1

Using the Wildcard Carrier Board

The Wildcard™ Carrier Board provides an interface between the QED Flash Board and up to eight Wildcard Modules. Wildcard Modules are small (2.5"x2.0"), stackable, and combinable modules that provide custom off-the-shelf solutions for your sensor measurement or real-time control system.

This document describes the capabilities of the Wildcard Carrier Board and presents an overview of the software drivers. A glossary of the software functions and complete hardware schematics are also included.

Wildcard Carrier Board Specifications	
Capacity:	Stacks up to eight Wildcard I/O modules, four on each of two Wildcard ports.
Memory:	Standard: 128K RAM & 128K Flash Expanded: 512K RAM & 512K Flash
Power:	Efficient 5V switching regulator provides up to x A of power to Wildcard boards.

Wildcard Carrier Board Hardware

The Wildcard Carrier Board comprises two Wildcard Port Connectors, a QED Digital I/O Connector, a QED Address/Data Connector, digital logic circuitry, switching power regulator, 128K RAM, and 128K flash memory. Jumpers are provided to disable the write protection of the flash memory and to select the power source of the Wildcard Carrier Board. The QED Digital I/O Connector and QED Address/Data Connector provide the interface to the host QED Flash Board while the Wildcard Module Port Connectors provide the interface to the Wildcard Modules.

Connecting to the QED Flash Board

To connect the Wildcard Carrier Board to the QED Flash Board or Panel-Touch Controller, follow these simple steps:

1. Setup and configure the QED Board as outlined in the document entitled "Getting Started with the QED Board".

2. With the power off, stack the Wildcard Carrier Board directly on the top side (the side with the QED-FORTH kernel ROM chip) of the QED Flash Board. **WARNING: There are three 40 pin sockets on the QED Board. Be sure to insert the QED Address/Data Connector on the Wildcard Carrier Board into the Address/Data Connector on the QED Board and the QED Digital I/O Connector on the Wildcard Carrier Board into the Digital I/O Connector on the QED Board.**

   CAUTION: The Wildcard Carrier Board does not have keyed connectors. Be sure to insert the Wildcard Carrier Board so that all pins are connected. The Wildcard Carrier Board and the QED Flash Board can be permanently damaged if the connection is done incorrectly.

Installing Wildcard Modules

The Wildcard Carrier Board can host as many as eight Wildcard Modules. Up to four modules can be stacked on each of the two module port connectors. A pair of jumpers (called the Module Select Jumpers), available on each module, specifies a 2-bit code that selects a unique address on the module port of the Wildcard Carrier Board. Module Port 0 provides access to modules 0–3 while Module Port 1 provides access to modules 4–7. Two modules on the same port cannot have the same address (jumper settings). Table 1-1 shows the possible jumper settings and the corresponding addresses.

Table 1-1 Jumper Settings and Associated Addresses

Module Port	Module Address	Installed Jumper Shunts
0	0	None
	1	J1
	2	J2
	3	J1 and J2
1	4	None
	5	J1
	6	J2
	7	J1 and J2

Power Regulation

The default configuration of the Wildcard Carrier Board uses an efficient 5-Volt on-board switching power regulator. The switching power regulator circuitry regulates the V+raw signal coming from the QED Board to power the Wildcard Carrier Board and supply up to 1 Amp of current for any attached Wildcard Modules. Typical modules use 85 mA of current. The V+raw input supply must be in the range of 7 to 14 VDC for the on-board switching regulator and power supply circuitry on the QED Board to work properly. A jumper across the two pins labeled “Int+5” (meaning Internal regulation for +5V) on JP1 selects the on-board switching power regulator. JP1 is located on the left edge of the Wildcard Carrier Board between the JTAG header and Module Port 1.

