Documentation for the Investment Technologies Mitsmini Microcontroller Board

© Dean Camera, 2004-2005

Version 1.0

Overview:

The Mitsmini is the latest in the Investment Technologies' line-up of microcontroller development boards. Forgoing the Atmel chips used in the previous ABC "Hotchip" boards in favour of a much more powerful Mitsubishi M16C core, the board gives unprecedented power to the user.

The Mitsubishi M30624FGMFP is a 16-bit microcontroller, boasting a 16 MHz clock (up to 16 million instructions per second), 8-channel 10-bit analogue-to-digital converter, and 2-channel digital-to-analogue converter.

This documentation should help strengthen where the ABC boards were weak - by providing an easy to use combination manual and reference to allow users of all experience to get started. The documentation has been broken up into sections for easy reference.

It should be stressed that this document is not a datasheet or technical manual; although advanced, it explains everything in an easy to understand manner for users of all expertise. The focus is mainly on hardware; for programming, please consult another software-centric manual, or read your chosen/preferred programming environment's help files (if included).

Documentation Contents:

CHAPTER 1: The Mitsmini Range

Mitsmini Kits available from Austrol:

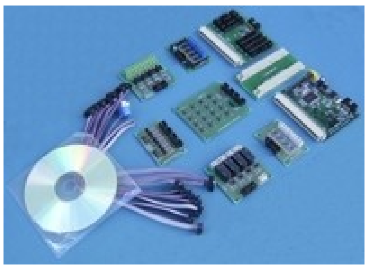
Austrol package the Mitsmini in several kits, each containing a different set of items. A brief description of the range is shown below.

Mitsmini Basic Starter Kit

The Starter Kit is ideal for persons familiar to microcontrollers and/or electronics. It includes the programming (serial) cable, the Mitsmini board, and the software CD. The Starter Kit would be ideal for Electronics enthusiasts with a stock of electronics parts, or for prototyping of a commercial device.

This is the "barebones" of Mitsmini kits; all external devices must be created and attached by the user.

Mitsmini Education Kit



The Educational Kit is perfect for persons new to microcontrollers. It contains a Software CD, Cables, and a wealth of add-on boards (see "Add-on Boards from Austrol" section) including the Mitsmini board and:

- Quad Relay Board
- 2-Way Bus Board
- MOSFET Transistor Board
- Opto-Isolated Board
- I/O Expander Board
- 4x4 Keypad Board
- NPN Transistor Board
- Quad non-isolated Input Board

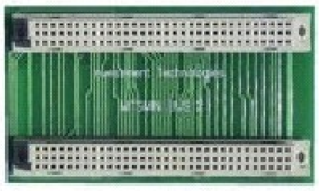
Mitsmini 2-way Bus Expansion Kit

This kit contains the Mitsmini microcontroller board, a 2-Way Bus Board, expander cables, Software CD and a board to make the Mitsmini outputs compatible with many of the Add-on boards originally designed for the ABCmini board also available from Austrol.

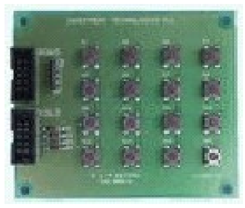
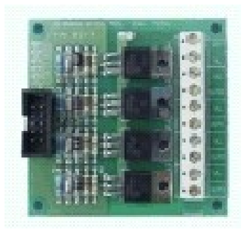
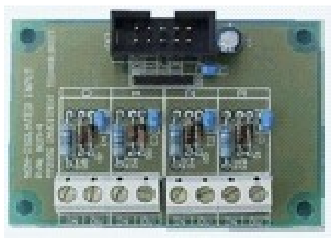
Mitsmini 5-way Bus Expansion Kit	
	Similar to the 2-Way Bus Expansion Kit, save for the bus board. This connector can accommodate up to five Mitsmini compatible boards.
Mitsmini RF Kit	
	A perfect kit for wireless applications. The RF Kit contains two Mitsmini microcontroller boards, two 5-Way bus boards, two I/O Converter Boards, Cables, Software CD and two RF wireless link boards.

Add-on boards available from Austrol:

To complement its range of kits, Austrol manufacture several separate add-on boards. You will need a bus board (5-way and 2-way boards shown below) to connect your Mitsmini to another or other boards.

5-way Bus Connector	
	This bord is designed to connect the Mitsmini to other Mitsmini add-on boards that use the 96-pin connector.
2-way Bus Connector	
	Similar to the 5-way bus board above, except that this will only connect a Mitsmini board to one other board.

RF Transceiver	
	This board will allow you to communicate wirelessly with other Mitsmini boards several meters away. You will need

	a minimum of two Mitsminis and two RF Boards to exchange information.
I/O Expander	
	The I/O expander will make the Mitsmini compatible with the older add-on boards designed for the predecessor of the Mitsmini, the ABCmini. 14 connectors plus an additional serial port (connected to UART 0) create a wealth of possibilities.
Keypad Module	
	Requires two I/O connectors. A 4x4 matrix of pushbuttons for user input. **(Requires the Mitsmini I/O Expander)**
Quad N-Channel Logic Level MOSFET	
	Although its name's a mouthful, this allows you to interface the Mitsmini to high-voltage and high-current devices. Driven by a standard logic I/O, this will drive up to 33V 2.5A (without additional heatsink). **(Requires the Mitsmini I/O Expander)**
Quad Non-Isolated Input	
	Some devices use a higher logic level than 5V, or you may wish to sense higher than logic level voltages on an I/O. This board will allow four inputs up to 24V each act as a digital signal. **(Requires the Mitsmini I/O Expander)**

NPN Open-Collector Relays	
	Requiring two I/O connectors (like the keypad module), this board will switch 8 light duty (less than 100ma) loads via 8 separate open-collector BC547 NPN transistors. **(Requires the Mitsmini I/O Expander)**
Quad Relay Module	
	Four relays allow medium-heavy loads (max 275VAC 5A or 35VDC 5A) to be switched via a single I/O connector. **(Requires the Mitsmini I/O Expander)**

CHAPTER 2: The Mitsubishi Chip

The M30624FGMFP microcontroller:

Belonging to the Mitsubishi M16C family, the M30624FGMFP offers power and flexibility in a tight surface-mount package.

All microcontrollers - just like a computer CPU - require a clock source. Each pulse from the clock tells the microcontroller to execute an instruction. The most popular clock source for Microcontrollers is the quartz crystal as they are widely available in many frequencies and are relatively cheap. Each Microcontroller model has different specifications and so a suitable clock frequency must be applied accordingly. The M30624FGMFP requires a 16MHz clock frequency - which will execute 16,000,000 cycles per second - but this can be internally divided by powers of two in software to give a custom amount of instructions executed per second. PIC micros automatically divide by 2 or 4, so may use a 16MHz clock but only execute 8,000,000 cycles per second.

The features of the M30624FGMFP are detailed below in point form. If you are unsure of a term or are new to microcontrollers, do not worry; these will be explained later in the documentation.

- CMOS (Complementary MOS) based
- 16MHz Clock – Speed can be divided by software
- 16-Bit Microcontroller – More powerful than the dated 8-bit Atmel Chips use in the ABCmini and ABCmaxi boards
- 10-Bit Analogue-to-Digital converter
- 256 Kb of Flash Rom
- 20 Kb or RAM
- Optimised for Low Power Consumption
- 25 Internal and 8 External Interrupts
- 5 output and 6 input 16-bit multifunction timers (for PWM, etc.)
- 3 serial UARTs
- 8 10-bit A/D converters
- 2 8-bit D/A converters
- 5ma maximum current per port pin

Interestingly, due to the 20Kb of RAM, you can run small programs straight from this memory, as opposed to programming the chip's ROM. As Flash ROM can only be programmed a limited amount of times, so this is great for development as RAM can be written to as many times as you want.

Like standard logic IC's, the M30624FGMFP runs on 5V logic and thus can be interfaced with other 5V logic ICs directly. Each port can act as either an input or an output, but some alternate pin features are unidirectional (either input or output).

Due to the microscopic size of the circuitry, the outputs can only use a tiny amount (5ma) of current each. Most applications will need a transistor (acting as a switch) to power devices. Light duty devices will require only a tiny standard transistor (such as the BC548) but heavy-duty applications (such as high-voltage or high-current) will need a large MOSFET transistor, or a standard transistor linked to a relay.

Most IC's (the M30624FGMFP is no exception) are static-sensitive. Friction or contact with statically charged materials may cause a charge to build up in the skin or clothes, and can be discharged as a high voltage (several kilovolts) into the delicate IC circuitry when the conductive pins or contacts are touched. Because of this, you should always wear an antistatic wrist strap connected to an earthed material (such as a tap) before working with the Mitsubishi IC or the Mitsmini board to dissipate these unwanted currents. Antistatic wrist straps are available from most electronics stores for $10-$20, or alternatively (this is not an ideal solution) you can touch an earthed object beforehand.

In addition, ensure you are not working on a conductive surface such as a metal workbench or table, as this will short out the board and/or power-supply and may cause personal injury.

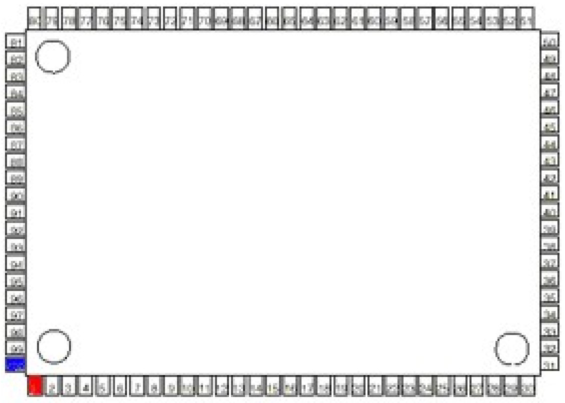
A note about Pin Doubling:

To squeeze as much features into a small package, almost all of the pins on the Mitsubishi microcontroller share their functionality with another feature. To enable such features as the A/D converter or the UART(s), you must sacrifice one or more digital I/O pins. If the extra feature is not enabled, the pin with default to its primary function.

