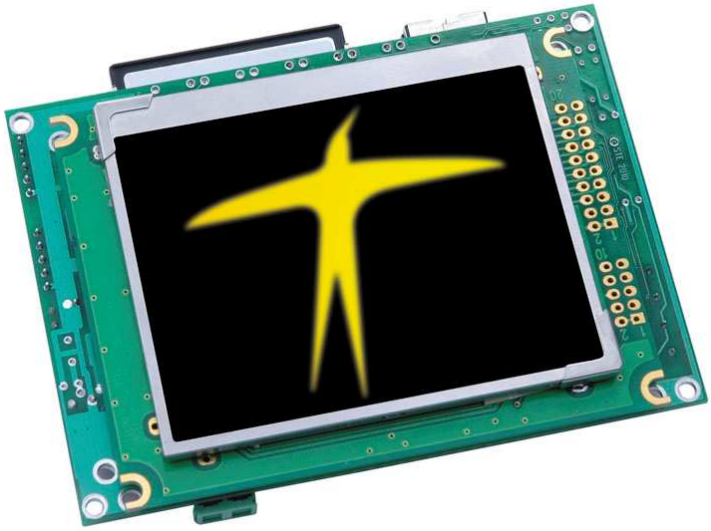
ET™

1/4 VGA Color Graphic TFT, Touchscreen, Push buttons, 100M BaseT Ethernet, RS232, CompactFlash, 16/24-bit ADC, DAC, Solenoid Drivers, and Relay



Technical Manual

TERN INC.
1950 5th Street, Davis, CA 95616, USA
Tel: 530-758-0180 Fax: 530-758-0181
Internet Email: sales@tern.com *http://www.tern.com*

Version 1.3

June 18, 2012

© 1993-2012 TERN INC.
1950 5th Street, Davis, CA 95616, USA
Tel: 530-758-0180 Fax: 530-758-0181
Email: sales@tern.com ***http://www.tern.com***

Important Notice

TERN is developing complex, high technology integration systems. These systems are integrated with software and hardware that are not 100% defect free. ***TERN products are not designed, intended, authorized, or warranted to be suitable for use in life-support applications, devices, or systems, or in other critical applications. TERN*** and the Buyer agree that ***TERN*** will not be liable for incidental or consequential damages arising from the use of ***TERN*** products. It is the Buyer's responsibility to protect life and property against incidental failure.
TERN reserves the right to make changes and improvements to its products without providing notice.
Temperature readings for controllers are based on the results of limited sample tests; they are provided for design reference use only.

Chapter 1: Introduction

1.1 Functional Description

The ***ET***™ is a high performance, low cost, C/C++ programmable controller supporting a color 3.5" TFT display. It is intended for networking industrial process control, high-speed data acquisition, and penal-mounting or hand-held user interface for many OEM products.

The 3.5" color TFT display has 320x240 pixels and a 71x54 mm viewing area. Each pixel uses 16-bit (5R, 6G, 5B) TFT color. The TFT embedded with a high performance graphic controller with 2D drawing engine to support high speed line, box, and circle drawing. Six mechanical push bottoms or a Touch Screen can be installed. Easy user interface software is available.

A Fast Ethernet Module can be installed to provide 100M Base-T network connectivity. This Ethernet module has a hardware LSI TCP/IP stack. It implements TCP/IP, UDP, ICMP and ARP in hardware. It has 16KB internal transmit and receiving buffer. The host can access the buffer via high speed DMA transfers. The hardware Ethernet module releases internet connectivity and protocol processing from the host processor. Software demo is available for connecting to Windows Internet Explorer.

A sigma-delta 24-bit ADC (LTC2448) offers 8 ch. differential or 16 ch. single-ended input channels. A peak single-channel output rate of 5 KHz can be achieved. A 16-bit parallel ADC (AD7655, 0-5V) supports ultra high-speed (1 MHz conversion rate) analog signal acquisition. The AD7655 can *simultaneous* sample on two channels of a total of 4 analog inputs. A 16-bit DAC (LTC2600) provides 8 analog output voltages (0-5V). Two channels of 12-bit DAC (DAC7612) can output 0-4.095V analog voltage.

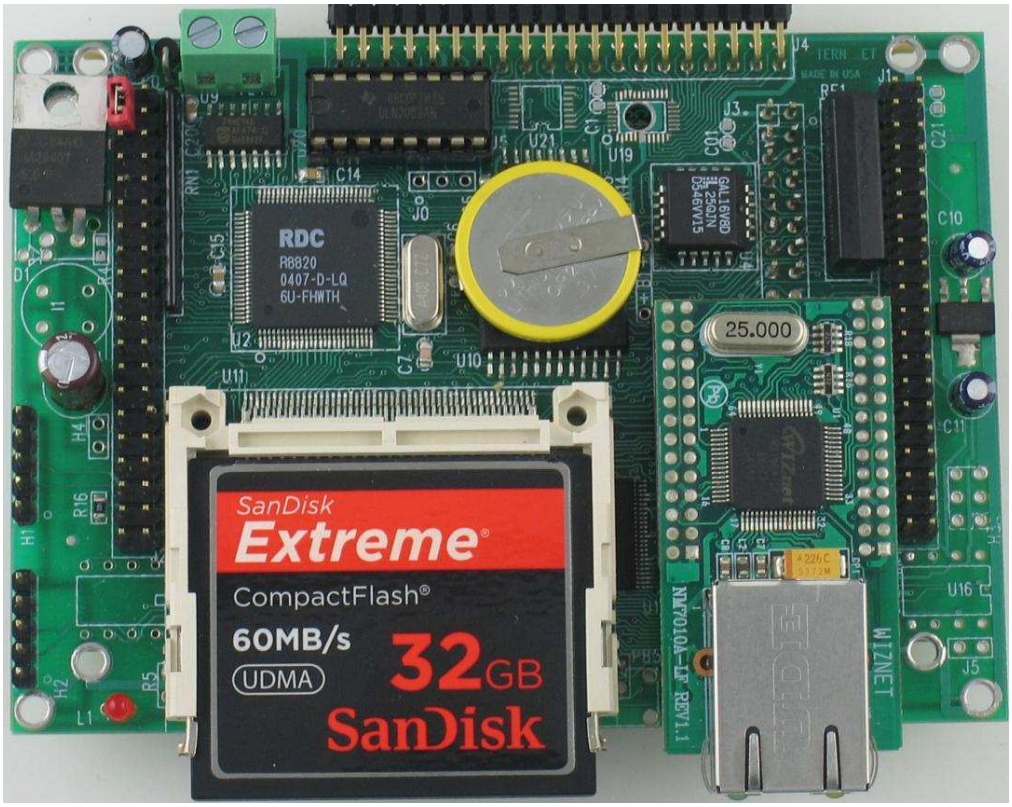
The ***ET*** supports CompactFlash cards with Windows compatible FAT file system support, allowing user easily transfer large amounts of data to or from a PC.

The ***ET*** features 16-bit ACTF Flash (256 KW) and battery-backed SRAM (256 KW). It also includes 3 timers, PWMs, PIOs, 512-byte serial EEPROM, two RS232 ports (One can be converted to RS485), 3 timer/counters, and a watchdog timer. The 16-bit counters can be used to count or time external events, up to 10 MHz, or to generate PWM outputs. A real time clock (RTC72423) is available.

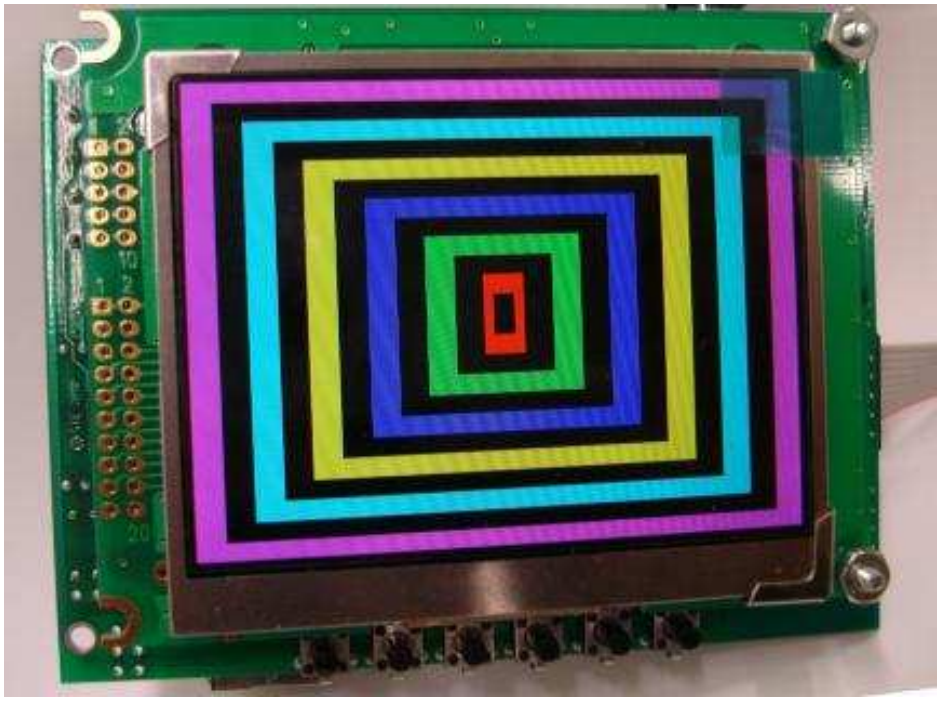
Seven high voltage sink drivers are installed, capable of sinking 350 mA at 50V per line, and they can directly drive solenoids, relays, or lights. A mechanical Reed relay provides reliable, fast switching contacts with a specification of 200 V, maximum 1 Amp carry current, 0.5 Amp switching, and 100 million times operation.

The ***ET*** runs on approximately 280mA at regulated 5V with TFT on. Optional switching regulator or low-drop regulator can provide power-off mode allowing micro-Amp DC power consumption. With default linear regulator, 9-12V, or Switching regulator, 9-24V DC, or low drop regulator, 5.1V-9V DC power can be used.

ET™ with CompactFlash and Ethernet.



ET™ with CompactFlash, Ethernet, 320x240 color TFT module and push buttons.