Alternatively, the Wildcard Carrier Board can be powered by the QED Board’s regulated +5 volt supply by simply moving the jumper across the two pins labeled “Ext+5” (meaning External regulation for +5V) on JP1. The Wildcard Carrier Board then receives its power from the 5-volt digital supply signal from

the QED Board. Use this option if no V+Raw supply is available (i.e. the QED Board is powered using a separate, regulated 5V supply). Do not configure the Wildcard Carrier Board to use the Ext+5 option if any of the Wildcard Modules in use require the V+Raw signal (such as the 24/7 Data Acquisition Module and the Analog I/O Module).

RAM

The Wildcard Carrier Board can access up to 512 Kbytes (sixteen 32 Kbyte pages) of on-board RAM. The RAM can be read and written but will not retain its contents when power is removed. 128 Kbytes (four 32 Kbyte pages) is the default amount that is installed on the board. Using the C Programming Language, data can be read from RAM using the kernel functions: `FetchChar()`, `FetchInt()`, `FetchLong()`, `FetchFloat()`, and `CmoveMany()`. Alternatively, data can be written to RAM using `StoreChar()`, `StoreInt()`, `StoreLong()`, `StoreFloat()`, and `CmoveMany()`. Using the Forth Programming Language, the RAM can be read via: `C@`, `@`, `2@`, `X@`, `F@`, and `CMOVE`; data can be written via `C!`, `!`, `2!`, `X!`, `F!`, and `CMOVE`. Consult the respective C and Forth glossaries for descriptions of these functions.

Flash

Flash memory is typically used to hold program code, constants, and data that does not need to be updated in real time. Flash memory retains its contents when power is removed. The Wildcard Carrier Board can access up to 512 Kbytes of on-board flash. 128 Kbytes is the default amount that is installed on the board. Flash memory is read using the standard memory operators listed in the prior section.

The following section describes the pre-coded software drivers that allow you to write to the on-board flash memory.

The on-board flash can be write-protected by removing the jumper across JP2. Jumper JP2 is located between the QED Address/Data Bus Connector and the flash memory chip.

Software

Pre-coded device drivers are provided to make it easy to use the flash memory on the Wildcard Carrier Board. This code is available as a pre-compiled “kernel extension” library to C and Forth programmers.

Overview of the Software Device Drivers

The function used to write to the flash memory on the Wildcard Carrier Board is `To_XFlash`. This function is similar to the `To_Flash` function used by kernel to program the flash memory on the QED Flash Board except it supports a wider range of flash memory types and sizes. Two other functions, `XDownload_Map` and `XStandard_Map`, allow you to compile and run code from the flash memory on the Wildcard Carrier Board.

Wildcard Carrier Board Flash Memory Map

Because code cannot be downloaded or compiled directly into flash memory on the Wildcard Carrier Board, the Wildcard Carrier Board flash memory map implements “page swapping” to provide a mechanism for getting the compiled code into the flash memory. There are two page-swap modes: one is called

the XStandard Map and the other is called the XDownload Map. As the names suggest, the XStandard Map is used during run-time, and the XDownload Map is used during downloading and compilation of Forth source code or C-Compiler S records from the PC to the QED Board. The two maps are very similar; the effect of changing from the XStandard to the XDownload map is to "swap" the locations of pages 0x50 to 0x5F between flash and RAM.

In the XStandard Map with the default 128K RAM and 128K Flash installed, the RAM starts at 0x0000 on page 0x40 and extends to address 0x7FFF on page 0x43. The flash memory starts at 0x0000 on page 0x50 and extends to address 0x7FFF on page 0x53. With 512K RAM and 512K Flash installed, the RAM starts at 0x0000 on page 0x40 and extends to address 0x7FFF on page 0x4F. The flash memory starts at 0x0000 on page 0x50 and extends to address 0x7FFF on page 0x5F.

In the XDtandard Map with the default 128K RAM and 128K Flash installed, the RAM starts at 0x0000 on page 0x50 and extends to address 0x7FFF on page 0x53. The flash memory starts at 0x0000 on page 0x40 and extends to address 0x7FFF on page 0x43. With 512K RAM and 512K Flash installed, the RAM starts at 0x0000 on page 0x50 and extends to address 0x7FFF on page 0x5F. The flash memory starts at 0x0000 on page 0x40 and extends to address 0x7FFF on page 0x4F. Table 1-2 summarizes the XDownload Map and the XStandard Map for both 128K RAM/128K Flash and 512K RAM/512K Flash configurations.