In the pin description table, the primary functions are shown in blue, with secondary functions shown in red, green, grey and pink.

Description of Pins:

The following table is the pin-out for the microcontroller's direct pins; i.e. not the Mitsmini's large white 96-pin connector. The pins start from the bottom-left (shown as red pin) and run around the outside in an anti-clockwise direction to pin 100 (shown as blue pin.)



Micro Pin	Description/Function(s)	Micro Pin	Description/Function(s)
1	Port 9.6 Serial Out #4 A/D Extended Input #1	51	Port 4.3 External Mem Address Bit 19
2	Port 9.5 Serial Clock #4 A/D Extended Input #0	52	Port 4.2 External Mem Address Bit 18
3	Port 9.4 D/A Converter 1 Timer B Input #4	53	Port 4.1 External Mem Address Bit 17
4	Port 9.3 D/A Converter 0 Timer B Input #3	54	Port 4.0 External Mem Address Bit 16
5	Port 9.2 Serial Out #3 Timer In #2	55	Port 3.7 External Mem Address Bit 15
6	Port 9.1 Serial In #3 Timer In #1	56	Port 3.6 External Mem Address Bit 14
7	Port 9.0 Serial Clock #3 Timer in #0	57	Port 3.5 External Mem Address Bit 13
8	Bus Select (L=16 Bit, H=8 Bit)	58	Port 3.4 External Mem Address Bit 12
9	CVNss (Connect to VCC)	59	Port 3.3 External Mem Address Bit 11
10	Port 8.7 External Secondary Timer Crystal In	60	Port 3.2 External Mem Address Bit 10
11	Port 8.6 External Secondary Timer Crystal Out	61	Port 3.1 External Mem Address Bit 9
12	/RESET (L for 20 cycles=Reset)	62	Vcc #2 (5V)
13	Xout (Crystal or Clock Source Out)	63	Port 3.0 External Mem Address Bit 8
14	Vss (0V)	64	Vss #2 (0V)
15	Xin (Crystal or Clock Source In)	65	Port 2.7 External Mem Address Bit 7
16	Vcc (5V)	66	Port 2.6 External Mem Address Bit 6
17	Port 8.5 NMI Interrupt (L=Active, Input only*)	67	Port 2.5 External Mem Address Bit 5
18	Port 8.4 External Interrupt #2 (L=Active)	68	Port 2.4 External Mem Address Bit 4
19	Port 8.3 External Interrupt #1 (L=Active)	69	Port 2.3 External Mem Address Bit 3
20	Port 8.2 External Interrupt #0 (L=Active)	70	Port 2.2 External Mem Address Bit 2
21	Port 8.1 Timer A #4 Input	71	Port 2.1 External Mem Address Bit 1
22	Port 8.0 Timer A #4 Output	72	Port 2.0 External Mem Address Bit 0
23	Port 7.7 Timer A #3 Input	73	Port 1.7 External Interrupt 5 (L=Active)
24	Port 7.6 Timer A #3 Output	74	Port 1.6 External Interrupt 4 (L=Active)
25	Port 7.5 Timer A #2 Input	75	Port 1.5 External Interrupt 3 (L=Active)
26	Port 7.4 Timer A #2 Output	76	Port 1.4
27	Port 7.3 Timer A #1 Input UART #2 /CTS & /RTS	77	Port 1.3
28	Port 7.2 Timer A #1 Output UART #2 Clock	78	Port 1.2
29	Port 7.1 Timer A #0 Input Timer B #0 Input	79	Port 1.1
	UART #2 RxD Serial Clock	80	Port 1.0
30	Port 7.0 Timer A #0 Output UART #2 TxD	81	Port 0.7
	Serial Data	82	Port 0.6
31	Port 6.7 UART #1 TxD	83	Port 0.5
32	Port 6.6 UART #1 RxD	84	Port 0.4
33	Port 6.5 UART #1 Clock	85	Port 0.3
34	Port 6.4 UART #1 /CTS & /RTS	86	Port 0.2
	UART #0 /CTS & CLKS	87	Port 0.1
35	Port 6.3 UART #0 TxD	88	Port 0.0
36	Port 6.2 UART #1 RxD	89	Port 10.7 A/D Input #7
37	Port 6.1 UART #1 Clock		Key Interrupt #3 (L=Active)
38	Port 6.0 UART #1 /CTS & /RTS	90	Port 10.6 A/D Input #6
39	Port 5.7 Serial Ready Serial Clock Out		Key Interrupt #2 (L=Active)
40	Port 5.6 External Mem Address Latch	91	Port 10.5 A/D Input #5
41	Port 5.5 External Mem Hold (L=Active)		Key Interrupt #1 (L=Active)
42	Port 5.4 External Mem Hold (State=Pin 41)	92	Port 10.4 A/D Input #4
43	Port 5.3 External Mem BCLK		Key Interrupt #0 (L=Active)
44	Port 5.2 External Mem Read (L=Active)	93	Port 10.3 A/D Input #3
45	Port 5.1 External Mem WRH & BHE (L=Active)	94	Port 10.2 A/D Input #2
46	Port 5.0 External Mem WRL & WR (L=Active)	95	Port 10.1 A/D Input #1
47	Port 4.7 Chip Select #3	96	Analogue Vss (0V)
48	Port 4.6 Chip Select #2	97	Port 10.0 A/D Input #0
49	Port 4.5 Chip Select #1	98	Analogue Voltage Reference
50	Port 4.4 Chip Select #0	99	Analogue VCC (5V)
		100	Port 9.7 A/D Trigger Serial Input #4

Table data sourced from the Mitsubishi microcomputers M16C / 62 Datasheet

* Port 8.5 is configurable as an input port only, due to the fact that it shares its second function with the /NMI interrupt.

The two internal memory types:

Inside the M30624FGMFP's chip package, there are two different types of memory. Flash ROM (Read-Only-Memory) is where the final program is stored; it is written by the computer and then run by the microcontroller each time power is applied. Flash requires no power to keep its contents, but has a limited amount of write-cycles – that of approximately 1000. After the flash has been reprogrammed over this limit, the memory may malfunction or become unresponsive to programming commands and the chip will need replacing. There is 256Kb of Flash inside the M30624FGMFP and this cannot be written to by the microcontroller's program – hence the "Read-Only" name.

RAM (Random-Access-Memory) is the microcontroller equivalent of short-term memory in humans; this is cleared when the power is removed and stores current variable values and other "on-the-fly" data. The M30624FGMFP's Bootloader can run programs straight out of RAM and doing so during development will ensure that in all likelihood you will never wear out the Flash ROM.

RAM can be written to as many times as you want, as it has no (or an incredibly high) write-cycle lifespan.

The Internal Bootloader:

The Mitsubishi chip contains a special alternate program that can be executed instead of the master program held in the flash memory. This program, called a Bootloader, allows the chip to execute programs held in the internal RAM memory or external Flash. This is a major benefit - as described above it prevents wearing of the internal Flash. To enable the Bootloader, you must download the program into the Mitsmini in Boot Loader Mode (see *The Included Software* chapter).

When a program is downloaded into the RAM, resetting will cause the Bootloader to automatically run the new program. It should be noted that programs running in Bootloader mode are restricted in size, as there is only 20Kb of RAM as opposed to the 256Kb of Flash ROM.

Bootloaders also allow you to place your programs in external memory; they will read and run the commands when set up to do so. External memory should be connected to the proper memory address and control pins on the microcontroller.

The Watchdog Timer:

Inside the M30624FGMFP, the oscillator circuit's (timing interval provided by the 16MHz external quartz crystal on the Mitsmini board) output is connected to two other circuits; the clock input of the main processing system, and the Watchdog timing system. This Watchdog circuit is executed completely separate from the rest of the chip, connected to the main system by only a handful of enable and settings lines.

The Watchdog behaves just like a real guard dog, resetting the microcontroller if not reset after a set number of cycles have elapsed. The Watchdog and the *reset after x cycles* parameter is configured via programming – it is not automatically turned on at chip start-up.

The Watchdog is very useful when running programs that contain loops which may repeat infinitely or when a glitch causes the microcontroller to “hang” – pause its program execution. It counts silently in the background, and if not reset within a certain amount of time, it assumes that the program execution has halted and resets the entire chip.

CHAPTER 3: The Mitsmini Board

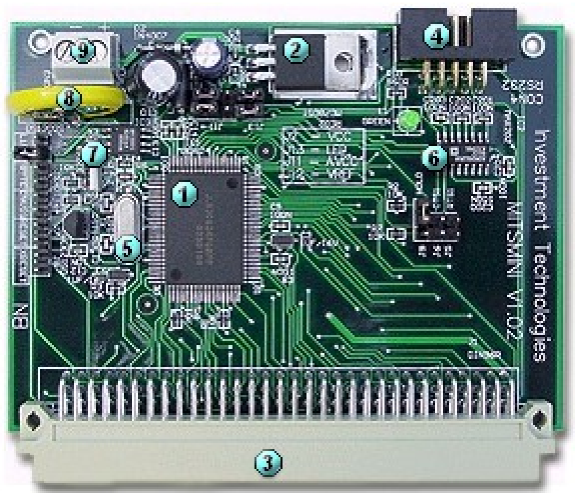
Overview of the Mitsmini board:

The Mitsmini board is based upon the Mitsubishi M16C microcontroller, instead of the (now) dated Atmel AT90S8535. This move has some large implications; namely the lack of support (and development environments) for the new micro, but has opened up a wealth of new features.

The M16C core is true 16-bit microcontroller, allowing twice the power and efficiency (when using 16-bit numbers) of the 8-bit AT90S8535 used in the ABC series. It has an astounding *87* I/O ports, allowing the control of up to 87 separate components at once. The Analogue-to-Digital converter is 8-channel 10-bit, creating very accurate measurements of analogue voltages.

