AP-514

APPLICATION NOTE

Embedded Intel386™ EX Processor Hamilton Hallmark Handheld Terminal (H4T) Reference Design

February 1996

Order Number: 272579-001

Information in this document is provided in connection with Intel products. Intel assumes no liability whatsoever, including infringement of any patent or copyright, for sale and use of Intel products except as provided in Intel's Terms and Conditions of Sale for such products.

Intel retains the right to make changes to these specifications at any time, without notice. Microcomputer Products may have minor variations to this specification known as errata.

*Other brands and names are the property of their respective owners.

†Since publication of documents referenced in this document, registration of the Pentium, OverDrive and iCOMP trademarks has been issued to Intel Corporation.

Contact your local Intel sales office or your distributor to obtain the latest specifications before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature, may be obtained from:

Intel Corporation
P.O. Box 7641
Mt. Prospect, IL 60056-7641

or call 1-800-879-4683

EMBEDDED Intel386™ EX PROCESSOR HAMILTON HALLMARK HANDHELD TERMINAL (H4T) REFERENCE DESIGN

CONTENTS

PAGE

CONTENTS

PAGE

1.0 INTRODUCTION

The H4T, or Hamilton Hallmark HandHeld Terminal, Reference Design was developed as a working application to enable shorter design cycles by providing a proven platform as a starting point. This design has been built, debugged, and tested. The H4T was designed by Hamilton Hallmark (an Avnet company) with Intel support and highlights the features of the embedded Intel386™ EX processor and Intel Boot Block Flash Memory. The design can be used "as is" or as a building block to enhance a specific solution.

Handheld terminal applications are typically characterized by palm-sized portable devices for specific uses. These vertical applications include portable POS, dataloggers, barcode scanners, communicators, and organizers that are used in a variety of high utility environments, including retail, service, warehouses, inventory control, shipping, package delivery, manufacturing, hospitals, law enforcement, and many others. The Intel386 architecture is ideal to provide sufficient computing power and low-cost solutions for such hand-held terminals. In general, software applications for handheld terminals have become more and more standardized on a PC-like environment with operating systems such as DOS; these applications can readily be developed on any PC systems with off-the-shelf tools.

The Intel386 EX processor is a single-chip system utilizing an on-board static Intel386 CX processor core and a host of integrated peripherals, including DMA and interrupt controllers, serial and parallel ports, chip selects, timers/counters, JTAG, and power/system management features. Its 26-bit addressing provides a large 64 MB memory address space. The H4T reference design also incorporated a number of additional technologies which may be used as building blocks for a myriad of applications.

2.0 PRODUCT DESCRIPTION

The battery powered H4T reference design is DOS compatible and uses a standard PC-like BIOS. It features several products and technologies:

- Embedded Intel386 EX Processor
- Intel 4 Mb Boot Block Flash Memory
- Complex PLD
- PCMCIA Slot
- RF Wireless Communications
- Barcode Wand
- Power Management

The H4T has the following features:

- 5.0V design, can be adapted to 3.3V
- 4x20 character LCD display
- 5x6 keypad and interface
- 1 Mb SRAM, 64 Kb x 16
- 4 Mb Flash Memory, 256 Kb x 16
- 115 Kb/s RF link
- 4.5″ x 6.5″ x 2″ (WxHxD) case size
- Powered by 6 AA-size NiCad batteries, 700 mAh, estimated 20 hrs typical use

The system block diagram is shown in Figure 1. It includes the Intel386 EX processor block, program storage memory, RAM, keypad, barcode input device, alpha numeric LCD display, PCMCIA interface and wireless RF link to a host computer. The device is handheld, battery operated and includes BIOS and embedded ROM-DOS with the PCMCIA slot acting as a solid state disk drive.

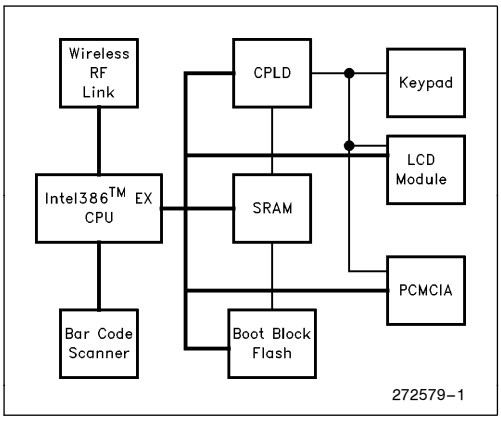

Figure 1. H4T Block Diagram

3.0 FUNCTIONAL DESCRIPTION OF THE Intel386™ EX MICROPROCESSOR

The Intel386 EX microprocessor (see Figure 2) is a fully static, 32-bit processor optimized for embedded applications. It features low power and low voltage capabilities, integration of many commonly used DOS-type peripherals, and a 32-bit programming architecture compatible with the large software base of Intel386 processors. The following sections provide an overview of the integrated peripherals.