Table 1-2 Wildcard Carrier Board Flash Memory Map. Italisized values describe the Memory Map with 512K Flash and 512K RAM installed (the default memory size is 128K Flash and 128K RAM).

XStandard Map		**XDownload Map**	
Flash Page	RAM Page	Flash Page	RAM Page
0x50	0x40	0x40	0x50
0x51	0x41	0x41	0x51
0x52	0x42	0x42	0x52
0x53	0x43	0x43	0x53
0x54	*0x44*	*0x44*	*0x54*
0x55	*0x45*	*0x45*	*0x55*
0x56	*0x46*	*0x46*	*0x56*
0x57	*0x47*	*0x47*	*0x57*
0x58	*0x48*	*0x48*	*0x58*
0x59	*0x49*	*0x49*	*0x59*
0x5A	*0x4A*	*0x4A*	*0x5A*
0x5B	*0x4B*	*0x4B*	*0x5B*
0x5C	*0x4C*	*0x4C*	*0x5C*
0x5D	*0x4D*	*0x4D*	*0x5D*
0x5E	*0x4E*	*0x4E*	*0x5E*
0x5F	*0x4F*	*0x4F*	*0x5F*

Software Development Using Flash Memory

The Wildcard Carrier Board vastly expands the amount of code storage of the QED Flash Board by adding up to 512k of Flash and 512K of RAM. The additional memory is useful for code, graphics, and data storage. For techniques on storing code into the additional memory on the Wildcard Carrier Board, please call Mosaic Industries at (510) 790-1255.

Installing the Wildcard Carrier Board Driver Software

The Wildcard Carrier Board device driver software is provided as a pre-coded modular runtime library, known as a "kernel extension" because it enhances the on-board kernel's capabilities. The library functions are accessible from C and Forth.

Mosaic Industries can provide you with a web site link that will enable you to create a packaged kernel extension that has drivers for all of the hardware that you have on your system. In this way, the software drivers are customized to your needs, and you can generate whatever combination of drivers you need. Make sure to specify the Wildcard Carrier Board Driver in the list of kernel extensions you want to generate, and download the resulting "packages.zip" file to your hard drive.

For convenience, a separate pre-generated kernel extension for the Wildcard Carrier Board is available from Mosaic Industries on the Demo and Drivers media (diskette or CD). Look in the Drivers directory, in the subdirectory corresponding to your hardware (QED, PanelTouch, or EtherSmart), in the Wildcard_Carrier_Board folder.

The kernel extension is shipped as a "zipped" file named "packages.zip". Unzipping it (using, for example, winzip or pkzip) extracts the following files:

- readme.txt - Provides summary documentation about the library.
- install.txt - The installation file, to be loaded to COLD-started QED Board.
- library.4th - Forth name headers and utilities; prepend to Forth programs.
- library.c - C callers for all functions in library; #include in C code.
- library.h - C prototypes for all functions; #include in extra C files.

Library.c and library.h are only needed if you are programming in C. Library.4th is only needed if you are programming in Forth. The uses of all of these files are explained below.

We recommend that you move the relevant files to the same directory that contains your application source code.

To use the kernel extension, the runtime kernel extension code contained in the install.txt file must first be loaded into the flash memory of the QED Board. Start QEDTerm with the QED board connected to the serial port and turned on. If you have not yet tested your QED board and terminal software, please refer to "Getting Started with the QED Board". Once you can hit enter and see the 'ok' prompt returned in the terminal window, type

```
COLD
```

to ensure that the board is ready to accept the kernel extension install file. Use the "Send File" menu item of the terminal to download the install.txt to the QED Board or Panel-Touch Controller.

Now, type

```
COLD
```

again and the kernel has been extended! Once install.txt has been loaded on a given board, it does not have to be reloaded (even if you revise your source code).

Using the Driver Code with C

Move the library.c and library.h files into the same directory as your other C source code files. After loading the install.txt file as described above, use the following directive in your source code file:

```
#include "library.c"
```

This file contains calling primitives that implement the functions in the kernel extension package. The library.c file automatically includes the library.h header file. If you have a project with multiple source code files, you should only include library.c once, but use the directive

```
#include "library.h"
```

in every additional source file that references the flash writing function.

Note that all of the functions in the kernel extension are of the _forth type. While they are fully callable from C, there are two important restrictions. First, _forth functions may not be called as part of a parameter list of another _forth function. Second, _forth functions may not be called from within an interrupt service routine unless the instructions found in the file named

\fabius\qedcode\forthirq.c

are followed. Also, the routines `To_XFlash`, `XDownload_Map`, and `XStandard_Map` are not reentrant. This means that you should not call this routine from multiple tasks unless you define a resource variable for writing to the flash memory.

NOTE: If your compiler was purchased before June 2002, you must update the files, qlink.bat and qmlink.bat in your \fabius\bin directory on your installation before using the kernel extension. You can download a zip file of new versions at

http://www.mosaic-industries.com/Download/new_qlink.zip

The two new files should be placed in c:\Fabius\bin. This upgrade only has to be done once for a given installation of the C compiler.

Using the Driver Code with Forth

After loading the install.txt file and typing `COLD`, use the terminal to send the "library.4th" file to the QED Board. Library.4th sets up a reasonable memory map and then defines the name header used by the Wildcard Carrier Board kernel extension. Library.4th leaves the memory map in the download map.

After library.4th has been loaded, the board is ready to receive your high-level source code files. Be sure that your software doesn't initialize the memory management variables DP, VP, or NP, as this could cause memory conflicts. If you wish to change the memory map, edit the memory map commands at the top of the library.4th file itself. The definitions in library.4th share memory with your Forth code, and are therefore vulnerable to corruption due to a crash while testing. If you have problems after reloading your code, try typing `COLD`, and reload everything starting with library.4th. It is very unlikely that the

kernel extension runtime code itself (install.txt) can become corrupted since it is stored in flash on a page that is not typically accessed by code downloads.

We recommend that your source code file begin with the sequence:

```
WHICH.MAP 0=
IFTRUE 4 PAGE.TO.RAM  \ if in standard.map...
       5 PAGE.TO.RAM
       6 PAGE.TO.RAM
      DOWNLOAD.MAP
ENDIFTRUE
```

This moves all pre-loaded flash contents to RAM if the QED Board is in the standard (flash-based) memory map, and then establishes the download (RAM-based) memory map. At the end of this sequence the QED Board is in the download map, ready to receive additional code.

We recommend that your source code file end with the sequence:

```
4 PAGE.TO.FLASH
5 PAGE.TO.FLASH
6 PAGE.TO.FLASH
STANDARD.MAP
SAVE
```

This copies all loaded code from RAM to flash, and sets up the standard (flash-based) memory map with code located in pages 4, 5 and 6. The `SAVE` command means that you can often recover from a crash and continue working by typing `RESTORE` as long as flash pages 4, 5 and 6 haven't been rewritten with any bad data.

Glossary

This glossary defines the `To_XFlash`, `XDownload_Map`, and `XStandard_Map` routines.

Overview of Glossary Notation

The keyword name of each glossary entry presented in this document is in **bold** typeface. Each function is listed with both a C-style declaration and a Forth-style stack comment declaration as described below. The "C:" and "4th:" tags at the start of the glossary entry distinguish the two declaration styles.

The Forth language is case-insensitive, so Forth programmers are free to use capital or lower case letters when typing keyword names in their program. Because C is case sensitive, C programmers must type the keywords exactly as shown in the glossary.

Each glossary entry starts with C-style and Forth-style declarations, and presents a description of the function.

The C declaration specifies that return data type before the function name, and lists the comma-delimited input parameters between parentheses, showing the type and a descriptive name for each.

The Forth declaration contains a "stack picture" between parentheses; this is recognized as a comment in a Forth program. The items to the left of the double-dash (--) are input parameters, and the item to the right of the double-dash is the output parameter. Forth is stack-based, and the first item shown is lowest

on the stack. The backslash (\ ) character is read as "under" to indicate the relative positions of the input parameters on the stack. In the Forth declaration, the parameter names and their data types are combined. All unspecified parameters are 16-bit integers. Forth promotes all characters to integer type.

The presence of both C and Forth declarations is helpful: the C syntax shows the types of the parameters, and the Forth declaration provides a descriptive name of the output parameter.

Glossary Entries

C: int **To_XFlash** (xaddr src_xaddr, xaddr dest_xaddr, uint num)
4th: **To_XFlash** (src_xaddr \ dest_xaddr \ num -- success_flag)
Transfers num bytes (0 <= num <= 65,535) starting at the specified source extended address, to the specified destination extended address in flash. The source may be anywhere in memory; it may even be in the flash that is being programmed. The destination must be in flash. Returns a flag equal to –1 if the programming was successful, or 0 if the programming failed. Reasons for failure include the write enable jumper is not installed or a destination that is not a programmable page in flash memory (page 0x60 for example). If any location in the flash memory is programmed more than 10,000 times, the cells may wear out causing a failure flag to be returned. If the standard 128K flash is installed on the Wildcard Carrier Board, writable flash pages include 0x50, 0x51, 0x52, 0x53. This function uses the 68HC11's on-chip RAM at hex B200 to B3CF to manage the write to the flash (the real time clock and C/Forth interrupt stack reserve the bytes at B3D0 to B3FF). The remaining on-chip RAM at B000 to B1FF remains available to the user. This routine is not re-entrant with respect to multitasking. This means that a multitasking application cannot support simultaneous flash programming by separate tasks unless a resource variable is defined and GET and RELEASE are used. This function can program flash memory on the Wildcard Carrier Board and/or the kernel on the QED Flash Board (pages 4,5,6,7, and 0x0D in the standard memory map).

C: void **XDownload_Map** (void)
4th: **XDownload_Map** (--)
Changes the state of the onboard CPLD to put the download memory map into effect on the flash-equipped Wildcard Carrier Boards. After execution of this routine, pages 0x50, 0x51, 0x52, and 0x53 are addressed to the onboard RAM, and pages 0x40, 0x41, 0x42, and 0x43 are addressed to the onboard flash (for the standard Wildcard Carrier Board memory configuration of 128K RAM and 128K Flash). This allows code (and Forth names) to be compiled into RAM on pages 0x50, 0x51, 0x52, and 0x53 and then transferred to flash using the `To_XFlash` function. To establish the standard memory map, see the glossary entry for `XStandard_Map`. Unlike the memory map on the QED Flash Board, a reset does not change the memory map state of the Wildcard Carrier Board and a power cycle sets the memory map to the standard map.

C: void **XStandard_Map** (void)
4th: **XStandard_Map** (--)
Changes the state of the onboard CPLD to put the standard memory map into effect on the flash-equipped Wildcard Carrier Boards. After execution of this routine, and upon each subsequent power cycle, pages 0x40, 0x41, 0x42, and 0x43 are addressed to the onboard RAM, and pages 0x50, 0x51, 0x52, and 0x53 are addressed to the onboard flash (for the standard Wildcard Carrier Board memory configuration of 128K RAM and 128K Flash). After code is downloaded to RAM and transferred to flash using the `To_XFlash` function, establishing the standard map allows code resident on pages 0x50, 0x51, 0x52, and 0x53 to be executed. To establish the download memory map, see the glossary entry for `XDownload_Map`. Unlike the memory map on the QED Flash Board, a reset does not

change the memory map state of the Wildcard Carrier Board and a power cycle sets the memory map to the standard map.

Schematics

Figure 1-2 On-Board Flash and RAM

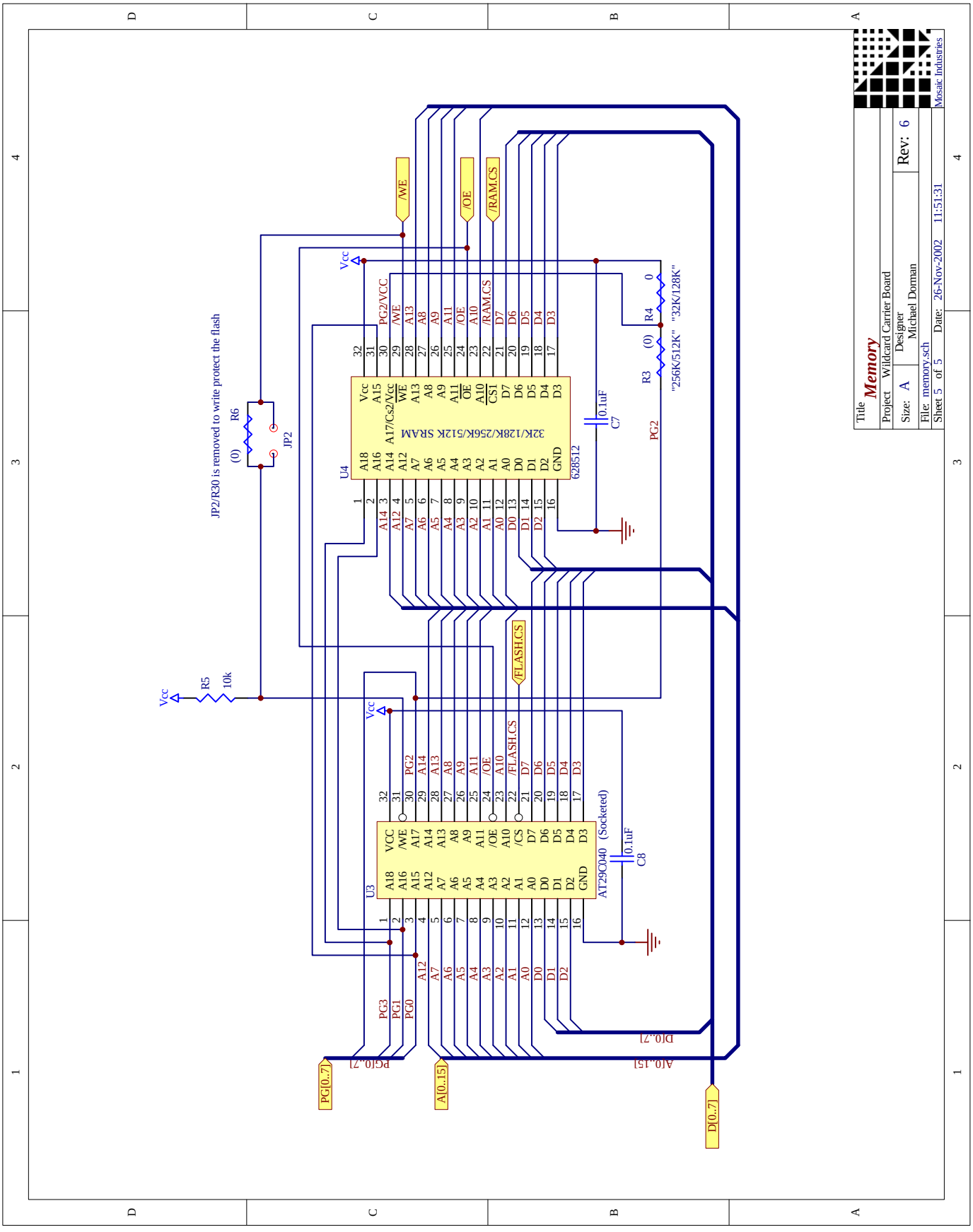