Two Digital-to-Analogue converters are provided on-board, as well as 3 Serial UARTs and PWM capability.

The main aspects of the Mitsmini board are numbered on the picture below. The descriptions for each item are shown on the corresponding number below the picture.



1: Mitsubishi M16C Microcontroller

2: Onboard 5V regulator (9-32V input)

3: 96 Pin Connector for I/O, etc.

4: Serial Connector

5: 16 MHz Crystal for the Mitsubishi Core

6: Serial Transceiver UART IC for serial communications

7: 32.768 KHz for Real Time Clock chip

8: Lithium-Ion Battery for external Real Time Clock chip

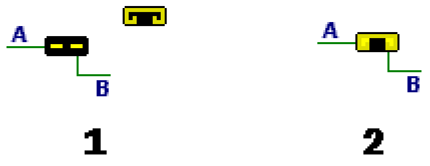
9: Power Input for the Mitsmini board (and on-board chips)

Setting up the Mitsmini's jumpers:

There are several groups of jumpers on the top of the Mitsmini board. It is important to set up these jumpers correctly before attempting to program or use the board.

A jumper consists of a pair of bare conductive pins, with a special connector designed to bridge them. This connector is a small piece of hollow plastic with a metal "U".

Jumpers can be quite fiddley and so it is easiest to remove or add jumpers with a pair of blunt tweezers.

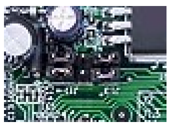


In the above diagram, the jumper is removed, and current cannot pass between points A and B. Once the jumper is inserted (as per diagram 2) current can be exchanged, creating an electrical pathway. Using this method, hardware “options” can be set on the board.

First, you must connect power to the board. You will need a “wall wart” power pack with an output voltage of 9-32V DC. Holding the Mitsmini board with the 96-pin connector down, the white power in screw-terminal is located above the battery (silver and yellow component). Connect your plugpack so that the positive wire is screwed into the correct terminal – the polarity is marked on the board. A diode protects the Mitsmini against reverse voltage, but it is best to get it right the first time (use a multimeter if you are unsure).

Section 1: Below the regulator

The jumpers in this section should be connected in a horizontal orientation.



Required Jumpers:

- **Power**

This jumper, the two pins from the top right, configures the board to use power from the on-board regulator. If this is not set, power can be applied externally via connector pin 1C (see *Description of connector pins* section).

- **VREF**

The VREF jumper supplies the reference voltage for the A/D converter. When the top-left two pins are jumpered, the A/D converter reference uses the 5V regulator. If you wish to use pin 2B (see *Description of connector pins* section) on the Mitsmini connector so you can apply an external reference voltage, jumper the second and third pin on the top-left of the jumper section.

- **AVCC**

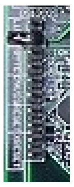
This jumper is identical in its configuration, except that it occupies the three bottom-left pins of the jumper section. The AVCC pin supplies the positive voltage for the A/D converter.

Optional Jumpers:

- **Power LED**

Although optional, this is a good idea for use in development. A single LED draws only 15ma or so, and gives a visual indication on when power is applied. These are the two bottom right pins of the jumper section.

Section 2: The left side of the board



This section of jumpers is located along the far-left side of the board, below the RTC backup battery. They have a small caption next to each jumper pair. The jumpers should be connected vertically.

Required Jumpers:

- **CNVSS**

When using internal memory, this jumper must be set. If you are running your program out of external memory, leave this jumper unconnected. This jumper should be use in conjunction with the "External Memory Enable" jumper.

Optional Jumpers:

- **LiBat**

Onboard the Mitsmini, to the top-left of the main Mitsubishi microcontroller, lays a small 8-pin IC. This IC is a Real-Time-Clock chip, connected to the microcontroller via an I2C connection (see *The I²C communication protocol*). To hold the time and date information in this chip while power is removed, this jumper must be set – connecting the on-board Lithium Ion battery to the RTC IC. This battery should last approximately 10 years when constantly connected.

- **Byte**

Since the Mitsmini's core is a 16-bit microcontroller, it makes sense that the external data bus(es) are also 16-bit. If you wish to interface an 8-bit device, setting this jumper will cause the micro to use an external 8-bit bus.

- **Reset**

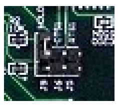
This is actually not a jumper. Bridging these two pins together momentarily will cause the micro to reset – this can be connected to a pushbutton for an external reset button.

- **XReset**

Short for "External Reset". Connecting this jumper will cause connector pin 29C (see *Description of connector pins* section) to act as an active low reset input. Applying GND momentarily to connector pin 29C when this jumper is set will cause the microcontroller to reset.

Section 3: To the right of the Microcontroller

Located underneath the UART chip, these jumpers control advanced UART and external memory functions.



Required Jumpers:

- **External Memory Enable**

This jumper tells the microcontroller whether to boot the program from internal or external Flash Memory (external memory must be connected to appropriate pins as shown on the table in the *Description of pins* section in the previous chapter). To select internal memory, the far-left middle and bottom pins must be jumpered, and for external memory the far-left middle and top pins must be jumpered.

Optional Jumpers:

- **CTS for UART1**

When designing programs that utilise the UART, you may wish to implement the RTS (Request to Send) and CTS (Clear to Send) signals. Jumpering the middle two pins in this section will connect the CTS signal to Port 6.0.

- **RTS for UART1**

As above, except the signal is RTS, and the jumper pins are the far-right top and bottom of the jumper section.

Description of Connector Pins:

The following table is the pin-out for the Mitsmini's connector. Assuming you are holding the board with the white connector downwards (with the component side up), the pins start from the bottom-right and run from right to left in lines of 32. Each line is given a descending letter reference, i.e. bottom row "A", middle row "B" and top row "C". On the Mitsmini board (as well as add-on boards with white 96-pin connectors), pin A1 is soldered into a square pad (hole), while the other pins are soldered to round holes.

Blue = Primary Function, Red = Second Function, Green = Third Function,
Pink = Forth Function, Grey = Fifth Function, Orange = Information/Notes

Mitsmini Connector	Micro Pin	Description/Function(s)
1A	100	Port 9.7 A/D Trigger Serial Input #4
2A	97	Port 10.0 A/D Input #0
3A	94	Port 10.2 A/D Input #2
4A	91	Port 10.5 A/D Input #5 Key Interrupt #1 (L=Active)
5A	88	Port 0.0
6A	85	Port 0.3
7A	82	Port 0.6
8A	79	Port 1.1
9A	76	Port 1.4
10A	73	Port 1.7 External Interrupt #5
11A	70	Port 2.2 External Mem Address Bit 2
12A	67	Port 2.5 External Mem Address Bit 5
13A	63	Port 3.0 External Mem Address Bit 8
14A	59	Port 3.3 External Mem Address Bit 11
15A	56	Port 3.6 External Mem Address Bit 14
16A	53	Port 4.1 External Mem Address Bit 17
17A	50	Port 4.4 Chip Select #0
18A	47	Port 4.7 Chip Select #3
19A	44	Port 5.2 External Mem Read (L=Active)
20A	41	Port 5.5 External Mem Hold (L=Active, Logic level forced via Mem Hold jumper)
21A	38	Port 6.0 UART #1 /CTS & /RTS
22A	35	Port 6.3 UART #0 TxD
23A	32	Port 6.6 UART #1 RxD
24A	29	Port 7.2 Timer A #1 Output UART #2 Clock
25A	26	Port 7.4 Timer A #2 Output
26A	23	Port 7.7 Timer A #3 Input
27A	20	Port 8.2 External Interrupt #0 (H=Active)
28A	17	Port 8.5 /NMI Interrupt (L=Active, held high via external 10k pull-up resistor)
29A	10	Port 8.7 External Secondary Timer Crystal In

Blue = Primary Function, Red = Second Function, Green = Third Function,
Pink = Forth Function, Grey = Fifth Function, Orange = Information/Notes

