Mosaic Industries

QED Digital I/O Board

APPLICATION NOTE
MI-AN-024

Summary

This application note describes how to use the QED Digital I/O Board. It describes the hardware capabilities of the board, presents coded software drivers in Forth and ANSI C, and includes complete hardware schematics.

QED Digital I/O Board Hardware Overview

The QED Digital I/O Board adds memory-mapped I/O to the QED Board. It provides 2 relay outputs, 8 opto-isolated high-current MOSFET outputs, and 8 opto-isolated high-voltage inputs. CMOS/TTL-compatible signals corresponding to the 8 outputs and 8 inputs are brought out to a logic level I/O connector with a pinout that is compatible with standard industrial I/O modules. Up to four Digital I/O Boards can be interfaced to the address/data bus of each QED Board embedded computer. CAUTION: Do not connect this board to the Digital I/O connector on the QED Board.

There are four connectors on the QED Digital I/O Board:

- A 40 pin "address/data bus header" connects to the QED Board Addr/Data connector.
- A 40 pin "field header" provides connections to two onboard relays, 8 opto-isolated inputs, 8 opto-isolated outputs, and their associated field grounds and snubber voltages.
- A 34 pin "logic level header" provides connections to 8 TTL/CMOS inputs and 8 TTL/CMOS outputs. This header provides a direct connection to standard 16-channel industrial AC and DC input/output modules made by Opto-22, Grayhill and others.
- A 3 pin screw terminal connector provides access to the input power terminals.

The Digital I/O Board can be powered directly from the QED Board via the V+raw signal on the address/data bus or from an external regulated 5 VDC or unregulated 6-25 VDC power source.

Memory Mapping And Page Select Jumpers

The QED Digital I/O Board connects to the QED board via the address/data connector and adds memory-mapped inputs and outputs[1]. The QED Board's 8 megabyte memory space comprises 256 pages. Each Digital I/O Board is mapped onto a 32 kilobyte page in the QED address space.

Configuration of page selection jumpers on the Digital I/O Board allows each board to be mapped to one of four pages: DC_H, DD_H, DE_H, or DF_H, where the $_H$ subscript indicates a hexadecimal number. The QED Board accesses the inputs and outputs on the Digital I/O Board by simply reading and writing to the appropriate addresses on the specified page.

The page is set by jumpers J0 and J1 which are located between the regulator and the power connector on the Digital I/O Board. Up to four Digital I/O Boards with unique page addresses can be interfaced to a single QED Board. Figure 1 shows the relationship between the jumper settings and the corresponding page in the QED Board's memory space.

J1	J0	Page
0	0	DC
0	1	DD
1	0	DE
1	1	DF

Figure 1. Jumper Settings For Page Mapping. A 1 indicates that the jumper is installed, and a 0 indicates that the jumper is not present.

The software presented below shows how to use a single Digital I/O Board configured for page DF_H. If you are going to use this software without modification, check that jumpers J0 and J1 are both installed so that the Digital I/O Board is mapped onto page DF_H.

Accessing the I/O by Reading and Writing to Specified Addresses

Only 4 bytes (at addresses 0000, 0001, 0002 and 0003) on the specified page are required to control all of the functions on the Digital I/O Board. Figure 2 summarizes the effect of reading from and writing to these control addresses.

[1] For the curious, the "Memory Mapped I/O" chapter in the QED Hardware Manual provides detailed information on how memory mapping is accomplished.

Address	Read/Write	Action
0000	Read	Relay 0 coil is de-energized; contacts are open
0000	Write	Relay 0 coil is energized; contacts are closed
0001	Read	Relay 1 coil is de-energized; contacts are open
0001	Write	Relay1 coil is energized; contacts are closed
0002	Read	Extra decoded output (pin 17) latches low
0002	Write	Extra decoded output (pin 17) latches high
0003	Read	Read the 8-bit input port and place results on the data bus
0003	Write	Write the contents of the data bus to the 8-bit output port

Figure 2. This table shows the action that results from reads and writes to the four addresses on the specified Digital I/O Board page as set by the jumpers J0 and J1. The "extra decoded output" is brought out to a through-hole on the board labeled EXTRA between the logic connector and the Address/Data connector.