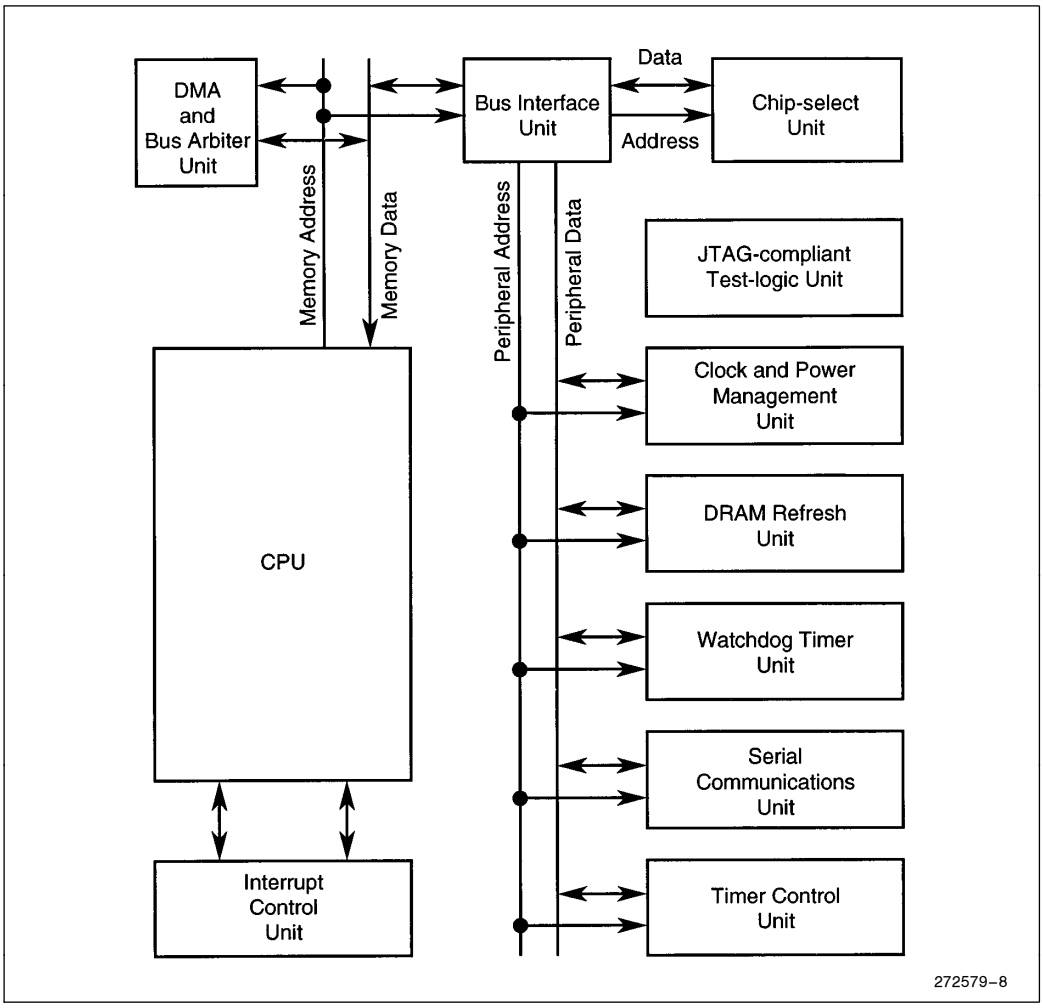

Figure 2. Intel386™ EX Microprocessor Block Diagram

3.1 Clock Generation and Power Management Unit

The clock generation circuit includes a divide-by-two counter, a programmable divider for generating a prescaled clock (PSCLK) and Reset circuitry. The CLK2 input provides the fundamental timing for the chip. It is divided by two internally to generate a 50% duty cycle Phase1 (PH1) and Phase 2 (PH2) for the core and integrated peripherals. For power management, separate clocks are routed to the core (PH1C/PH2C) and the peripheral modules (PH1P/PH2P).

Two Power Management modes are provided for flexible power-saving options. During Idle mode, the clocks to the CPU core are frozen in a known state (PH1C low and PH2C high), while the clocks to the peripherals continue to toggle. In Powerdown mode, the clocks to both core and peripherals are frozen in a known state (PH1C low and PH2C high). The Bus Interface Unit will not honor any DMA, DRAM refresh, or HOLD requests in Powerdown mode because the clocks to the entire device are frozen.

3.2 Chip Select Unit

The Chip Select Unit (CSU) decodes bus cycle address and status information and enables the appropriate chip-selects. The individual chip-selects become valid in the same bus state as the address and become inactive when either a new address is selected or the current bus cycle is complete.

The CSU is divided into eight separate chip-select regions, each of which can enable one of the eight chip-select pins. Each chip-select region can be mapped into memory or I/O space. A memory-mapped chip-select region can start on zero or any $2^{(n+1)}$ Kbyte address location (where n = 0–15, depending upon the mask register). An I/O-mapped chip-select region can start on zero or any $2^{(n+1)}$ byte address location (where n = 0–15, depending upon the mask register). The size of the region is also dependent upon the mask used.

3.3 Interrupt Control Unit

The Intel386 EX microprocessor's Interrupt Control Unit (ICU) contains two 8259A modules connected in a cascade mode. The 8259A modules make up the heart of the ICU. These modules are similar to the industry-standard 8259A architecture.

The Interrupt Control Unit directly supports up to eight external (INT7:0) and up to eight internal (IR7:0) interrupt request signals. Pending interrupt requests are posted in the Interrupt Request Register, which contains one bit for each interrupt request signal. When an interrupt request is asserted, the corresponding Interrupt Request Register bit is set. The 8259A module can be programmed to recognize either an active-high level or a positive transition on the interrupt request lines. An internal Priority Resolver decides which pending interrupt request (if more than one exists) is the highest priority, based on the programmed operating mode. The Priority Resolver controls the single interrupt request line to the CPU. The Priority Resolver's default priority scheme places IR0 as the highest priority and IR7 as the lowest. The priority can be modified through software.

Besides the eight interrupt request inputs available to the Intel386 EX microprocessor, additional interrupts can be supported by cascaded external 8259A modules. Up to four external 8259A units can be cascaded to the master through connections to the INT3:0 pins. In this configuration, the interrupt acknowledge (INTA#) signal can be decoded externally using the ADS#, D/C#, R/W#, and M/IO# signals.

3.4 Timer/Counter Unit

The Timer/Counter unit on the Intel386 EX microprocessor has the same basic functionality as the industry-standard 82C54 counter/timer. It provides three independent 16-bit counters, each capable of handling clock inputs up to 8 MHz. This maximum frequency must be considered when programming the input clocks for the counters. Six programmable timer modes allow the timers to be used as event counters, elapsed-time indicators, programmable one-shots, and in many other applications. All modes are software programmable.

3.5 Watchdog Timer Unit

The Watchdog Timer (WDT) unit consists of a 32-bit down-counter that decrements every PH1P cycle, allowing up to 4.3 billion count intervals. The WDTOUT pin is driven high for sixteen CLK2 cycles when the down-counter reaches zero (the WDT times out). The WDTOUT signal can be used to reset the chip, to request an interrupt, or to indicate to the user that a ready-hang situation has occurred. The down-counter can also be updated with a user-defined 32-bit reload value under certain conditions. Alternatively, the WDT unit can be used as a bus monitor or as a general-purpose timer.

3.6 Asynchronous Serial I/O Unit

The Intel386 EX microprocessor's asynchronous serial I/O (SIO) unit is a Universal Asynchronous Receiver/Transmitter (UART). Functionally, it is equivalent to the National Semiconductor NS16450 and INS8250. The Intel386 EX microprocessor contains two asynchronous serial channels.

The SIO unit converts serial data characters received from a peripheral device or modem to parallel data and converts parallel data characters received from the CPU to serial data. The CPU can read the status of the serial port at any time during its operation. The status information includes the type and condition of the transfer operations being performed and any errors (parity, framing, overrun, or break interrupt).

Each asynchronous serial channel includes full modem control support (CTS#, RTS#, DSR#, DTR#, RI#, and DCD#) and is completely programmable. The programmable options include character length (5, 6, 7, or 8 bits), stop bits (1, 1.5, or 2), and parity (even, odd, forced, or none). In addition, it contains a programmable baud rate generator capable of DC to 512 Kbaud.

