



JED MICROPROCESSORS PTY LTD

173 Boronia Rd, Boronia, (PO Box 30), Victoria, 3155, Australia.

Phone: +61 3 9762 3588, Fax: +61 3 9762 5499.

Web site: <http://www.jedmicro.com.au> email: jed@jedmicro.com.au

JED AVR575 single board computer/datalogger

(Ed Schoell, Jan 2, 2002)

Background

Over the last few years JED has designed a number of general purpose single board computers which have found application in control and communication applications as well as data logging to on-board memory. These boards have been programmable in C or BASIC and have used either the AVR family of microcontrollers or the Wilke BASIC Tiger range.

Some of these designs have been standard designs, being sold to "all comers" and appearing in print and www site advertising, but a larger range of boards have been designed by JED as customer-defined and customer-funded custom boards. Often custom boards have come out of standard boards, where a customer has wanted special functions added to standard boards, eg adding a GPS interface or a custom encoder interface to a standard board.

The AVR575 board described in this document is targeted at the real low end of the price curve ... it is a simple board with limited I/O designed to fit below the other AVR and Tiger boards, and provide a low cost, standard board which is customised by what JED loads onto the board during assembly, and without overhead to price it above what a fully customised boards would sell for, and allowing these customisations to be made without any development costs.

Thus, this board does have provision for a considerable amount of I/O, typically, only a small part of this is loaded in most applications. The size of the CPU FLASH memory and data storage memory capacity is customisable as well. About the only thing non-customisable on the AVR575 is the size of the printed circuit board itself: this is always 3.940" by 2.950".

CPU Options

The CPU on this board is an AVR 44 pin TQFP device, and can be chosen according to the required amount of CPU FLASH memory, CPU RAM memory and CPU EEPROM memory required, from the following options:

- **AT90LS8535-4AC**, giving 8K bytes of program FLASH memory, 512 bytes of CPU RAM, and 512 bytes of CPU EEPROM memory;
- **ATmega163-8AC**, giving 16K bytes of program FLASH memory, 512 bytes of CPU RAM, and 1K bytes of CPU EEPROM memory.

More CPUs are planned with this pin configuration, so these would work on this board as well.

CPU port functions

The pinout of this device has four, 8-line ports, and these are used in a variety of ways:

Port **PA0 ... PA7** is an input-only port in an 8535 CPU, and while usually used for analog inputs for the 8-channel Analog to Digital Converter, these inputs can be read as logic inputs readable as an 8-bit port as well.

Pins **PA0 ... PA3** can be used as keyboard inputs, if a keyboard scan option is installed.

(In an Atmega163 CPU, these ports can be driven as an output port as well, so special board loadings which needed more outputs could use this port to provide from 1 to 8, 5 volt swing, 20mA sinking outputs.)

Port **PB0 ... PB4** are used to address the chip decoder for the (optional) 1 to 8 EEPROM memory devices on the boards, and are also used to scan an (optional) keyboard, in conjunction with scan inputs on PA0 ... PA3. They are also used to scan an (optional) 8-way DIP switch, read into port PD3.

Port **PC0 ... PC2** and **PC4 ... PC7** are used to drive the control lines and 4 data lines to an (optional) LCD display, typically a 4 line by 20 character device which has the same mounting holes as the AVR575 circuit board.

PC3 is used as an (optional) 1-Wire interface to Dallas/Maxim 1-Wire system devices such as identification buttons, temperature sensors, or simple I/O devices across a 1-Wire network.

PD0, PD1 are serial RXD and TXD respectively, used for serial communications via the RS232 or (optional) RS485 serial interfaces. Port **PD2** is used as the TX-Enable for the RS485 interface, if installed. (It can also be used as one option of the Real Time Clock chip select.)

PD3 is the input for the dip switch scanner.

PD4 ... PD7 are four drive outputs to four (optional) MDT3055VL power FETs. Options for these are discussed below.