Reading address 0003 on the specified page reads the input port. Thus a C assignment or a Forth C@ (character fetch) statement can be used to acquire the input data. Writing to address 0003 on the specified page writes to the output port. Thus a C assignment or a Forth C! (character store) statement can be used to output the data. When controlling the onboard relays, the act of reading or writing the specified address actuates the relay, and the actual data present on the data bus is immaterial.

It is important to note that once a signal is sent to the outputs, it cannot be read back from the same address. The software drivers presented later in this document are constructed to store the prior outputs in a RAM variable so that this factor does not limit performance in any way.

Powering the Digital I/O Board

The power connector on the QED Digital I/O Board allows you to connect a regulated +5 volt DC supply or an unregulated 6-25 VDC supply. Alternatively, the QED Digital I/O Board can be powered directly from the QED board by installing jumper J2 which connects Vin on the QED Digital I/O Board to V+Raw on the QED Board. The Digital I/O Board is typically shipped with jumper J2 installed.

CAUTION: If a power source is supplied to the QED Digital I/O Board at Vin, make sure that J2 is removed! Otherwise damage to the boards and/or the power supplies may occur.

Output Status After a Power-up or Reset

After a power-up or hardware reset, the relay coils are de-energized and the relay contacts are open. The MOSFET outputs are off (not conducting load current), and the active-low output signals at the logic level header are in their inactive states at +5 volts.

Relays

The two onboard relays are named Relay 0 and Relay 1. Each Single Pole Single Throw Normally Open (SPST-NO) relay is capable of handling 3 A at up to 30 VDC or 250 VAC. The contact connections of Relay 0 are named R0.COM (the "common" terminal) and R0.NO (the "normally open" terminal). Figure 3 shows how to connect to Relay 0 with an AC or DC load; connections to Relay1 are made in a similar manner.

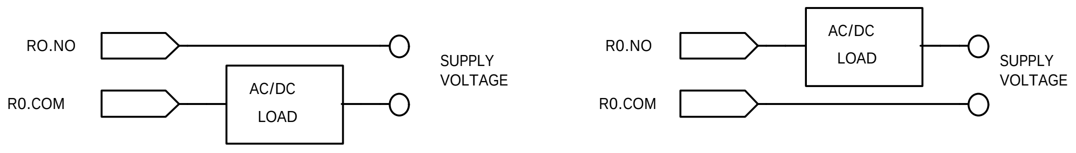


Figure 3. Two ways to connect an onboard relay to an AC or DC load.

Relay Noise Suppression Circuitry

Noise suppression and high voltage snubbing at the relay contacts is performed by onboard circuitry which includes a resistive/capacitive (RC) filter in parallel with a "ZNR" transient suppression diode. The combination of the RC circuit and the ZNR performs arc suppression for both AC and DC loads, greatly reducing radiated electromagnetic interference and increasing the effective lifetime of the relay contacts. The ZNR supplied with the board is Panasonic ERZ-C05DK201U (available from Digi-Key) which has a 130 VAC/170VDC maximum working voltage. If you need to switch higher voltage loads, replace the socketed ZNR with another Panasonic ZNR rated at the required voltages. For example, to control a 250 VAC motor you could install Panasonic part number ERZ-C05DK391U which has a 250 VAC/320VDC maximum working voltage.

Outputs

The QED Digital I/O Board adds 8 outputs to the QED board. Depending on which header you use to access the outputs, you can interface to high current opto-isolated MOSFET outputs, or CMOS/TTL logic level outputs.

Connecting To The High Current Outputs

The high current MOSFET outputs are opto-isolated and capable of sinking 3 amps per channel at up to 50 VDC. The outputs are protected by snub diodes to prevent damage from high-voltage inductive spikes.

These MOSFET outputs control DC loads only. If you need to control AC loads, use the relay outputs described above.