3.7 Synchronous Serial I/O Unit

The Synchronous Serial I/O (SSIO) unit provides for simultaneous, bidirectional communications. It consists of a transmit channel, a receive channel, and a dedicated baud rate generator. The transmit and receive channels can be operated independently (with different clocks) to provide non-lockstep, full-duplex communications; either channel can originate the clocking signal (Master Mode) or receive an externally generated clocking signal (Slave Mode).

The SSIO provides numerous features for ease and flexibility of operation. With a maximum clock input of 12.5 MHz to the baud rate generator (assuming 25 MHz device operation), the SSIO can deliver a baud rate of 6.25 Mbits per second. Each channel is double buffered. The two channels share the baud rate generator and a multiply-by-two transmit and receive clock. The SSIO supports 16-bit serial communications with independently enabled transmit and receive functions and gated interrupt outputs to the interrupt controller.

3.8 Parallel I/O Unit

The Intel386 EX microprocessor has three 8-bit, general-purpose I/O ports. All port pins are bidirectional, with CMOS-level input and outputs. All pins have both a standard operating mode and a peripheral mode (a multiplexed function), and all have similar sets of control registers located in I/O address space. Ports 1 and 2 provide 8 mA of drive capability, while port 3 provides 16 mA.

3.9 DMA and Bus Arbiter Unit

The Intel386 EX microprocessor's DMA controller is a two-channel DMA; each channel operates independently of the other. Within the operation of the individual channels, several different data transfer modes are available. These modes can be combined in various configurations to provide a very versatile DMA controller. Its feature set has enhancements beyond the 8237 DMA family; however, it can be configured such that it can be used in an 8237-like mode. Each channel can transfer data between any combination of memory and I/O with any combination (8 or 16 bits) of data path widths. An internal temporary register that can disassemble or assemble data to or from either an aligned or a nonaligned destination or source optimizes bus bandwidth.

The bus arbiter, a part of the DMA controller, works much like the priority resolving circuitry of a DMA. It receives service requests from the two DMA channels, the external bus master, and the DRAM Refresh controller. The bus arbiter requests bus ownership from the core and resolves priority issues among all active requests when bus mastership is granted.

Each DMA channel consists of three major components: the Requestor, the Target, and the Byte Count. These components are identified by the contents of programmable registers that define the memory or I/O device being serviced by the DMA. The Requestor is the device that requires and requests service from the DMA controller. Only the Requestor is considered capable of initializing or terminating a DMA process. The Target is the device with which the Requestor wishes to communicate. The DMA process considers the Target a slave that is incapable of controlling the process. The Byte Count dictates the amount of data that must be transferred.

3.10 Refresh Control Unit

The Refresh Control Unit (RCU) simplifies dynamic memory controller design with its integrated address and clock counters. Integrating the RCU into the processor allows an external DRAM controller to use chip-selects, wait state logic, and status lines.

The Intel386 EX microprocessor's RCU consists of four basic functions. First, it provides a programmable-interval timer that keeps track of time. Second, it provides the bus arbitration logic to gain control of the bus to run refresh cycles. Third, it contains the logic to generate row addresses to refresh DRAM rows individually. And fourth, it contains the logic to signal the start of a refresh cycle.

Additionally, it contains a 13-bit address counter that forms the refresh address, supporting DRAMs with up to 13 rows of memory cells (13 refresh address bits). This includes all practical DRAM sizes for the Intel386 microprocessor's 64 Mbyte address space.

3.11 JTAG Boundary Scan Unit

The JTAG Boundary Scan Unit provides access to the device pins and to a number of other testable areas on the device. It is fully compliant with the IEEE 1149.1 standard and thus interfaces with five JTAG-dedicated pins: TRST#, TCK, TMS, TDI, and TDO. It contains the Test Access Port (TAP) finite-state machine, a 4-bit instruction register, a 32-bit identification register, a single-bit bypass register, and an 8-bit test mode register. The JTAG unit also contains the necessary logic to generate clock and control signals for the chains that reside outside the JTAG unit itself: the SCANOUT and Boundary Scan chains.

Since the JTAG unit has its own clock and reset signals, it can operate autonomously. Thus, while the rest of the microprocessor is in Reset or Powerdown, the JTAG unit can read or write various register chains. This feature can be used, for example, to write to the test mode register while the rest of the chip is in Reset or Powerdown. Then when the microprocessor exits Reset or Powerdown, it will enter the specified test mode.

4.0 PARTS SELECTION

Successful handheld battery-operated systems must be small, lightweight, reliable, and capable of operating for an extended period of time from a single battery charge. All parts were chosen with these guidelines in mind.

4.1 CPU

The Intel386 EX processor is a single-chip system utilizing an onboard static Intel386 CX processor or core and a host of integrated peripherals, including DMA and interrupt controllers, serial and parallel ports, chip selects, timers and counters, JTAG, power/system management features. Its 26-bit addressing provides 64 MB memory address space.

4.2 Memory

The 28F400BX, a 256Kx16 boot block Flash Memory, is used for program storage in the H4T reference design. The part is compact and capable of holding BIOS, ROM-DOS and initial application program while supporting in circuit reprogrammability.

The system RAM is a Micron Technologies MT5C64K16A, 64Kx16 SRAM. SRAM, rather than DRAM or pseudo-static RAM, was chosen for the system memory for several reasons. First, the small form factor required by a handheld device requires that the physical size of the entire memory sub-system be a minimum. Second, although DRAMs and pseudo-static RAMs would have been less costly in terms of dollars, they both presented an unacceptable trade-off in terms of power consumption. Third, a design criterion was to maximize the operating time per battery charge. Since the 386EX processor is a fully static part, completely stopping the CPU clock whenever processing was not necessary would represent the lowest possible power consumption. However, with the processor completely stopped there would be no mechanism available to accomplish a refresh of RAM contents.

The x16 configuration was selected for both the Flash Memory and SRAM in order to eliminate the requirement for generating the BS8# signal, which would have required additional logic, and to reduce the power consumption. (Power consumption for a x16 configuration is significantly lower than that for the x8. This is due to the need for additional bus cycles for both program fetches and 16 bit data read/writes when using parts with the x8 configuration.)

4.3 Data Buffers

The 28F400BX Flash Memory requires data bus buffers to eliminate data bus contention due to data hold time specifications. A 74FCT16245 data bus buffer is used, minimizing board space requirement and part count.

The PCMCIA interface has warm and hot insertion capabilities, and buffers are required to isolate the PCMCIA connector during card insertion and removal. x16 parts were used where possible to reduce part count and minimize PCB space.

4.4 User Interfaces and Additional Parts

The primary user interface is a 5x6 keypad, and the display device is a 4-row by 20-character alphanumeric LCD module. The keypad, LCD module and PCMCIA interfaces require additional logic to generate required timing and control sequences. These functions are implemented in the register-rich Lattice 1024 (CPLD) architecture.

The system requirements also call for an RF link and barcode wand; the H4T reference design implements a National RF module and a Hewlett-Packard HBSW-8200, respectively. These two components are connected to the 2 asynchronous serial ports on the CPU. The RF module requires full RS-232 drive levels, so a Maxim MAX233 is used to provide level translation. **The barcode wand has simple interface requirements.** Data signal output voltage from the barcode wand is 0 to +5V and is inverted. A comparator is used as an inverter to correct this polarity problem. Another comparator, the LP339 quad comparator, generates the reset pulse and drives the FET for power control of the RF module.

5.0 POWER SUPPLY CONSIDERATIONS

A battery pack of six AA size NiCd cells with a total capacity of 700 mAh is used as the power source. These six cells produce a nominal operating voltage of 7.2 volts and "end-of-charge" voltage of 6.0 volts. This allows the use of a step-down switching power supply for the +5 volt power with an efficiency in excess of 90% when averaged over the expected operating conditions. Even though major portions of the design are typically turned off during normal operation, worst case considerations require the power supply to deliver approximately 2 amps peak current at 5 volts. A +12V supply is required for programming a PCMCIA Flash Memory card and the 28F400BX for firmware updates. A Maxim MAX713 is the battery charging controller. This provides a switching constant current battery charger that will charge the battery pack in an hour or less without generating excessive heat. A Maxim MAX746 is used as the +5V supply and a Maxim MAX734 supplies the +12V for Flash Memory programming.

The CPU has sufficient output pins for additional levels of power control. In order to help maximize the operating time per battery charge, the CPU controls the power to the RF module, barcode scanner and the EL backlight for the LCD display, in addition to controlling the V_{PP} voltage for programming the Flash Memory. **Logic level P channel FETs were selected to implement these switches.**

The CPU includes an extremely flexible chip select unit. After reset, the upper chip select is automatically selected with 15 wait states and an internal ready# signal generated. This makes interfacing to the 28F400BX for the boot code completely "glueless". UCS# is used for the chip select, RD# is connected directly to the device. The BYTE# input of the 28F400BX connects to V_{CC} to force the part to respond in 16-bit mode. If 8-bit wide ROMs are chosen for the boot code, the BS8# signal must be generated whenever the ROM is accessed. See Figure 3 for one method of generating the BS8# signal.

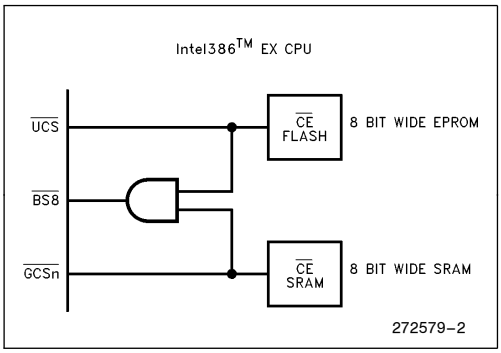

Figure 3. BS8# Generation

When connecting an SRAM to the CPU, WR# timings must be observed. The WR# signal is de-asserted at the same time that the address lines are allowed to change states for the next cycle. This will violate the address hold times for most SRAM devices. Intel386 EX processor designs should qualify WR# with the READY# signal so that when the READY# signal is asserted, the system level WR# signal is de-asserted. Figure 4 shows one method of conditioning the WR# signal with the READY# signal. On the H4T reference design this function is performed within the Lattice 1024 CPLD.

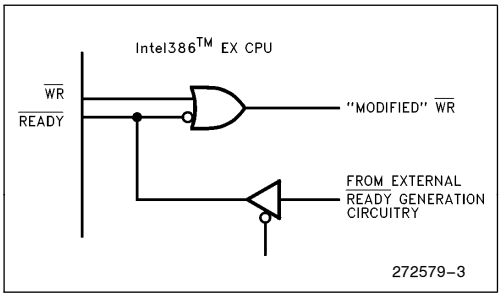

Figure 4. WR# Modification

Although the H4T reference design does not require entry into protected mode to run native code, this may be necessary in many applications. The Intel386 EX processor provides a register on board to cause a fast CPU reset required to support such functions. The PORT92 register resides at address 0092h and is described in the *Intel386™ EX Embedded Microprocessor Hardware Reference Manual.*

In many DOS applications, UCS# is used as the ROM BIOS chip select and is remapped into the 1 MB DOS space for computability. However, entry into protected mode invokes the fast CPU reset which returns the UCS# configuration registers to their initialized state. It is convenient to gate and map an additional GCS# pin (which will not be affected by the fast CPU reset) for protected mode applications. Figure 5 demonstrates a simple method to accomplish this task.

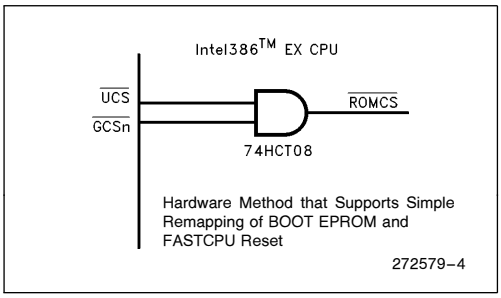

Figure 5. Generating Chip Select for Protected Mode Operation (Port 92 Reset)

6.0 CPLD FUNCTIONALITY

The CPLD has several functions. First, it provides substantial power savings by gating the CPU processor clock. Second, it contains the necessary state machines and control logic to drive the LCD display and row/scan decode circuitry for the keypad. Finally, it provides additional decoding and special functions for the PCMCIA port, protected mode operation, etc. Important functions of the ispLSI 1024 CPLD are detailed above in Figure 6.

In a typical non-static core processor design, the CPU requires a constant clock source and polls I/O during idle or slow periods. In this design, the processor is almost always idle, and simply marks time if special power saving modes are not employed. In order to reduce power consumption to an absolute minimum, the E3X processor clock static clock is completely disabled, restarting only when there is keypad activity. In other applications, this "wake-up" might be generated by a barcode scan or another activity.

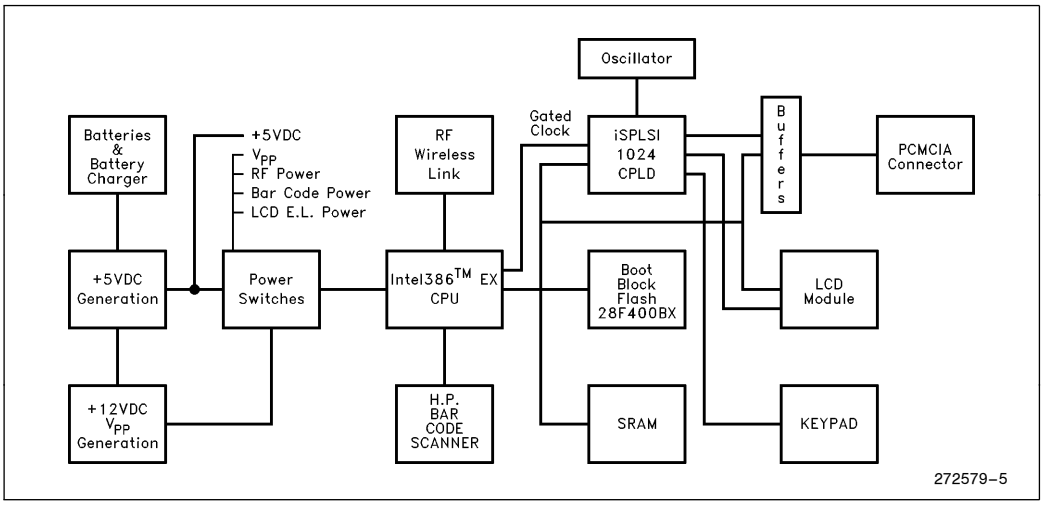

Figure 6. Detailed H4T Functional Block Diagram

As seen both in Figure 7 and Equation 1, a 50MHz clock input signal is gated with the term KR5 to enable the clock to the processor. Once the processor has completed a task, it disables itself under software control by clearing the KR5 signal. The KR5 activate and deactivate operation is described in Section 6.1.

Equation 1

```
FIFTY_QUAL = FIFTYMHZ * KR5
```

6.1 Keypad Controller Logic

The keypad used in the H4T reference design contains 30 keys configured in a 5x6 matrix. There are 6 column decode signals, KC[0:5], which act as inputs to the CPLD and 5 row output lines, KR[0:4], which are driven by the CPLD. Each of the KC lines is pulled high externally.

After initialization, the KR lines are set to a default low state and the processor clock is disabled. When a key is pressed, one or more corresponding KC lines will also be pulled low. A logic term in the CPLD called *KCQUAL* is activated when the change of state is detected. This term sets an internal register bit KR5, enabling the processor's input clock. The processor, now enabled, must scan the keypad to find out which key has been pressed. This is done by generating a delayed I/O read (i.e. longer than the keypad debounce specification) of the buffered KC lines, D[0:5]. If a KC line is still low, the I/O read decodes the keypad column. Each KR bit is then set sequentially by the processor (i.e. walking ones) while the status of the KC signals are monitored. Once the low KC line changes back to a high state, the row decode has been accomplished and the ASCII value assigned to the key is determined via a look-up table in memory. As soon as the processor has completed executing its code, it resets the KR lines (including KR5) to zero, which disables its own input clock. See Equations 2.

6.2 LCD Display Controller

The H4T reference design display is a 20 character by 4 line, 5x7 dot matrix LCD module. Built into the display is an on-board controller with display data RAM, a custom character-generating RAM, and an ASCII character-generating ROM containing 192 characters. It is through the display controller that the CPU can write, read, and set up the module. There is an 11 pin interface consisting of 8 data lines and 3 control signals built into the module to allow for communications between the controller and the CPU. The data lines are buffered by a 74FCT16245 while the three control signals are generated in the CPLD. The control signals are Enable, R/W# (Read/Not Write), and RS (Register

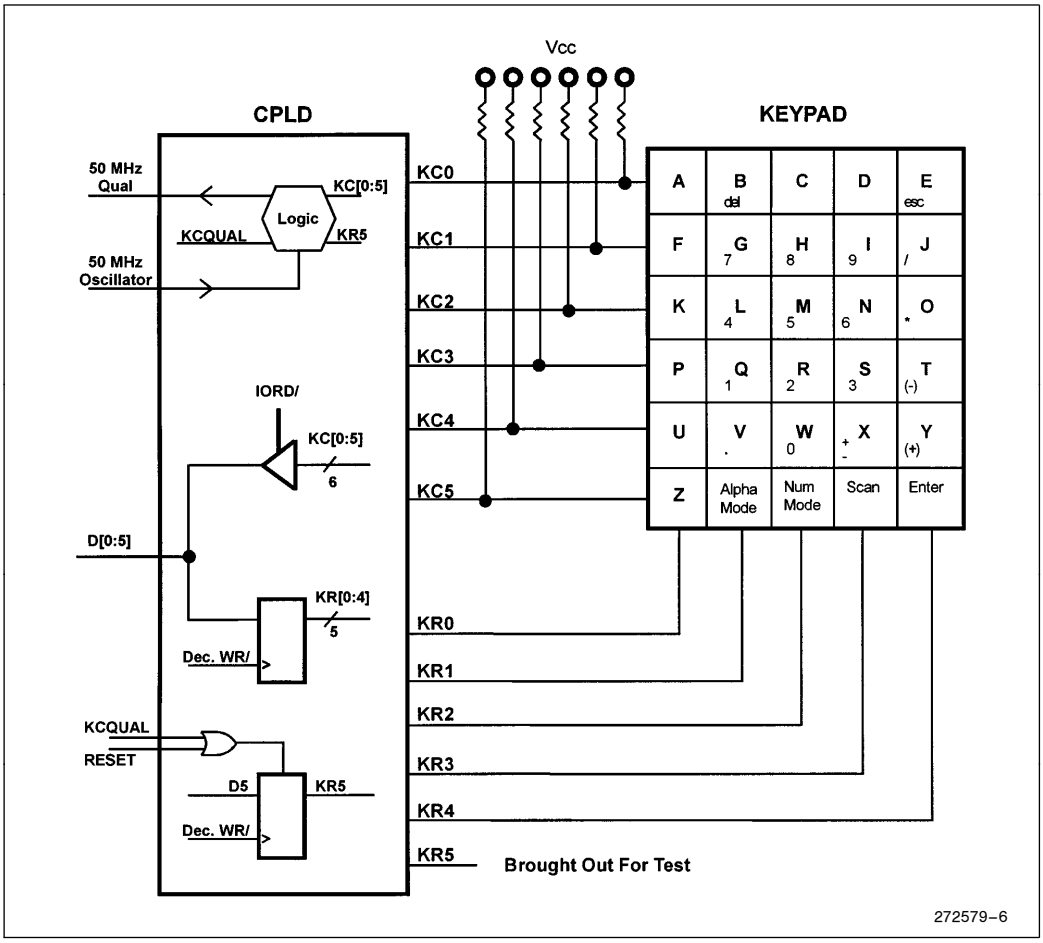

Figure 7. Keypad/CPLD Interface

Equations 2

```
KCQUAL            = /(KC0 * KC1 * KC2 * KC3 * KC4 * KC5)
DB[0..5]          = /GCS0 * KC[0..5] * /IORD
DB[0..5].TRST     = /IORD
KR[0..5]       :  = D[0..5]
KR[0..4].CLKF     = /GCS0 * /IOWR * FIFTYMHZ      :dec. write
KR5.CLKF          = /GCS0 * /IOWR * /FIFTYMHZ     :dec. write
```

Select). R/W# and RS are used by the LCD as a 2-bit code to determine which of four functions to perform as shown in Table 1.

Table 1. LCD Module Control Line Matrix

RS	R/W#	Data	Description
0	0	00000001 0000001x 1xxxxxxx	Clear Display Return Cursor Home Character Generator RAM Functions
0	1	Address Counter	Read Busy Flag / Address Counter
1	0	Write Data	Write Data To Character Generator RAM or Data RAM
1	1	Read Data	Read Data From Character Generator RAM or Data RAM

The Enable signal, which has a minimum pulse width of 450 ns, tells the LCD controller to decode the state of R/W# and RS. The CPLD contains a 6-bit resettable counter for creating this signal when the processor performs an I/O read/write operation at address 310-31Fh. In the H4T, GCS1 (Global Chip Select 1) has been initialized for this purpose, and is set for 31 wait states which allows for a normal LCD R/W cycle (1 uS min) without the need for ready generation logic. The Boolean equations used to accomplish the R/W# and RS decode are shown in Equations 3.

LCD initialization code is included as part of the boot source code file H4T.ASM on the floppy disk in the H4T Reference Design Kit. A simplified LCD Display timing diagram is shown in Figure 8.

Equations 3

```
/LCDRS = /GCS1 * /CA1
/LCDRW = /GCS1 * /MIO * /WR
LCDENBL      = Q5 * /Q4        ; decoded from 6 bit counter
             + Q5 * /Q3        ; = 560nS at fastest clock frequency
             + Q5 * Q4 * Q3 * /Q2
Q0           := ~Q0
Q1           := Q1 @ Q0
Q2           := Q2 @ (Q0 * Q1)
Q3           := Q3 @ (Q0 * Q1 * Q2)
Q4           := Q4 @ (Q0 * Q1 * Q2 * Q3)
Q5           := Q5 @ (Q0 * Q1 * Q2 * Q3 * Q4)
Q[0..5].RSTF = GCS1
Q[0..5].CLKF = FIFTYQ
```

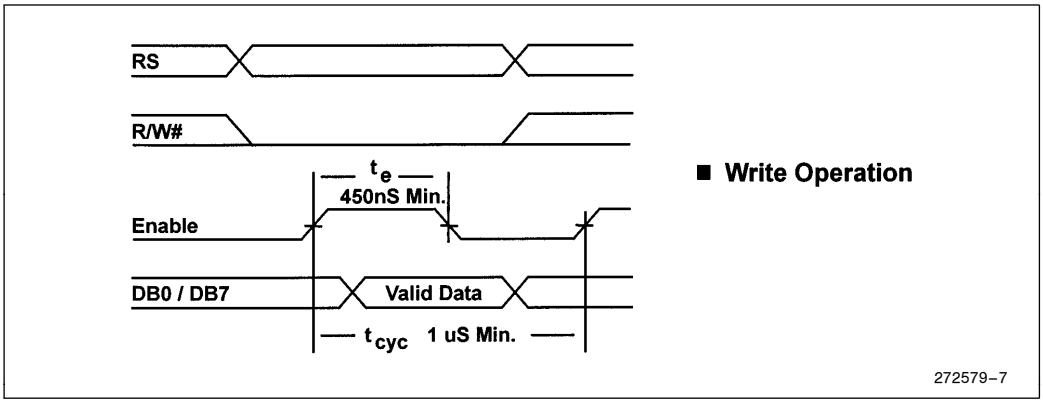

Figure 8. Simplified LCD Display Timing Diagram

6.3 PCMCIA Interface

Communications between the Intel386 EX processor and the PCMCIA port consist of a 16-bit bi-directional buffered data port, with latched addressing and six control signals generated by the CPLD. In order to access the PCMCIA port an I/O Write operation to 330h is used to address the signal ADLATCH which strobes in the upper 10 bits of the port address. Subsequent port operations are accomplished by driving GCS2 which has been initialized as an I/O chip select for address 320h to 32Fh. The CPU accesses the PCMCIA port in a page mode fashion in order to maintain backward compatibility with earlier software applications and the 64K segmentation of the original Intel 8086 microprocessor.

Warm and hot insertion capability is provided by monitoring the two port Card Detect lines, CD1 and CD2. Output buffers are tri-stated until the logical NAND of these two signals (/CD12OUT) is activated. See Equations 4.

Equation 4

```
/PRD     = /GCS2 * /RD
/PWR     = /GCS2 * /WR
/E2      = /GCS2 * /BHE
/E1      = /GCS2 * /BLE
/CD12OUT = /CD1  * /CD2
ADLATCH  = /GCS3 * ADS
```

6.4 Additional Logic

There are a few additional decoded signals created in the CPLD (see Equation 5 below):

ROMCS : Used for accessing the flash memory.

WROUT : Used externally as the write signal.

IORD : Used internally for specifying a read to any I/O port.

IOWR : Used internally for specifying a write to any I/O port.

Equation 5

```
/ROMCS = /GCS4 * /UCS
/WROUT = /WR * RDY
/IORD  = /MIO * /RD
/IOWR  = /MIO * /WROUT
```

7.0 PERIPHERALS

7.1 Barcode Wand Interface

A Hewlett Packard HBSW-8200 Barcode Wand is used as the scanning mechanism due to its simplified RS-232 interface, rugged design, and ease of programming. The wand supports communications rates up to 9600 baud, with programmable start/stop bits. Programming is accomplished by applying power and scanning preset barcodes in the users manual supplied with the wand. The output is straight 7-bit ASCII and no special application programs are needed.

7.2 RF Module

The H4T REFERENCE DESIGN incorporates an RF module from National Semiconductor. The AirShare™ module provided a full RS-232C interface and accepts ASCII characters for transmission and reception. Transmission range is limited from 25 feet to 30 feet.

8.0 SOFTWARE CONSIDERATIONS

When the CPU is activated from powerdown, the only chip select line enabled is UCS. As is typical of the Intel architecture processor family, the processor will look for its first instruction at the top of memory, which in this case is 3FFFFF0h (64 MB address space).

The standard first instruction is typically a far jump to a set of initialization routines. For example, in the case of a typical PC BIOS implementation the jump may be executed to the bottom of the current 64K segment where registers and chipsets are initialized. The BIOS ROM is then remapped to the top of the first 1 MB of the processor address range.

The process is similar, with only a few minor modifications to initialize the CPU. The far jump is required, but a few on-board housekeeping tasks must be taken care of before the UCS pin, I/O port, and pin configurations can be initialized and remapped. Compiler directives and macros enable the jump as illustrated below where **FarJmp** is a macro which is called again when re-mapping the UCS pin.

Code Fragment 1

```
;=== This is the hardware entry point after a reset ===

org 0fff0h                      ;top of the 64k flash eprom block

starting_point:
         FarJmp 0f000h, start   ;jump to bottom of block w/offset start

FarJmp macro seg, off           ;the org of start is zero, so
       db   0eah                ;jump is to 0f000:0000
       dw   off
       dw   seg
endm

.code                           ;assembler directive defines beginning of
       org  0h                  ;code segment
start:
.................
.................
.................

;**** one more housekeeping chore

      in al, 92h                ;only care about bits 0 and 1
      and al, 0feh              ;bit 0 controls internal core only reset
      or al, 02h                ;bit 1 is A20Gate
      out 92h, al               ;set A20Gate = 1 which drives A20 low
                                ;fast CPU reset (used to access protected mode)
                                ;is disabled
```

8.1 I/O Space Configuration

The Intel architecture processor family allows the designer to set aside a specific I/O space which may be up to 64K in length. After reset, all on-board peripherals and registers are initialized to known states, with the serial ports, interrupt controllers, DMAs, and timers all mappped to their standard DOS compatible addresses.

Any of these peripherals, chip selects, and other board functions may be remapped to new I/O address spaces by writing to specific configuration registers as documented below. These configuration registers are internal to the CPU and are mapped into expanded I/O space near the top of the 64K range from F000h to F800h. However, access to the configuration registers is only made available after entering the **expanded I/O space mode** which requires the specific write sequence shown in Code Fragment 2.

This sequence sets the ESE bit, which is required before any other mapping procedures may be invoked. Once the ESE bit has been enabled, DOS compatible peripherals may be remapped out of DOS space by setting the appropriate REMAPCFG register bits, and then altering the associated I/O configuration registers to reflect the starting address and range. Setting of I/O port parameters (serial port baud rates etc.), enabling specific port pins, and mapping of chip selects may also then be accomplished.

The previous code fragment demonstrated how to open the expanded I/O space. Once that is accomplished, the internal peripheral configuration registers may be freely accessed. An example to address internal registers while providing another level of power reduction is shown below. The code used to remap the boot EPROM into a DOS compatible area is shown in Code Fragment 3.

Since this is a bootblock Flash Memory (28F400BX), there are numerous 64K segments with this code residing in the EPROM itself in the last 64K block @ location 70000h to 7FFFFh. Code will continue to execute in the same 64K segment while remapping ROM chip upper select pin (UCS) to the first megabyte of address space for DOS compatibility.

When the CPU is instructed to set the starting (or lower range) address for the UCS to 512K (mov ax, 0008h) the code segment register (CS) is changed to 8000, which corresponds to the first 64K block in the 28F400BX, not the last. Since the next few instructions (see Code Fragment 4) have already been loaded into the pre-fetch cue, an immediate far jump can be initiated to the last 64K, restoring the CS register in the ROM.

Code Fragment 2

```
;open expanded i/o space
;this special procedure is required to set the ESE (Expanded I/O Space
;Enable) bit in the REMAPCFG register

 mov ax,0080h              ' disable interrupts
 xchg al,ah
 out 23h,al
 xchg al,ah
 out 22h,al
 out 22h,ax
```

Code Fragment 3

```
;shut off watchdog timer clock reduce power/disable dog

        mov dx, 0f4cah          ;Register: WDTSTATUS
        mov al, 01h             ;Stop all watchdog clocks; set CLKDIS bit
        out dx, al

;remap ucs (28F400bx)
;start address is 80000h region => range 512k bytes (256k words)
;2 wait states

        mov dx,0f438h           ;Register: UCSADL
        mov ax,0302h            ;define UCS as memory CS, bus size = 16
        out dx,ax                       ;disable external ready, 2 wait states
                                ;set lower 5 bits of region address

        mov dx,0f43ah           ;Register: UCSADH
        mov ax,0008h            ;set start address to 512k
        out dx,ax

        mov dx,0f43eh           ;Register: UCSMSKH
        mov ax,03ffh            ;don't mask out any addresses as invalid YET
        out dx,ax

        mov dx,0f43ch           ;Register UCSMSKL
        mov ax,0f801h           ;complete mask.
        out dx,ax               ;Enable chip select
```

Code Fragment 4

```
;currently executing code @ (3FF)F000:offset
;CS was changed to 8000:offset by mov ax, 0008 @ 0f43ah
;Calling the macro FarJmp and passing the parameters F000, offset (The label
;Remap) will reset CS to F000 before pre-fetch cue is empty

        FarJmp 0F000h, Remap
remap:
        mov dx,0f43eh
        mov ax,0007h              ;NOW set range to 512K
        out dx,ax
```

8.2 Global Chip Select Register Configuration

The CPU contains many internal registers which are used to configure devices such as the on-board I/O ports and global chip select registers. The internal registers are mapped into expanded I/O space from f000h to f800h. Once expanded I/O space has been opened, port and global chip select initialization may begin.

The typical process to initialize a global chip select pin requires four I/O writes to 16-bit registers which determine parameters such as bus widths, number of wait states, starting address and address range. The decision to place the chip select in either memory or I/O space is also selected.