Mitsmini Connector	Micro Pin	Description/Function(s)
30A	5	Port 9.2 Serial Out #3 Timer In #2
31A	2	Port 9.5 Serial Clock #4 A/D Extended Input #0 RTC Chip I²C Data I/O
32A	1	Port 9.6 Serial Out #4 A/D Extended Input #1 RTC Chip /Reset
1B	N/A	VCC
2B	98	Analogue Voltage Reference (AREF) (See Jumper section)
3B	95	Port 10.1 A/D Input #1
4B	92	Port 10.4 A/D Input #4 Key Interrupt #0 (L=Active)
5B	89	Port 10.7 A/D Input #7 Key Interrupt #3 (L=Active)
6B	86	Port 0.2
7B	83	Port 0.5
8B	80	Port 1.0
9B	77	Port 1.3
10B	74	Port 1.6 External Interrupt 4 (L=Active)
11B	71	Port 2.1 External Mem Address Bit 1
12B	68	Port 2.4 External Mem Address Bit 4
13B	65	Port 2.7 External Mem Address Bit 7
14B	60	Port 3.2 External Mem Address Bit 10
15B	57	Port 3.5 External Mem Address Bit 13
16B	54	Port 4.0 External Mem Address Bit 16
17B	51	Port 4.3 External Mem Address Bit 19
18B	48	Port 4.6 Chip Select #2
19B	45	Port 5.1 External Mem WRH & BHE (L=Active)
20B	42	Port 5.4 External Mem Hold (State=Pin 41)
21B	39	Port 5.7 Serial Ready Serial Clock Out
22B	36	Port 6.2 UART #1 RxD
23B	33	Port 6.5 UART #1 Clock RTC Chip I²C Clock
24B	30	Port 7.0 Timer A #0 Output UART #2 TxD Serial Data
25B	27	Port 7.3 Timer A #1 Input UART #2 /CTS & /RTS
26B	24	Port 7.6 Timer A #3 Output
27B	21	Port 8.1 Timer A #4 Input
28B	18	Port 8.4 External Interrupt #2 (H=Active)
29B	11	Port 8.6 External Secondary Timer Crystal Out
30B	6	Port 9.1 Serial In #3 Timer In #1
31B	3	Port 9.4 D/A Converter 1 Timer B Input #4
32B	N/A	GND
1C	N/A	Power (Used as power input when onboard regulator disabled – see Jumper section)
2C	N/A	AVCC (Only active when jumper set – see Jumper section)
3C	96	Analogue Vss (0V)
4C	93	Port 10.3 A/D Input #3
5C	90	Port 10.6 A/D Input #6 Key Interrupt #2 (L=Active)
6C	87	Port 0.1
7C	84	Port 0.4
8C	81	Port 0.7
9C	78	Port 1.2
10C	75	Port 1.5 External Interrupt 3 (L=Active)
11C	72	Port 2.0 External Mem Address Bit 0
12C	69	Port 2.3 External Mem Address Bit 3
13C	66	Port 2.6 External Mem Address Bit 6
14C	61	Port 3.1 External Mem Address Bit 9
15C	58	Port 3.4 External Mem Address Bit 12
16C	55	Port 3.7 External Mem Address Bit 15
17C	52	Port 4.2 External Mem Address Bit 18
18C	49	Port 4.5 Chip Select #1
19C	46	Port 5.0 External Mem WRL & WR (L=Active, held high via external 10k pull-up resistor)
20C	43	Port 5.3 External Mem BCLK
21C	40	Port 5.6 External Mem Address Latch
22C	37	Port 6.1 UART #1 Clock
23C	34	Port 6.4 UART #1 /CTS & /RTS UART #0 /CTS & CLKS
24C	31	Port 6.7 UART #1 TxD
25C	28	Port 7.2 Timer A #1 Output UART #2 Clock
26C	25	Port 7.5 Timer A #2 Input
27C	22	Port 8.0 Timer A #4 Output
28C	19	Port 8.3 External Interrupt #1 (H=Active)
29C	12	/RESET (L=Reset, requires XReset jumper to be connected – see Jumper section)
30C	7	Port 9.0 Serial Clock #3 Timer in #0
31C	4	Port 9.3 D/A Converter 0 Timer B Input #3
32C	N/A	GND

Table data sourced from the Mitsubishi microcomputers M16C / 62 Datasheet

CHAPTER 4: Bits, Bytes and Communication

Internal Registers:

When data is being manipulated inside the microcontroller, it is placed inside a register. A register (shorthand for “shift register”) is a special hardware circuit that can hold a single byte of information in binary form. In an 8-bit microcontroller, a register can hold a value between binary 00000000 (decimal 0) and binary 11111111 (decimal 255). If a larger number is being calculated/manipulated, two registers can be placed together to hold a number between decimal 0 and 65535.

Since the M30624FGMFP is a 16-bit microcontroller, the registers can hold 16 bits (each bit is a single digital “switch” that can hold a 0 or 1 value) by itself – giving a range of decimal 0 to 65535 – or, like the 8-bit micro, two can be strung together to form a 32-bit number between decimal 0 and 4294967295.

When performing an operation, the microcontroller’s processor uses registers. For example, if the microcontroller was instructed to multiply 16 with 3, and then subtract 8, it would:

- Clear a free register
- Store the value 16 to the previously cleared register
- Clear a second free register
- Store the value 3 to the second cleared register
- Multiply the two registers – store result in the first register
- Clear the second register
- Store 8 in the second cleared register
- Subtract the second register from the first
- Store the result in the first register

Even though the task is only two simple mathematical operations, the microcontroller requires 9 clock cycles to perform it. The Mitsubishi micro runs at 16MHz – 16 million cycles per second so each instruction would take approximately $62.5^{*10^{\wedge-9}}$ seconds (0.0000000625 seconds per cycle) – so the 9 cycles would take approximately 5.6 millionths of a second to complete.

There are over a hundred device registers inside the M30624FGMFP, although some are reserved and cannot be read or written. Each function (such as the A/D) has one or more registers – in the case of the A/D, there are several control registers to activate the A/D channels, and several registers to read back the A/D results on each channel.

From a programming point of view, only several registers are used often. R0 to R3 are data registers used for arithmetic (mathematical) operations, PORT0 to PORT10 and PIN0 to PIN10 registers control the I/O pull-up resistors (see *Pull-up and pull-down resistors*) and values. Other registers, such as the A/D control and value registers, are used less often, but are equally important.

Several other registers are used “behind the scenes” by the microcontroller’s CPU. The PC (program counter) register is 20 bits long and holds the address (memory location) of the current program instruction, and the Flag register holds such information as whether the previous arithmetic operation resulted in a result of 0. The address registers A0 and A1 (used for address-related commands) and the data registers R0 to R3 (used for arithmetic calculations) come in two “banks”. While R0 may contain one value in bank 1, it may also simultaneously hold a different value in the second bank. Bit 4 of the flag register selects between the two register banks; if the flag is set to 0 the first bank is selected and if the flag is set to 1 the second register bank is selected. By switching between these banks, you can have the equivalent of six data registers and four address registers, although all the registers switch banks at the same time; you can’t switch the bank of a single register.

The mathematic registers R0 to R3 are all 16-bit, but R0 and R1 can be broken into two 8-bit sections, labelled R0H or R1H (for high, bits 8-16 of the register) and R0L or R1L (for low, bits 0-7 of the register).

The PORT, PIN and DDR registers:

The DDR register controls the function of the port. Setting a DDR bit high will set the corresponding port bit as an output, while a low value will configure the pin as an input. Each port has its own DDR register, named DDR0 for port 0, DDR1 for port 1, and so on.

To read the value of an input pin, you must use the PIN register. For example, if Port 0.3 was an input; you would read its value from third bit of the PIN0 register. When the pin is configured as an input, the PORT register controls the pull-up on that port pin. As with the DDR register, there is a separate PIN register for each port.

Outputs use only the PORT and DDR registers (the PIN register does not affect the port’s function). Setting a bit high on a PORT register will set the corresponding port pin high.

Mode	PORT	PIN	DDR
Output (High)	1	N/A	1
Output (Low)	0	N/A	1
Input (No Pull-up)	0	(Pin Value)	0
Input (Pull-up Enabled)	1	(Pin Value)	0

You should be very careful when switching an output pin to an input pin, as the PORT register may be set to an unexpected state, causing problems in the program’s execution.

Port 8.5 does not have a bit in the DDR8 register; this pin shares its second function with the /NMI interrupt and has no internal output circuitry.

Microcontroller Interrupts:

When programming, there will be times when you wish you could execute calculations, and then automatically branch off into a subroutine when a condition is met, without having to waste extra MCU clock cycles checking the status of a pin or variable. Interrupts were created to solve this; an interrupt will pause the program when its condition is met (running a pre-defined subroutine) before returning to its place inside the original program.

There are two types of interrupt, Software and Hardware. Hardware interrupts are "fired" when an interrupt pin on the microcontroller changes from high to low, or a peripheral's (such as the A/D converter) status changes. Software interrupts include timer overflows, etc.

You will need to enable interrupts in your program, and then enable each individual interrupt you intend to use (to prevent other unused interrupts from firing and creating errors).

The following is a table showing the main interrupts supported by the Mitsubishi micro. "L" is an abbreviation of Low Logic Value (digital 0), and "H" is an abbreviation of High Logic Value (digital 1). Some of the interrupts are assigned to a software interrupt number, listed in the *INT #* column. You can fire these interrupts via your program, allowing you to manually run the same subroutines that the corresponding hardware interrupt executes.

Type	INT #	Abbreviation	Name	Condition
Software		UND	Undefined Instruction	UND Instruction is executed.
Software		OFL	Arithmetic Overflow	INT0 command is executed after an arithmetic operation causes an overflow.
Software	0	BRK	BRK Execution	BRK Instruction is executed.
Software	N/A	INT (0 to 63)	Software Interrupt	INT command is executed. Software interrupts 0-31 are shared with some peripheral interrupts.
Hardware		/RESET	Reset Command	Occurs when L is placed on the /RESET pin.
Hardware		/NMI	NMI Interrupt	Occurs when L is applied to the /NMI pin.
Hardware		WDT	Watchdog Timer	Fired when the Watchdog timer exceeds its limit (see Watchdog section).
Hardware		/KI (0 to 3)	Key Interrupts	Fired when one of the 4 external Key Interrupt pins' logic changes from H to L.
Hardware	14	ADC	A/D Conversion	Fired when the A/D performs a conversion.
Hardware	17	UART 0 Trans	Serial Transmission	Occurs when UART0 transmits data.
Hardware	18	UART 0 Rec	Serial Reception	Occurs when UART0 receives data.
Hardware	19	UART 1 Trans	Serial Transmission	Occurs when UART1 transmits data.
Hardware	20	UART 1 Rec	Serial Reception	Occurs when UART1 receives data.
Hardware	15	UART 2 Trans	Serial Transmission	Occurs when UART2 transmits data.
Hardware	16	UART 2 Rec	Serial Reception	Occurs when UART2 receives data.
Hardware	21	TA 0	Timer A	Fires when one of the various Timer A events occur.
Hardware	22	TA 1		
Hardware	23	TA 2		
Hardware	24	TA 3		
Hardware	25	TA 4		
Hardware	26	TB 0	Timer B	Fires when one of the various Timer B events occurs.
Hardware	27	TB 1		
Hardware	28	TB 2		
Hardware	5	TB 3		
Hardware	6	TB 4		
Hardware	7	TB 5		
Hardware	29	/INT 0	External Interrupt Pins	Occurs when one of the 6 external interrupt pins changes from H to L.
Hardware	30	/INT 1		
Hardware	31	/INT 2		
Hardware	4	/INT 3		
Hardware	9	/INT 4		
Hardware	8	/INT 5		

Table data sourced from the Mitsubishi microcomputers M16C / 62 Datasheet

Introduction to Digital Communication:

The RS-232 serial communication protocol is a standard for device-to-device communications. Most computers have a RS-232 communication port - also known as a COM or Serial port - although some modern computers are forgoing such "legacy" connections as the parallel (Printer) and serial ports in favour of USB.