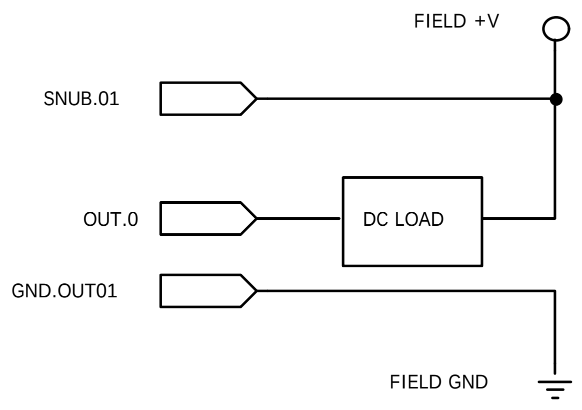
Figure 4 shows how to connect a DC load (motor, solenoid, etc.) to the high current outputs. The field voltage supply can be up to 50 VDC. Pairs of outputs share the same field ground and snub voltage connections, meaning that you can control loads powered by up to four supplies with widely varying power and ground levels. For convenience and to assure full 3 A per output drive capability, each shared ground is brought out to two separate pins on the field header. A glance at the header schematic on the "Power and Connectors" page of the schematics at the end of this document clarifies this point: the "GND.OUT01" signal is brought out to pins 5 and 7, "GND.OUT23" signal is brought out to pins 11 and 13, "GND.OUT45" signal is brought out to pins 15 and 17, and "GND.OUT67" signal is brought out to pins 21 and 23.

Note that the snub signal must be connected to a positive field supply voltage. This voltage must be between 5 V and 50 V for the output to function at its rated 3 ampere output current. This is true for all loads, whether they are resistive or inductive.

If you are testing the MOSFET outputs, remember that you must connect a load to the output in order to see a voltage change as the output changes state. To perform simple non-isolated testing to verify the functionality of the board, you could connect a resistor (say, 1 k•) between the output and any convenient field supply (such as +5V), connect the same field supply to the output's snub signal, and connect the output's field ground to the Ground connection of the +5V supply. Then, when an output bit is off, no current will flow in the load resistor, and a voltmeter or scope will show that its voltage is high (at +5V if you used this as the field supply). If you use the software at the end of this document to turn an output bit on, current will flow through the output resistor, and your voltmeter or scope will show that the corresponding voltage falls to near zero.

Figure 4.

This diagram shows how to connect any DC load to a high current output. GND.OUT01 is a shared field ground connection for the two outputs OUT.0 and OUT.1, and SNUB.01 is a shared snub voltage connection for OUT.0 and OUT.1. The snub signal must be connected to the field supply voltage for the output signal to function.



The field connector on the QED Digital I/O board is rated for currents up to 3 amperes and it should mate with a connector that is capable of carrying the required load current. For example, plugging the Digital I/O Board into the Backplane Board in the QED Industrial Control System provides full 3 ampere field connections and routes the output signals to convenient screw terminals. Note that standard insulation-displacement ribbon cable connectors rated at only 1 ampere per contact.

Connecting to the Logic Level Outputs

CMOS and TTL logic level outputs are available at the logic level header. The logic level outputs are active low. That is, if your software writes a 1 to an output bit, the output voltage of the corresponding bit at the logic level header will be 0 volts, and if your software writes a 0 to an output bit, the output voltage at the logic level header will be +5 volts.

You can think of the logic level outputs as a latched output port that drives the high current MOSFET outputs. Because the logic level signals share the latch circuitry with the MOSFET outputs, you must choose whether you want to use the MOSFET outputs or the logic level outputs; you cannot use both simultaneously. The socketed opto-isolator chips U3 and U7 should be removed if you are using the logic header. This disconnects the logic level circuitry from the high current/high voltage field I/O connector.

Inputs

The QED Digital I/O Board adds 8 inputs to the QED board. Depending on which header you use to connect to the inputs, you can interface to high voltage opto-isolated inputs or CMOS/TTL logic level inputs.

Connecting to the High Voltage Inputs

The high voltage inputs are capable of sensing 3-50VDC. These inputs are opto-isolated and pairs of input channels share the same ground. This allows you to sense voltages referenced to as many as 4 different ground levels. Figure 5 illustrates how to connect an input voltage to the QED Digital I/O Board. When the input voltage is 3-50V, a logical 1 input will be read. When the input voltage is less than 0.4V, a logical 0 input will be read.

