTILITY . This will correctly restore the Wildcard copy of the text screen and return to the menu with everything intact.

When you have written your utility code save the source file on a COPY of the supplied disc, and alter A.U by adding at line 1245

```
1245          .IN YOUR.FILE
```

where YOUR.FILE is the name of your source file. Assemble A.U and it will create a binary called U. To produce a final utility type

```
BLOAD U,A$28ØØ
BLOAD UL
BSAVE YOUR.NAME,A$25ØØ,L$BØØ
```

where YOUR.NAME is the name of the utility you want. We suggest that you call all the utilities UTILITY.######### (######### is the name of your choice) so that they can be recognised.

EXAMPLES

The easiest way to demonstrate how to write a utility is with an example. By pure coincidence there just happen to be two example utilities on the disc supplied. Neither of then do anything useful, but they show the principles involved. They are called U.DEMO1 and U.DEMO2.

U.DEMO1 displays a message on the screen and returns to the Wildcard menu only when ESC is pressed.

U.DEMO2 takes the contents of high res screen one and copies it to high res screen two.

Start by listing out both demonstrations source files. If they are commented as well as we believe their operation should be obvious. Try assembling them by including them in A.U and combining the binary with UL as described above. They should work correctly. From now on you are on your own.

One final warning; once a utility is in the Wildcard it is there for good unless the utility itself sets the location UTILITY to zero. Until this happens pressing U from the Wildcard menu will run the utility and not load it. Happy programming.

SUPPORT

You are not completely on your own. We will try and help if we can with any problems. Please remember that for most non-trivial problems all we can do is look through the ROM listing for the answer, which you can try just as well. Just because we produced the product doesn't mean we understand how it works.

It you do telephone with a problem you will not receive an immediate answer. The programmers live two thousand light years from the nearest telephone and all communication with them has to take place via hyper-spacial warp. It is better if you write describing the help you need exactly and enclosing any relevant source listings. We will reply, even if all we say is that we cannot help. Any reply you receive should contain information on our entire product range. Please read it. If there is anything there that you want we need the business.

THOUGHT FOR THE DAY

Real programmers don't comment their code.
If it was hard to write it should be hard to understand.