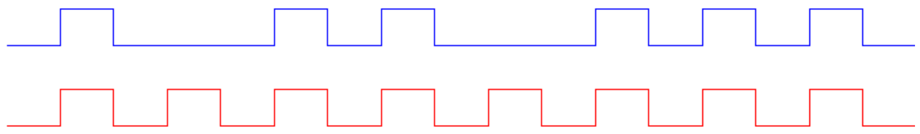
Digital communication takes the form of a series of 1's and 0's. A digital "1" is the logic voltage (in the case of the Mitsubishi M30624FGMFP chip, it's 5V, or 12V for RS-232 serial) and a "0" is ground (negative power terminal) on the Mitsmini, or $^{-}$12V for RS-232 serial. To translate signals from a series of 1's and 0's, a language called binary is used.

Binary groups data into a byte consisting of several bits, where a bit is a single piece of digital data – either 1 or 0. These bits – similar to miniature light switches - are then translated into a decimal number between 0 and 255. This number is converted to one of the 255 different standard characters from the ASCII table. When several bytes are converted into ASCII and grouped, they form numbers, words and symbols.

To transmit data between devices, the characters are first translated into binary, and sent along a wire to the receiving device, which then converts them back into ASCII. In some cases (such as the microcontrollers) the binary will form a lot of nonsense words and strange characters because it is not text communication, but machine code. Machines do not use ASCII characters for programs and instructions, rather the raw binary.

To ensure that both devices are in sync when receiving data, a clock signal is used. The clock line is a second wire that transmits pulses at a regular interval. When a clock pulse is received, the receiving device checks to see if the data line is high (logic 1) or low (logic 0).

Below is an example of a piece of digital communication (8 bits). The clock line is shown in red, while the data is shown in blue.



1	**0**	**1**	**1**	**0**	**1**	**1**	**1**
$1^{2\wedge7}$	$0^{2\wedge6}$	$1^{2\wedge5}$	$1^{2\wedge4}$	$0^{2\wedge3}$	$1^{2\wedge2}$	$1^{2\wedge1}$	$1^{2\wedge0}$

The above data line has transmitted the binary information "10110111" to the receiving device. This is enough data to form a byte of 8 bits, making the decimal equivalent of 183 - ASCII character full stop (period). Communication with a clock signal is useful when the communication speed varies, but if the speed is known, a clock signal is unnecessary. Devices communicating with a computer (via serial link) are configured to run at a preset transmission speed; data is sent in bytes, with a "start bit" (high bit) signalling the start of a byte transfer, and a "stop bit" ending it. The amount of start bits and stop-bits are customized by the device's software.

To prevent incorrect information (corrupt data) from being used by the receiving device, a checksum is added to the end of each block of data. A checksum is a single byte, the value of which is the sum of all the bytes sent in the previous block. Once the checksum has been transferred, the receiver can check to ensure the data it has received is valid – if not, it requests the data again.

The I^{2}C Communication Protocol:

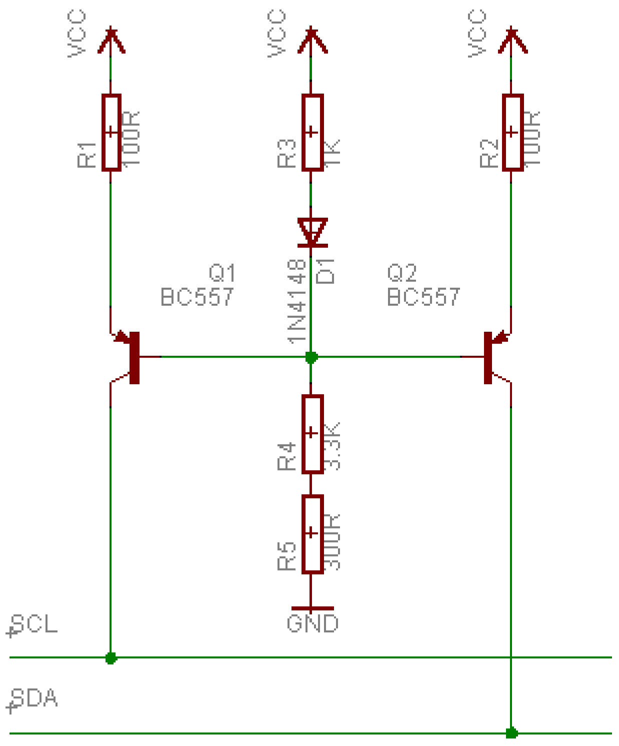
The *Inter-Integrated Circuit Bus* is a popular method of data transmission to and from IC's in a circuit. Designed by Phillips, its original purpose was to create a simple interface between IC's in their TV circuits to reduce PCB complexity; many manufacturers have since implemented the technology into everyday chips. The I^{2}C bus utilizes two wires that connect the serial clock (SCK) and serial data (SDA) pins on each I^{2}C enabled chip to each other. Each I^{2}C chip can be in either master or slave mode, although it is most common to have a single master device controlling one or more slaves. There can be a maximum total of 112 devices on each bus, each with a unique address.

A master device (such as a microcontroller) would send data down an I^{2}C bus to a slave device (such as a port extender chip), which would then react to the sent data. The I^{2}C bus has the advantage of being simple to implement (only two wires between components) and manufacturers produce many different types of low-cost I^{2}C enabled chips. I^{2}C chips will not work with the RS-232 (standard serial) communication protocol and vice-versa. Both lines (SCK and SDA) must be pulled high by two 4.7k resistors to the 5V rail.

Because there are several devices connected to a single bus, you must address the device that you wish to communicate with. Datasheets that come with the I^{2}C-enabled chip will state the address as a 7-byte binary number (for example "1011011"). Devices such as I^{2}C EEPROMS made by a single manufacturer will all have the same address, leading to problems when several of the same chips are placed on the one single bus. To overcome this, there may be several address bits that are user selectable, indicated by a letter instead of a digit in the address (e.g. "1101xx1"). These chips will come with pins labelled "A" followed by a number (such as "A0" or "A3"). Connecting the external "A" pins to GND or VCC will cause the address bits to become 0 or 1's, preventing conflicts.

The real-time-clock chip on the Mitsmini board is connected to the Mitsubishi microcontroller via an I^{2}C bus – see *Description of connector pins*.

If your I^{2}C bus is very long, an I^{2}C terminator is required to extend the maximum length (from master device to last slave device) to ~80cm. The original schematic was designed by Detlef Queck, and recreated by myself in the Eagle schematic editor.



The January 2004 edition of *Silicon Chip* has a “Picaxe-18X 4-Channel Datalogger” project, which has a more detailed explanation of the I²C bus (that cannot be added here due to copyright reasons).

CHAPTER 5: Connecting your Mitsmini

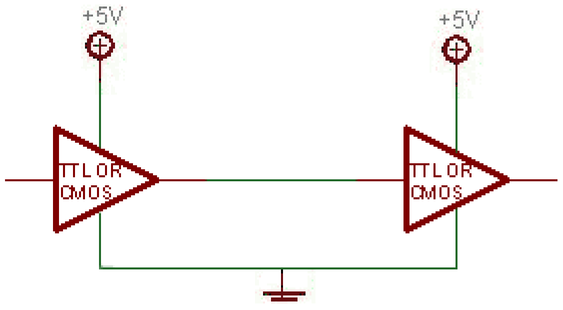
Sinking and Sourcing current:

A device is said to be sinking current when the current flows from the positive (5V) line, through the device to ground, while sourcing current is the opposite (current goes from the positive line, through the device and down to ground). When devices are driven by current sinking, they are activated by “reverse-logic”, where a low (logic 0) signal will switch on the device, and a high will deactivate it.

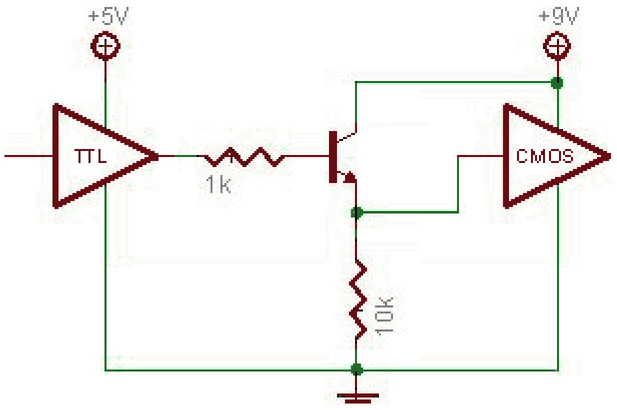
Interfacing to other ICs:

It is possible to directly interface your Mitsmini’s output to other ICs, without any other hardware. The Mitsmini uses a 5V logic interface that is compatible with almost every IC on the market. There are two main types of IC's; CMOS (Complementary MOS) chips, upon which the M30624FGMFP microcontroller is based, and the older TTL (Transistor-Transistor Logic) IC technology.

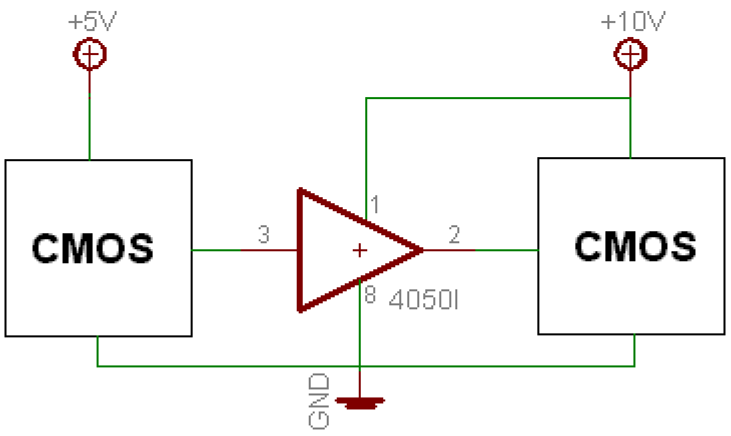
TTL IC’s run on 5V power supplies, while most (NOT the M30624FGMFP microcontroller) CMOS IC’s will run on a supply ranging between 3 and 15V. The Mitsubishi micro will directly connect to both types, provided that both chips share the same ground and voltage as each other. A chip-to-chip interfacing example is shown below.