Optional socketed onboard pull-up resistors (labeled OPR0 to OPR7 on the schematic and the Digital I/O Board silkscreen) allow you to monitor the state of a contact closure device such as a switch. For example, to interface a contact closure device to input channel IN1, install a 1K• 1/4 watt resistor in the OPR1 socket next to the "I1" silkscreen label, and connect the contact closure device as shown in Figure 6. The input will be read as a logical 0 when the switch contact is open (contacts not connected), and it will be read as a logical 1 when the device is closed (contacts connected).

To sense AC voltage, you can replace the socketed opto-isolator chips U1 and U2 with NEC part PS2504-4. This is an AC input opto-isolator capable of sensing positive and negative voltages. The input will be a logical 1 whenever the field voltage is greater than +3 V or less than -3 V, so near the zero crossings of the AC waveform the input will be read as a logical 0 even though the AC signal is present. A simple time-averaging software function can be applied to determine if the AC waveform is active or not. In any case, use appropriate input clamping circuitry if necessary to make sure that you <u>do not exceed the 50V limit on the inputs</u>.

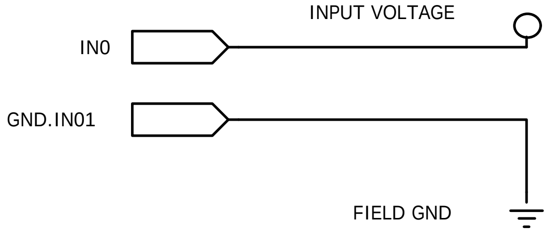


Figure 5. Connecting an input voltage to the QED Digital I/O Board.

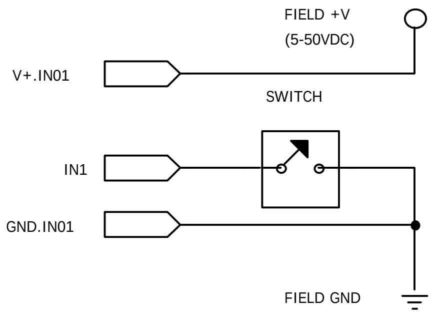


Figure 6. Connecting a contact closure device such as a switch to the QED Digital I/O Board. To provide the proper voltage swing at the input, install a 1K• 1/4 watt resistor in the OPR1 socket (next to the "I1" label) on the board. The input will be read as a logical 0 when the switch contact is an open circuit and as a logical 1 when the switch is closed.

Connecting to the Logic Level Inputs

CMOS and TTL logic level inputs can be sensed at the logic level header. The inputs from the logic header are active low. That is, the inputs read by the application software are the logical complement of the signal levels present at the logic level connector.

Because the logic level signals share the input buffer circuitry with the high voltage inputs, you must choose whether you want to use the high voltage inputs or the logic level inputs; you cannot use both simultaneously. The socketed opto-isolator chips U3 and U7 must be removed if you are using the logic header; this disconnects the logic level circuitry from the high voltage field I/O connector.

Interfacing To Industry-Standard I/O Modules

You can use the logic level header to connect the Digital I/O Board to an 8- or 16-module industrial I/O rack made by Opto-22, Grayhill, and others. These racks typically have sockets for 8, 16 or 24 I/O modules that can acquire AC or DC inputs or control AC or DC outputs. A 34 conductor ribbon cable with a 34 pin dual row connector on one end and a 50 pin dual row connector on the other is required to connect the Digital I/O Board to the industrial I/O rack. The 16 unconnected pins on the 50 pin connector are used only on 24-module racks which are not supported by the Digital I/O Board.

Industrial I/O modules are controlled by active low (inverting) logic. As described above, the circuitry on the QED Digital I/O Board creates active low logic at the logic level header. The result of these two inversions is quite simple: we use positive logic to write to and read from the Industrial I/O modules. In other words, to turn on an output module, your application program simply writes a logical 1 to the corresponding output bit. Likewise, if an input to a module is active (for example, a +5 V input to a 0-5V input module), the corresponding bit will be read as a 1 by your application software.

The industrial I/O module rack can be supplied with +V from the QED Digital I/O Board. Typically a jumper must be installed on the module rack to supply +5V to the industrial I/O module rack via the ribbon cable connector, and this is described in the documentation that accompanies the I/O rack.