Figure 1-2 On-Board Flash and RAM

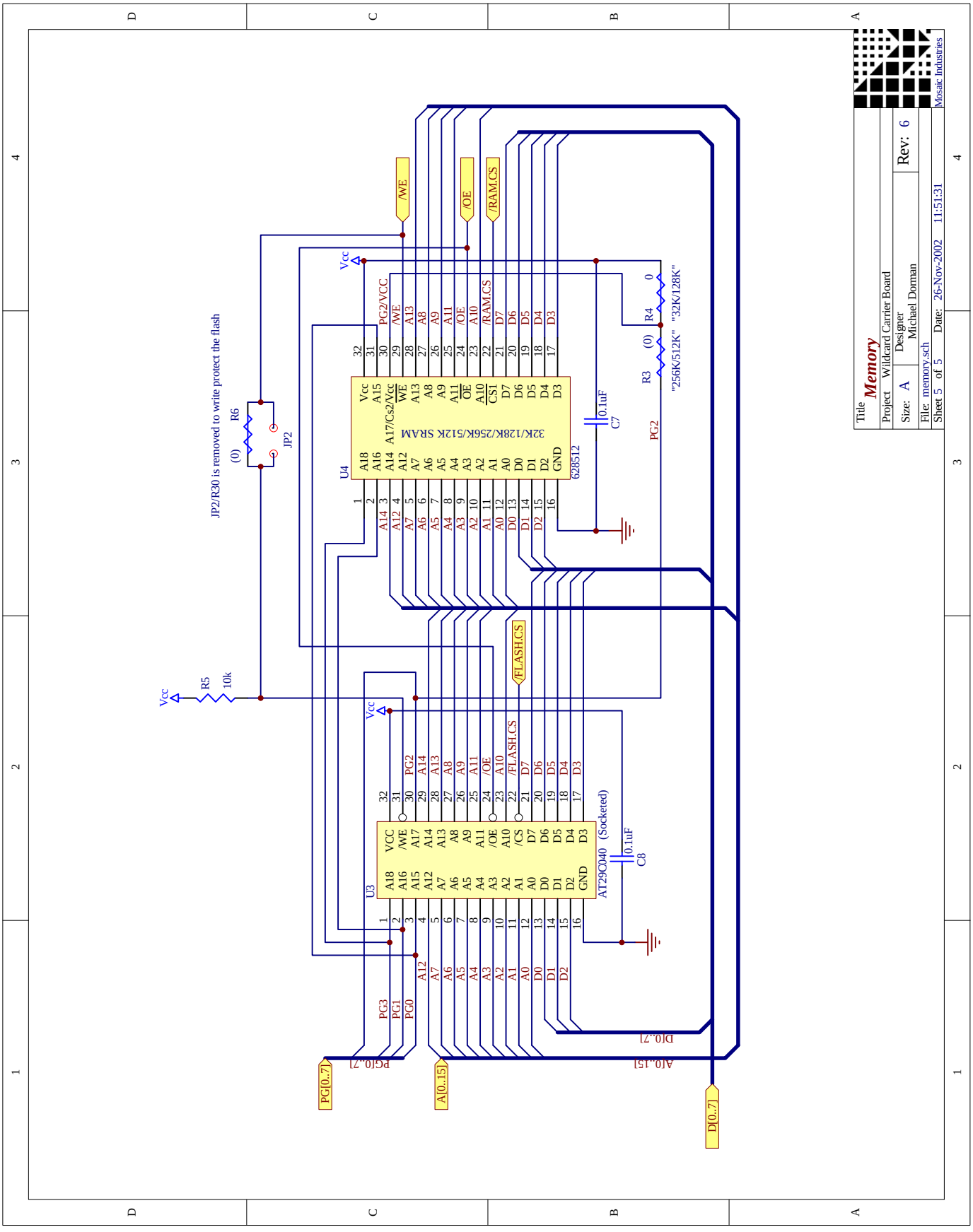

Figure 1-3 Wildcard Module Port Connectors