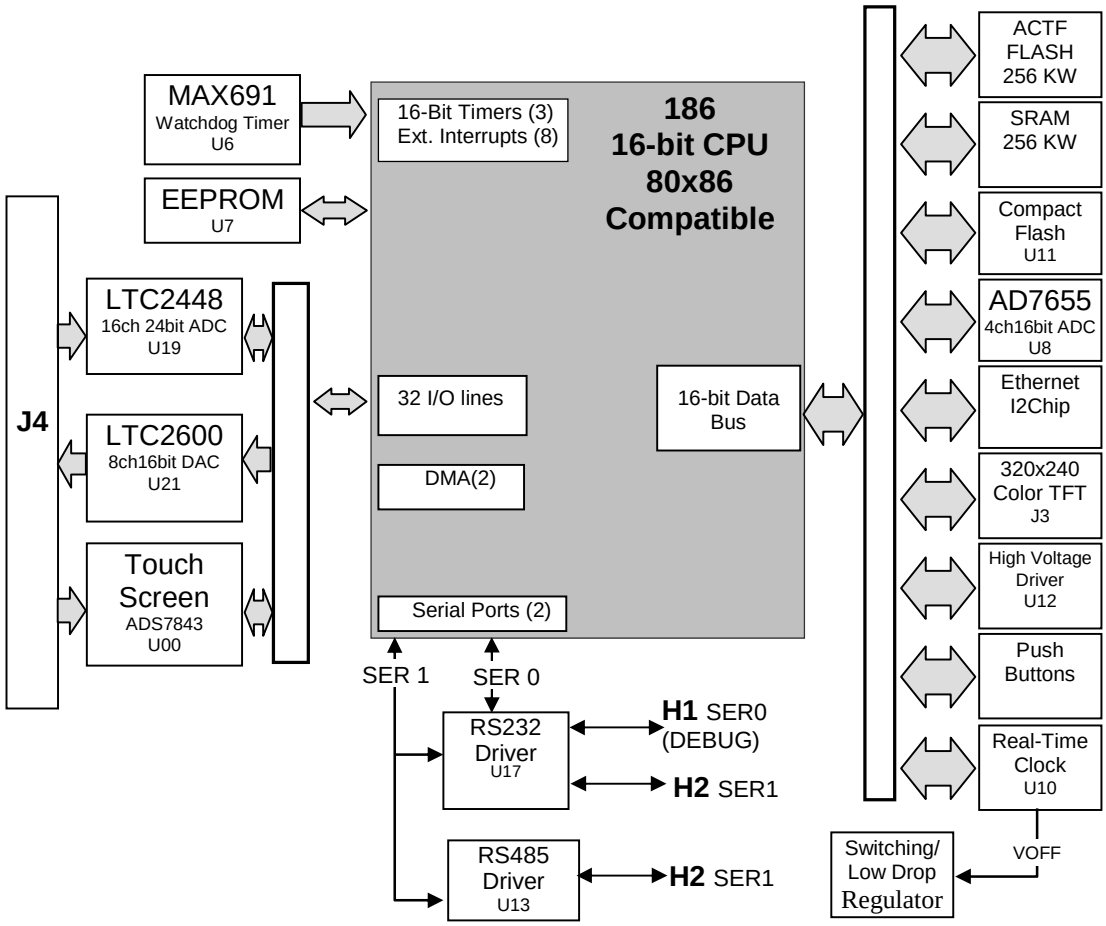


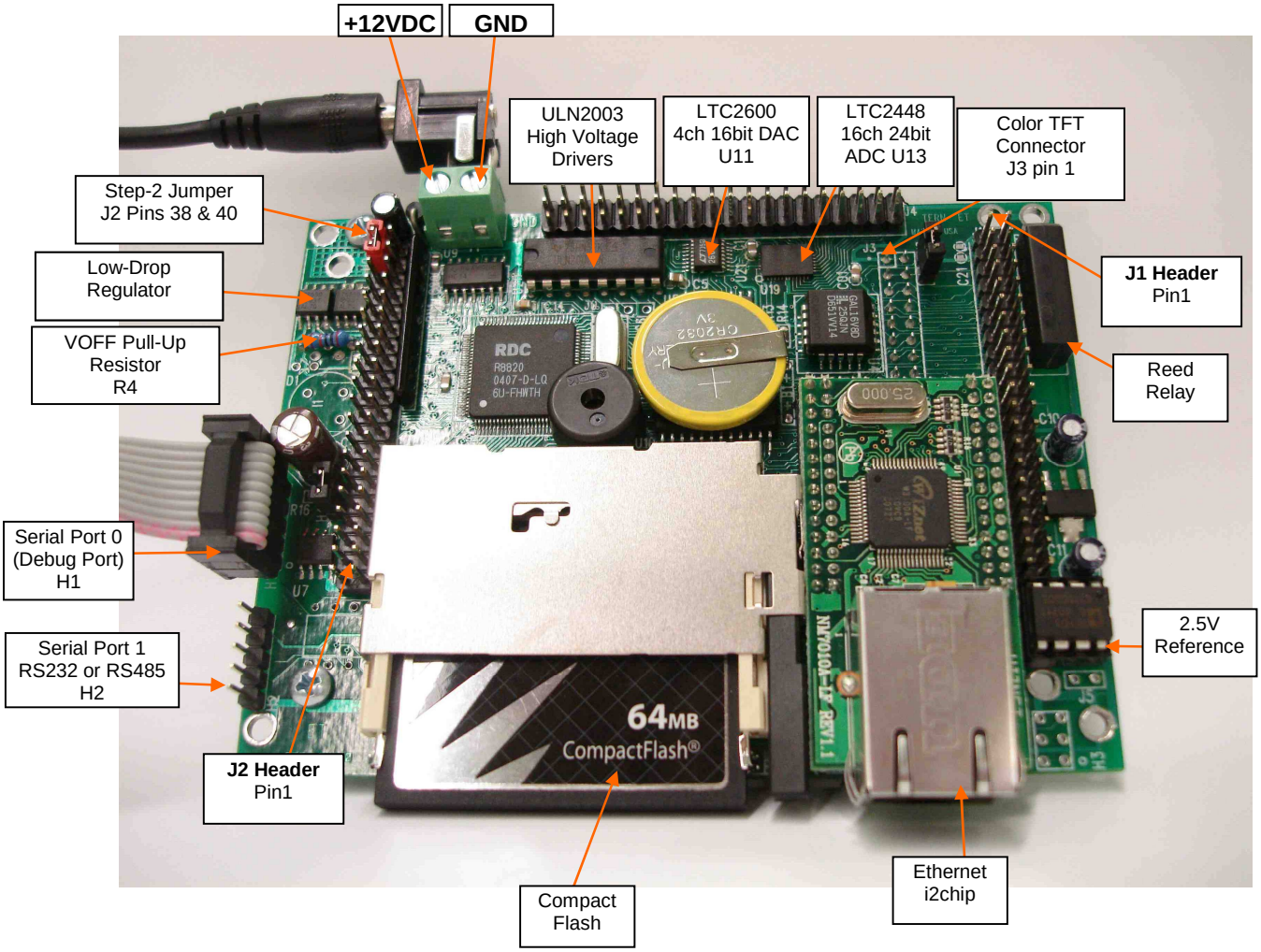
Figure 1.1 Functional block diagram of the ET™

Features:

- 4.2 x 3.1", 280 mA at 5V with TFT on. Boot time < 1 second.
- 40 or 80 MHz, 16-bit CPU, program in C/C++
- 256 KW 16-bit Flash, 256 KW 16-bit SRAM, 512 bytes EE
- 3.5" Color TFT (71x54 mm), Touch screen or 6 Keys
- 12+ I/Os, Real-time clock, 2 serial ports, PWM, counters
- 16 ch 24-bit ADC, 4 ch 16-bit ADC, 8 ch. 16-bit DAC
- Hardware TCP/IP stack for 100M Base-T Ethernet
- CompactFlash card with FAT file system support
- 2 RS232/RS485, 7 Solenoid drivers, 1 Reed Relay

1.2 Physical Description

Below shows the physical description of the ET™.



Programming Overview

An "ACTF Boot Loader" resides in the top protected sector of the 256KW on-board flash chip (29F400). At power-on/reset, the ACTF Utility will check the STEP 2 jumper (J2 pins 38 & 40). If the STEP 2 jumper is installed, the "jump address" located in the on-board serial EEPROM will be read out and the CPU will jump to that address for immediate execution. A DEBUG kernel (already pre-programmed at the factory) can be downloaded and programmed into the flash starting at address 0xFA000. Using the ACTF Utility, the "GFA000 <enter>" command will set the jump address to 0xFA000. The command will also run the DEBUG kernel, preparing the ET™ for communication with the Paradigm C/C++ IDE for downloading and debugging applications. The following diagrams show the procedure for programming the ET™. Steps include preparing the ET™ for debugging, debugging the ET™, standalone field test, and production.

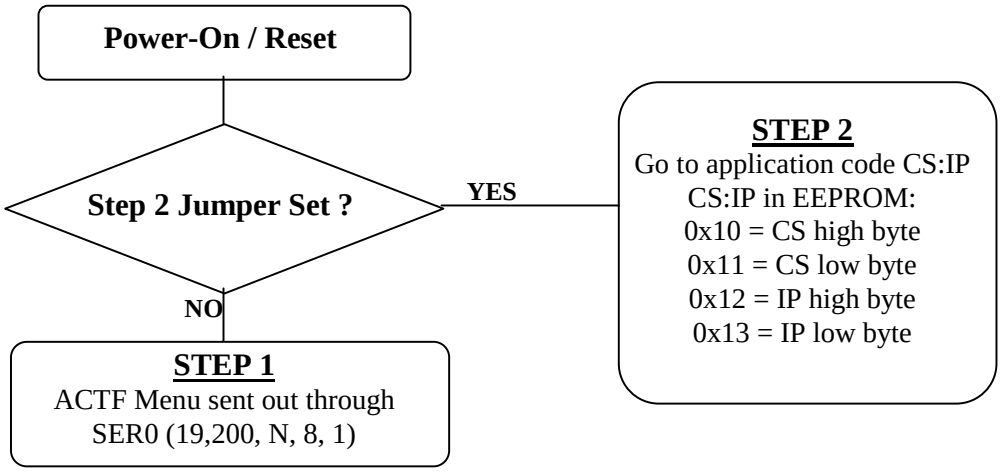


Figure 1.2 Flow Chart of ACTF Operation

By default, the DEBUG kernel has been loaded into the ACTF flash at the factory for your convenience. You may proceed directly to STEP 1: Debugging.

Preparation for Debugging:

This had already been done at the factory! You may proceed to STEP 1: Debugging. This step is only required if you have completed STEP 3 and would like to return to STEP 1.

● Connect the ET™ (SER0, H2) to PC (COMx) via serial debug cable provided with the EV-P/DV-P. Using the Windows "Hyper Terminal", create a serial link based on 19,200, 8 bits, 1 stop, no parity.
● Power on the ET™ WITHOUT the STEP 2 jumper installed (J2 pins 38 & 40). The ACTF text MENU should be sent out via serial port to "Hyper Terminal".
● Use the "D <enter>" command to initiate download. Select Transfer -> Send File, and select \tern\186\rom\ae86\l_tdrem.hex. Use the "G04000 <enter>" command to execute this script.
● Select Transfer -> Send File to select \tern\186\rom\ae86\ee40_115.hex (40MHz) or ee80_115.hex (80MHz). This is the debug kernel. Use the "GFA000 <enter>" command to set jump address and execute the debug kernel. The LED will blink twice and remain on.
● Set the STEP 2 jumper (J2 pins 38 & 40). The ET™ is now ready to communicate with the Paradigm C/C++ IDE for debugging and application development.

Step 1: Debugging:

● Launch the Paradigm C/C++ IDE. Select File -> Open. Chose the project file \tern\186\samples\et\et.pdl.
● Use samples within the "et.pdl" project to create application. Download, run, and debug application.

Step 2: Standalone Field Test:

● After completing STEP 1, by default, your application resides in the battery-backed SRAM starting at address 0x10000.
● Remove STEP 2 jumper and setup Hyper Terminal link with ET™. (Open Windows "Hyper Terminal" program. Set for 19,200, 8 bits, 1 stop, no parity).
● At power-on, ACTF menu will be sent to Hyper Terminal. Use the "G10000 <enter>" command to execute application. Set STEP 2 jumper (J2 pins 38 & 40). At every power-on/reset, application at 0x10000 will execute.
● Complete STANDALONE FIELD TEST. If return to STEP 1 is required, remove STEP 2 jumper and use the "GFA000 <enter>" command to run debug kernel to prepare to setup for communication with Paradigm C/C++ IDE.

Step 3: Production:

The DV-P Kit is required for this step. If you do not have the DV-P Kit, visit http://tern.com/devkit.htm for upgrade details.

● Refer to Section 3.3 of the ACTF technical manual, found in the \tern_docs\manuals directory. Here you will find details on generating an ACTF downloadable HEX file based upon you application.
● Remove the STEP 2 jumper and create serial link using Hyper Terminal (19,200, N, 8, 1). At power-on/reset, you will see the ACTF menu at Hyper Terminal. Use the "D <enter>" command to initiate download process. Select Transfers -> Send File, and select \tern\186\rom\ae86\l_29f40.hex.
● This file will erase the flash and prepare the flash to accept ACTF downloadable application HEX file. Use the "G04000 <enter>" command to run script. Flash will be ready for application.
● Select Transfer -> Send File to select your ACTF downloadable application HEX file. Upon completion, use the "GC0000 <enter>" command to execute application. This command also sets the jump address to point you application in flash. Set STEP 2 jumper (J2 pins 38 & 40). At power-on/reset application will execute.

There is no ROM socket on the ET™. The user's application program must reside in the SRAM (starting at address of 0x10000 by default based on \tern\186\config\186.cfg) for debugging in STEP 1, reside in the battery-backed SRAM for standalone field testing in STEP 2, and finally be programmed into the on-board flash for a complete product. For production, the user must produce an ACTF-downloadable HEX file for the application based on the DV-P Kit. From the ACTF Utility, use the command "GC0000 <enter>" to point to the user's application code in the flash. The STEP 2 jumper must installed for every production-version board.

1.3 Minimum Requirements for ET™ System Development

Minimum Hardware Requirements

- PC or PC-compatible computer with serial COMx port that supports 115,200 baud
- ET™ controller
- Debug Serial Cable (RS232; DB9 connector for PC COM port and IDE 5x2 connector for controller)
- Center Negative Wall Transformer

Minimum Software Requirements

- TERN EV-P installation CD-ROM and a PC running: Windows 95/98/2000/ME/NT/XP/VISTA/7

With the EV-P, you can program and debug the ET™ in Step One and Step Two, but you cannot run Step Three. To generate an application Flash File and complete a project, the development kit, DV-P, is required. The EV-P kit can be upgraded to the DV-P Kit. See http://tern.com/devkit.htm for details.

Chapter 2: Installation

2.1 Software Installation

Please refer to the "software_kit.pdf" technical manual on the TERN installation CD, under tern_docs\manual\software_kit.pdf, for information on installing software.

2.2 Hardware Installation

Overview

- Connect PC-IDE serial cable:
  For debugging (STEP 1), place IDE connector on SER0 with red edge of cable at pin 1. This DEBUG cable is a 10-pin IDE to DB9 cable, made by TERN.
- Connect wall transformer:
  Connect 9V wall transformer to power and plug into power jack using power jack adapter supplied with EV-P/DV-P Kit

Hardware installation consists primarily of connecting the microcontroller to your PC.

2.2.1 Connecting the ET to the PC

Fig 2.1, 2.2 and 2.3 show the location of the debug serial port and power jack. The ET is linked to the PC via a serial cable (DB9-IDE) which is supplied with TERN's EV-P / DV-P Kits.

The ET communicates through SER0 by default. Install the 5x2 IDC connector on the SER0 H1 5x1 pin header. ***IMPORTANT:*** *Note that the* ***red*** *side of the cable must point to pin 1 of the H1 header and the pins connect to the top row of the 5x2 IDC connector.* The DB9 connector should be connected to one of your PC's COM Ports (COM1 or COM2).

2.2.2 Powering-on the ET

By factory default setting:
1) The RED STEP2 Jumper is installed. (Default setting in factory)
2) The DEBUG kernel is pre-loaded into the on-board flash starting at address of 0xFA000. (Default setting in factory)
3) The EEPROM is set to jump address of 0xFA000. (Default setting in factory)

Connect +9-12V DC to the DC power terminal. With the low-drop regulator option, the input voltage can go as low as 5.1V DC. The screw terminal at the corner of the board is the positive voltage input and the other terminal is GND (see figure for details). A power jack adapter (seen below) is included with the TERN EV-P/DV-P kit. It can be used to connect the output of the power jack adapter and the ET. Note that the output of the power jack adapter is center negative.

The on-board LED should blink twice and remain on, indicating the debug kernel is running and ready to communicate with Paradigm C++ TERN Edition for programming and debugging.

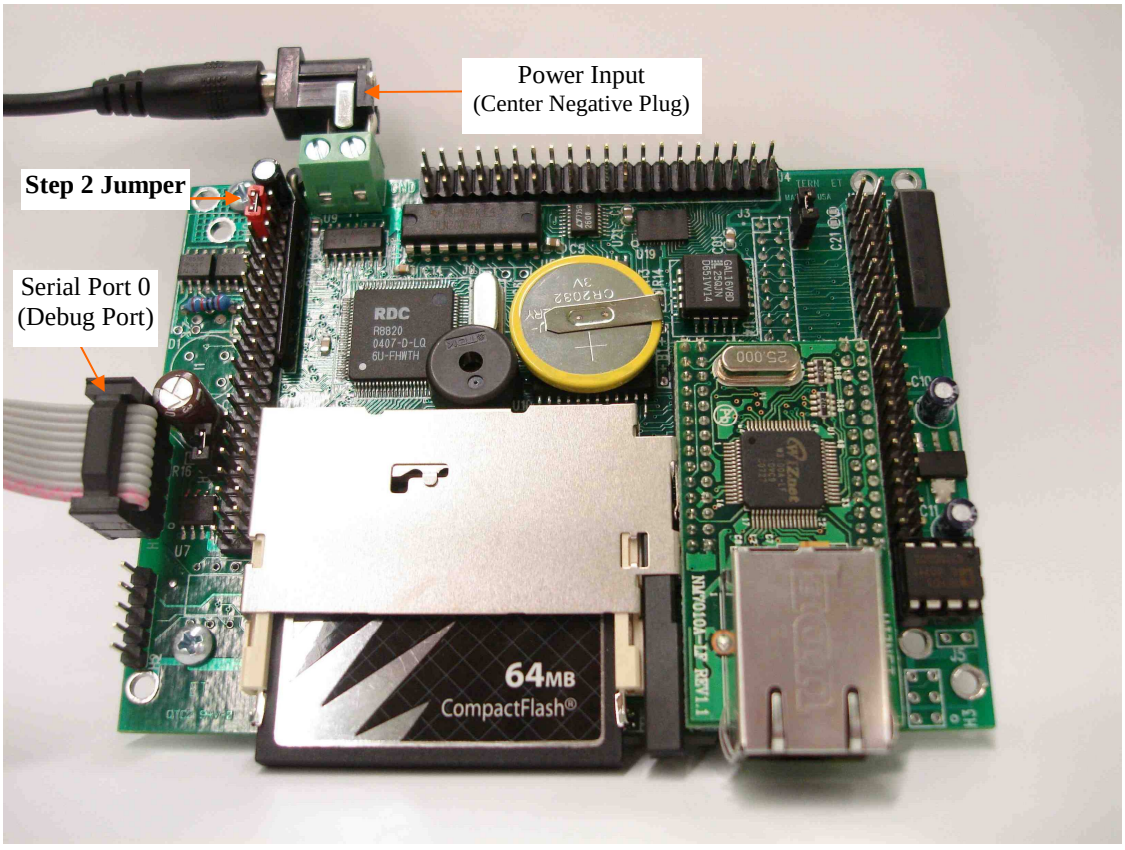


Figure 2.1 Locations of STEP2 Jumper, LED, Power input and DEBUG port

Chapter 3: Hardware

3.1 Am186ES/R8820/IA186/R1120 - Introduction

The Am186ES is based on industry-standard x86 architecture. The Am186ES controllers uses 16-bit external data bus, are higher-performance, more integrated versions of the 80C188 microprocessors which uses 8-bit external data bus. In addition, the Am186ES has new peripherals. The on-chip system interface logic can minimize total system cost. The Am186ES has two asynchronous serial ports, 32 PIOs, a watchdog timer, additional interrupt pins, a pulse width demodulation option, DMA to and from serial ports, a 16-bit reset configuration register, and enhanced chip-select functionality.

There are a total of four compatible CPU chips can be used in the **ET™**:
R8820 from RDC is a drop-in replacement 5V, 40MHz chip for the AM186ES. Connecting J0.1=J0.2.
AM186ES(AMD, 5V, 40 MHz), R8820(RDC, 5V, 40 MHz), IA186ES(INNOVASIC, 5V, 40 MHz) and R1120(RDC, 3.3V, 80 MHz).
The multiple sources of the CPU can support longer life time of the **ET™** product. The technical specifications and discussions in this manual are based on AM186ES.

By default, the **ET™** uses 5V 40 MHz R8820 and low power 55-70 ns SRAM.
Optional 3.3V 80 MHz R1120 can be installed.
At 80 MHz, the low power 55 ns SRAM with battery backup works fine but will not be able to support DMA operation.
A fast 10/15/25 ns SRAM (Not low power) can be used to support zero wait state and DMA operation at 80 MHz, but the backup battery will be drain in few days.
There are three pads on the PCB for battery. One pad is ground, while the other two allow a 3V backup lithium battery to be installed in two different positions:
1) If the battery's positive lead is installed in the pad which is further away from the RTC and CompactFlash, it supports the RTC only. No battery backup for the SRAM.
2) If the battery's positive lead is installed in the pad which is closer to the RTC, it supports both RTC and SRAM.

3.2 Am186ES – Features

3.2.1 Clock and crystal

Due to its integrated clock generation circuitry, the Am186ES microcontroller allows the use of a times-one crystal frequency. The design achieves 40 MHz CPU operation, while using a 40 MHz crystal.

The system CLKOUTA signal is routed to J1 pin 4, default 40 MHz for EL40.

CLKOUTA remains active during reset and bus hold conditions. The initial function ae_init(); disables CLKOUTA and CLKOUTB with clka_en(0); and clkb_en(0);

You may use clka_en(1); to enable CLKOUTA=CLK=J1 pin 4.

The R8820 uses a 40 MHz crystal.

By default the 3.3V R1120 uses a 20 MHz crystal. The CPU speed is software programmable with the PLL.

At power-on, the on-board ACTF Flash programs the R1120 running at 20 MHz system clock, so a 9600 baud (instead 19,200 baud) is used for the ACTF menu.

Debug kernels for Paradigm C++ TERN Edition are available:

c:\tern\186\rom\ae86\EE40_115.hex, or c:\tern\186\rom\ae86\EE80_115.hex

The EE40_115.hex will allow 40 MHz **ET^{TM}** talk to Paradigm C++ TERN Edition at 115,200 baud. The EE80_115.hex will run the **ET^{TM}** based on R1120 at 80 MHz.

By default, the EE40_115.hex is pre-programmed for the 40 MHz **ET^{TM}**.

User can use software to setup the CPU speed:

```
outport(0xfff8,0x0103);  // PLLCON, 20MHz crystal, 0103=40 MHz, 0107=80MHz
```

3.2.2 External Interrupts and Schmitt Trigger Input Buffer

There are eight external interrupts: INT0-INT6 and NMI.

/INT0, J2 pin 8, used by touch screen reader ADS7843.
/INT1, J2 pin 6, free to use.
INT2, J2 pin 19, free to use
/INT3, J2 pin 21, free to use
/INT4, J2 pin 33, used by 100M BaseT Ethernet
INT5=P12=DRQ0, used by **ET^{TM}** as clock for multiple devices
INT6=P13=DRQ1, J2 pin 11, free to use
/NMI, J2 pin 7

Some of external interrupt inputs, /INT0, 1, 3, 4 and /NMI, are buffered by Schmitt-trigger inverters (U9, 74HC14), in order to increase noise immunity and transform slowly changing input signals to fast changing and jitter-free signals. As a result of this buffering, these pins are capable of only acting as input.

These buffered external interrupt inputs require a falling edge (HIGH-to-LOW) to generate an interrupt.

The **ET^{TM}** uses vector interrupt functions to respond to external interrupts. Refer to the Am186ES User's manual for information about interrupt vectors.

3.2.3 Asynchronous Serial Ports

The Am186ES CPU has two asynchronous serial channels: SER0 and SER1. Both asynchronous serial ports support the following:

- Full-duplex operation
- 7-bit, 8-bit, and 9-bit data transfers
- Odd, even, and no parity
- One stop bit
- Error detection
- Hardware flow control
- DMA transfers to and from serial ports
- Transmit and receive interrupts for each port
- Multidrop 9-bit protocol support
- Maximum baud rate of 1/16 of the CPU clock speed
- Independent baud rate generators

The software drivers for each serial port implement a ring-buffered DMA receiving and ring-buffered interrupt transmitting arrangement. See the samples files ***s1_echo.c*** and ***s0_echo.c***.

Important Note: For 80MHz EL80, DMA functions are not available when by default low power 55 ns SRAM is installed. If install a 25 ns SRAM, 80MHz **ET^{TM}** can have all DMA functions, but it will drain the backup battery fast.

3.2.4 Timer Control Unit

The timer/counter unit has three 16-bit programmable timers: Timer0, Timer1, and Timer2.

Timer0 and Timer1 are connected to external pins:

Timer0 output = P10 = J2 pin 12
Timer0 input = P11 = U7 EEPROM pin 5
Timer1 output = P1 = J2 pin 29
Timer1 input = P0 = J2 pin 20

Timer0 input P11 is used and shared by on-board EEPROM, LED, and HitWD, not recommended for other external use.

The timer can be used to count or time external events, or can generate non-repetitive or variable-duty-cycle waveforms.

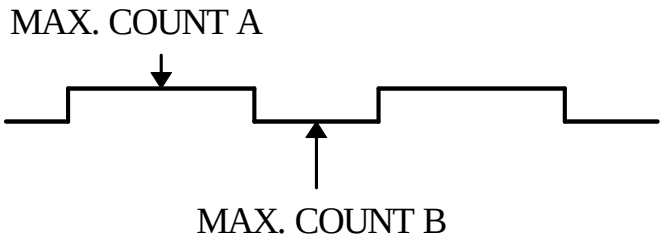
Timer2 is not connected to any external pin. It can be used as an internal timer for real-time coding or time-delay applications. It can also prescale timer 0 and timer 1 or be used as a DMA request source.

The maximum rate at which each timer can operate is 10 MHz, since each timer is serviced once every fourth clock cycle. Timer output takes up to six clock cycles to respond to clock or gate events. See the sample programs ***timer02.c*** and ***ae_cnt1.c*** in the **`tern\186\samples\ae`** directory.

3.2.5 PWM outputs and PWD

The Timer0 and Timer1 outputs can also be used to generate non-repetitive or variable-duty-cycle waveforms. The timer output takes up to 6 clock cycles to respond to the clock input. Thus the minimum timer output cycle is 25 ns x 6 = 150 ns (at 40 MHz).

Each timer has a maximum count register that defines the maximum value the timer will reach. Both Timer0 and Timer1 have secondary maximum count registers for variable duty cycle output. Using both the primary and secondary maximum count registers lets the timer alternate between two maximum values.



Pulse Width Demodulation can be used to measure the input signal's high and low phases on the /INT2=J2 pin 19.

3.2.6 Power-save Mode

The **ET™** can be used for low power consumption applications. The power-save mode of the Am186ES reduces power consumption and heat dissipation, thereby extending battery life in portable systems. In power-save mode, operation of the CPU and internal peripherals continues at a slower clock frequency. When an interrupt occurs, it automatically returns to its normal operating frequency.

3.3 Am186ES PIO lines

The Am186ES has 32 pins available as user-programmable I/O lines. Each of these pins can be used as a user-programmable input or output signal, if the normal shared function is not needed. A PIO line can be

configured to operate as an input or output with or without a weak pull-up or pull-down, or as an open-drain output. A pin's behavior, either pull-up or pull-down, is pre-determined and shown in the table below.

After power-on/reset, PIO pins default to various configurations. The initialization routine provided by TERN libraries reconfigures some of these pins as needed for specific on-board usage, as well. These configurations, as well as the processor-internal peripheral usage configurations, are listed below in Table 3.1.

PIO	*Function*	*Power-On/Reset status*	ET™ *Pin No.*	ET™ *Initial*
P0	Timer1 in	Input with pull-up	J2 pin 20	Input with pull-up
P1	Timer1 out	Input with pull-down	J2 pin 29; Beeper	Input with pull-down
P2	/PCS6/A2	Input with pull-up	J2 pin 24	Input with pull-up
P3	/PCS5/A1	Input with pull-up	J2 pin 15	Input with pull-up
P4	DT/R	Normal	J2 pin 38	Input with pull-up (Step 2)
P5	/DEN/DS	Normal	J2 pin 30	Input with pull-up
P6	SRDY	Normal	J2 pin 35; J03.2	Input with pull-down
P7	A17	Normal	U3 pin 22	A17
P8	A18	Normal	U3 pin 23	A18
P9	A19	Normal	J2 pin 10	A19
P10	Timer0 out	Input with pull-down	J2 pin 12	Input with pull-down
P11	Timer0 in	Input with pull-up	U7.5	Input with pull-up
P12	DRQ0/INT5	Input with pull-up		LED/ HWD/ADC/DAC
P13	DRQ1/INT6	Input with pull-up	J2 pin 11	Input with pull-up
P14	/MCS0	Input with pull-up	J2 pin 37; U4.2	PAL
P15	/MCS1	Input with pull-up	J2 pin 23	Input with pull-up
P16	/PCS0	Input with pull-up	J1 pin 19	/PCS0
P17	/PCS1	Input with pull-up	J4 pin 2; U4.9	PAL
P18	CTS1/PCS2	Input with pull-up	J2 pin 22	Input with pull-up
P19	RTS1/PCS3	Input with pull-up	J2 pin 31; U13.3	Input with pull-up
P20	RTS0	Input with pull-up	J2 pin 27; RE1.3	Input with pull-up (Relay)
P21	CTS0	Input with pull-up	J2 pin 36; U20.7	Input with pull-up (HV)
P22	TxD0	Input with pull-up	J2 pin 34	TxD0
P23	RxD0	Input with pull-up	J2 pin 32	RxD0
P24	/MCS2	Input with pull-up	J2 pin 17	Input with pull-up
P25	/MCS3	Input with pull-up	J2 pin 18	Input with pull-up
P26	UZI	Input with pull-up	J2 pin 4; U21	Input with pull-up* (DAC)
P27	TxD1	Input with pull-up	J2 pin 28	TxD1
P28	RxD1	Input with pull-up	J2 pin 26	RxD1
P29	/CLKDIV2	Input with pull-up	J2 pin 3; U00.15	Input with pull-up* (TS)
P30	INT4	Input with pull-up	J2 pin 33; JP1.2	Input with pull-up (ET)
P31	INT2	Input with pull-up	J2 pin 19	Input with pull-up

* Note: P26 and P29 must NOT be forced low during power-on or reset.

Table 3.1 I/O pin default configuration after power-on or reset

Three external interrupt lines are not shared with PIO pins:

INT0 = J2 pin 8
INT1 = J2 pin 6
INT3 = J2 pin 21

The 32 PIO lines, P0-P31, are configurable via two 16-bit registers, PIOMODE and PIODIRECTION. The settings are as follows:

MODE	PIOMODE reg.	PIODIRECTION reg.	PIN FUNCTION
0	0	0	Normal operation
1	0	1	INPUT with pull-up/pull-down
2	1	0	OUTPUT
3	1	1	INPUT without pull-up/pull-down

ET™ initialization on PIO pins in **ae_init()** is listed below:

```
outport(0xff78,0xe73c);     // PDIR1, TxD0, RxD0, TxD1, RxD1, P16=PCS0, P17=PCS1
outport(0xff76,0x0000);     // PIOM1
outport(0xff72,0xec7b);     // PDIR0, P12,A19,A18,A17,P2=PCS6=RTC
outport(0xff70,0x1000);     // PIOM0, P12=LED
```

The C function in the library **ae_lib** can be used to initialize PIO pins.

void ***pio_init***(char bit, char mode);

Where bit = 0-31 and mode = 0-3, see the table above.

Example: ***pio_init***(12, 2); will set P12 as output

pio_init(1, 0); will set P1 as Timer1 output

void ***pio_wr***(char bit, char dat);

pio_wr(12,1); set P12 pin high, if P12 is in output mode

pio_wr(12,0); set P12 pin low, if P12 is in output mode

unsigned int ***pio_rd***(char port);

pio_rd (0); return 16-bit status of P0-P15, if corresponding pin is in input mode,

pio_rd (1); return 16-bit status of P16-P31, if corresponding pin is in input mode,

Some of the I/O lines are used by the **ET™** system for on-board components (Table 3.2). We suggest that you not use these lines unless you are sure that you are not interfering with the operation of such components (i.e., if the component is not installed).

You should also note that the external interrupt PIO pins INT2, 4, 5, and 6 are not available for use as output because of the inverters attached. The input values of these PIO interrupt lines will also be inverted for the same reason. As a result, calling ***pio_rd*** to read the value of P31 (**INT2**) will return 1 when pin 19 on header J2 is pulled low, with the result reversed if the pin is pulled high.

Signal	Pin	Function
P1	Timer1 out	Beeper
P4	/DT	STEP2 jumper
P12	DRQ0/INT5	LED, EEPROM, watch dog, ADC, DAC, touch screen
P14	/MCS0	PAL
P17	/PCS1	PAL
P19	RTS1/PCS3	RS485
P20	RTS0	Reed relay
P21	CTS0	High voltage eriver
P26	UZI	DAC
P29	/CLKDIV2	Touch screen
P30	INT4	Ethernet interrupt

Table 3.2 I/O lines used for on-board components

3.4 I/O Mapped Devices

3.4.1 I/O Space

External I/O devices can use I/O mapping for access. You can access such I/O devices with ***inportb***(port) or ***outportb***(port,dat). These functions will transfer one byte or word of data to the specified I/O address. The external I/O space is 64K, ranging from 0x0000 to 0xffff.

The default I/O access time is 15 wait states. You may use the function void ***io_wait***(char wait) to define the I/O wait states from 0 to 15. The system clock is 25 ns (or 50 ns), giving a clock speed of 40 MHz (or 20 MHz). Details regarding this can be found in the Software chapter, and in the Am186ES User's Manual. Slower components, such as most LCD interfaces, might find the maximum programmable wait state of 15 cycles still insufficient. Due to the high bus speed of the system, some components need to be attached to I/O pins directly.

For details regarding the chip select unit, please see Chapter 5 of the Am186ES User's Manual. The table below shows more information about I/O mapping.

I/O space	Location	Usage
0x0000-0x00FF	J1 pin 19=P16	USER*
0x0100-0x010F	U5	HV, ADC, DAC, TS
0x0120	U8	AD7655 Convert & Read B
0x0124	U8	AD7655 Read A
0x0140	U10	RTC 72423
0x0160	U12	Push buttons
0x01e0	U11	CF IO Base

*PCS0 may be used for other TERN peripheral boards.

To illustrate how to interface the **ET™** with external I/O boards, a simple decoding circuit for interfacing to an 82C55 parallel I/O chip is shown in Figure 3.1.

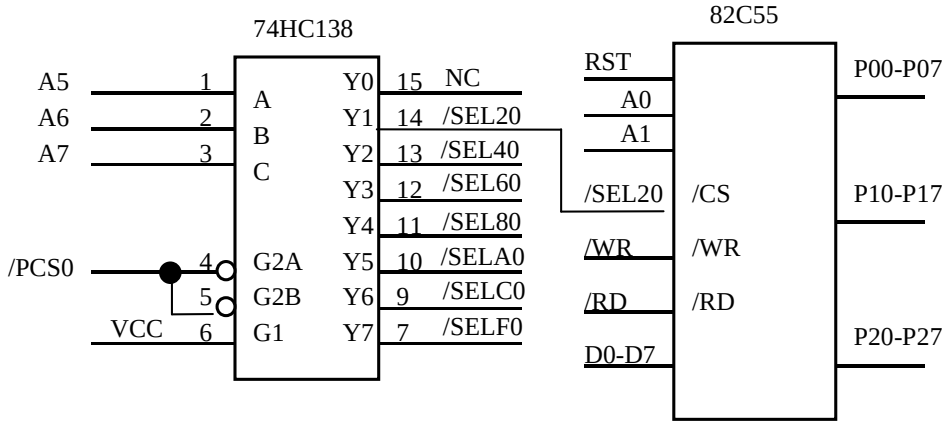


Figure 3.1 Interface to external I/O devices

The function **`ae_init()`** by default initializes the /PCS0 line at base I/O address starting at 0x00. You can read from the 82C55 with ***inportb(0x020)*** or write to the 82C55 with ***outportb(0x020,dat)***. The call to ***inportb(0x020)*** will activate /PCS0, as well as putting the address 0x00 over the address bus. The decoder will select the 82C55 based on address lines A5-7, and the data bus will be used to read the appropriate data from the off-board component.

3.5 Other Devices

A number of other devices are also available on the **ET™**. Some of these are optional, and might not be installed on the particular controller you are using. For a discussion regarding the software interface for these components, please see the Software chapter.

3.5.1 On-board Supervisor with Watchdog Timer

The MAX691/LTC691 (U6) is a supervisor chip. With it installed, the **ET™** has several functions: watchdog timer, battery backup, power-on-reset delay, power-supply monitoring, and power-failure warning. These will significantly improve system reliability.

Watchdog Timer

The watchdog timer is activated by setting a jumper on J5 of the **ET™**. The watchdog timer provides a means of verifying proper software execution. In the user's application program, calls to the function **hitwd()** (a routine that toggles the P12=HWD pin of the MAX691) should be arranged such that the HWD pin is accessed at least once every 1.6 seconds. If the J5 jumper is on and the HWD pin is not accessed within this time-out period, the watchdog timer pulls the WDO pin low, which asserts /RESET. This automatic assertion of /RESET may recover the application program if something is wrong. After the **ET™** is reset, the WDO remains low until a transition occurs at the WDI pin of the MAX691. When controllers are shipped from the factory the J5 jumper is off, which disables the watchdog timer.

The Am186ES has an internal watchdog timer. This is disabled by default with **ae_init()**.

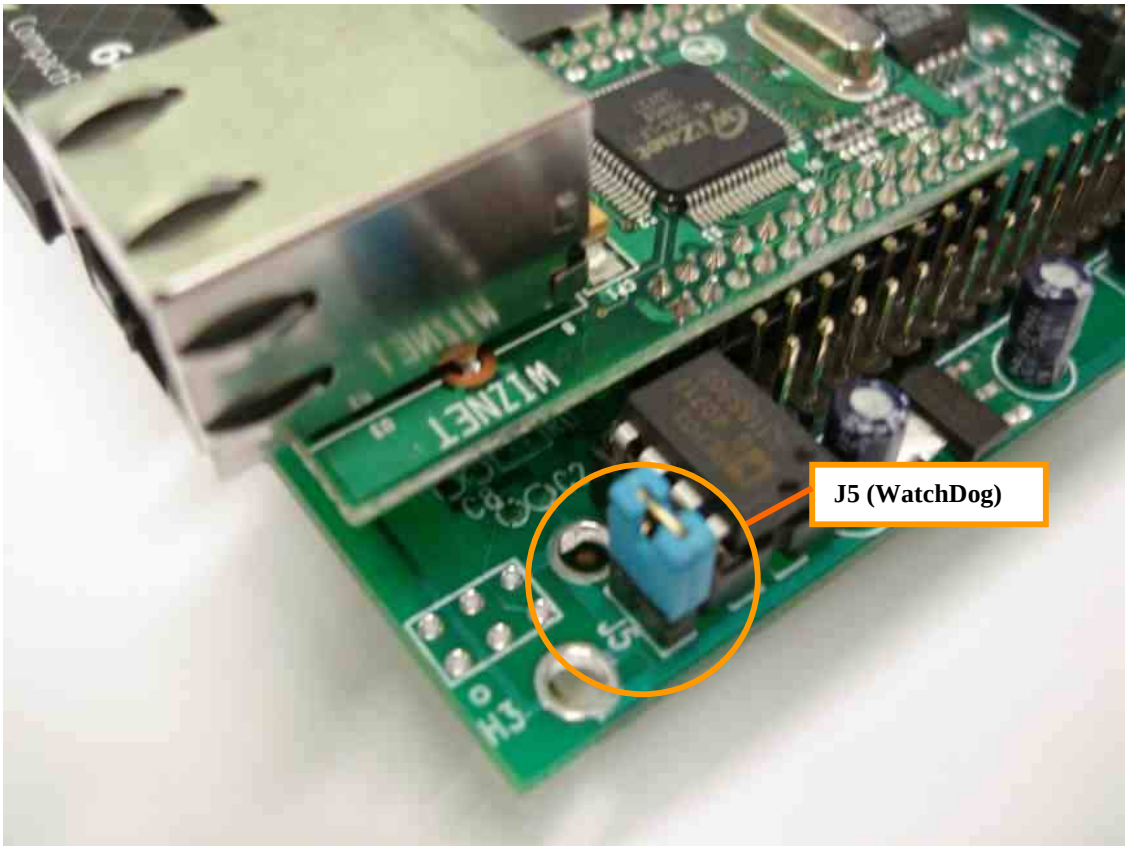
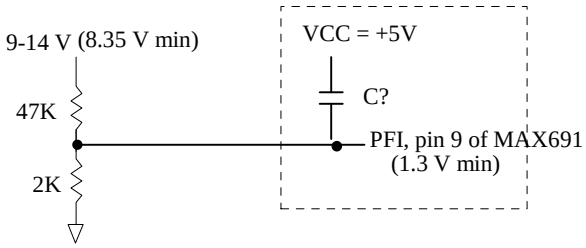


Figure 3.2 Location of watchdog timer enable jumper

Power-failure Warning

The supervisor supports power-failure warning and backup battery protection. When power failure is sensed by the PFI=J1.11, pin 9 of the MAX691 (lower than 1.3 V), the PFO is low. The PFI pin 9 of 691 is directly shorted to VCC by default. In order to use PFI externally, cut the trace and bring the PFI signal out. You may design an NMI service routine to take protect actions before the +5V drops and processor dies. The following circuit shows how you might use the power-failure detection logic within your application.



Using the supervisor chip for power failure detection

Battery Backup Protection

The backup battery protection protects data stored in the SRAM and RTC. The battery-switch-over circuit compares VCC to VBAT (+3 V lithium battery positive pin), and connects whichever is higher to the VRAM (power for SRAM and RTC). Thus, the SRAM and the real-time clock RTC72423 are backed up. In normal use, the lithium battery should last about 3-5 years without external power being supplied. When the external power is on, the battery-switch-over circuit will select the VCC to connect to the VRAM.

3.5.2 EEPROM

A serial EEPROM of 128 bytes (24C01), 512 bytes (24C04), or 2K bytes (24C16) can be installed in U7. The **ET™** uses the P12=SCL (serial clock) and P11=SDA (serial data) to interface with the EEPROM. The EEPROM can be used to store important data such as a node address, calibration coefficients, and configuration codes. It typically has 1,000,000 erase/write cycles. The data retention is more than 40 years. EEPROM can be read and written by simply calling the functions **ee_rd()** and **ee_wr()**.

A range of lower addresses in the EEPROM is reserved for TERN use. Details regarding which addresses are reserved, and for what purpose, can be found in Appendix C of this manual. Refer to **c:\tern\186\samples\ae\ae_ee.c** for sample code concerning the EEPROM.

3.5.3 Real-time Clock RTC72423

If installed, the real-time clock RTC72423 (EPSON, U10) is mapped in the I/O address space 0x0140. It must be backed up with a lithium coin battery. The RTC is accessed via software drivers ***rtc_init()*** or ***rtc_rd()***.

It is also possible to configure the real-time clock to raise an output line attached to an external interrupt, at 1/64 second, 1 second, 1 minute, or 1 hour intervals. This can be used in a time-driven application, or the **VOFF** signal can be used to turn on/off the power supply. To use the VOFF feature on the ET, the switching or low drop regulator option and pull up resistor R4 must be installed.

See the VOFF sample program **`tern\186\samples\et\et_voff.c`** for details.

3.5.4 Reed Relay

One Reed Relay can be installed on the **ET™** at location RE1. The relays offer high speed switching compared to electromechanical relays, a specification of 200 V, maximum 1 Amp carry current, 0.5 Amp switching, and 100 million times operation. The relay is driven by /RTS0 (P20). The relay can be configured to enable and disable the TFT back lighting.

See **tern\186\samples\et\et_relay.c** and **\tern_docs\parts\relay9007.pdf** for details.

3.5.5 High-Voltage, High-Current Drivers

ULN2003A has high voltage, high current Darlington transistor arrays, consisting of seven silicon NPN Darlington pairs on a common monolithic substrate. All channels feature open-collector outputs for sinking 350 mA at 50V, and integral protection diodes for driving inductive loads. Peak inrush currents of up to 600 mA sinking are allowed. By default, U20 provides high-voltage driver outputs.

These outputs may be paralleled to achieve high-load capability, although each driver has a maximum continuous collector current rating of 350 mA at 50V. The maximum power dissipation allowed is 2.20 W per chip at 25 degrees C (°C). The common substrate G is routed to GND. All current sinking in must return to GND. A heavy gauge (20) wire must be used to connect a GND terminal to an external common ground return. K connects to the protection diodes in the ULN2003A chips and should be tied to highest voltage in the external load system. K can be connected to an unregulated on board +12V via J4 pin 32. **ULN2003A is a *sinking* driver.** An example of typical application wiring is shown below.

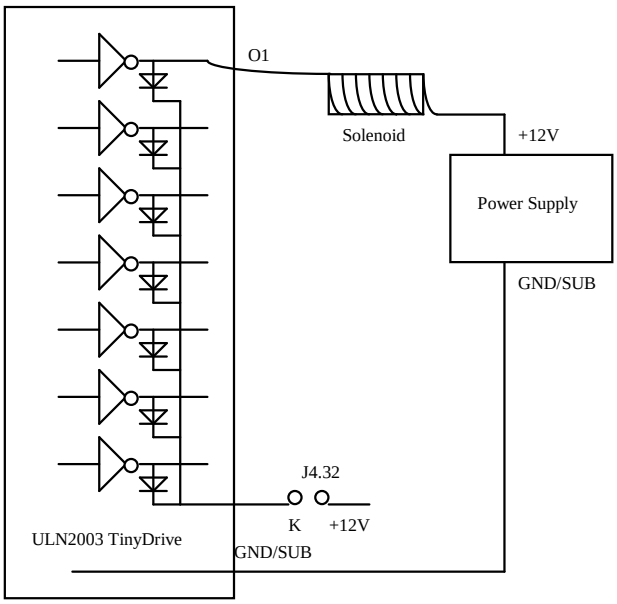


Figure 3.3 Drive inductive load with high voltage/current drivers.

3.5.6 16-bit parallel high speed ADC (AD7655)

The unique 16-bit parallel ADC (AD7655, 0-5V) supports ultra high-speed (1 MHz conversion rate) analog signal acquisition. The AD7655 contains two low noise, high bandwidth track-and-hold amplifiers that allow *simultaneous* sampling on two channels. Each track-and hold amplifier has a multiplexer in front to provide a total of 4 channels analog inputs. The parallel ADC achieves very high throughput by requiring

only two CPU I/O operations (one start, one read) to complete a 16-bit ADC reading. With a precision external 2.5V reference, the ADC accepts 0-5V analog inputs at 16-bit resolution of 0-65,535.

See sample program **\tern\186\samples\et\et_ad16.c** for details on reading the ADC. The sample program is also included in the pre-built sample project; **\tern\186\samples\et\et.pdl**.

Refer to the data sheet for additional specifications; \tern_docs\parts\ad7655.pdf.

3.5.7 24-bit, 16-channel ADC(LTC2448)

A 24-bit LTC2448 sigma-delta ADC can be installed. The LTC2448 chip offers 8 ch. differential or 16 ch. single-ended input channels. Variable speed/resolution settings can be configured. A peak single-channel output rate of 5 KHz can be achieved.

The LTC2448 switches the analog input to a 2 pf capacitor at 1.8MHz with an equivalent input resistance of 110K ohm. The ADC works well directly with strain gages, current shunts, RTDs, resistive sensors, and 4-20mA current loop sensors. The ADC can also work well directly with thermocouples in the differential mode. By default, a precision reference with a internal temperature sensor(LT1019, 2.5V) is installed, providing local temperature measurement for thermocouple applications.

The software source sample code on TERN CD, **c:\tern\186\samples\et\et_ad24.c**, allows user to modify the input reading resolution. For digital inputs, only one byte reading is needed. Also see Chapter 4 for software channel / hardware pin details.

3.5.8 16-bit, 8-channel DAC(LTC2600)

The LTC2600 is an eight channel 16-bit digital-to-analog converter (DAC) in an SO-8 package. It is complete with a rail-to-rail voltage output amplifier capable of driving up to 15mA. It uses a 3-wire SPI compatable serial interface and has an output range of 0-REF volts, making 1 LSB equal to REF/65535 V. The reference voltage input is by default shorted to VCC. The REF voltage must be greater than GND and less than VCC. The DAC outputs are routed to the J4 pin header, pins 19-26.

The DAC is installed on the **ET™** at location U21 and uses P26 as the chip select. The synchronous serial interface is used to send data to the device. Refer to the sample code, **\tern\186\samples\et\et_da.c** for an example on driving the DAC.

The sample is also included in the pre-built sample project **\tern\186\samples\et\et.pdl.**

Refer to the DAC data sheet for additional specifications; **\tern_docs\parts\ltc2600.pdf.**

3.5.9 CompactFlash Interface

By utilizing the compact flash interface on the **ET™**, users can easily add widely used 50-pin CF standard mass data storage cards to their embedded application via RS232, TTL I2C, or parallel interface. TERN software supports Linear Block Address mode, 16-bit FAT flash file system, RS-232, TTL I2C or parallel communication. Users can write files to the CompactFlash card or read files from the CompactFlash card. Users can also transfer files to a PC via the CF reader port.

CF cards can also be used as a means to store images and data to be displayed onto the LCD. This allows users to have access to unlimited images to be used in an application in conjunction with the LCD. As

dicussed above, the AM186ES suuports DMA to allow images/data to be transferred directly to the image buffer for increased speed.

Sample code and function prototypes are available to assist in creating applications which use the file system to access the CF. Refer to the target **\tern\186\samples\et\fs_cmds1.axe**. This sample uses the source code **\tern\186\samples\flashcore\fs_cmds1.c**. Also, for a complete listing of file system function prototypes and data types, refer to the header files "fileio.h" and "filegeo.h" found the **\tern\186\include** directory.

3.5.10 100 MHz BaseT Ethernet

An WizNet™ Fast Ethernet Module can be installed to provide 100M Base-T network connectivity. This Ethernet module has a hardware LSI TCP/IP stack. It implements TCP/IP, UDP, ICMP and ARP in hardware, supporting internet protocol DLC and MAC. It has 16KB internal transmit and receiving buffer which is mapped into host processor's direct memory. The host can access the buffer via high speed DMA transfers. The hardware Ethernet module releases internet connectivity and protocol processing from the host processor. It supports 4 independent stack connections simultaneously at a 4Mbps protocol processing speed. An RJ45 8-pin connector is on-board for connecting to 10/100 Base-T Ethernet network. A software library is available for Ethernet connectivity.

See target **\tern\186\samples\et\et_http.axe** for a demo of this module (utilizing the CompactFlash interface).

Figure 3.4 WizNet Ethernet Module.

3.5.11 Power Supplies

The **ET™** supports three different 5V DC regulators: linear regulator, switching regulator, and low drop regulator.

- Linear 5V regulator, input voltage is 9-12V DC.
- Switching 5V Regulator with VOFF, input voltage is 9-24V DC
- Low Drop regulator with VOFF, input voltage can be as low as 5.1V power

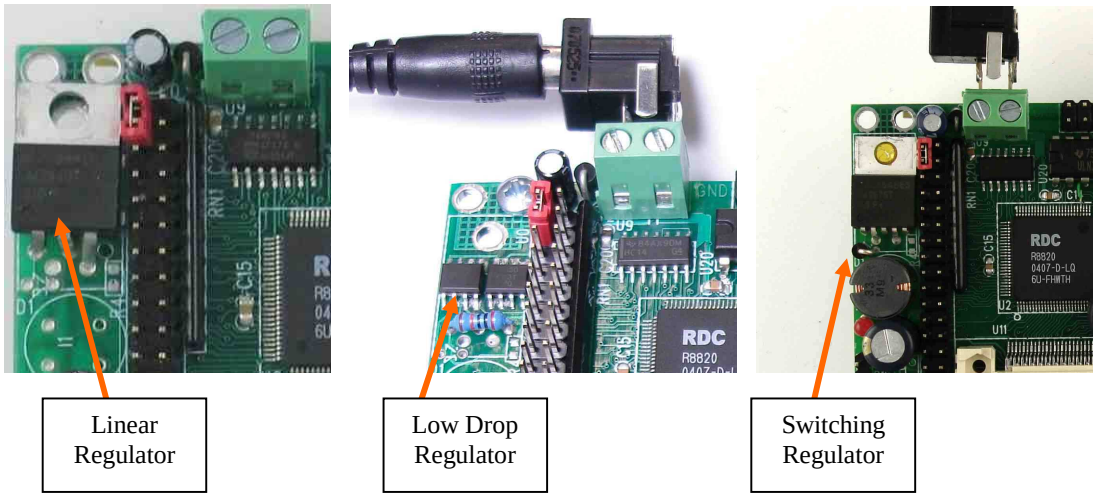


Figure 3.5 Power regulator options.

The switching regulator and the low drop regulator can be turned on or off using the VOFF pin in conjunction with the real time clock STD pin. In order to use VOFF, a pull-up resistor at R4 must be installed. See the VOFF sample program **`tern\186\samples\et\et_voff.c`** for details.

3.5.12 320x240 Pixel TFT with Touch Screen or Push-buttons

The 3.5” color TFT display has 320x240 pixels and a 71x54 mm viewing area. Each pixel uses 16-bit (5R, 6G, 5B) TFT color. The TFT is embedded with a high performance graphic controller with a 2D drawing engine to support high speed line, box, and circle drawing. Six mechanical push bottoms or a Touch Screen can be installed. Easy user interface software is available. See samples ***tft_top.c***, ***et_grid.c***, ***et_img.c***, and ***et_key.c*** in **\tern\186\samples\et** for software details regarding the LCD and push-buttons.

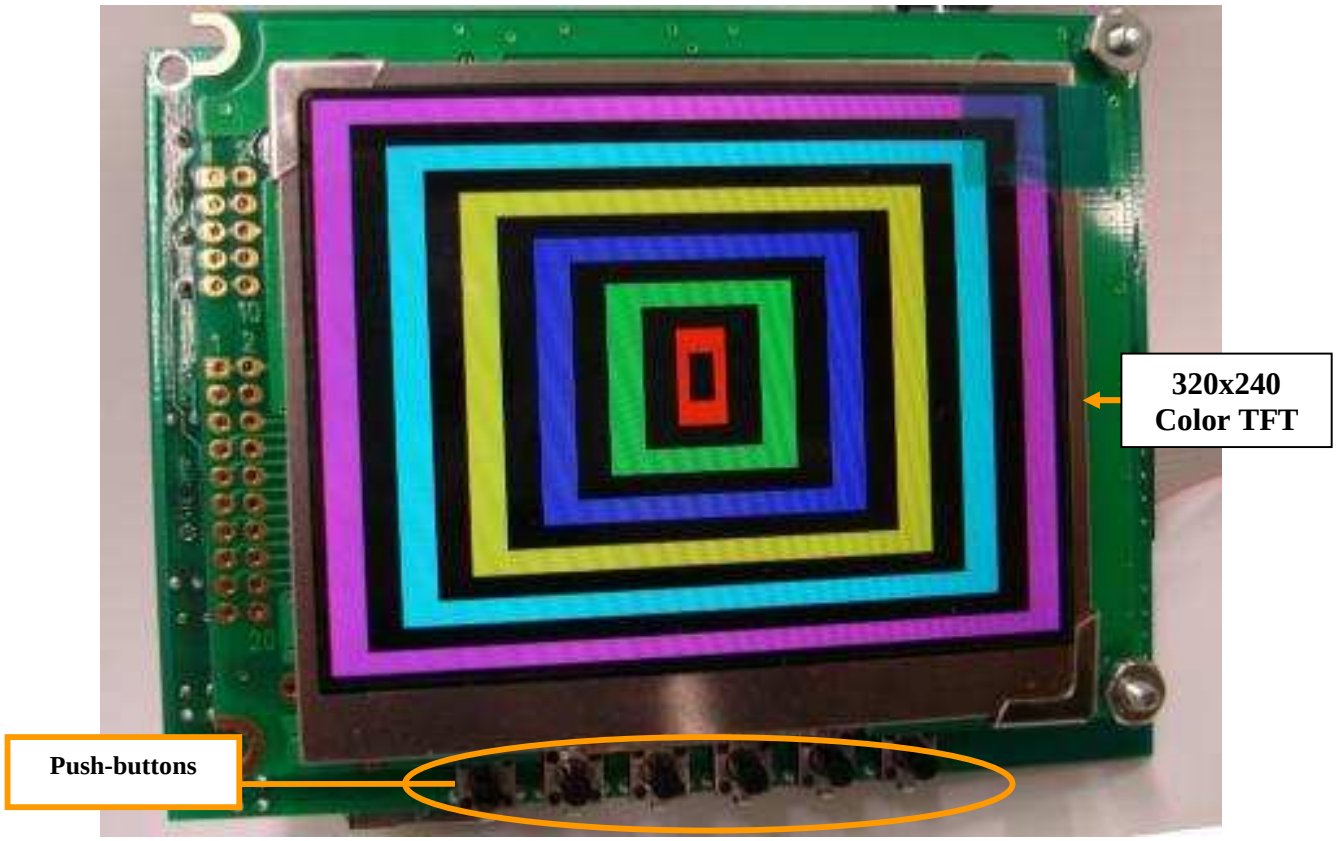


Figure 3.6 Color TFT and Push-buttons shown.

3.6 Headers and Connectors

3.6.1 Expansion Headers J1 and J2

There are two 20x2 0.1 spacing headers for expansion. Most signals are directly routed to the Am186ES processor. These signals are 5V only, and any out-of-range voltages will most likely damage the board.

J2 Signal			
GND	40	39	VCC
P4	38	37	P14
/CTS0	36	35	P6
TXD0	34	33	/INT4
RXD0	32	31	/RTS1
P5	30	29	P1
TXD1	28	27	/RTS0
RXD1	26	25	
P2	24	23	P15
/CTS1	22	21	/INT3
P0	20	19	INT2
P25	18	17	P24
	16	15	P3
	14	13	
P10	12	11	P13
A19	10	9	
/INT0	8	7	/NMI
/INT1	6	5	N2
P26	4	3	P29
GND	2	1	S2

J1 Signal			
VCC	1	2	GND
	3	4	CLK
	5	6	GND
	7	8	D0
VOFF	9	10	D1
	11	12	D2
D15	13	14	D3
/RST	15	16	D4
RST	17	18	D5
P16	19	20	D6
D14	21	22	D7
D13	23	24	GND
	25	26	A7
D12	27	28	A6
/WR	29	30	A5
/RD	31	32	A4
D11	33	34	A3
D10	35	36	A2
D9	37	38	A1
D8	39	40	A0

3.6.2 DAC/ADC/HV Header J4

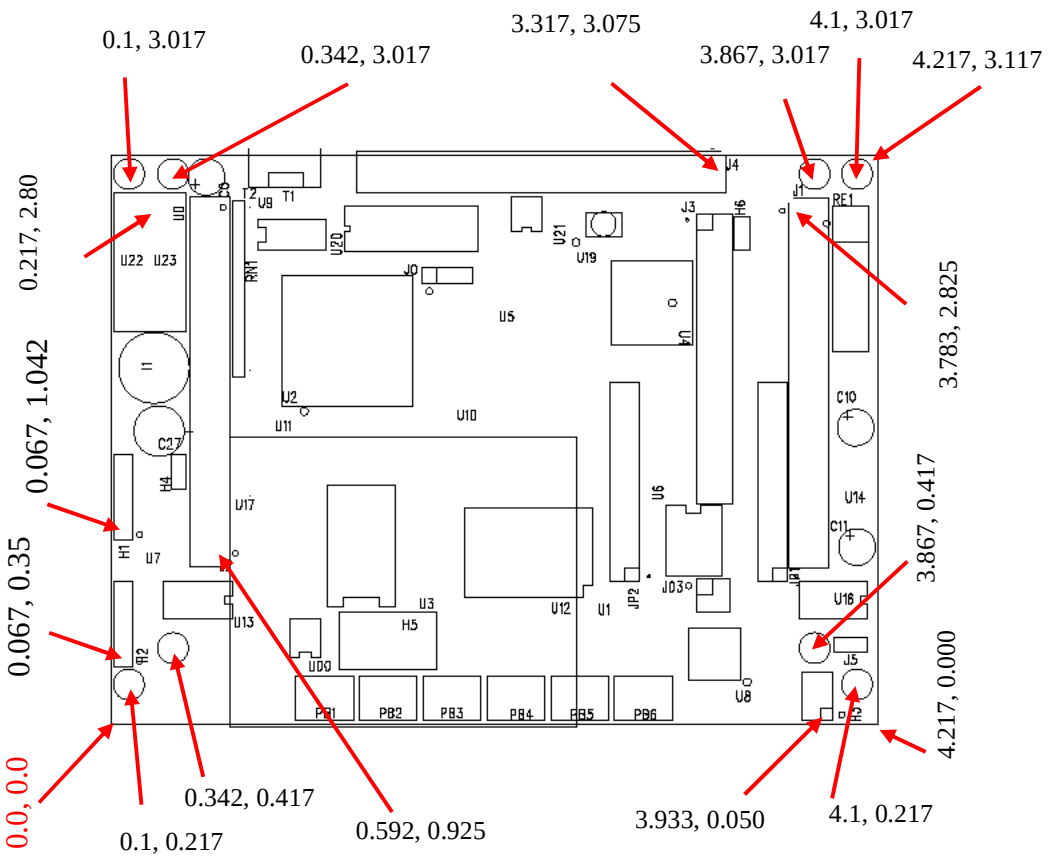
J4 Signals			
B01	1	2	B00
B03	3	4	B02
B05	5	6	B04
B07	7	8	B06
B09	9	10	B08
B11	11	12	B10
B13	13	14	B12
B15	15	16	B14
GND	17	18	GND
V1	19	20	V2
V3	21	22	V4
V5	23	24	V6
V7	25	26	V8
S1	27	28	N1
VCC	29	30	PWR
GND	31	32	
HV7	33	34	K
HV5	35	36	HV6
HV3	37	38	HV4
HV1	39	40	HV2

3.6.3 Jumpers

Jumper Setting	*Function*
J2 pin 38 = J2 pin 40	Step 2
J5 pin 1 = J5 pin 2	Watchdog
H4 pin 1 = H4 pin 2	VOFF
H6 pin 1 = H6 pin 2	TFT Power

ET Layout

The **ET PCB** measures 4.217 x 3.117 inches. All dimensions are in inches. For mounting holes, arrows point to the center of the mounting hole.



Appendix B: Software Glossary

The following is a glossary of library functions for the A-Engine86.

---

void ae_init(void) ae.h

Initializes the AM188ES processor. The following is the source code for ***ae_init()***

```
outport(0xffa0,0xc0bf);      // UMCS, 256K ROM, 3 wait states, disable AD15-0
outport(0xffa2,0x7fbc);      // 512K RAM, 0 wait states
outport(0xffa8,0xa0bf); // 256K block, 64K MCS0, PCS I/O
outport(0xffa6,0x81ff);      // MMCS, base 0x80000
outport(0xffa4,0x007f); // PACS, base 0, 15 wait

outport(0xff78,0xe73c);     // PDIR1, TxD0, RxD0, TxD1, RxD1, P16=PCS0, P17=PCS1=PPI
outport(0xff76,0x0000);     // PIOM1
outport(0xff72,0xec7b);     // PDIR0, P12,A19,A18,A17,P2=PCS6=RTC
outport(0xff70,0x1000);     // PIOM0, P12=LED

outportb(0x0103,0x9a);     // all pins are input, I20-23 output
outportb(0x0100,0);
outportb(0x0101,0);
outportb(0x0102,0x01);     // I20=ADCS high
clka_en(0);
enable();
```

Reference: led.c

---

void ae_reset(void) ae.h

Resets AM188 processor.

---

void delay_ms(int m) ae.h

Approximate microsecond delay. Does not use timer.

```
Var:  m – Delay in approximate ms
```

Reference: led.c

---

void led(int i) ae.h

Toggles P12 used for led.

```
Var:  i - Led on or off
```

Reference: led.c

---

void delay0(unsigned int t) ae.h

Approximate loop delay. Does not use timer.

```
Var:  m – Delay using simple for loop up to t.
```

Reference:

---

void pwr_save_en(int i) ae.h

Enables power save mode which reduces clock speed. Timers and serial ports will be effected. Disabled by external interrupt.

```
Var:  i – 1 enables power save only.  Does not disable.
```

Reference: ae_pwr.c

---

void clka_en(int i) ae.h

Enables signal CLK respectively for external peripheral use.

```
Var: i – 1 enables clock output, 0 disables (saves current when
disabled).
```

Reference:

---

void hitwd(void) ae.h

Hits the watchdog timer using P03. P03 must be connected to WDI of the MAX691 supervisor chip.

Reference: *See Hardware chapter of this manual for more information on the MAX691.*

---

void pio_init(char bit, char mode) ae.h

Initializes a PIO line to the following:
mode=0, Normal operation
mode=1, Input with pullup/down
mode=2, Output
mode=3, input without pull

```
Var:  bit – PIO line 0 - 31
      Mode – above mode select
```

Reference: ae_pio.c

---

void pio_wr(char bit, char dat) ae.h

Writes a bit to a PIO line. PIO line must be in an output mode
mode=0, Normal operation
mode=1, Input with pullup/down
mode=2, Output
mode=3, input without pull

```
Var:  bit – PIO line 0 - 31
      dat – 1/0
```

Reference: ae_pio.c

---

unsigned int pio_rd(char port) ae.h

Reads a 16 bit PIO port.

```
Var:  port – 0: PIO 0 - 15
             1: PIO 16 – 31
```

Reference: ae_pio.c

---

void outport(int portid, int value) dos.h

Writes 16-bit *value* to I/O address *portid*.

```
Var:  portid – I/O address
      value – 16 bit value
```

Reference: ae_ppi.c

---

void outportb(int portid, int value) dos.h

Writes 8-bit *value* to I/O address *portid*.

```
Var:  portid – I/O address
      value – 8 bit value
```

Reference: ae_ppi.c

---

int inport(int portid) dos.h

Reads from an I/O address *portid*. Returns 16-bit value.

```
Var:  portid – I/O address
```

Reference: ae_ppi.c

---

int inportb(int portid) dos.h

Reads from an I/O address *portid*. Returns 8-bit value.

```
Var:  portid – I/O address
```

Reference: ae_ppi.c

---

int ee_wr(int addr, unsigned char dat) aeee.h

Writes to the serial EEPROM.

```
Var:  addr – EEPROM data address
      dat - data
```

Reference: ae_ee.c

---

int ee_rd(int addr) aeee.h

Reads from the serial EEPROM. Returns 8-bit data

```
Var:  addr – EEPROM data address
```

Reference: ae_ee.c

---

int ae_ad12(unsigned char c) ae.h

Reads from the 11-channel 12-bit ADC. Returns 12 bit AD data of the previous channel.
In order to operate ADC, I20,I21,I22 must be output and P11 must be input.
P11 is shared by RTC, EE. It must left high at power-on/reset.
Unipolar:
Vref- = 0x000
Vref+ = 0xfff
Use 1 wait state for Memory and I/O without RDY, < 300 us execution time
Use 0 wait state for Memory and I/O with VEP010, < 270 us execution time

```
Var:  c – ADC channel.
                  c = {0 … a}, input ch = 0 – 10
                  c = b,       input ch = (vref+ - vref-) /2
                  c = c,       input ch = vref-
                  c = d,       input ch = vref+
                  c = e,       software power down
```

Reference: ae_ad12.c

---

void io_wait(char wait) ae.h

Setup I/O wait states for I/O instructions.

```
Var:  wait – wait duration {0…7}
      wait=0, wait states = 0, I/O enable for 100 ns
      wait=1, wait states = 1, I/O enable for 100+25 ns
      wait=2, wait states = 2, I/O enable for 100+50 ns
      wait=3, wait states = 3, I/O enable for 100+75 ns
      wait=4, wait states = 5, I/O enable for 100+125 ns
      wait=5, wait states = 7, I/O enable for 100+175 ns
      wait=6, wait states = 9, I/O enable for 100+225 ns
      wait=7, wait states = 15, I/O enable for 100+375 ns
```

Reference:

---

void rtc_init(unsigned char * time) ae.h

Sets real time clock date, year and time.

```
Var:  time – time and date string
      String sequence is the following:
            time[0] = weekday
            time[1] = year10
            time[2] = year1
            time[3] = mon10
            time[4] = mon1
            time[5] = day10
            time[6] = day1
            time[7] = hour10
            time[8] = hour1
            time[9] = min10
            time[10] = min1
            time[11] = sec10
            time[12] = sec1
unsigned char time[]={2,9,8,0,7,0,1,1,3,1,0,2,0};
/* Tuesday, July 01, 1998, 13:10:20 */
```

Reference: rtc_init.c

---

int rtc_rd(TIM *r) ae.h

Reads from the real time clock.

```
Var:  *r – Struct type TIM for all of the RTC data
      typedef struct{
            unsigned char sec1, sec10,  min1, min10, hour1, hour10;
            unsigned char day1, day10, mon1, mon10, year1, year10;
            unsigned char wk;
            } TIM;
```

Reference: rtc.c

---

void t2_init(int tm, int ta, void interrupt far(*t2_isr)()); ae.h

void t1_init(int tm, int ta, int tb, void interrupt far(*t1_isr)());
void t0_init(int tm, int ta, int tb, void interrupt far(*t0_isr)());

Timer 0, 1, 2 initialization.

```
Var:  tm – Timer mode. See pg. 8-3 and 8-5 of the AMD CPU Manual
      ta – Count time a (1/4 clock speed).
      tb – Count time b for timer 0 and 1 only (1/4 clock).
           Time a and b establish timer duty cycle (PWM). See
           hardware chapter.
      t#_isr – pointer to timer interrupt routine.
```

Reference: timer.c, timer1.c, timer02.c, timer2.c, timer0.c timer12.c

---

void nmi_init(void interrupt far (* nmi_isr)()); ae.h
void int0_init(unsigned char i, void interrupt far (*int0_isr)());
void int1_init(unsigned char i, void interrupt far (*int1_isr)());
void int2_init(unsigned char i, void interrupt far (*int2_isr)());
void int3_init(unsigned char i, void interrupt far (*int3_isr)());
void int4_init(unsigned char i, void interrupt far (*int4_isr)());
void int5_init(unsigned char i, void interrupt far (*int5_isr)());
void int6_init(unsigned char i, void interrupt far (*int6_isr)());

Initialization for interrupts 0 through 6 and NMI (Non-Maskable Interrupt).

```
Var:  i – 1: enable, 0: disable.
      int#_isr – pointer to interrupt service.
```

Reference: intx.c

---

void s0_init(unsigned char b, unsigned char* ibuf, int isiz, ser0.h
unsigned char* obuf, int osiz, COM *c) (void);
void s1_init(unsigned char b, unsigned char* ibuf, int isiz, ser1.h
unsigned char* obuf, int osiz, COM *c) (void);

Serial port 0, 1 initialization.

```
Var:  b – baud rate. Table below for 40MHz and 20MHz Clocks.
      ibuf – pointer to input buffer array
      isiz – input buffer size
      obuf – pointer to output buffer array
      osiz – ouput buffer size
      c – pointer to serial port structure. See AE.H for COM
      structure.
```

b	baud (40MHz)	baud (20MHz)
1	110	55
2	150	110
3	300	150
4	600	300
5	1200	600
6	2400	1200
7	4800	2400
8	9600	4800
9	19200	9600
10	38400	19200

11	57600	38400
12	115200	57600
13	23400	115200
14	460800	23400
15	921600	460800

Reference: s0_echo.c, s1_echo.c, s1_0.c

---

void scc_init(unsigned char m1, unsigned char m2, unsigned char b, unsigned char* ibuf,int isiz, unsigned char* obuf,int osiz, COM *c) scc.h

Serial port 0, 1 initialization.

Var: m1 = SCC691 MR1
m2 = SCC691 MR2
b – baud rate. *Table below for 8MHz Clock.*
ibuf – pointer to input buffer array
isiz – input buffer size
obuf – pointer to output buffer array
osiz – ouput buffer size
c – pointer to serial port structure. *See* AE.H *for* COM *structure.*

m1 bit	**Definition**
7	(RxRTS) receiver request-to-send control, 0=no, 1=yes
6	(RxINT) receiver interrupt select, 0=RxRDY, 1=FIFO FULL
5	(Error Mode) Error Mode Select, 0 = Char., 1=Block
4-3	(Parity Mode), 00=with, 01=Force, 10=No, 11=Special
2	(Parity Type), 0=Even, 1=Odd
1-0	(# bits) 00=5, 01=6, 10=7, 11=8

m2 bit	**Definition**
7-6	(Modes) 00=Normal, 01=Echo, 10=Local loop, 11=Remote loop
5	(TxRTS) Transmit RTS control, 0=No, 1= Yes
4	(CTS Enable Tx), 0=No, 1=Yes
3-0	(Stop bit), 0111=1, 1111=2

b	**baud (8MHz)**
1	110
2	150
3	300
4	600
5	1200
6	2400
7	4800
8	9600
9	19200
10	31250
11	62500
12	125000
13	250000

Reference: s0_echo.c, s1_echo.c, s1_0.c

int putser0(unsigned char ch, COM *c); ser0.h
int putser1(unsigned char ch, COM *c); ser1.h
int putser_scc(unsigned char ch, COM *c); scc.h

Output 1 character to serial port. Character will be sent to serial output with interrupt isr.

```
Var:  ch – character to output
      c – pointer to serial port structure
```

Reference: s0_echo.c, s1_echo.c, s1_0.c

---

int putsers0(unsigned char *str, COM *c); ser0.h
int putsers1(unsigned char *str, COM *c); ser1.h
int putsers_scc(unsigned char ch, COM *c); scc.h

Output a character string to serial port. Character will be sent to serial output with interrupt isr.

```
Var:  str – pointer to output character string
      c – pointer to serial port structure
```

Reference: ser1_sin.c

---

int serhit0(COM *c); ser0.h
int serhit1(COM *c); ser1.h
int serhit_scc(COM *c); scc.h

Checks input buffer for new input characters. Returns 1 if new character is in input buffer, else 0.

```
Var:  c – pointer to serial port structure
```

Reference: s0_echo.c, s1_echo.c, s1_0.c

---

unsigned char getser0(COM *c); ser0.h
unsigned char getser1(COM *c); ser1.h
unsigned char getser_scc(COM *c); scc.h

Retrieve 1 character from the input buffer. Assumes that *serhit* routine was evaluated.

```
Var:  c – pointer to serial port structure
```

Reference: s0_echo.c, s1_echo.c, s1_0.c

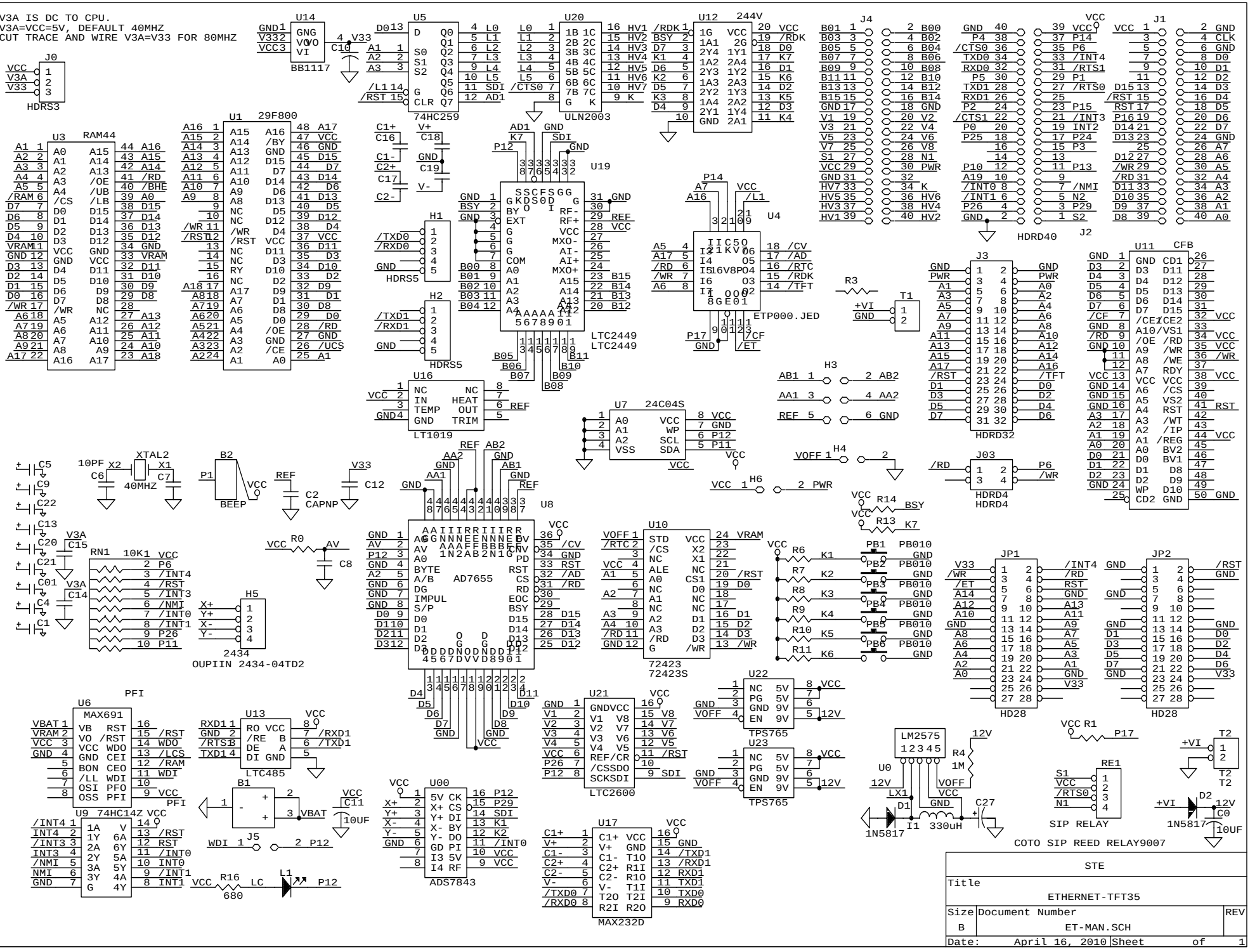
---

int getsers0(COM *c, int len, unsigned char *str); ser0.h
int getsers1(COM *c, int len, unsigned char *str); ser1.h
int getsers_scc(COM *c, int len, unsigned char *str); scc.h

Retrieves a fixed length character string from the input buffer. If the buffer contains less characters than the length requested, *str* will contain only the remaining characters from the buffer. Appends a '\0' character to the end of *str*. Returns the retrieved string length.

```
Var:  c – pointer to serial port structure
      len – desired string length
      str – pointer to output character string
```

Reference: ser1.h, ser0.h for source code.


V3A IS DC TO CPU.
V3A=VCC=5V, DEFAULT 40MHZ
CUT TRACE AND WIRE V3A=V33 FOR 80MHZ
U14
BB1117
U5
74HC259
U20
ULN2003
U12
244V
J4
J2
HDRD40
J1
J0
HDRS3
U3
RAM44
U1
29F800
U19
LTC2449
LTC2449
H1
HDRS5
H2
HDRS5
U16
LT1019
U4
ETP000.JED
R3
T1
J3
HDRD32
U11
CFB
U7
24C04S
H3
U8
AD7655
XTAL2
40MHZ
B2
BEEP
RN1
10K1
H5
2434
OUPIIN 2434-04TD2
R0
U10
72423
72423S
R6
R7
R8
R9
R10
R11
PB1
JP1
JP2
HD28
HD28
J03
HDRD4
HDRD4
H4
H6
R14
R13
U6
MAX691
PFI
U13
LTC485
U9
74HC14Z
B1
J5
R16
680
L1
U00
ADS7843
U21
LTC2600
U22
TPS765
U23
TPS765
U17
MAX232D
U0
LM2575
R4
1M
1N5817
I1
330uH
C27
R1
P17
RE1
SIP RELAY
T2
D2
1N5817
COTO SIP REED RELAY9007
STE
ETHERNET-TFT35
ET-MAN.SCH
April 16, 2010