If you are trying to interface a TTL IC which runs on a lower voltage than the CMOS IC it connects to, a simple interface circuit may be required (shown below).

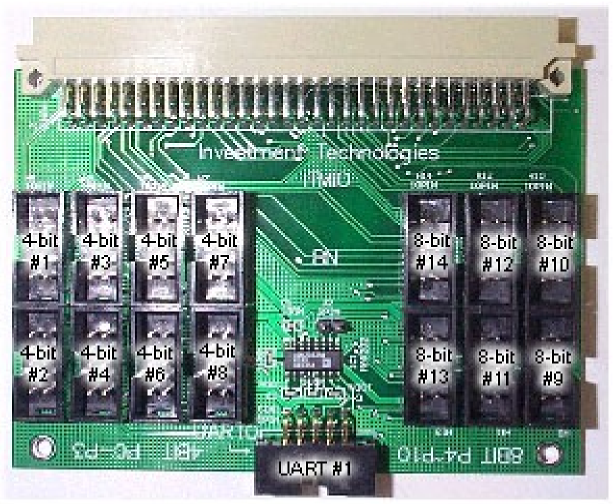


Similarly, if you want to connect the M30624FGMFP to a CMOS chip with a higher supply voltage, you will need to use a modified version of the above example (using the M30624FGMFP instead of a TTL IC and a pull-down resistor appropriate for the second IC’s supply voltage), or using a 4050 buffer IC as shown below.



Connecting to devices via the I/O Expander board:

Austrol’s I/O expander board connects to your Mitsmini via a 2-way or 5-way bus board and its 96-pin connector. It contains eight 4-bit and six 8-bit connectors onboard, plus a second RS-232 serial UART (see *Introduction to Digital Communication* section). Assuming you are holding the board with the 96-pin connector upwards, the 4-bit ports are located to the left, the 8-bit ports to the right, and the serial port in the bottom-centre. Each of the 4-bit and 8-bit ports are marked as “H1”, “H2”, etc. on the I/O expander’s board.



The serial port (microcontroller UART1)

The pins on the serial connector run from right-to-left, with pin 1 at the top-right (square pad). The pin-out for the I/O serial connector is as follows.

Connector Pin	Micro Pin	Function
1		
2	32	Serial Receive (From Computer)
3	31	Serial Transmit (From Microcontroller)
4		
5	N/A	Ground
6		
7	*See Text*	RTS (Request to Send) signal
8	*See Text*	CTS (Clear to Send) signal
9		
10		

Unlike the serial connector on the Mitsmini board, the RTS and CTS signals are connected together via the jumper (if fitted) on the I/O expander board. Some computer programs may require these signals but the jumper may be omitted if desired.

The 4-bit connectors

If you have purchased add-on boards from Austrol that were originally designed for the ABCmini board (and have small black connectors instead of the 96-pin white one found on the Mitsmini), you should plug these boards into the eight 4-bit connectors. If you have not purchased the add-on board connector cables from Austrol, you can make your own by connecting two female 10-pin IDC connectors “straight through” with ribbon cable. The four-bit connectors each have 5 pins connected to GND, one pin connected to VCC, and four pins connected to ports on the M30624FGMFP microcontroller. The pin-outs for each of the connectors are shown below. In each case, the connector pins start from the top-left of each connector, running from left-to-right.

An exception is the LED output board. Due to the limited current that can be supplied by the Mitsubishi chip’s pins, the 3ma will not have sufficient power to drive the LEDs directly, which use about 15-20ma each. Attempting to interface the LED board with the Mitsmini may cause permanent damage to the M30624FGMFP.

Connector 1

Connector Pin	Micro Pin	Function
1	N/A	VCC (5V)
2	N/A	Ground (0V)
3	88	Port 0.0
4	N/A	Ground (0V)
5	87	Port 0.1
6	N/A	Ground (0V)
7	86	Port 0.2
8	N/A	Ground (0V)
9	85	Port 0.3
10	N/A	Ground (0V)

Connector 2

Connector Pin	Micro Pin	Function
1	N/A	VCC (5V)
2	N/A	Ground (0V)
3	84	Port 0.4
4	N/A	Ground (0V)
5	83	Port 0.5
6	N/A	Ground (0V)
7	82	Port 0.6
8	N/A	Ground (0V)
9	81	Port 0.7
10	N/A	Ground (0V)

Connector 3

Connector Pin	Micro Pin	Function
1	N/A	VCC (5V)
2	N/A	Ground (0V)
3	80	Port 1.0
4	N/A	Ground (0V)
5	79	Port 1.1
6	N/A	Ground (0V)
7	78	Port 1.2
8	N/A	Ground (0V)
9	77	Port 1.3
10	N/A	Ground (0V)

Connector 4

Connector Pin	Micro Pin	Function
1	N/A	VCC (5V)
2	N/A	Ground (0V)
3	76	Port 1.4
4	N/A	Ground (0V)
5	75	Port 1.5
6	N/A	Ground (0V)
7	74	Port 1.6
8	N/A	Ground (0V)
9	73	Port 1.7
10	N/A	Ground (0V)

Connector 5		
Connector Pin	**Micro Pin**	**Function**
1	N/A	VCC (5V)
2	N/A	Ground (0V)
3	72	Port 2.0
4	N/A	Ground (0V)
5	71	Port 2.1
6	N/A	Ground (0V)
7	70	Port 2.2
8	N/A	Ground (0V)
9	69	Port 2.3
10	N/A	Ground (0V)

Connector 6		
Connector Pin	**Micro Pin**	**Function**
1	N/A	VCC (5V)
2	N/A	Ground (0V)
3	68	Port 2.4
4	N/A	Ground (0V)
5	67	Port 2.5
6	N/A	Ground (0V)
7	66	Port 2.6
8	N/A	Ground (0V)
9	65	Port 2.7
10	N/A	Ground (0V)

Connector 7		
Connector Pin	**Micro Pin**	**Function**
1	N/A	VCC (5V)
2	N/A	Ground (0V)
3	63	Port 3.0
4	N/A	Ground (0V)
5	61	Port 3.1
6	N/A	Ground (0V)
7	60	Port 3.2
8	N/A	Ground (0V)
9	59	Port 3.3
10	N/A	Ground (0V)

Connector 8		
Connector Pin	**Micro Pin**	**Function**
1	N/A	VCC (5V)
2	N/A	Ground (0V)
3	58	Port 3.4
4	N/A	Ground (0V)
5	57	Port 3.5
6	N/A	Ground (0V)
7	56	Port 3.6
8	N/A	Ground (0V)
9	55	Port 3.7
10	N/A	Ground (0V)

The 8-bit connectors

Six 8-bit connectors are located on the I/O board, each connected to a full 8-bit port on the M30624FGMFP microcontroller. Each of these connectors allow for more complex or larger devices to be connected to the Mitsmini. As with the 4-bit connectors, the pins start from the top-left and run from left-to-right.

Connector 9		
Connector Pin	**Micro Pin**	**Function**
1	N/A	VCC (5V)
2	54	Port 4.0
3	53	Port 4.1
4	52	Port 4.2
5	51	Port 4.3
6	50	Port 4.4
7	49	Port 4.5
8	48	Port 4.6
9	47	Port 4.7
10	N/A	Ground (0V)

Connector 10		
Connector Pin	**Micro Pin**	**Function**
1	N/A	VCC (5V)
2	46	Port 5.0
3	45	Port 5.1
4	44	Port 5.2
5	43	Port 5.3
6	42	Port 5.4
7	41	Port 5.5
8	40	Port 5.6
9	39	Port 5.7
10	N/A	Ground (0V)

Connector 11		
Connector Pin	**Micro Pin**	**Function**
1	N/A	VCC (5V)
2	30	Port 7.0
3	29	Port 7.1
4	28	Port 7.2
5	27	Port 7.3
6	26	Port 7.4
7	25	Port 7.5
8	24	Port 7.6
9	23	Port 7.7
10	N/A	Ground (0V)

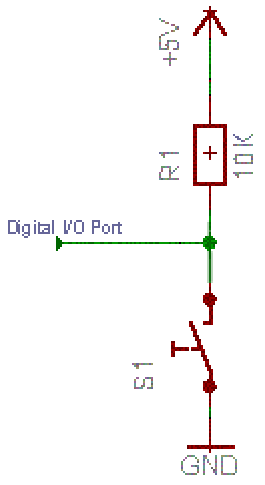
Connector 12		
Connector Pin	**Micro Pin**	**Function**
1	N/A	VCC (5V)
2	22	Port 8.0
3	21	Port 8.1
4	20	Port 8.2
5	19	Port 8.3
6	18	Port 8.4
7	17	Port 8.5
8	33	Port 6.5
9	12	/RESET
10	N/A	Ground (0V)

Connector 13		
Connector Pin	**Micro Pin**	**Function**
1	N/A	VCC (5V)
2	7	Port 9.0
3	6	Port 9.1
4	5	Port 9.2
5	4	Port 9.3
6	3	Port 9.4
7	2	Port 9.5
8	1	Port 9.6
9	100	Port 9.7
10	N/A	Ground (0V)