I/O modules 0 - 7 must be output modules and modules 8 - 15 must be input modules. The socketed opto-isolator chips U1 and U2 (for the high-voltage inputs) and U3 and U7 (for the high-current outputs) on the QED Digital I/O Board must be removed when controlling industrial I/O modules via the logic level connector.

Software Overview And Example

Software drivers to control the QED Digital I/O Board are presented in this section. Commented source code is presented in both QED-Forth and ANSI C. Use of the C version requires the Windows-based QED Control C programming environment.

This software supports up to four QED Digital I/O Boards. Each Board has a unique address and a corresponding "board identifier" constant defined near the start of the source code. The examples at the end of the code assume that "BOARD3" at page DF_H is in use. To run the examples without changing the source code, make sure that both jumpers J0 and J1 are installed (see Figure 1) to locate your Digital I/O Board at page DF_H. If you wish to map the board onto a different page, set J0 and J1 accordingly and pass the appropriate board identifier to the driver routines.

Controlling the Relays

The relays are turned on and off by reading and writing to addresses on the specified page as described in Figure 2. Four simple functions make it easy to turn Relay0 and Relay1 on and off; each function expects a board identifier constant (BOARD0, BOARD1, BOARD2, or BOARD3) as an input parameter. The RELAY0.SHADOW and RELAY1.SHADOW (in C: Relay0_Shadow and Relay1_Shadow) locations in RAM keep track of the current state of each relay.

For example, to activate RELAY1 on BOARD3, execute the QED-Forth command:

```
BOARD3 RELAY1.ON
```

If you have compiled the C version, your code can call the C function as:

```
Relay1_On(BOARD3)
```

or you can interactively execute the C function from your terminal by typing:

```
Relay1_On( int BOARD3)
```

If you listen carefully, you can hear the relay "click" on, and you can check that the relay is on by using an ohmmeter to verify continuity between pins 3 and 4 (R1.COM and R1.NO) on the field header.

Controlling The Outputs

In the code below, the DIGITAL.STORE (in C: DigitalStore) function expects a data byte and a board identifier as input parameters. The routine uses C! (in C: StoreChar()) to output an 8-bit value to the output port at address 0003 on the Digital I/O Board's specified page, and it also saves the value in a "shadow ram" location. Read/modify/write operations are performed by the convenient DIGITAL.SET.BITS and DIGITAL.CLEAR.BITS functions (named DigitalSetBits and DigitalClearBits in C) which expect a mask byte and a board identifier as input parameters.

Note that you can not use the standard QED-Forth functions SET.BITS, CLEAR.BITS, TOGGLE.BITS or CHANGE.BITS to control I/O on the Digital I/O Board, because these standard read/modify/write operators assume that data can be "read back" from the output port. Likewise, the Control C library functions SetBits(), ClearBits(), ToggleBits() and ChangeBits() can not be used for the same reason.

For example, to cause OUT0 on BOARD3 to start sinking current, execute the QED-Forth command:

```
HEX 01 BOARD3 DIGITAL.STORE
```

If you have compiled the C version, your code can call the C function as:

```
DigitalStore(0x01, BOARD3)
```

or you can interactively execute the C function from your terminal by typing:

```
DigitalStore( char 01, int BOARD3)
```

Then, if you have connected the appropriate load, snub voltage and ground signals as described in the "Connecting To The High Current Outputs" section above, you will notice that the voltage at OUT0 (pin 6 on the field header) goes low as the MOSFET turns on and conducts current.

Reading The Inputs

In the code below, the DIGITAL.FETCH (in C: DigitalFetch) function reads and returns an 8-bit value from the input port at address 0003 on the Digital I/O Board's specified page. For example, if voltage inputs are connected to the field header (and opto-isolator chips U1 and U2 are installed), each 1 bit returned represents a voltage level of 3-50 VDC on the corresponding input.

Initialization

The INIT.DIGITAL.IO (in C: InitDigitalIO) function expects a board identifier input parameter. It sets all of the outputs inactive and initializes the "shadow" ram locations. It must be executed after each reset or restart and before any of the read/modify/write operations are performed on the digital I/O.