Port PA0 ... PA7 Analog/Digital Input system from connectors J9 and J10

There are eight general-purpose inputs for this card ... four are on connector J9, a 5 pin connector which has a ground on pin 1, and four on J10, a 6 pin connector with a ground on pin 1 and a Vcc 5 volt output on pin 6. These connectors are usually 0.15" spaced Phoenix screw terminals.

(An alternate load of the board puts an J11, an 11-pin, 0.1" spaced connector in place of J9 and J10, which allows Panduit/Molex/AMP crimp connectors to be used in place of screw terminal Phoenix types. These connectors allow for much faster system assembly and card replacement.)

(A second alternative is to load Phoenix plug-in screw terminal connectors for J9 and J10, with either vertical or right-angle mounts possible. This allows a board to be quickly replaced without unscrewing and screwing wires into terminals.)

All inputs via these two connectors go to a series of 8 socket strips (or holes into which fixed resistors can be soldered by JED). Each input has 7 or 8 holes for leads and this allows a variety of input signal processing to be performed.

Voltage reference for ADC: If Link 1 is jumpered and nothing is installed in location U2, the ADC converts with 1024 steps from 0 ... 5 volts Vcc. If U2, a 4.096 volt reference is installed the input range for the ADC is 0 to 4.092 volts for a count of 000 to 3FFh, and one bit is 4 mV exactly. 4.000 volt input returns a count of 1000 decimal.

Pullups: Resistors installed between pins 1 and 2 of each of the 8 strips allow for pullups on the input lines:

- If digital input mode is being used, this allows the input device to be a contact closure to ground, eg a switch or relay contact or NPN transistor or N-channel FET, eg in an opto-isolator, proximity switch or PLC output to be detected. Typical pull-up values to 5 volt are 4k7 or 1k5;
- If analog input mode is being used, this pullup can source current into a sensor or voltage divider. Thus a thermistor from the input pin to ground can be supplied with current. A variable resistive sensor (eg a fuel tank level sender) can also be measured in this way, with a top resistor installed with a similar value to the maximum resistance of the sensor, the voltage input is then 0 to 2.5 volts from min to max.

Pulldowns: Resistors installed between pins 2 and 3 of each of the 8 strips allow for pull-downs to ground on the input lines:

- In digital mode, this pulls an OFF line to ground and a positive voltage greater than 2.4 volts can be sensed as an ON. (Voltage input values are set by a following voltage divider ... see below;
- In analog mode, this supplies a load for an analog current source. If sensing a 4-20mA loop the value of 200 Ohms gives a voltage into the analog inputs of 0.8v to 4.0 volts. If sensing a lower current source such as an AD590 temperature to current transducer, a 1K resistor will give 2.73 volts with the 0 degrees Centigrade current of 2.73mA.

Voltage-divider resistor pair: Two resistors can be installed in positions 4 ... 5 (in series) and 6 ... 7 (to ground) of all eight positions. In both analog and digital applications, this pair divides whatever voltage is presented to the board by the ratio of the resistors. Values for these should be chosen so as not to load the input source excessively. If the input voltage range is within 0 ... 5.0 volts, no resistor to ground is needed.

So, if a 12 volt signal is being sensed a three to one divider is needed, so values of 7K5 and 15k are appropriate, and available from the 0.1% metal film range in 15ppm precision, giving a 0 ... 15 volt input range.

If a 24 volt digital signal is being sensed, 5% or 1% resistors from the more common range are usable, eg 22K series and 4K7 to ground gives a 28 volt maximum.

Clamp diodes are provided to ground and Vcc on all 8 lines, and these protect the microprocessor port PA input pins. If the current is limited to 100mA into these diodes, the input maximum voltage is more limited by thermal heating of the resistors. Pulse over-voltages and electrostatic hits will be well-protected with series resistors over 4K7 in value.

Keyboard option input resistors: If a scanned keyboard is desired, the lower four strips have positions for four resistors, typically 4K7, between positions 7 ... 8. These are used in conjunction with port pins PD0 ... PD3 to scan a 16-keyboard. (It is possible to use larger keyboards at the sacrifice of more analog/digital port inputs.) If keyboard resistors are installed on strips for inputs 1 ... 4, no other resistors should be installed in those strips.