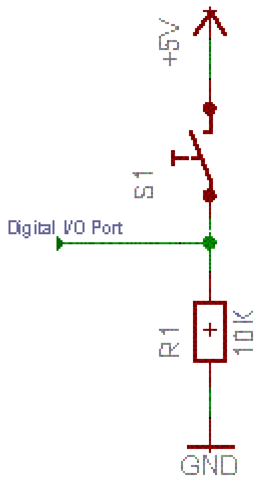
Connector 14		
Connector Pin	**Micro Pin**	**Function**
1	N/A	VCC (5V)
2	97	Port 10.0
3	95	Port 10.1
4	94	Port 10.2
5	93	Port 10.3
6	92	Port 10.4
7	91	Port 10.5
8	90	Port 10.6
9	89	Port 10.7
10	N/A	Ground (0V)

Pull-up and Pull-down resistors:

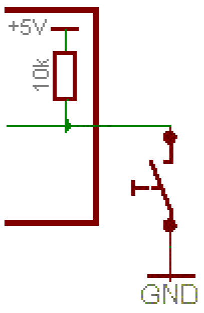
For the Mitsubishi M16C chip to register digital inputs correctly (this does not apply when reading from the A/D inputs), either an external or internal pull-up resistor must be used. A pull-up resistor maintains a positive voltage at the port input (logic 1) until dissipated by a short to ground (Logic 0). The standard value for a M30624FGMFP pull-up resistor is 10K ohms. A typical pull-up resistor connected with a pushbutton looks like this:



When the pushbutton in the circuit is not closed, the port will read logic 1 due to the trickle of voltage produced via the resistor. Pushing the button will make the port read logic 0, as the voltage at the resistor is dissipated to ground. The logic can be reversed by swapping the position of the pushbutton and resistors, as shown in the next picture:



This example will read logic 0 when not pushed, or logic 1 when pushed. The M30624FGMFP chip also includes internal 10K pull-up resistors on all ports (except port 8.5, see *The PORT, PIN and DDR registers*), which can be enabled via software. You will need to consult your programming environment's manual for the commands to do this. If you are using a sensor that changes resistance after a stimulus, such as a CDS cell light-sensor, you can place this in the position of the pushbutton and connect it to an A/D port for an exact reading or a standard logic port to give a high or low value after the sensors value exceeds a value.



Above is a rather simplistic view of the internal pull-up resistors. In practice, the port schematics are quite a bit more complex, as the transistors that turn on and off the pull-ups, logic circuits and data registers are not shown. When connecting sensors that use the Mitsubishi chip's internal pull-up resistors, you must connect the sensor to GND, not VCC. For example, a switch connected to the port must be connected directly to GND for the pull-ups to work. Below is the full internal circuit diagram of a standard I/O port, courtesy of the Mitsubishi M16C datasheet.

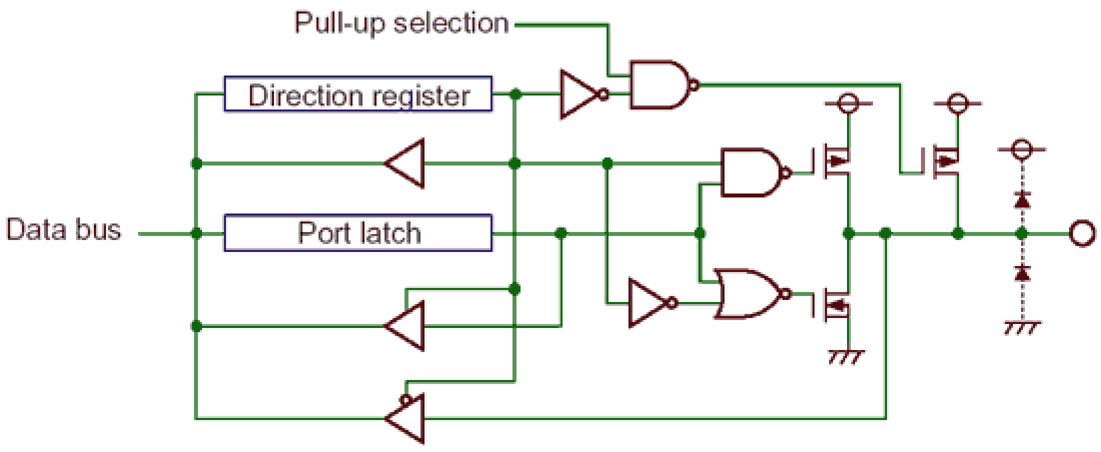
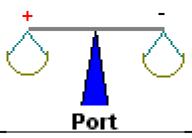
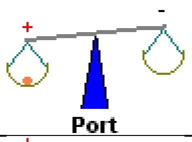
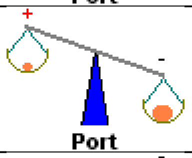
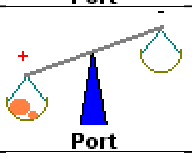
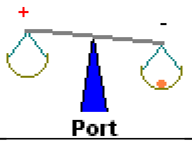
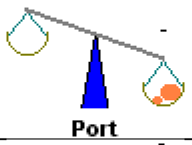
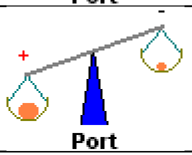


Image Source: Mitsubishi microcomputers M16C / 62 Datasheet

Microcontrollers use a "balance" system to determine whether a port is logic 0 or 1. A good analogy of this is a set of old-fashioned scales with positive at one end and negative at the other. The port is read as logic 1 if there is more current entering the microcontroller (from VCC) than leaving it (via GND). Even if the current is tiny (like the milliamps supplied via the pull-up resistors) the port will still read logic 1 so long as current exceeds the amount of current flowing to ground. Pressing the pushbutton on the pull-up example above forces all current to ground, outweighing the pull-up's milliamps of current and thus sending the port low.

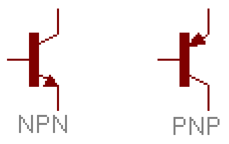
Condition	Graphical Representation	Pin Logic
High Impedance (no input)		Fluctuating between High and Low
Pull-up, no input		High
Pull-up, low logic input		Low
Pull-up, high logic input		High
Pull-down, no logic input		Low
Pull-down, low logic input		Low
Pull-down, high logic input		High

Transistors:

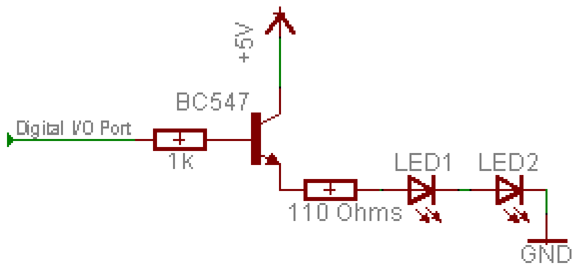
As previously mentioned, the ports on the Mitsmini can only drive devices at 5ma per port pin. While this is fine for direct IC-to-IC communications, a small hobby motor may take half an amp or more, and large devices may require several amps. Attempting to power such devices – even a standard LED requires at least 25ma – would result in the immediate destruction of your microcontroller and possibly the surrounding components.

The easiest way to interface the ports with external devices is to use standard or MOSFET transistors. Transistors can serve two main purposes, as a switch or as an amplifier. When used in conjunction with the Mitsubishi microcontroller the transistors act as switches to turn on or off external devices.

Standard Transistors



Standard transistors come in thousands of models, each with different specifications but they can be grouped into two types, PNP and NPN. In the examples below, the BC547 NPN signal transistor is used. The BC547 has a maximum current load of 100ma, enough to drive approximately 4 normal LEDs with a current limiting resistor. Below is an example of two LEDs being driven from a BC547.



You will need to calculate the required transistor turn-on resistor value (1k in the above example) yourself as each transistor has its own specifications. This information can be found on the transistor's datasheet, which is either supplied with the transistor or it (usually) can be downloaded off the Internet.

To calculate the required resistance for the LEDs, Ohms law must be used. For those not familiar with ohms law, it states that V=IR, I=V/R and R=V/I (where V is volts, I is Amps and R is Ohms). In practical terms, this means that the missing value (V, I, or R) can be calculated via the formulas and the two other values.

A standard LED's current is typically around 25ma at about 1.5V for normal brightness. You should check your chosen LED's datasheet to obtain the specific values required.

The transistor's output in the above schematic is 5V. With two LEDs in series each consuming 1.5V each, this results in a 3V total voltage drop across the two LEDs. In practise this will differ slightly due to losses in the transistor, which is typically around 200mV or so.

With a 3Vdrop, that leaves 2V across the resistor. Due to all the parts being in series, the current flowing through each component is identical – 25mA. Since we now know two parts of the equation – the voltage across the resistor as well as the current flowing through the resistor – we can use Ohm's law to calculate the resistor's value:

V= I x R
R= V / I
= 2V / .025A
= 80 ohms

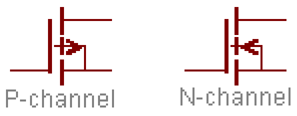
Since 100 ohms is a standard resistor value, it can be bought and used in the circuit. If you need a non-standard resistor value, you could add two resistors together in series (add the values), or just use the next standard value up or down, so long as the current stays within the acceptable range printed on your LED's datasheet.

An other thing you must be careful about when using resistors is the maximum power that can be dissapated through the resistor (measured in Watts). To find the power being dissapated in our above example, we must use the P=VI formula:

P = V x I
= 2 x .025
= .05 Watts

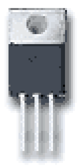
This is a very small amount of power, and can easily be dissapated by a standard ¼ watt (maximum) resistor.

MOSFET Transistors



While standard transistors will drive several LEDs or other small devices, the 100ma or so supplied by them is insufficient to drive large devices, such as large DC motors whose current requirements are usually several Amps. MOSFET transistors are real workhorses, some providing up to 60 Amps, depending on the specifications. Like normal transistors, MOSFETs come in two different types, P-channel and N-channel. In this segment, N-channel MOSFETs are used.

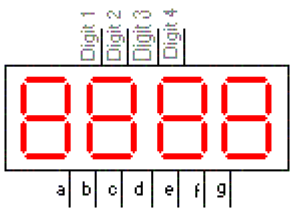
While most standard NPN and PNP transistors come in the TO-92 "half-moon" package, MOSFETs are typically contained in the TO-220 package as used for the Mitsmini's 5V power regulator.