QED-Forth Driver Code

```
HEX                 \ use hexadecimal base during compilation
11 WIDTH !          \ store 11 characters in each name header
4 USE.PAGE          \ initialize memory map; this can be deleted if the
                    \ memory map has already been initialized by other code.

ANEW DIGITALIO.DRIVERS  \ a forget marker to facilitate re-loading the code

\ *************** ADDRESS AND PAGE CONSTANTS ***************

\ These are four "board.id" parameters that are passed to the driver routines;
\ Note that the value of each board.id is simply the board's page:
  DC  CONSTANT    BOARD0           \ Page with J1 and J0 removed
  DD  CONSTANT    BOARD1           \ Page with J1 removed, J0 installed
  DE  CONSTANT    BOARD2           \ Page with J1 installed, J0 removed
  DF  CONSTANT    BOARD3           \ Page with J1 and J0 installed
      \ Note: Digital I/O Boards are shipped from the factory as "BOARD3"
      \       with both jumpers J1 and J0 installed.


\ here are the relevant addresses on the board page:
0000  CONSTANT    RELAY0.ADDRESS    \ Address of Relay 0
0001  CONSTANT    RELAY1.ADDRESS    \ Address of Relay 1
0003  CONSTANT    IO.ADDRESS        \ Address of input and output ports
\ 0002 CONSTANT   EXTRA.OUTPUT.ADDRESS    \ not available at the connector

\ **************** SHADOW RAM ******************

\ here are definitions of "shadow" ram locations that allow read-back
\ of previously written output values:
2VARIABLE   OUTPUT.SAVE \ 1 byte for each of 4 digital i/o boards
2VARIABLE   RELAY0.SAVE \ 1 byte for each of 4 digital i/o boards
2VARIABLE   RELAY1.SAVE \ 1 byte for each of 4 digital i/o boards

: OUTPUT.SHADOW   ( board.id -- xaddr.of.shadow.byte )
      BOARD0 -                ( offset -- )
      OUTPUT.SAVE ROT XN+     ( -- xaddr.of.shadow.byte )
      ;

: RELAY0.SHADOW   ( board.id -- xaddr.of.shadow.byte )
      BOARD0 -                ( offset -- )
      RELAY0.SAVE ROT XN+     ( -- xaddr.of.shadow.byte )
      ;

: RELAY1.SHADOW   ( board.id -- xaddr.of.shadow.byte )
      BOARD0 -                ( offset -- )
      RELAY1.SAVE ROT XN+     ( -- xaddr.of.shadow.byte )
      ;
\ **************** RELAY CONTROL *****************

: RELAY0.ON       ( board.id -- )         \ Turns on Relay 0
\ Writing to RELAY0.ADDRESS on the board.page turns on RELAY0
      >R 0 RELAY0.ADDRESS R@ C!
      FF R> RELAY0.SHADOW C!              \ set the shadow variable byte
;
```

```
: RELAY0.OFF      ( board.id -- )         \ Turns off Relay 0
\ Reading from RELAY0.ADDRESS on the board.page turns off RELAY0
      >R RELAY0.ADDRESS R@ C@
      DROP                          \ drop the unneeded fetched contents
      00 R> RELAY0.SHADOW C!        \ clear the shadow variable byte
;

: RELAY1.ON       ( board.id -- )         \ Turns on Relay 1
\ Writing to RELAY1.ADDRESS on the board.page turns on RELAY1
      >R 0 RELAY1.ADDRESS R@ C!
      FF R> RELAY1.SHADOW C!              \ set the shadow variable byte
;

: RELAY1.OFF      ( board.id -- )         \ Turns off Relay 1
\ Reading from RELAY1.ADDRESS on the board.page turns off RELAY1
      >R RELAY1.ADDRESS R@ C@
      DROP                          \ drop the unneeded fetched contents
      00 R> RELAY1.SHADOW C!        \ clear the shadow variable byte
;

\ **************** WRITE TO THE OUTPUT PORT ******************

: DIGITAL.STORE   ( byte\board.id -- )
      \ Sends byte of data to output and shadow
      2DUP                          \ duplicate data & board.page
      IO.ADDRESS SWAP C!            \ Write value to port
      OUTPUT.SHADOW C!              \ Store value in shadow location
      ;

: DIGITAL.SET.BITS ( mask.byte\board.id -- )
      \ For each mask bit that equals 1, sets corresponding output bit high
      DUP OUTPUT.SHADOW C@          ( mask.byte\board.id\prior.contents -- )
      ROT OR                        \ OR the mask with the shadow memory
      SWAP                          ( data\board.id -- )
      DIGITAL.STORE                 \ Send value to outputs
;

: DIGITAL.CLEAR.BITS ( mask.byte\board.id -- )
      \ For each mask bit that equals 1, sets corresponding output bit low
      DUP OUTPUT.SHADOW C@    ( mask.byte\board.id\prior.contents -- )
      ROT COMPLEMENT AND      \ AND the shadow with the complement of mask
      SWAP                    ( data\board.id -- )
      DIGITAL.STORE           \ Send value to outputs
;

\ **************** READ THE OUTPUT PORT ******************
: DIGITAL.FETCH   ( board.id -- byte )    \ Reads the value of the Inputs
      IO.ADDRESS SWAP C@
;
\ **************** INITIALIZE ******************
: INIT.DIGITAL.IO  ( board.id -- )  \ Initializes board and shadow memory
      LOCALS{ &board.id }
      0 &board.id DIGITAL.STORE           \ Initialize all outputs low
      &board.id RELAY0.OFF                \ Turn relays off
      &board.id RELAY1.OFF
;
```

QED-Forth Example

If a motor is connected to OUT7 as in Figure 4, the following words will turn the motor on and off. Recall that the hexadecimal value 80_H has its top bit (bit 7) set and all other bits clear.

```
\ **************** EXAMPLES OF USE *****************

HEX
80    CONSTANT    MOTOR7.MASK             \ Output channel 7 mask

: TURN.MOTOR.ON ( -- )
      MOTOR7.MASK BOARD3 DIGITAL.SET.BITS       \ Sets bit 7 on the output
;

: TURN.MOTOR.OFF ( -- )
      MOTOR7.MASK BOARD3 DIGITAL.CLEAR.BITS     \ Clears bit 7 on the output
;

\ Here are some valid commands.  We assume that we are in HEX base and that
\ we use BOARD3 (J0 and J1 are both installed):
\ BOARD3 INIT.DIGITAL.IO            \ do this first
\ FF BOARD3 DIGITAL.STORE           \ set all outputs high
\ FF BOARD3 DIGITAL.CLEAR.BITS      \ set all outputs low
\ BOARD3 RELAY0.ON
\ BOARD3 RELAY0.OFF
\ TURN.MOTOR.ON
\ TURN.MOTOR.OFF
```

Control C Driver Code and Example

Enter this code into a file using WinEdit, then click on the Make (hammer) icon and send the resulting .txt download file to the QED Board using the Windows terminal. You can then interactively execute each function from your terminal — remember to type a space after the opening left parenthesis in each interactive function call.

```
#include <\mosaic\allqed.h>         // include header files

// *************** ADDRESS AND PAGE DEFINITIONS ***************

#define     BASE_DIGITAL_IO_PAGE  0xDC    // Page with J1 and J0 removed

// These are 4 "board.id" parameters that are passed to the driver routines;
// The value of each board.id is the board's offset from the base_page:
#define     BOARD0  0               // Jumpers J1 and J0 removed
#define     BOARD1  1               // J1 removed, J0 installed
#define     BOARD2  2               // J1 installed, J0 removed
#define     BOARD3  3               // J1 and J0 installed
      // Note: Digital I/O Boards are shipped from the factory as "BOARD3"
      //       with both jumpers J1 and J0 installed.

#define MAX_DIGITAL_BOARDS  4 // up to 4 digital I/O boards per QED Board

// here are functions that return a calculated 32-bit
// extended addresses (xaddr) for the specified Digital I/O Board:

_Q xaddr Relay0Address(int board_id)      // Relay0 xaddress
{     return(0x10000 * (BASE_DIGITAL_IO_PAGE + board_id));
}           // shift board addr into upper 16bits; lower 16 bits = 0000


_Q xaddr Relay1Address(int board_id)      // Relay1 xaddress
{     return(0x10000 * (BASE_DIGITAL_IO_PAGE + board_id) + 0001);
}           // shift board addr into upper 16bits; lower 16 bits = 0001

_Q xaddr IOAddress(int board_id)          // Input & output port xaddress
{     return(0x10000 * (BASE_DIGITAL_IO_PAGE + board_id) + 0003);
}           // shift board addr into upper 16bits; lower 16 bits = 0003


// *************** SHADOW RAM *****************


// Allocate "Shadow" ram arrays for the output port and relays
// that allow read-back of previously written output values:
static uchar Output_Shadow[MAX_DIGITAL_BOARDS];
static uchar Relay0_Shadow[MAX_DIGITAL_BOARDS];
static uchar Relay1_Shadow[MAX_DIGITAL_BOARDS];
```

```
// **************** RELAY CONTROL *****************

_Q void Relay0_On(int board_id)
      // Writing to Relay0Address turns on RELAY0
{     StoreChar( 0, Relay0Address(board_id));