Vcc-output: A 5-volt output (with EMC filtering) is provided on J10, pin 6 to power sensors or outside signal conditioning devices. Fahrenheit or Centigrade temperature sensors such as the National LM34 or LM35 can be powered from this pin to ground and the signal connected to an input. These output 10mV per degree, so give good resolution considering one bit of the ADC is 4mV with a 4.096 volt reference installed.

Port B systems for memory access, DIP switch and keyboard

Non-volatile memory system

Port B lines PB0 ... PB2 send a chip address to a 1 or 8 decoder which selects one of up to eight EEPROM devices, usually Atmel AT25F1024. These devices have 1Mbits of non-volatile data addressed as 128K bytes. Communications with these devices is via the SPI interface, and the access to the chip is activated by sending an address to PB0 ... PB2, and taking PB4 LOW.

The reason this chip is used on this board is that it is writable byte by single byte, with no page modes or ram sector buffers necessary. This allows records of logged data of any arbitrary length to be written, eg 11 bytes, 19 bytes or just one byte at a time. The memory has 10,000 write/erase cycles. After data is logged to the EEPROM memory, and after it is read out, it can be erased in blocks of 32K bytes at a time, or the whole of each chip can be erased with one instruction. Byte write time is 100 microseconds maximum, and erase time is 1.1 seconds maximum per sector. Writes can only be performed into erased sectors.

(Note; non-volatile RAM is available to the user in the RTC ... this RAM can be written and erased an infinite number of times, so is useful to hold a non-volatile "next address" pointer, rather than use EEPROM locations in either the AT25F1024 devices or the CPU EEPROM, both of which have a cycle limit. This is discussed below in the logger application guide.)

Option switch input

The 8-bit DIP option switch is read bit by bit, by running port PB0 ... PB2 down values from 0 to 7, and at each step reading data from port PD3. If a switch is closed, a LOW will be read for that bit as it is addressed by PB0 ... PB2. This value could be used for a network address, calibration data, logging time intervals, or whatever the imagination comes up with.

Keyboard scanner (optional)

A four by four keyboard matrix can be scanned by the AVR575.

The four rows of the key matrix are scanned with four FETs driven from port PB0 ... PB3. 4K7 ohm resistors installed in sockets L4, L5, L6 and L7 in holes 7 ... 8 read the columns 1, 2, 3, and 4 into PA0 ... PA3. (The pullup resistors on this port should be enabled in the CPU for these 4 bits). Taking a PB0 ... PB3 port line high selects a row, and if any key on that row is depressed, the corresponding bit is read as a LOW into PA0 ... PA3. If keyboard resistors are installed on strips for inputs 1 ... 4, no other resistors should be installed in those strips. Connector J12 is the keyboard connector, and this is physically compatible with the 3 x 4 telephone-style keyboards available from JED. The keyboard is conveniently located just below the LCD display, if it is also mounted on the rear of the board.

(It is possible to use larger keyboards at the sacrifice of more analog/digital port inputs.)

A typical keyboard system scan is initialised by an interrupt from an internal timer at, say, 10 millisecond intervals. A scan runs through the ports PB0 ... PB3, and reads back 4 values which can then be decoded to find which key was pressed. If a key is valid for successive 10 ms scans it is assumed to be a settled key and the value used. If it was only there for one scan, it could be a bounce and should be ignored.

None of the offered compilers have inbuilt keyboard scan functions, so users will have to write their own. (JED may cover this in a future application note.)

Port C LCD display interface (Optional)

Port PC0 ... PC2 and PC4 ... PC7 are used to drive the E, R/W* and E control lines and 4 data lines to an (optional) LCD display, typically a 4 line by 20 character device which has the same mounting holes as the AVR575 circuit board. The display is used in multiplex mode to save processor pins, where the 8-bit data byte is split into two 4-bit nibbles written consecutively.

The LCD display can either mount on the back of the AVR575, or can connect via a Panduit/Molex/AMP crimp connector.

