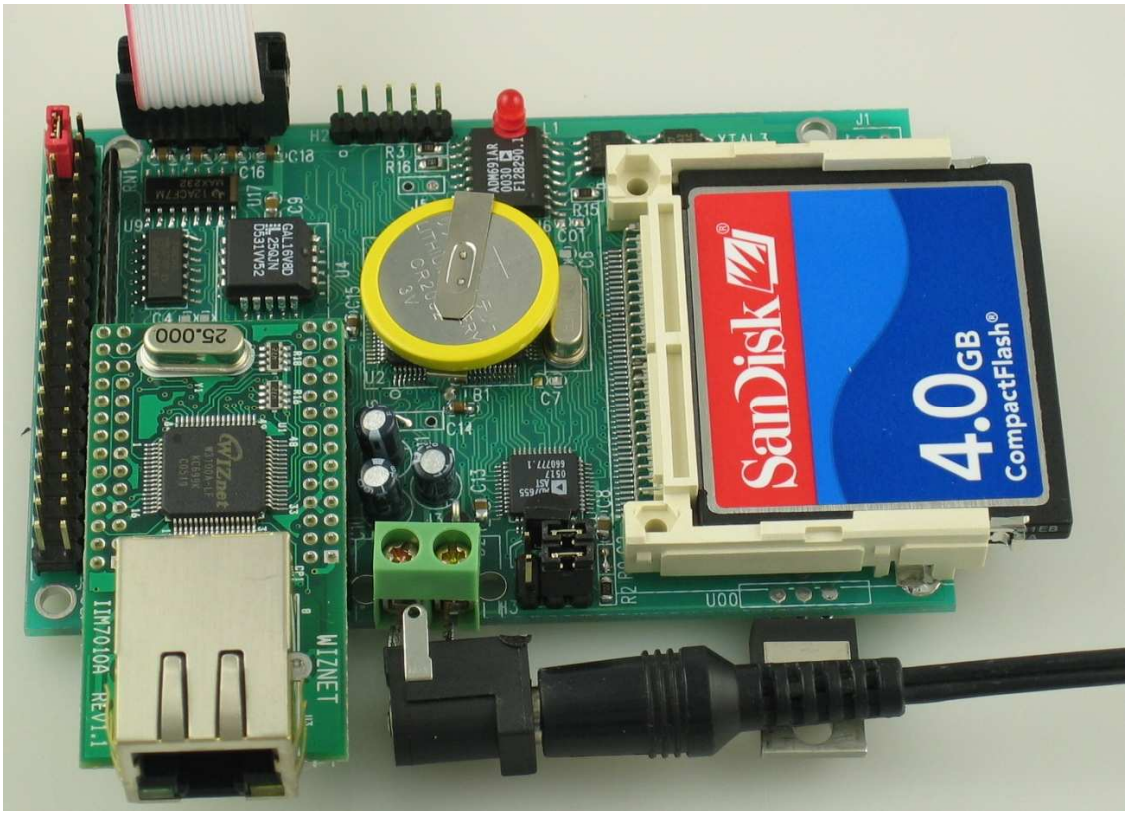
E-Engine™

16-bit Controller, 100M BaseT Ethernet, USB, RS232, CompactFlash, and
16-bit parallel high speed ADC



Technical Manual

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

Version 2.0

October 21, 2010

© 1993-2010 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

Measuring 3.6 x 2.3 inches, the ***E-Engine™*** (***EE***) is a high performance, low cost, C/C++ programmable controller. It is intended for networking application including industrial process control, high-speed data acquisition, and especially ideal for OEM applications.

The ***EE*** can use any one of these 3 16-bit CPU chips: 40 MHz AM186ES(5V), or 40 MHz RDC R8820(5V) or 80MHz RDC R1120(3.3V).

An 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, which represents a huge improvement over software-based TCP/IP stacks. No processor cycles are used to track packet transmission/retransmission, timeouts, etc. The resulting system can easily handle transmissions in the 100K bytes per second+ range in real world applications. 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. Software libraries and demo project are available for Ethernet connectivity.

A high-performance USB stack chip to provide an easy to program USB 1.1/2.0 slave interface. The onboard hardware fully handles USB stack processing, and provides for high-speed bi-directional 8-bit parallel communication. The hardware interface includes 384 bytes of FIFO transmit buffer, and 128 bytes of FIFO for the receiving buffer, making this an ideal low-overhead solution for all embedded applications. No USB specific firmware programming is required on the controller side.

The ***EE*** features fast execution times through 16-bit ACTF Flash (256 KW) and battery-backed SRAM (256 KW). It also includes 3 timers, PWMs, 20+ PIOs, 512-byte serial EEPROM, two UARTs, 3 timer/counters, and a watchdog timer. The three 16-bit timers can be used to count or time external events, up to 10 MHz, or to generate non-repetitive or variable-duty-cycle waveforms as PWM outputs. The PIO pins are multifunctional and user programmable.
A serial real timer clock (DS1337, Dallas) is a low power clock/calendar with two time-of-day alarms and a programmable square-wave output.

Two RS232 channels of full-duplex asynchronous receivers and transmitters are on-board. The UARTs incorporate 9-bit mode for multi-processor communications.

A 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 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.

The ***EE*** supports low cost, removable, up to 2 GB mass storage CompactFlash cards with onboard CompactFlash interface. User can store and transfer large amounts of data with a PC, via a CF card with TERN's FAT filesystem software support.
The ***EE*** can be powered by USB, or regulated 5V, or unregulated 9V DC power with on-board 5V regulator installed.

The ***EE*** provides a true 16-bit data bus for SRAM, Flash, ADC, Ethernet, and a J1 20x2 expansion header. The *EE* is an ideal upgrade for the A-Engine, V25-Engine, 386-Engine, or R-Engine providing increased reliability, networking functionality, and performance. They have the similar mechanical dimensions, pin outs, software drivers, and both are programmed using Paradigm C++ TERN Edition Evaluation Kit (EV-P) or Development Kit (DV-P).

The ***EE*** can be integrated into an OEM product as a processor core component. It also can be used to build a smart sensor, or can act as a node in a distributed microprocessor system.

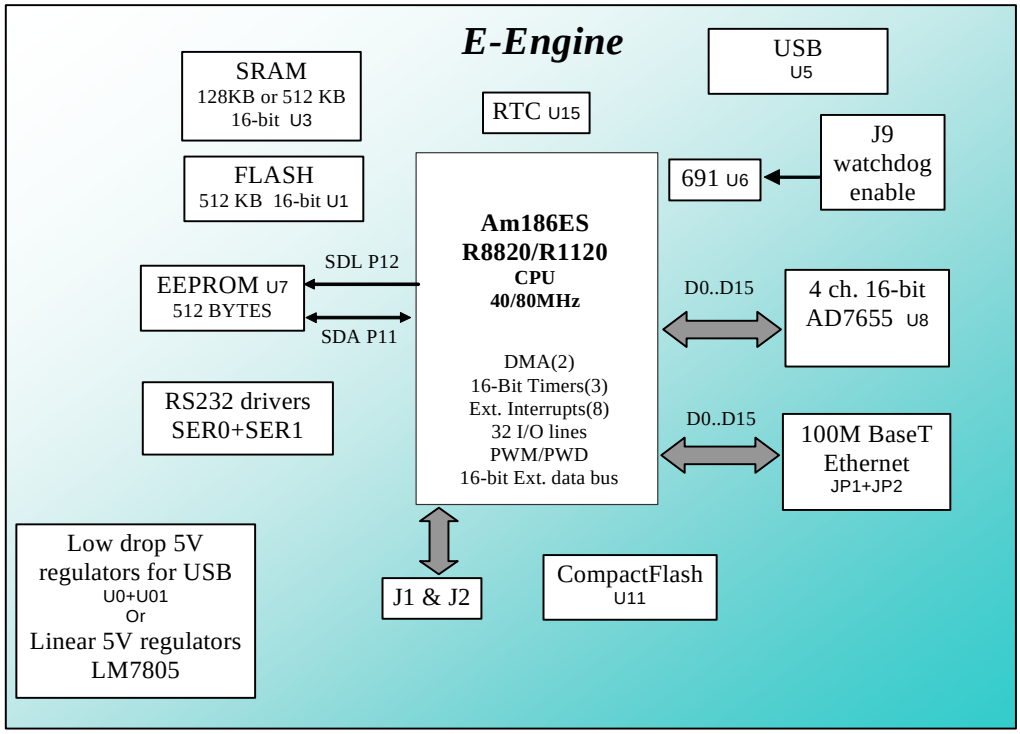


Figure 1.1 Functional block diagram of the E-Engine

The ***EE*** supports on-board 512 KB 16-bit Flash and up to 512 KB 16-bit battery-backed SRAM. The on-board ACTF Flash has a protected boot loader and can be easily programmed in the field via serial link. Users can download a kernel into the Flash for remote debugging. With the DV-P Kit support, user application codes can be easily field-programmed into and run out of the Flash.

A 512-byte serial EEPROM is included on-board. Two DMA-driven serial ports from the Am186ES support high-speed, reliable serial communication at a rate of up to 115,200 baud. All serial ports support 8-bit and 9-bit communication.

There are three 16-bit programmable timers/counters and a watchdog timer. Two timers can be used to count or time external events, at a rate of up to 10 MHz, or to generate non-repetitive or variable-duty-cycle waveforms as PWM outputs. Pulse Width Demodulation (PWD), a distinctive feature, can be used to measure the width of a signal in both its high and low phases. It can be used in many applications, such as bar-code reading.

The ***EE*** *has* 32 user-programmable, multifunctional I/O pins from the CPU. Schmitt-trigger inverters are provided for six external interrupt inputs, to increase noise immunity and transform slowly-changing input signals into fast-changing and jitter-free signals. A supervisor chip with power failure detection, a watchdog timer, an LED, and expansion ports are on-board.

Features:

- 3.6 x 2.3 x 1”, 200 mA at 5V for 80 MHz
- 40 or 80 MHz, 16-bit CPU, program in C/C++
- 256 KW 16-bit Flash, 256 KW 16-bit SRAM, 512 bytes EE
- 20+ TTL I/Os, Real-time clock, 2 serial ports, PWM, counters
- 4 ch 16-bit parallel high speed ADC (AD7655)
- Hardware TCP/IP stack for 100M Base-T Ethernet
- CompactFlash card with FAT file system support

1.2 Physical Description

The physical layout of the E-Engine is shown in Figure 1.2.

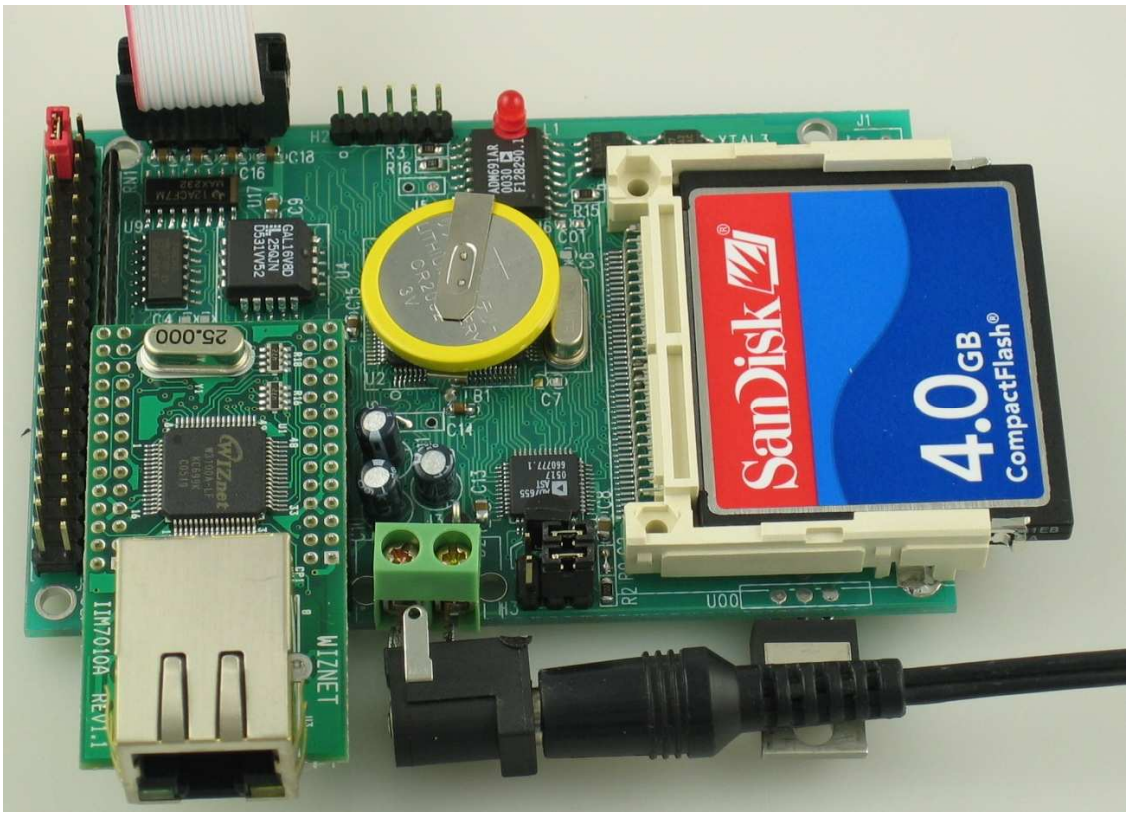


Figure 1.2 Physical layout of the E-Engine

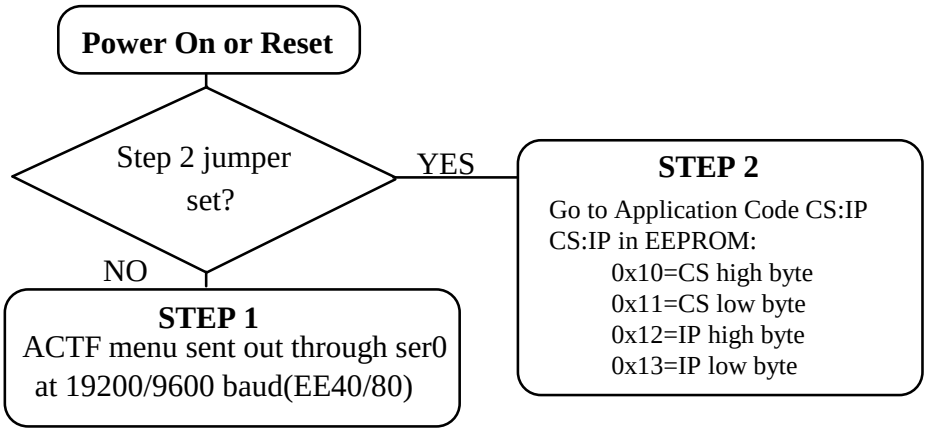


Figure 1.3 Flow chart for ACTF operation

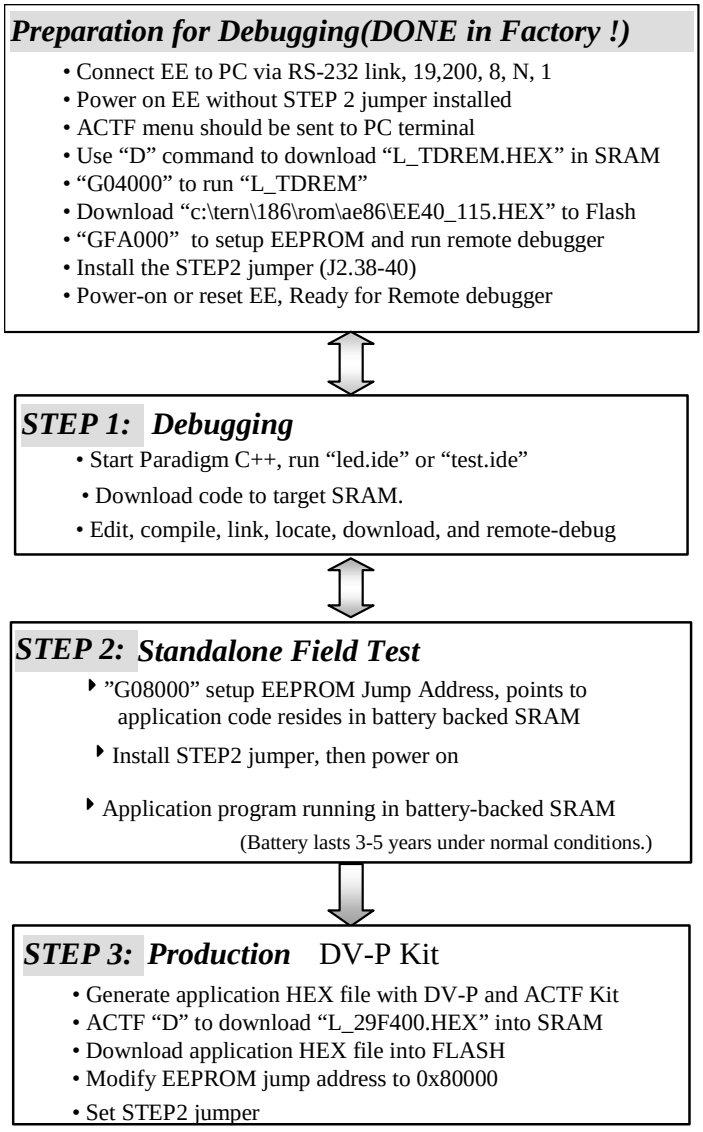
The "ACTF boot loader" resides in the top protected sector of the 512KB on-board Flash chip (29F400).

By default, in the factory, before shipping, the DEBUG kernel (UE40_115.hex) is pre-loaded in the Flash starting at 0xFA000, and the RED STEP2 jumper is installed, ready for Paradigm C++ debugger. User does not need to download a DEBUG kernel to start with.

At power-on or RESET, the "ACTF" will check the STEP 2 jumper. If STEP 2 jumper is not installed, the ACTF menu will be sent out from serial port0 at 19200 baud for a EE40, or 9600 baud for a EE80.
If the STEP 2 jumper is installed, the "jump address" located in the on-board serial EE (see App. B) will be read out and then jump to that address. A DEBUG kernel "EE40_115.hex" for the EE40 or "EE80_115.hex" for the EE80 can be downloaded, residing in "0xFA000" of the 512KB on-board flash chip.
The "EE40_115.hex can also be downloaded into an EE80 for easier running all demo projects, which are designed for running 40MHz.

1.3 E-Engine Programming Overview

Steps for product development:



There is no ROM socket on the EE. The user's application program must reside in SRAM for debugging in STEP1, reside in battery-backed SRAM for the standalone field test in STEP2, and finally be programmed into Flash for a complete product. For production, the user must produce an ACTF-downloadable HEX file for the application, based on the DV-P+ACTF Kit. The "STEP2" jumper (J2 pins 38-40) must be installed for every production-version board.

Step 1 settings

In order to talk to EE with Paradign C++, the EE must meet these requirements:

1) EE40_115.HEX or EE80_115.HEX must be pre-loaded into Flash starting address 0xfa000.

2) The SRAM installed must be large enough to hold your program.

For a 32K SRAM, the physical address is 0x00000-0x07fff

For a 128K SRAM, the physical address is 0x00000-0x01ffff
For a 512K SRAM, the physical address is 0x00000-0x07ffff

3) The on-board EE must have a Jump Address for the EE40_115.HEX with starting address of 0xfa000.

4) The STEP2 jumper must be installed on J2 pins 38-40.

For further information on programming the E-Engine, refer to the manual on the TERN CD under: tern_docs\manuals\software_kit.pdf.

The ***EE*** works with most TERN expansion boards including the P50, P100, P300, MotionC, MMC, and Eye0.

The E-Engine is connecting to a EYE0 Camera

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 (H1) with red edge of cable at pin 1. This DEBUG cable is a 10-pin IDE to DB9 cable, made by TERN (See Appendix D).
- 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 to the PC

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

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

2.2.2 Powering-on the EE™

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. The DC 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.

(See next page for connection diagram).

2.2.3 Connecting the EE™

The proper connections required to debug the board (through Paradigm software).

H1 (Ser 0) is a 5x1 pin header. Use the back row of the IDE cable's female header to connect to H1. (See Appendix D)

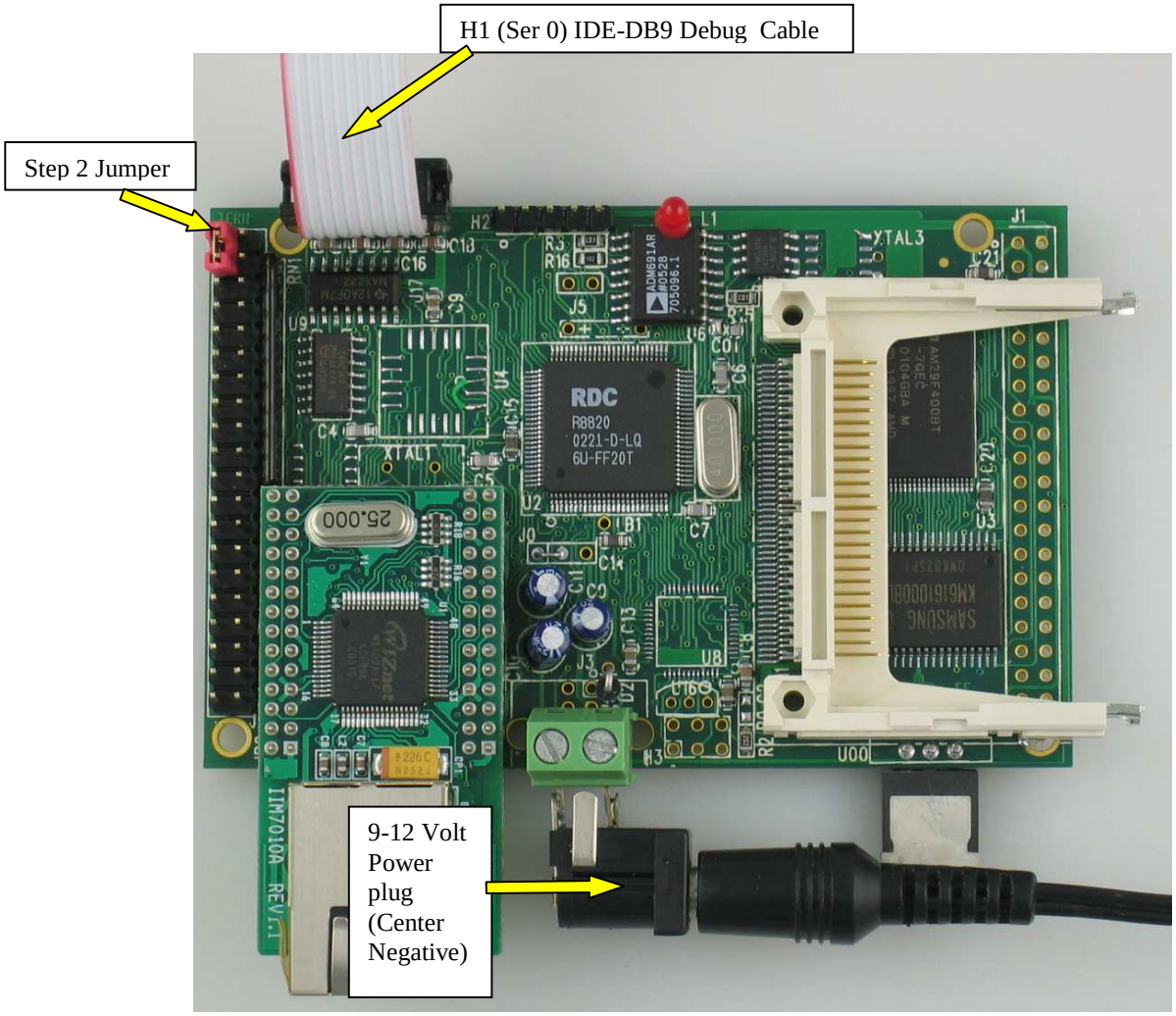


Figure 2.1: Debug Cable (Ser0), Power Plug, and Step 2 Jumper shown

NOTE: Remember to watch for the "**double blink**" off the LED. This indicates the **Debug Kernel** has been loaded with the **jump address** pointing to it. This is mandatory to commence downloading code through the Paradigm environment.

Chapter 3: Hardware

3.1 Am186ES/R8820/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.

R8820 is a drop-in replacement 5V, 40MHz chip for the AM186ES. Connecting J0.1=J0.2.

R1100 is a 80MHz, 3.3V chip can be installed on the E-Engine with J0.2=J0.3.

By default, the E-Engine uses 5V 40 MHz R8820 and low power 55-70 ns SRAM with battery backup.
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 pads is ground, and the other two pads allowing a 3V backup lithium battery be installed in two different positions:
1) The battery's positive lead is installed in the pad which is away from the RTC, supporting the RTC only. No battery backup for the SRAM.
2) The battery's positive lead is installed in the pad which is closer to the RTC, supporting both RTC and SRAM.
In the future, when the fast (10 ns) and low low standby power SRAM is available, then 80 MHz E-Engine can have both RTC and SRAM with battery backup plus the DMA, zero wait state operation.
User can use sample program c:\tern\186\samples\ee\rdc_id.c to read the ID register(0xfff4), in order to identify RDC CPU type.

R1100=0xC5D9, R1120=0x85D9, R8820/30=0x04D9(xxD9)

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 EE40.

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 ACTF manu.

Two debug kernels are available:

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

The EE40_115.hex will run the R1120 at 40 MHz, and the EE80_115.hex will run the R1120 at 80 MHz.

By default, the EE80_115.hex is pre-programmed for the 80 MHz E-Engine.

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, free to use.
/INT1, J2 pin 6, free to use.
INT2, J2 pin 19, RTC DS1337 alarm
/INT3, J2 pin 21, free to use
/INT4, J2 pin 33, used by 100M BaseT Ethernet
INT5=P12=DRQ0, J2 pin 5, used by E-Engine as output for LED/EE/HWD
INT6=P13=DRQ1, J2 pin 11, used by USB TXE
/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 E-Engine 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 EE80, DMA functions are not available when by default low power 55 ns SRAM is installed. If install a 25 ns SRAM, 80MHz EE can have all DMA functions, but it will drain the backup battery fast. Two battery positive pads allowing the battery be installed:

1) Support both RTC and low power SRAM, or

2) Support only RTC.

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 EE pin 5
Timer1 output = P1 = J2 pin 29
Timer1 input = P0 = J2 pin 20

Timer0 input P11 is used and shared by on-board EE, 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.

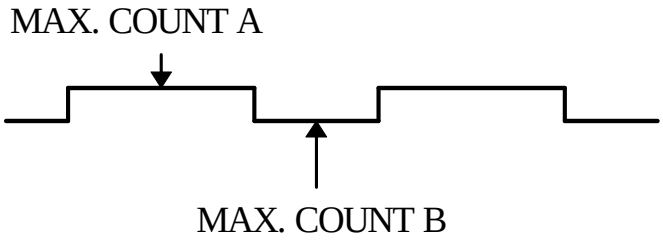
Timer2 is not connected to any external pin. It can be used as an internal timer for real-time coding or time-delay applications. It can also prescale timer 0 and timer 1 or be used as a DMA request source.

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 E-Engine 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*	*E-Engine Pin No.*	*E-Engine 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	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	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 EE pin 5	Input with pull-up
P12	DRQ0/INT5	Input with pull-up	U8 pin 3	Output for LED/EE/HWD
P13	DRQ1/INT6	Input with pull-up	J2 pin 11	Input with pull-up(USB)
P14	/MCS0	Input with pull-up	J2 pin 37	Input with pull-up(ET)
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	J2 pin 13	USB, ADC
P18	CTS1/PCS2	Input with pull-up	J2 pin 22	Input with pull-up
P19	RTS1/PCS3	Input with pull-up	J2 pin 31	Input with pull-up
P20	RTS0	Input with pull-up	J2 pin 27	Input with pull-up
P21	CTS0	Input with pull-up	J2 pin 36	Input with pull-up
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	Input with pull-up*
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	Input with pull-up*
P30	INT4	Input with pull-up	J2 pin 33	Input with pull-up
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

E-Engine 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
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 E-Engine 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**
P14	/MCS0	100M BaseT Ethernet
P4	/DT	STEP2 jumper
P11	Timer0 input	Shared with RTC, EE data input
P12	DRQ0/INT5	Output for LED or U7 serial EE clock or Hit watchdog
P13	/INT6	USB TXE

Signal	Pin	Function
P17	/PCS1	USB, ADC
P22	TxD0	Default SER0 debug
P23	RxD0	Default SER0 debug
/INT4	J2 pin 33	Ethernet interrupt, if U8 is installed

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	Select	Usage
0x0100-0x0114	/AD	AD7655: U8 ADC
0x0140		USB

3.5 Other Devices

A number of other devices are also available on the E-Engine. 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 E-Engine 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 E-Engine. 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 E-Engine 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()**.

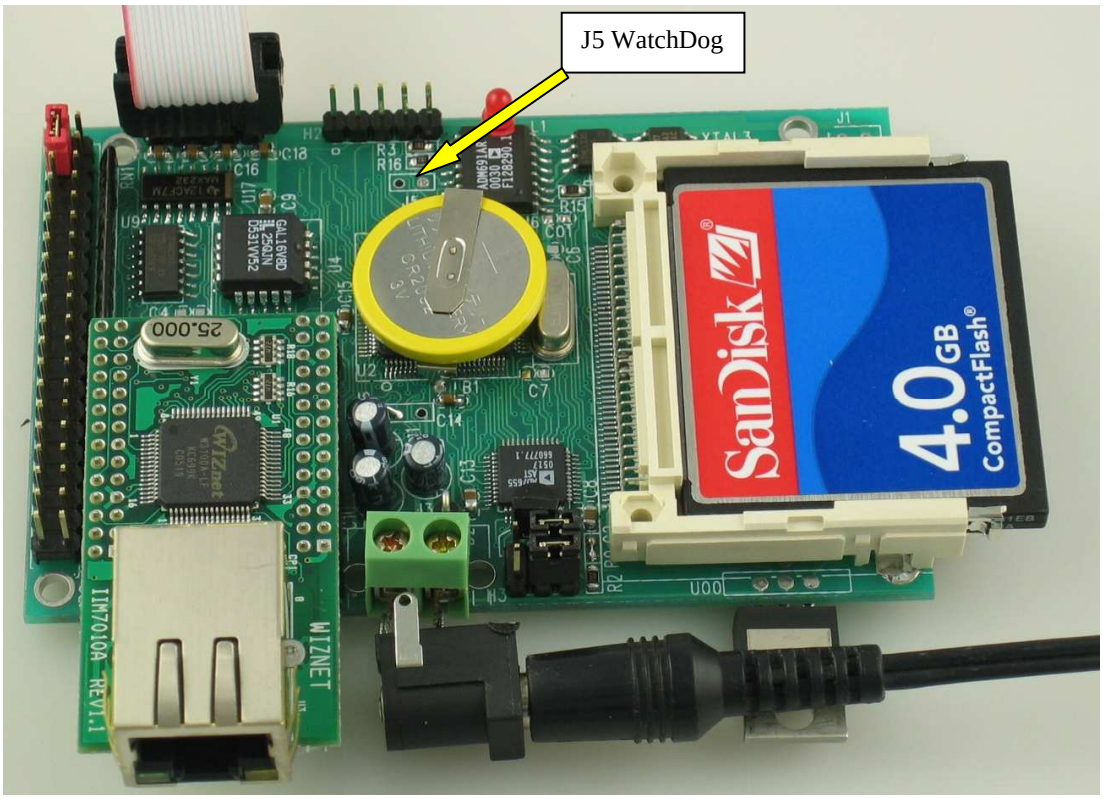
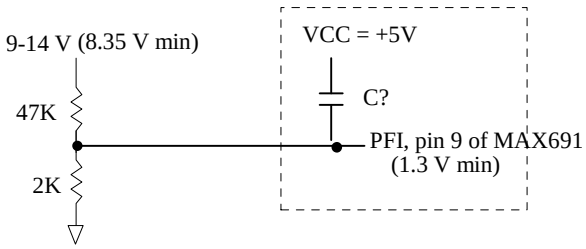


Figure 3.1 Location of watchdog timer enable jumper

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 MAX691 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 DS1337 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 E-Engine 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()**. (Note: ee in these functions stands for EEPROM, not E-Engine)

The EEPROM uses the data input signal line, P11. 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 B of this manual.

3.5.3 Real-time Clock DS1337

The DS1337 serial real-time clock is a low-power clock/calendar with two programmable time-of-day alarms and a programmable square-wave output. Address and data are transferred serially via a 2-wire, bidirectional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The data at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either 24-hour or 12-hour format with AM/PM indicator.

The RTC is accessed via software drivers *rtc1_init()* and *rtc1_rds()*, which have been specifically written for this chip. Refer to sample code in the tern\186\samples\fn directory for fn_rtc.c

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 controller using an external switching power supply.

3.5.4 AD7655, 16-bit parallel high speed ADC

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 analog input channels. 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\ee\ee_ad.c for details on reading the ADC. The sample program is also included in the pre-built sample project; \tern\186\samples\ee\ee.ide.

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

3.5.5 USB

The ***EE™*** integrates a high-performance USB stack chip to provide an easy to program USB 1.1/2.0 slave interface. The onboard hardware fully handles USB stack processing, and provides for high-speed bi-directional 8-bit parallel communication. The hardware interface includes 384 bytes of FIFO transmit buffer, and 128 bytes of FIFO for the receiving buffer, making this an ideal low-overhead solution for all embedded applications.
The EE USB exposes a slave USB interface, and connects to a PC via USB-B connector. For connection to the TERN controller, the CUSB relies on the J1 expansion header compatible with most TERN controllers. No USB specific firmware programming is required on the controller side. The USB interface is seen as a transparent parallel FIFO buffer tasked with transferring data back and forth with the remote host. The only control signals needed are “ready to transmit” and “data received” signals, readily available to your C/C++ application running on the TERN controller.

Royalty-free software drivers are provided for most Windows environments (XP, 2000, NT, 98). These field proven USB software drivers eliminates the requirement for Windows USB driver development. Two types of USB software drivers are available: VCP and D2xx. The VCP (Virtual Com Port) driver supports up to 300 K bytes per second transfer rate, and allowing the device to be accessed transparently on the PC side through traditional COM port software. The D2xx (USB direct driver and DLL) drivers can support up to 1M bytes per second. Additional utilities available from third-party sources allow the USB interface to be programmed with unique service and product ID numbers, allowing the unit to be transparently integrated into OEM applications.

3.5.6 100 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.

3.5.7 Power Supplies

The E-Engine can be powered by 3 ways:

1) Regulated external 5V DC power via J2.39=VCC and J2.40=GND, or J1.1=VCC and J1.2=GND.

2) Unregulated external 5.2V DC to 9V DC power via USB socket while two low drop 5V regulators (U0, U01) are installed. The EE can also be powered via USB port from a PC via a USB A-B cable. User can also power the EE by apply external DC via the USB socket.

3) Unregulated 9V to 12V DC power via two pin screw ternimals(T2) while a 5V linear regulator(LM7805, U00) is installed. The screw terminals are in the place of the USB socket, so the USB socket not be installed. There is a polarity protect diode installed for the screw terminal input DC power. The LM7805 is rated for 1A current, and can take as high as 35V. However, due to the linear regulation, all the input voltage has to drop to 5V, if the voltage drop with the current (200 mA) is generating a lot of heat.

The E-Engine requires good regulated 3.3V DC power for the Ethernet and R1120 CPU. Also requires regulated 5V DC power for the rest circuit. There is a 3.3V regulater (U14) on board.

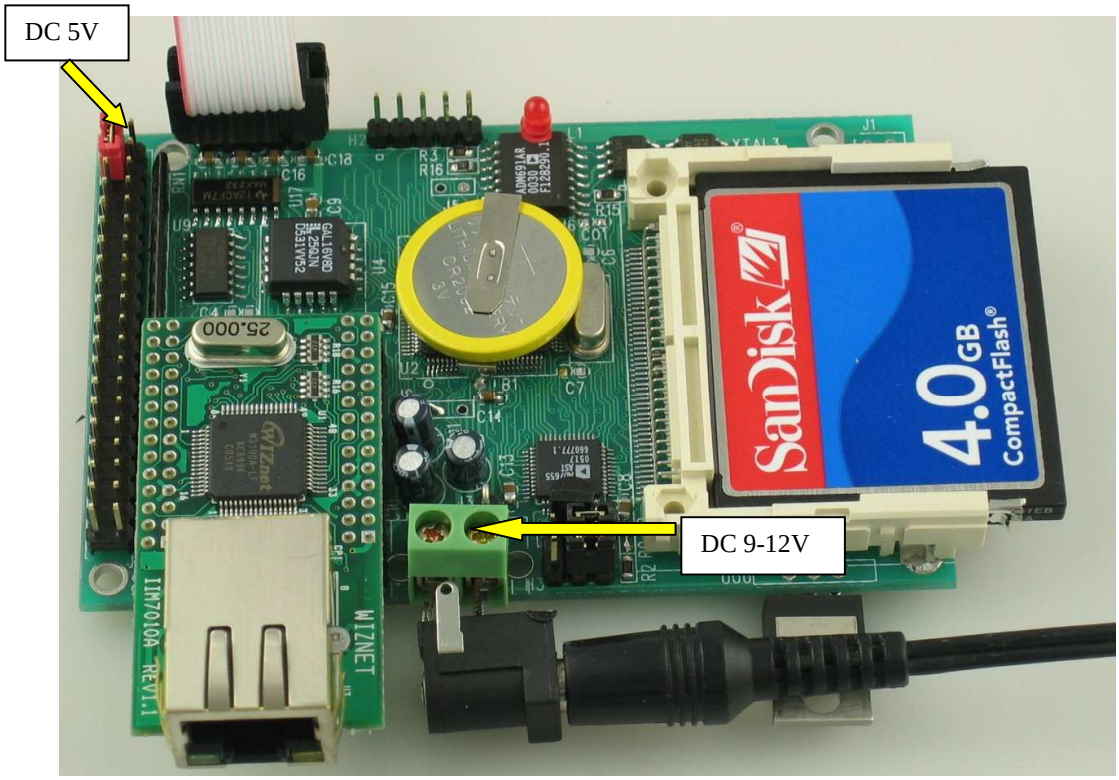


Figure 3.2 Power supplies to E-Engine

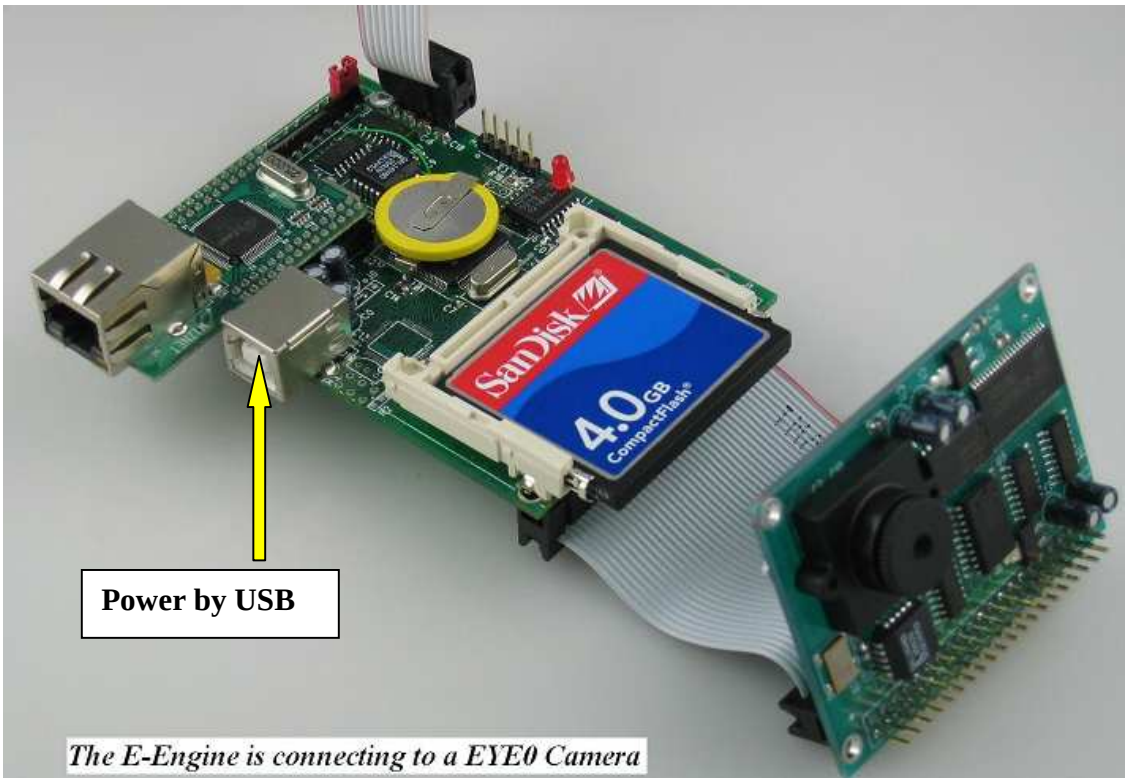


Figure 3.3 Power supply through USB to E-Engine

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	
P26	4	3	P29
GND	2	1	

J1 Signal			
VCC	1	2	GND
	3	4	CLK
	5	6	GND
	7	8	D0
VOFF	9	10	D1
PFI	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 H3 Connector for ADC7655

H3			
AB1	1	4	AB2
AA1	2	5	AA2
GND	3	6	REF

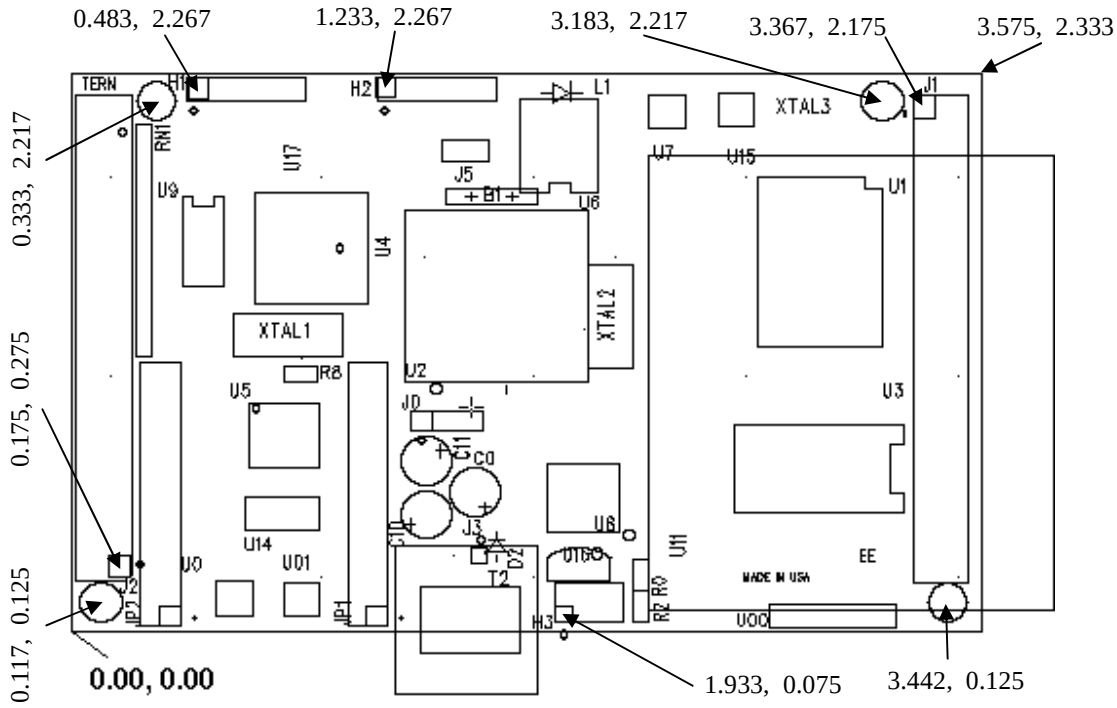
3.6.3 Jumpers

The following is a list of jumpers and connectors on E-Engine.

Name	**Size**	**Function**	**Possible Configuration**
J1	20x2	Main expansion port, A0-A7, D0-D15, /WR, /RD, P16,PFI,RST	
J2	20x2	Main expansion port, P*xx*, handshaking from SER2-9	Step 2 Jumper -> J2.38 = J2.40
J5	2x1	Watchdog timer	Enabled if Jumper is on Disabled if jumper is off

Appendix A: E-Engine Layout

All dimensions are in inches.



Appendix B: Serial EEPROM Map

Part of the on-board serial EEPROM locations are used by system software. Application programs must not use these locations.

0x00	Node Address, for networking		
0x01	Board Type	00	VE
		10	CE
		01	BB
		02	PD
		03	SW
		04	TD
		05	MC
0x02			
0x03			
0x04	SER0_receive, used by ser0.c		
0x05	SER0_transmit, used by ser0.c		
0x06	SER1_receive, used by ser1.c		
0x07	SER1_transmit, used by ser1.c		
0x10	CS high byte, used by ACTR™		
0x11	CS low byte, used by ACTR™		
0x12	IP high byte, used by ACTR™		
0x13	IP low byte, used by ACTR™		
0x18	MM page register 0		
0x19	MM page register 1		
0x1a	MM page register 2		
0x1b	MM page register 3		

Appendix C: Software Glossary

The following is a glossary of library functions for the E-Engine.

---

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

---

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 rtc1_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: fn_rtc.c

---

int rtc1_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: fn_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

.

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

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

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

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

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.

Appendix D: Debug Cable Diagram

The *E-Engine*^{TM} SER0 RS232 port (H1) is a 5x1 pin-header. Make sure pin 1 of the cable is connecting to the pin 1 of the SER0(H1). You will only be using pins 1,3,5,7,9 row of this header onto H1.

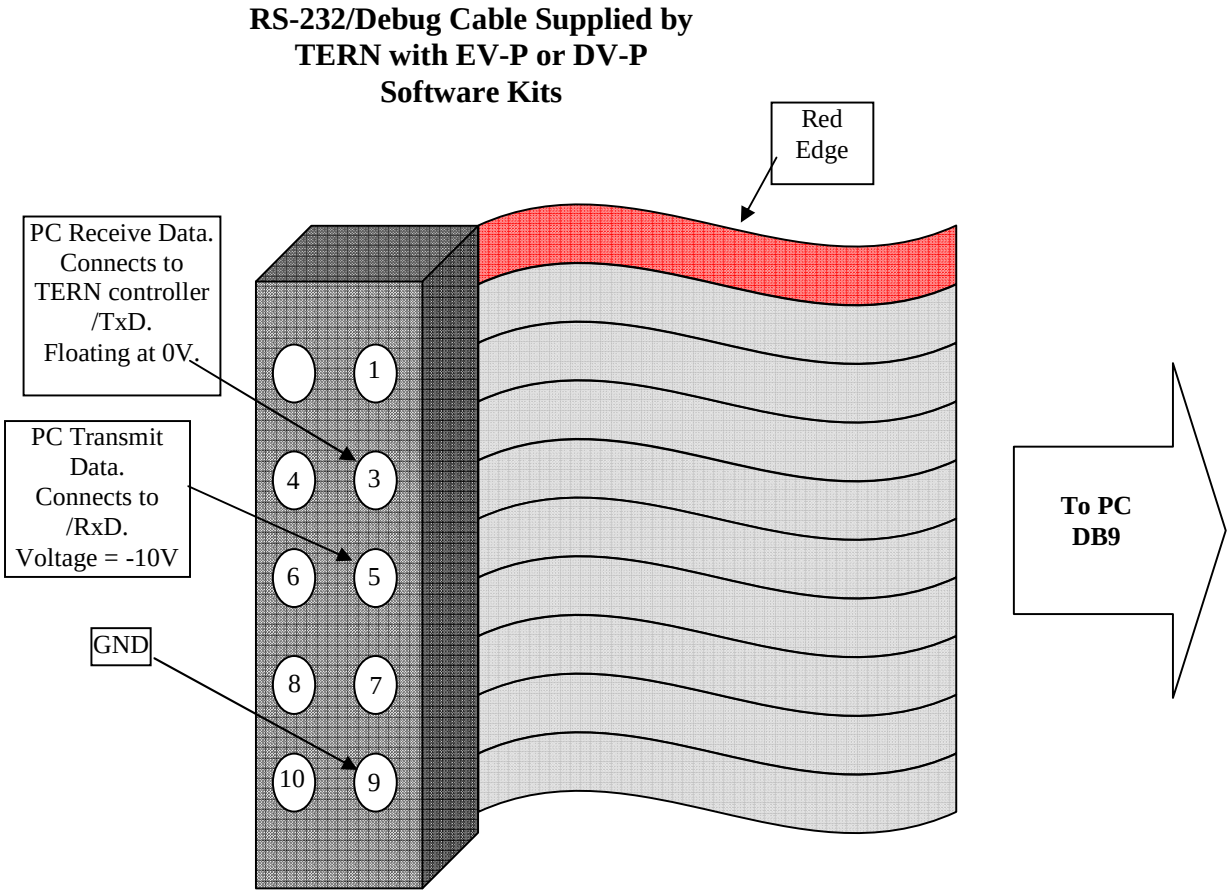
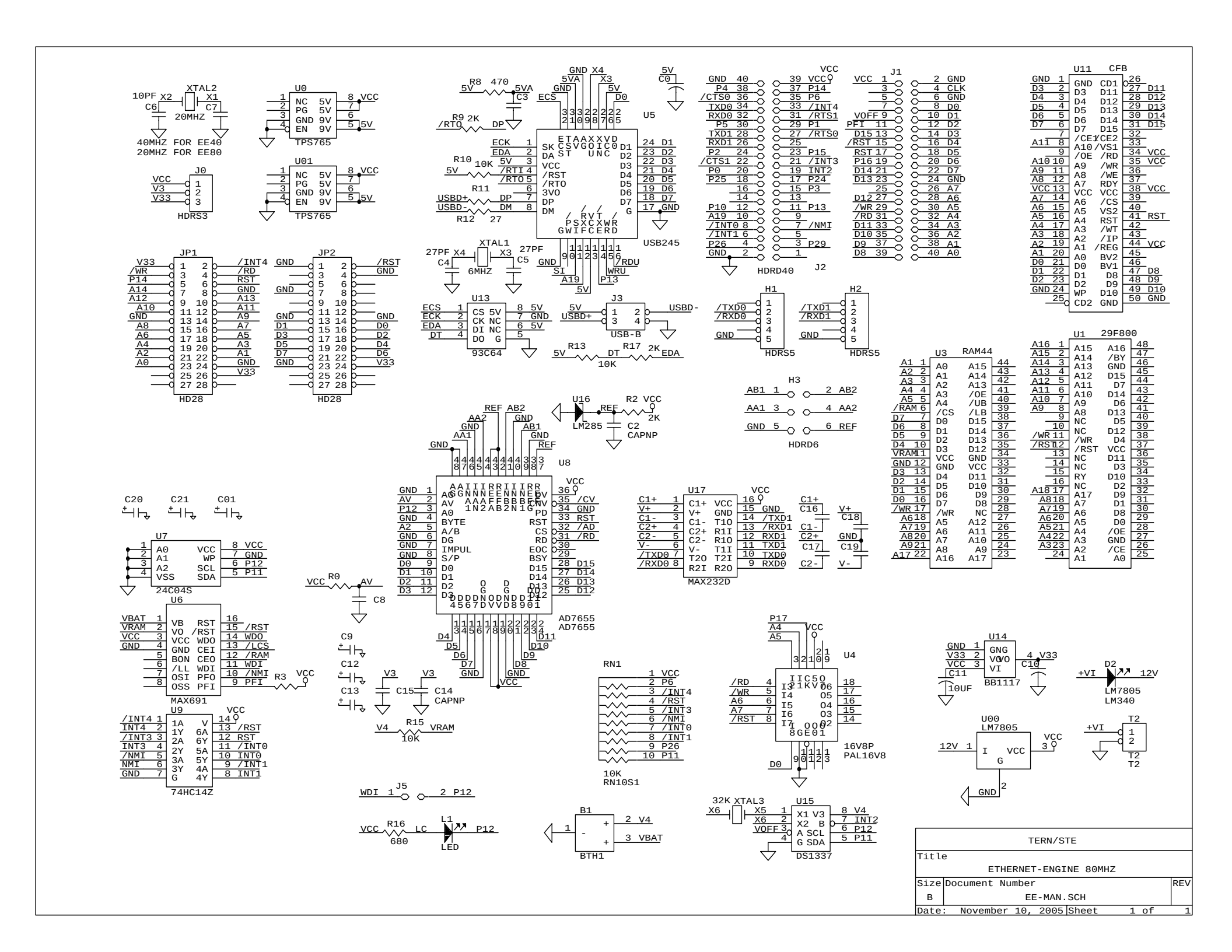


Diagram of Debug Cable (5x2, 10-pin socket shown)


XTAL2
10PF X2 X1
C6 C7
20MHZ
40MHZ FOR EE40
20MHZ FOR EE80
U0
NC 5V
PG 5V
GND 9V
EN 9V
TPS765
U01
TPS765
J0
HDRS3
JP1
JP2
HD28
HD28
R8 470 5VA
C3
R9 2K /RTO DP
R10 10K 5V
/RTI
/RTO
R11
USBD+ DP
USBD- DM
R12 27
U5
ETAAXXVD
SK CSVGOIC0
DA ST UNC
VCC
/RST
/RTO
3VO
DP
DM
/ RVT /
PSXCXWR
GWIFCERD
USB245
XTAL1
27PF X4 X3 27PF
C4 6MHZ C5
U13
CS 5V
CK NC
DI NC
DO G
93C64
J3
USB-B
R13 10K
R17 2K
U16
LM285
R2 2K
C2
CAPNP
U8
AA2
AB2
AA1
AB1
REF
U7
24C04S
C20
C21
C01
U6
MAX691
R3
R0
C8
C9
C12
C13
C14
C15
CAPNP
R15 10K
U9
74HC14Z
J5
WDI
P12
R16 680
L1
LED
B1
BTH1
RN1
RN10S1
U4
16V8P
PAL16V8
U15
DS1337
32K XTAL3
U17
MAX232D
C16
C17
C18
C19
H1
H2
HDRS5
H3
HDRD6
J1
J2
HDRD40
U3
RAM44
U1
29F800
U11
CFB
U14
BB1117
10UF
D2
LM7805
LM340
U00
LM7805
T2
TERN/STE
Title
ETHERNET-ENGINE 80MHZ
Size B
Document Number
EE-MAN.SCH
REV
Date: November 10, 2005
Sheet 1 of 1