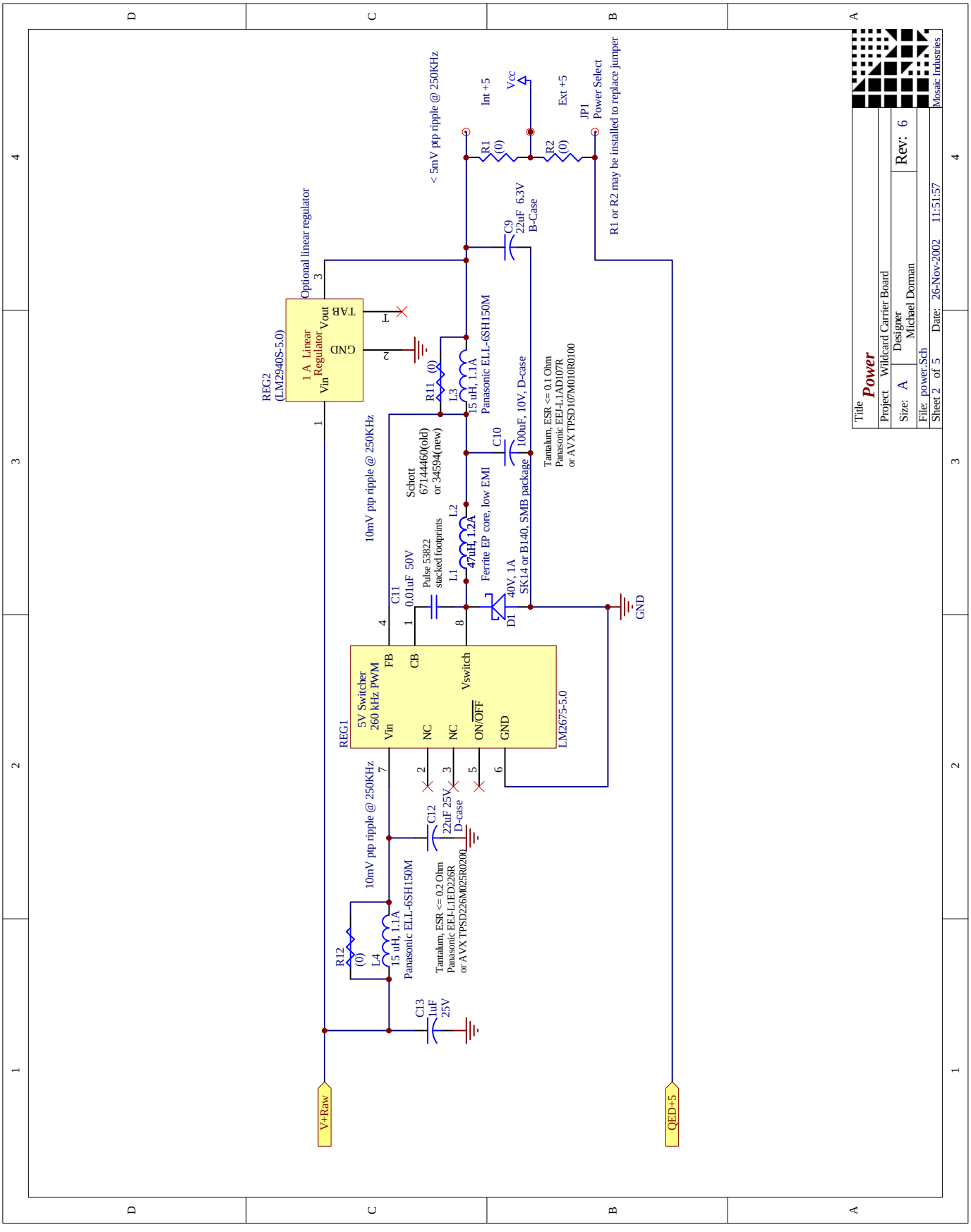

Figure 1-4 Switching Regulator Power Supply