Bias for the LCD is supplied by VR1, a trim pot used to set optimum viewing angle of the LCD. Jumper L11 is used to select either a positive BIAS or a negative bias, with respect to ground. L11 linked 1 ... 2 sets a positive bias, and L11 linked 2 .. 3 uses a negative 9 volts from the RS232 interface to supply a negative bias.

This particular port and pinout is supported by the standard AVR compilers which supply built-in drivers to suit, initialising the display and providing programmers with simple commands to put data to the screen.

Backlight to the display can be hardwired ON or can be controlled by F14, an output FET connected to CPU port PD7. It can be driven from regulated 5 volts or with an alternate board load with a larger value and power rated resistor, from input power (to save heat in the 5 volt regulator). Resistor values need to be chosen to trade off LED brightness, LED current rating and LED life, resistor heat dissipation and power consumption.

Expansion port option: If the LCD is not needed, the LCD connector J2 and seven pins of port C of the CPU can have user-defined (JED built) or user designed and built boards either piggy-backing onto the AVR575, or connected to the AVR575 via wires via a Panduit/Molex/AMP crimp connector.

Dallas/Maxim 1-Wire interface

PC3 is used as an (optional) 1-Wire interface to Dallas/Maxim 1-Wire system devices such as identification buttons, temperature sensors, or simple I/O devices across a 1-Wire network. This interface uses a bi-directional port pin on the CPU, and is driven by the standard C and BASIC compilers which generate support code for 1-Wire devices.

This port is protected against ESD hits on the 1-Wire line by a DS9503, 7.5 volt Zener diode with a “snap-back” characteristic, providing 27Kv of ESD protection, measured by method IEC801-2.

Serial I/O for the AVR575 from PD0 ... PD2

RS232

The UART in the CPU used on this board pins out to PD0 (RX in) and PD1 (TX out). To communicate via RS232, U5, a MAX221 device translates the CMOS levels from the CPU to RS232 levels, using a capacitive charge pump to generate the +/- 9 volt levels. This device has a useful feature for systems where power consumption must be minimal ... the charge pumps used for the RS232 voltage generation are controlled by a “connection detection” circuit inside the MAX221, which drops power consumption to under 1 microamp when there is no RS232 voltage source connected to the MAX221 RX input. When connected to valid RS232 signals the voltage generator powers up within 250 microseconds, allowing normal communications.

An RX and a TX line only are provided, and no connections are made to handshake lines. Users should provide their own loopback of handshake lines if they are needed. Jumper L12 allows the connections to the TX and RX lines to be swapped, allowing the AVR575 to look like a Data Terminal Equipment (DTE) to a Data Communications Equipment (DCE), eg a Modem or GSM data phone.

If this board is to be connected to a PC or laptop, eg to download logged data, it needs to look like the DCE (Modem) to the PC, which is always wired as a DTE.

The connections for L12 are as follows:

AVR575 is DCE	PC is DTE
TX on D9 pin 2	RX on Pin 3
Link L12 pin 1 ... 2	Link L12 pin 3 ... 4

AVR575 is DTE	Modem is DCE
TX on D9 pin 3	RX on Pin 2
Link L12 pin 1 ... 3	Link L12 pin 2 ... 4

Again, there are options for the connector for RS232 I/O. Instead of a 0.15" Phoenix screw terminal block, either horizontal or vertical Phoenix plug-in screw terminals, or J6, a 3 pin, 0.1" Panduit/Molex/AMP crimp connector can be used in place of screw terminal Phoenix types.

RS232 links: To support RS232, link **L1 pins 1 ... 2** must be linked to connect the RS232 receiver into the CPU.

Two links, called FORCE-ON and FORCE-OFF are solder jumpers on the back of the board, one of which must be connected to force the RS232 charge jumper voltage generators to run continuously, and not shut down if the RS232 cable is disconnected. This would be done if the V- generator is needed to supply a negative bias on a wide temperature range LCD display. Link solder-link 3 in this case to force V- available always.

If not needed, link solder-link 2, so the V- generator is switched, to save power.

RS485 (Optional)

An RS485 option can be installed on the AVR575, by placing a MAX487(E) or LTC485 into location U17. This provides a two-wire (plus ground) RS485 half-duplex system to the AVR575 board. Connection is via J8, a three pin screw terminal 0.15" Phoenix connector. Either horizontal or vertical Phoenix plug-in screw terminals, or J13, a 3 pin, 0.1" Panduit/Molex/AMP crimp connector can be used in place of screw terminal Phoenix types.

With RS485 communication, it is necessary for the TX enable of the transmitter to be turned ON before the sending of the first start bit, and turned off after the last stop bit is finished. In this design the TX-EN line is controlled by port PD2 on the CPU, setting a HIGH for TX-EN, and this must be done by the actual transmitting software. The AVR UART system actually has a third interrupt to flag the end of transmission, at which time the TX-EN signal can be set LOW, if no further bytes need to be sent.

This line is initially pulled LOW by R8, so that an off-line system turning ON does not disrupt network traffic. users should take care to initialise that bit of port D to a LOW before setting the data direction bit for that port to a TX.

RS485 links: To support RS485, link **L1 pins 2 ... 3** must be linked to connect the RS485 receiver into the CPU.

The RS485 receiver can be active at all times, or only active when the RS485 transmitter is OFF (ie this board is disabled during transmission, and does not hear its own transmissions)

Link **L2 pins 1 ... 2** is used to select the RS485 receiver on all the time. **L2 pins 2 ... 3** disables the receiver during AVR575-transmission.

Note: if RS485 is enabled, Solder-link 5 should NOT be connected, and if RTC is installed, Solder-link 4 should be made to enable the RTC from PD4.

Digital I/O from Port D

As discussed above, **PD0, PD1** are used for the serial communications port. Port **PD2** is used as the TX-Enable for the RS485 interface, if installed. (It can also be used as one option of the Real Time Clock chip select.)

PD3 is the input for the dip switch scanner.

PD4 ... PD7 are four drive outputs to four (optional) MDT3055VL power FETs. These FETs are a TO251 device with logic level switching thresholds and built-in clamp Zener for flywheel catch for driving inductive loads. While the FETs are rated at 12 Amps and 60 volts, the tracks on the board limit the recommended maximum continuous current to 1 Amp each.

If the FETs are not installed, the mounting position can have series resistors installed, typically 4K7, and the pins can be four more logic inputs to the CPU. Optional pull-up/pull-down resistors can be installed for these inputs at location RP1.

(The **PD4** pin can also be used to drive the RTC chip enable, if **PD2** is used by the RS485.)

The **PD7** driver FET can be used to drive the LCD backlight via a resistor which can drive it from regulated 5 volts or with an alternate board load, from input power (to save heat in the 5 volt regulator).

Real-Time-Clock (Optional)

The RTC is a Dallas/Maxim DS1305E, which has its own 32Khz crystal, and can also optionally control a FET in series with the input power, to give the board control of its own power ON and OFF functions.

This clock chip communicates with the CPU using the SPI interface on PB5 ... PB7 as MOSI, MISO and CLK, and is enabled for SPI communications with either PD2 or PD4. Solder-links 5 (PD2) or 4 (PD4) select which is used.

This RTC chip has multiple counters and 96 bytes of battery backed RAM, which can be used for any useful function, eg storage of next available address and which EEPROM size.

The RTC is powered from a Lithium battery which is soldered in, and which has a shelf life of >10 years. Current drain is 400 nA max, so the 280mAH capacity should last 81 years.

RTC controlled power switching (Optional)

The alarm registers in the DS1395E are used for controlling the interrupt to the power switching system.

If no power switching is needed, a two-pin connector is installed at J14, just pins 1 and 2, and the board is always operating when input power is connected. (Note: the CPU can go to various Idle, Power-down and Power-save modes to reduce power consumption without powering the whole board down.)

If power switching is needed, components F6, F5 and F7 are loaded, and J14 or J4 are installed as three-pin connectors, with pin 3 in each case being power input via the switching FET, F7, controlled by the RTC. If a data-logging system uses RTC controlled power-up and power-down, if it is desired to force the system ON for setup or up-load of saved data, this is done by having power applied directly to pin 3 and having a front panel switch jumpering pins 2 and 3 of J14/J4 applying unswitched power to the CPU, starting it up, allowing commands via a keyboard, via the RS232 connection or via a switch command input on the DIP switch.

Again, there are options for the connector for the power connector. Instead of a 0.15" Phoenix screw terminal block, either horizontal or vertical Phoenix plug-in screw terminals, or J4, a 2 or 3 pin 0.1" Panduit/Molex/AMP crimp connector can be used in place of screw terminal Phoenix types.

While no compiler writers have specific drivers for the DS1305E, the SPI communications are included in the compiler functions.

Power supply regulator

Power for the board is regulated to 5 volt by a three-terminal regulator, U16. Several different devices can be installed here, trading off low dropout, with 18 volts input max (LM2931) or 35 volt maximum in, but 3 volt dropout (LM78M05 or LM340).

If a low-dropout device is used, it is possible to power the board from a 6 volt gell cell rechargeable, possibly solar power backed, for long term data logger installations. This part is also suitable for use in vehicles if the autpowerup parts are omitted, as these regulators, while they will not run on a 24 volt industrial supply, will withstand load dumps and reverse voltages of 60 volts in auto applications.

If higher inputs, such as 24 volt industrial rails are used, an LM78M05 or LM340 are ideal.

Typical data logging sequence

Setup: A location in the RTC RAM area is used as an address pointer to the next available address in the memory. This is set to 000000 at the start, and the EEPROM memory is erased. (read into CPU RAM (This is set to zero after the previous data is unloaded and the EEPROM memory is erased.) Somewhere in non-volatile memory, either in the RAM or in the EEPROM in the CPU will need to be a maximum address value.

Logging operation: First, check there is capacity left by adding the next address pointer value to the record size and verifying it is not larger than the maximum address value. If it is, either stop, or erase the oldest sector and wrap around.

Data is read at time intervals determined by the RTC, and a record accumulated in the CPU RAM, which might consist of a time stamp, or a record number, a number of analog readings, a number of input status readings compressed into a byte, and anything else necessary.

The address pointer is then read from the RTC RAM using an SPI sequence to CPU RAM. This pointer is then used to work out which EEPROM chip is used (divide address by 128K), and enable that chip.

The data record is then written byte by byte to the EEPROM using the address pointer as the index.

Issue a write enable instruction (WREN) and write each byte in turn, sending in the PROGRAM command followed by three bytes of address, followed by one byte of data. The writing of data occurs when the CS line is raised after bit 0 of the data. A status read command (RDSR) can return the status of the write operation, to determine when to send the next byte.

After each byte, increment the 24-bit address in CPU RAM. (Note that if non-binary length records are being written, it is necessary to recalculate which chip is to get the next byte after each write, as the address may have incremented over a chip boundary during a record.)

After all bytes of the current record have been written, the address pointer has been incremented to what will be the first address of the next record. Write this back to the battery backed RAM in the RTC, and then set the RTC to the next wake-up time. Resetting the alarm in the RTC will close down the board (if it is using RTC-controlled power switching. If logging is being done when some external action triggers it, power will probably be on all the time, but the CPU could be in an idle mode. An internal timer could be programmed to take the CPU out of idle mode and look for an event at desired intervals, say 10 times a second, and be idle the rest of the time.

Data unloading sequence: When the logger is to be interrogated, it must be powered externally, bypassing the clock-controlled power switch (see above) and in some way switched to command mode, eg by connecting it to a PC and sending a command in serially telling it to dump data, record by record, until the top address data is sent. A handshake operation for this is probably best, where a command is sent in, and one record sent in response to each command. The software on the PC gets a string of bytes and knows from the location in the record what each byte is, and can save it on disk or process it in a spread-sheet. At the end, an erase command is sent in, and in response the AVR575 would erase all the EEPROMs and set the address pointer bytes back to 000000, ready for the next run.