TO-220 Package

Although your MOSFET may have specifications that allow a large amount of voltage and current to flow through it, each MOSFET is only able to dissipate a certain amount of heat before malfunctioning and burning out. To prevent this, you should use adequate heat-sinking to remove the excess heat. Your MOSFET specifications will have a *Diss* or P_{tot} measurement showing the maximum wattage of power that the transistor can handle.

Multiplexing LEDs:

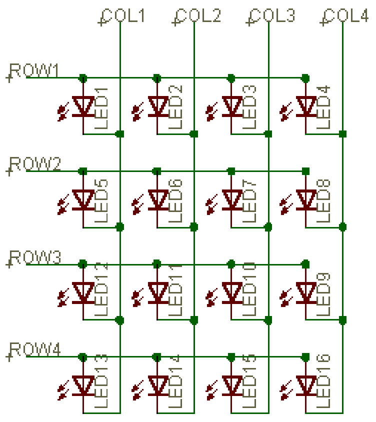
In many applications, you will need a circuit that can drive many LEDs at once. Regardless of the amount of I/O's a microcontroller has, you will inevitably run out – or it will become impractical to program so many ports. Consider the following display:



This has four LED digits, with the active digit selected by a negative voltage to the digit pins at the top (assuming it's a common cathode display), and a positive voltage on the segments at the bottom. You should use a transistor (not a buffer chip – *see above section for standard transistor use*) for the digit pins, as each digit will require a maximum of 140ma if all seven segments on at the same time – more than a TTL or CMOS buffer chip can supply.

Because of current limitations, each segment should be attached to a port either by another transistor or a logic buffer chip. This is compact and easy to use, but has the disadvantage of only being able to light one digit at a time. Fast microcontrollers (such as the Mitsubishi chip used in the Mitsmini) can overcome this problem using a method called multiplexing. When a device is multiplexed, it uses limited pins to control multiple devices by scanning the devices many times per second.

Each segment's transistor or logic buffer is connected to a separate I/O pin, as is each digit pin's transistor. When programming, you should make the Mitsmini send Digit 1's transistor high and light the segments you want Digit 1 to have, then switch off Digit 1 and select Digit 2 and light it's segments. Continue this all the way through the number of digits in the group, lighting each digit for only a fraction of a second. When each digit it shown (or "refreshed") many times a second (15-30 refreshes per second) a phenomenon called "Persistence Of Vision" causes the display to appear as if all of the digits are on with the correct numbers, without any flickering – the same system employed on TV and monitor screens. If the refresh rate is too slow, the display will flicker. This method, when used with 7-segment displays, uses 8 I/O pins for one digit (one full port) plus one I/O pin for each additional digit.



The multiplexing method can be applied to dot-matrix displays as well (above). With this method, LED's are "addressed" individually. Writing the letter A on this 4x4 dot-matrix display would light the following LEDs:

Column 1	**Column 2**	**Column 3**	**Column 4**
Row 1	Row 1	Row 1	Row 1
Row 2	Row 3	Row 3	Row 2
Row 3			Row 3
Row 4			Row 4

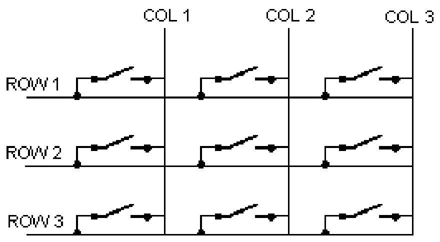
In this method, the columns take on the same role as the Digit Pins in the 7-Segment Display. A Column is held high, while the rows to be lit are held low (non-lit rows are neutral). Each column is turned on, and its LEDs lit until all rows have been show, in which case the cycle repeats for another “refresh” cycle.

LED’s can be purchased packaged in a “matrix” connection like this TC07-11HWA. This has internal connections like the LED matrix shown above. You can add several together to create a moving or static display.

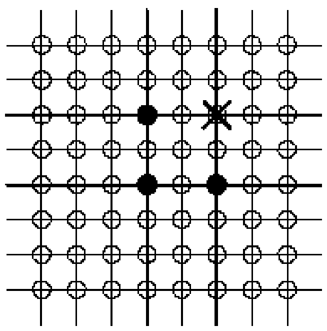
When using the LEDs in this manner, remember that transistors and appropriate current limiting resistors are required on both the rows and columns.

Multiplexing Pushbuttons:

Another form of multiplexing is used with pushbutton keypads. These have a number of buttons joined into a matrix, so that many buttons can be scanned using only a few ports. The M30624FGMFP is instructed to put a voltage on each of the rows, while another pin reads the column. A voltage of 5 volts (logic 1) indicates a pushed button, while no voltage (logic 0) indicates an un-pushed button. A typical push-button matrix looks like this:



There are two methods of scanning for pushed keys. One way is to send all of the row ports low and read the column ports (with pull-ups enabled), and then send all of the column ports high and read the row ports (with pull-ups enabled). After both scans, you can use code to determine the pressed key(s). This is fast, but can lead to “ghost” keys, when several keys are pressed:



Sourced from Atmel application notes (www.atmel.com)

Here the key marked with an "X" is determined as "pressed" by the microcontroller when using the first method, due to connections made while the other shaded keys are pushed. The second method is slower than the first. It requires sending the row and column ports low one-after-another to determine all the keys pressed with accuracy – but at a slower rate. To prevent ghost keys, you should connect a diode in series with each button on the matrix.

Remember that buttons and switches "bounce" after being pressed/released, giving multiple key-press signals if a delay of about 30ms between readings is not added.

Pulse Width Modulation:

Most microcontrollers have the facility to perform Pulse Width Modulation (PWM) commands to control the speed of motors. In short, PWM rapidly cycles an output port at a desired frequency at the supply voltage (5v) to speed up or slow down a motor. Since digital I/O's cannot output an analogue voltage (variable voltage), the workaround uses the fact that rapidly pulsing the motor at several kilohertz will cause it to rotate, with a speed proportional to the frequency (similar to a stepper-motor).

PWM has the added benefit of using the motor's full torque at variable speeds, unlike analogue voltages outputted by the M30624FGMFP's two internal digital to analogue converters. To connect a motor to a PWM output, attach one of the wires to a MOSFET transistor connected to an I/O port, and the other wire to the negative supply. To operate the motor, switch the port on with a PWM command.

You can also use PWM for a limited range of other devices. PWM commands to LEDs will change the apparent brightness, as the LED is cycled on and off too fast for it to build up to full power. This has the same effect as supplying the LED with a lower voltage than required to shine at full brightness, but does not change the supplied voltage (or current), which stays at 5v and ~20ma respectively. Most new mobile phones use pulse-width modulation to make the backlit screen fade in and out.

CHAPTER 6: Appendix

Glossary of Terms:

Some terms used in this documentation may be unfamiliar to new users or people previously unexperienced in microcontrollers/digital electronics and are described below.

Analogue IC

Analogue IC's (such as Op-Amps) are still used in electronics, but have given way to digital in recent times. Analogue signals are signals that vary between 0V and the supply voltage. Analogue is useful in audio electronics, but due to its susceptibility to noise it is no longer used for communication purposes.

Digital IC

Digital IC's are now widely used in modern electronics. Digital communication is more reliable than its analogue counterpart (more immune to noise) and subsequently is the choice for most modern communication methods. Digital data can be either high (digital "1") or low (digital "0"). A high digital signal is denoted by the logic voltage (usually 5V) used on the digital chip, while a low signal is 0V (ground).

I/O Port

A single pin on a microcontroller that can be set to act as an input (receiving data), or an output. A digital I/O port can be read by the microcontroller if it is in input mode, or it can be set high or low if it is configured as an output.

This can also refer to an entire group of I/O ports which make up a port byte. Such port groups usually carry names of descending alphabetical letters or numerical numbers.

Microcontroller

A digital integrated circuit that combines several items such as RAM, FLASH, Shift-Registers, etc. into a single package. A microcontroller can be programmed, and can control several devices simultaneously. There are many types of microcontrollers on the market, the most popular of which are from the brands of ATMEL and PIC.

PWM

Short for Pulse-Width-Modulation, this is mostly used for motor speed control. A PWM signal is a digital signal pulsed from high to low at a high frequency (several KHz), with a duty cycle (on and off-time) varied by the microcontroller. Varying the duty cycle causes a standard motor to rotate at a slower speed than normal at full torque.

Serial UART

A serial UART (Universal Asynchronous Receiver/Transmitter) is a method of Digital Communication, sending data to another device equipped with a UART.

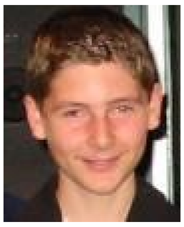
Surface Mount

Tiny components that are assembled via machines onto the top layer of a PCB. SM components are a standard in the electronics industry as they allow large circuits to be contained in a small area. Almost every component used in commercial electronics products today is surface mount. SM devices are very hard to solder or remove by hand, and so most hobbyists use through-hole components.

Through Hole

Standard components that are mounted on the top layer of a PCB, but are soldered to the underside via small holes (hence the name). Through hole technology is old and generally only used in hobbyist electronics or specialist components (such as regulators, MOSFETs, etc.), most of which require more power than fine surface-mount tracks can supply.

About the Author:



I have spent many hours compiling this documentation. Please feel free to email me with any comments/questions/abuse/ideas on this document, or the Mitsmini (and related equipment). Address all emails to "dean_camera@hotmail.com" (without the quotes). I would love to hear from you.

I would also like to hear about:

- **Stories**
- **Circuits**
- **Links to relevant Websites**
- **Technical Information**
- **Corrections**
- **Anything else related to this document/microcontrollers**

Check out my other projects at my website: http://home.pacific.net.au/~sthelena.

Disclaimer: