The Multi-I/O expansion board gives users the ability to add analog inputs and outputs, UART capability (for GPS or modem) and isolated high current outputs to the Flashlite 386Ex. Available in several configurations, each Multi-IO board can add up to 8 channels of 12 bit A/D, 4 channels of 12 bit D/A, 8 output drivers and 2 UARTs. If more I/O is required, several boards can be added to the system using 10 pin IDC style ribbon cables. A library of C and QuickBasic functions is supplied to eliminate the hassle of programming devices on a serial bus.

Features:

Analog to Digital:
- 12 bit resolution
- 4.096V reference (1mV/count)
- Single-ended or differential modes

Digital to Analog:
- 12 Bit
- Voltage output (0-4.095 V)
- Current output (0-20 mA)

UARTs:
- TTL outputs, RS-232 tolerant inputs
- 115K max speed
- Configurable interrupt (IRQ5 or IRQ6)
- 8 byte FIFO

Drivers:
- 1 Amp transistors
- 60V max Vce
- 2500 V optical isolation

System requirements:

- Flashlite386Ex SBC
- Interface Cable
- Stacking standoffs

Configuration:

Five jumpers allow configuration of each Multi-I/O board. Depending on the features of the board, some of the jumpers may not be required. Jumpers configure the direction of signals on J2 and J4 (BUS1 and BUS2), allow selection of the hardware interrupt associated with the UARTs, and connection of the UART1 transmit data line to either J11 or J13.

J3 controls the direction of signals on the BUS1 and BUS2 connectors. Shunts should be installed between pins 1-3 and 2-4 or 3-5 and 4-6. If only one IO board is installed, and that board plugs directly into the Flashlite, these jumpers are not important. When adding additional IO boards to the system or connecting the IO board with a cable to the Flashlite, BUS1 will transmit to a second board if J3 has 3-5 and 4-6 shorted. BUS2 will transmit data when J3 has 1-3 and 2-4 shorted. Signals will be received on the remaining BUS connector.

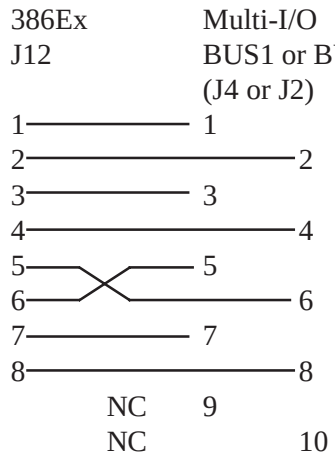
Use J10 and J19 to select between IRQ5 and IRQ6 for the UARTs. Short pins 1-2 for IRQ5 or 3-5 for IRQ6. The serial port can be used without interrupt connections.

When UART1 signals are expected on J11, J12 1-2 should be shorted. Short J12 2-3 when connecting to J13.

Installation:

Mount the IO board to the back of the Flashlite 386Ex, securing it to the 3/8" standoffs with 6-32 screws or male standoffs. Be sure to orient the IO board so that the 10 pin bus connectors (Jxx and Jxx) are opposite Jxx on the Flashlite. Connect the IO board to the Flashlite using interface cable p/n xx-xxxx. See the cable drawing for wiring information.

http://www.jkmicro.com
Davis, CA USA (530) 297-6073



Connector Pinouts:

J7 / AD1	
Vref	1
AD0	2
AD1	3
GND	4

J8 / AD2	
Vref	1
AD2	2
AD3	3
GND	4

J16 / AD3	
Vref	1
AD4	2
AD5	3
GND	4

J17 / AD4	
Vref	1
AD6	2
AD7	3
GND	4

J5 / DA1	
Vout1	1
Iout0	2
GND	3

J6 / DA2	
Vout2	1
Iout2	2
GND	3

J14 / DA3	
Vout3	1
Iout3	2
GND	3

J15 / DA4	
Vout4	1
Iout4	2
GND	3

J9 / OUT1	
Out1+	1
Out1-	2
Out2+	3
Out2-	4
Out3+	5
Out3-	6
Out4+	7
Our4-	8

J18 / OUT2	
Out5+	1
Out5-	2
Out6+	3
Out6-	4
Out7+	5
Out7-	6
Out8+	7
Out8-	8

J11 / UART1	
CTS	1
RxD	2
TxD	3
RTS	4
GND	5

J20 / UART2	
CTS	1
RxD	2
TxD	3
RTS	4
GND	5

J2 and J4 (BUS1 and BUS2)			
Data	1	2	CLK
Reset/	3	4	CS IN(OUT)
Vcc	5	6	GND
IRQ5	7	8	IRQ6
Vcc	9	10	GND

Connectors:

Interface connectors use Molex KK style headers and housings.
J5, J6, J14, J15 require 3 pin Molex housings, p/n 22-01-3037
J7, J8, J16, J17 require 4 pin Molex housings, p/n 22-01-3047
J11, J20 require 5 pin Molex housings, p/n 22-01-3057
J9, J18 require 8 pin Molex housings, p/n 22-01-3087
The above housings accept crimp terminals for 18-30 gauge wire, Molex p/n: 08-50-0114

Software:

DRIVER.OBJ contains functions callable from C or QuickBASIC. This library allows easy access to the various functions of the Multi-I/O board. The C and BASIC functions have common names and act identically. The assembly source code (DRIVER.ASM) and C header file (DRIVER.H) are also included. The object file may be regenerated using TASM.

All values passed and returned from the library functions are 16 bits wide. Many functions do not use all 16 bits, but the complete word is required. When the complete word is not used, data fills the low order bits and the high order bits are unused and ignored. In the following descriptions, C function syntax is shown first and BASIC syntax, where different, is shown in brackets ([ ]).

int GetVersion(*void*) [GetVersion(*data*)]
Returns the version number of the driver library.
Least significant digit is minor revision.
Most significant digits are major revision.

void PutA2DChannel(*address*)
Sets the channel to get data from on the next call to GetA2DChannel( ).
address refers to a device (or channel). Address values are sequential, starting at 0, for each type of device.

int GetA2D(*void*) [GetA2D(*data*)]
Returns the conversion results from previously selected A/D

void PutD2AChannel(*address*)
Selects a D/A channel.
address refers to a device (or channel). Address values are sequential, starting at 0, for each type of device.

void PutD2A(*data*)
Writes output value to the selected D/A channel.
data is a value read from an A/D or written to a D/A. Only the low 12 bits are used.

void PutDriverChannel(*address*)
Selects driver.
address refers to a device (or channel). Address values are sequential, starting at 0, for each type of device.
Each driver bit is addressed individually.

void PutDriver(*state*)
Turns the selected driver ON or OFF.
state is the value written to a driver: 0=OFF, non-zero = ON.

Basic UART functions:
These functions send and receive characters with 8 data bits, 1 stop bit and no parity bits. If other data formats are required, see the enhanced UART functions. The UART baud rate must be set during initialization, there is no power-on default. UART interrupt functions are not currently supported by the driver library.

void PutUARTChannel(*address*)

Selects a UART channel.
address refers to a device (or channel). Address values are sequential, starting at 0, for each type of device.

void PutUARTBaud(*baud*)

Sets the baud rate on the selected UART channel. This must be called before attempting to use a UART, there is no default power-on baud rate.
baud is a value from the table below that sets the UART baud rate

Function Parameter	Baud Rate
203	230.4k baud
115	115.2k baud
57	57.6k baud
38	38.4k baud
19	19.2k baud
9600	9600 baud
4800	4800 baud
2400	2400 baud
1200	1200 baud
600	600 baud

void PutUARTChar(*character*)

Waits until UART transmitter is ready and sends a character to the selected UART channel.
character is the value to be transmitted by a UART. The value should be in the low order bits.
This is a blocking function.

int GetUARTChar(*void*) [GetUARTChar(*data*)]

Waits until the selected UART has received a character and returns data from the receive buffer.
This is a blocking function.

int GetUARTRxStatus(*void*) [GetUARTRxStatus(*data*)]

Checks the receiver status on the selected UART. Returns 1 if a character is waiting.

int GetUARTTxStatus(*void*) [GetUARTTxStatus(*data*)]

Checks the transmitter status on the selected UART. Returns 1 if the transmit buffer is empty .

Enhanced UART functions:
These functions allow complete control of the UART and its configuration registers. All functions rely on PutUARTChannel( ) to select the desired UART. See the MAX3100 data sheet for information on these registers. Use of these functions may change the behavior of the basic UART functions. To reset the a UART for use with the basic functions, write a hex E000 to the Write Configuration register, then set the baud rate using PutUARTBaud( ).

void PutUARTWC(*word*)

Writes a new value to the UART Write Configuration register and driver shadow register.
word is a 16 bit value as described in Table 1 of the MAX3100 data sheet.

void PutUARTWD(*word*)

Writes a value to the UART Write Data register.
word is a 16 bit value as described in Table 3 of the MAX3100 data sheet.

int GetUARTRC(*void*) [GetUARTRC(*data*)]
Reads a 16 bit value from the UART Read Configuration register

int GetUARTRD(*void*) [GetUARTRD(*data*)]
Reads a 16 bit value from the UART Read Data register

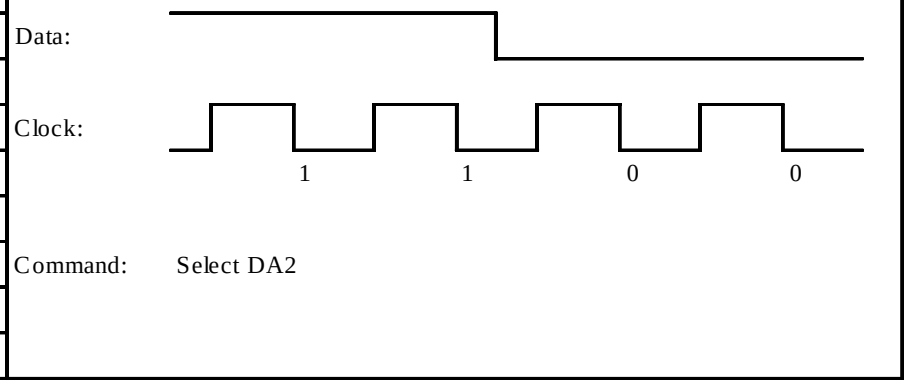
Hardware Notes:

The Multi-I/O board operates on a serial bus implemented using 4 processor I/O lines for clock, data, reset and chip select signals. Two interrupt lines are available for use with the UARTs. The data signal is bidirectional, reset and CS are active low, and data must be valid on the falling edge of the clock signal.

To reset the boards, assert the reset line. Boards must be reset in order to change the selected channel.

To select a device, a start bit and a 3 bit address are clocked to the board. There are 6 selectable devices on each half of the board: AD write, AD read, UART, DA1 write, DA2 write, and DRIVER write. The second half of the board is selected using CS6 and additional boards are selected using CS7. CS6 and CS7 are only specified on the first (left) half of the board. Data is clocked MSB first when selecting a board.

1 000	AD write
1 001	AD read
1 010	UART
1 011	DA1 write
1 100	DA2 write
1 101	DRIVER write
1 110	Right side select
1 111	Next board select



After a device has been selected, information is written to (or read from) the device as per the device specification.

The Driver circuit consists of a shift register and a latch. Data is clocked in MSB first and latched on the falling edge of chip select.

UART signals are 0 and 5 volts. These signals may not work with a true RS-232 device depending on the sensitivity of its receiver. The UART inputs (Rx and CTS) are diode protected and current limited so that RS-232 level voltages will not damage the device.

When using the Digital to Analog converters in voltage output mode, it may be necessary to install a resistor (approximately 10k ohm) between pins 2 and 3 of the output. This will reduce noise on the voltage output.