      Relay0_Shadow[board_id] = TRUE;           // set the shadow
}

_Q void Relay0_Off(int board_id)
      // Reading from Relay0Address turns off RELAY0
{     FetchChar(Relay0Address(board_id));       // discard the result
      Relay0_Shadow[board_id] = FALSE;          // clear the shadow
}

_Q void Relay1_On(int board_id)
      // Writing to Relay1Address turns on RELAY1
{     StoreChar( 0, Relay1Address(board_id));
      Relay1_Shadow[board_id] = TRUE;           // set the shadow
}

_Q void Relay1_Off(int board_id)
      // Reading from Relay1Address turns off RELAY1
{     FetchChar(Relay1Address(board_id));       // discard the result
      Relay1_Shadow[board_id] = FALSE;          // clear the shadow
}


// **************** WRITE TO THE OUTPUT PORT *****************

_Q void DigitalStore(uchar data, int board_id)
      // Sends byte of data to output and shadow
{     StoreChar( data, IOAddress(board_id) );   // Write value to port
      Output_Shadow[board_id] = data;     // Store value in shadow location
}

_Q void DigitalSetBits(uchar mask, int board_id)
      // For each mask bit that equals 1, sets corresponding output bit high
{     DigitalStore(Output_Shadow[board_id] | mask, board_id);
}                                    // OR mask with shadow

_Q void DigitalClearBits(uchar mask, int board_id)
      // For each mask bit that equals 1, sets corresponding output bit low
{     DigitalStore(Output_Shadow[board_id] & ~mask, board_id);
}                                    // And shadow with complement of mask

// **************** READ THE OUTPUT PORT *****************

_Q uchar DigitalFetch(int board_id)
      // Reads the value of the Inputs
{     return(FetchChar(IOAddress(board_id)));
}
```

```
// **************** INITIALIZE *****************

_Q void InitDigitalIO(int board_id)
      // Initializes Digital I/O Board and shadow memory
{     DigitalStore(0, board_id);          // Initialize all outputs low
      Relay0_Off(board_id);               // Turn relays off
      Relay1_Off(board_id);
}

// ****************** Example and main() *****************

// If a motor is connected to OUT7 on BOARD3 as in Figure 4,
// the following functions will turn the motor on and off.
// Recall that the hexadecimal value 0x80 has its top bit (bit 7) set
// and all other bits cleared.
// Note that both jumpers J0 and J1 must be installed to identify this
// digital I/O Board as "BOARD3".

// The following are valid interactive commands
// that you can type from the terminal to test the I/O on BOARD3:
//          DigitalStore( char 0F, int 3)
//          DigitalClearBits( char 03, int 3)
//          Relay0_On( int 3)
//          DigitalFetch( int 3)


#define     MOTOR7_MASK 0x80              // Output channel 7 mask

_Q void TurnMotorOn(void)
{     DigitalSetBits(MOTOR7_MASK,BOARD3);       // Sets bit 7 of the output
}

_Q void TurnMotorOff(void)
{     DigitalClearBits(MOTOR7_MASK,BOARD3);     // Clears bit 7 of the output
}

void main(void)
{     InitDigitalIO(BOARD3);
      TurnMotorOn();
}
```

The information provided herein is believed to be reliable; however, Mosaic Industries assumes no responsibility for inaccuracies or omissions. Mosaic Industries assumes no responsibility for the use of this information and all use of such information shall be entirely at the user's own risk.