Table 2. Memory Configuration Register Bit Assignments

Reg. Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADH	xx	xx	xx	xx	xx	xx	A25	24	23	22	21	A20	A19	A18	A17	A16
ADL	A15	14	13	12	11	CASM	BS16	MEM	RDY	xx	xx	WS4	WS3	WS2	WS1	WS0
MSKH	xx	xx	xx	xx	xx	xx	A25	24	23	22	21	A20	A19	A18	A17	A16
MSKL	A15	14	13	12	11	CMSM	xx	xx	xx	xx	xx	xx	xx	xx	xx	CSEN

Table 3. I/O Configuration Register Bit Assignments

Reg. Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADH	xx	xx	xx	xx	xx	xx	A15	14	13	12	11	10	09	08	07	06
ADL	A05	04	03	02	01	CASM	BS16	MEM	RDY	xx	xx	WS4	WS3	WS2	WS1	WS0
MSKH	xx	xx	xx	xx	xx	xxx	A15	14	13	12	11	10	09	08	07	06
MSKL	A05	04	03	02	01	CMSM	xxx	xxx	xx	xx	xx	xx	xx	xx	xx	CSEN

8.3 Chip Select Initialization

An example of how to configure GCS6 to be the lower memory chip select so on board SRAM can be used to facilitate procedures, data tables, and the stack is shown below. Most of the code fragments will pass parameters to a macro and utilize register names as they appear in the *Embedded Intel386™ Microprocessor Hardware Reference Manual*. An equate statement for each register name may be found in the file EQU.INC which is included on the floppy disk in the H4T Reference Design Kit.

The macro, SetE3XRegWord receives the parameters and performs the functions shown in Code Fragment 5.

Code Fragment 5

```
SetE3XRegWord MACRO reg,val
mov dx, reg      ;move I/O address
                 ;into DX register
mov ax, val
out dx, ax
ENDM
```

The equate statement for GCS6 indicates the extended I/O address for each register.

Code Fragment 6

```
CS6ADL    EQU  0f430h
CS6ADH    EQU  0f432h
CS6MSKL   EQU  0f434h
CS6MSKH   EQU  0f436h
```

The entire initialization of the GCS6 registers is shown in Code Fragment 7.

Code Fragment 7

```
;starting addr = 0k
;region = 256K
;0000:0000 => 3FFF:FFFEh
;2 wait states

SetE3XRegWord     CS6ADL, 0302h
SetE3XRegWord     CS6ADH, 0000h
SetE3XRegWord     CS6MSKH, 0003h
SetE3XRegWord     CS6MSKL, 0f801h
```

A similar process may be utilized to initiate other global chip select pins. Several more examples are shown in the assembly file H4T.ASM, included on the floppy disk in the H4T Reference Design Kit.

8.4 Port Pin Activation

Many pins on the CPU have multiple functions. For example, a port pin may be configured for input or output, to act as an open drain driver, or have a specific peripheral related function. Because of this, it is necessary to configure a number on-board port registers to define both the function of the pin and to enable and disable the internal drivers. This is done via the four Port Configuration Registers: PINCFG, P1CFG, P2CFG, and P3CFG.

The series of I/O writes shown in Code Fragment 8 will configure the pins to support the SIO functions, and enable the global chip select functions used in the H4T reference design.

8.5 Leaving Expanded I/O Space

Leaving the expanded I/O space will safeguard the data which has been written to these and other registers. Code to close the expanded I/O space is shown in Code Fragment 9.

Code Fragment 9

```
;close expanded i/o space

        mov al,00h
        out 23h,al
```

8.6 Stack Initialization

The stack segment is initialized as shown in Code Fragment 10.

Code Fragment 8

```
;**********
;these are byte wide registers and thus, use the macro SetE3XRegByte to
;accomplish the initialization macros are documented in the file MACRO.inc
;
;enable select serial port 0 (COM1) pins through register P1CFG
;RTS0 DTR0 DSR0
;not using DCD0, RI00, LOCK#, or HOLD/HLDA
;**********
SetE3XRegByte P1CFG, 00001110b    ; Was PINCFG0 addr. 0F820h

;enable CTS0, TXD0, RXD0, GCS3-0
SetE3XRegByte P2CFG, 11101111b    ; Was PINCFG1 addr. 0F822h

;enable CTS1, TXD1, DTR1, RTS1 ... RXD1 and DSR1 are enabled by default
SetE3XRegByte PINCFG, 00001111b   ; Was PINCFG3 addr 0F826h

;enable internal reference clock for baud rate generator instead of
;external 1.832mhz source
SetE3XRegByte SI0CFG, 00000011b
```

Code Fragment 10

```
;**********
;init stack to live at 3000:0000 where top of stack
;is segment 3000 or 3000 x 10h ==> 30000h
;this is at the top of SRAM (since the stack grows down) and will be
accessed by the GCS6/LCS
;pin initialized earlier
;**********
        mov ax, 2fffh
        mov ss, ax
```

8.7 Serial Port Configuration

The Embedded Intel386 EX processor has two integrated asynchronous serial ports. Each UART is equivalent to the National Semiconductor NS16450 and INS8250A. Upon reset, the CPU comes up in a PC-AT ISA compatible mode, where the serial port and other ISA-compatible peripherals can be found at the normal DOS addresses for COM1 and COM2. (3F8h to 3FFh and 2F8h to 2FFh, respectively). The standard register set for each port is also located at the proper DOS address and is shown below in Table 4.

Table 4. Synchronous I/O Port Registers

SIO 0 Register Addresses (COM1 / DOS Addr.)	SIO 1 Register Addresses (COM2 / DOS Addr.)	DLAB Bit Value	Register Accessed
03F8h	02F8h	0	Receive Buffer (RBRx)
03F8h	02F8h	0	Transmit Buffer (THRx)
03F9h	02F9h	0	Interrupt Enable (IERx)
03F8h	02F8h	1	Divisor Latch / Low Byte
03F9h	02F9h	1	Divisor Latch / High Byte
03FAh	02FAh	X	Interrupt ID (IIRx)
03FBh	02FBh	X	Line Control (LCRx)
03FCh	02FCh	X	Modem Control (MCRx)
03FDh	02FDh	X	Line Status (LSRx)
03FEh	02FEh	X	Modem Status (MSRx)
03FFh	02FFh	X	Scratch Pad (SCRx)

Code to initialize a 16450 is shown in Code Fragment 11. The register names and addresses follow the standard format used in programming a PC I/O card. The example below shows initialization of Port 0 as 9600 baud, no parity, 8 bits data, 1 stop bit.

The same procedure may be used to calculate baud rates and initial control bits for COM2, by using the *ASYNCHRONOUS SERIAL CHANNEL 1 - SLOT 0 (DOS) ADDRESSES* listed in the file EQU.INC, included on the floppy disk in the H4T Reference Design Kit.

NOTE:
If serial ports are relocated to non-DOS space, the *ASYNCHRONOUS SERIAL CHANNEL - SLOT 15* (Non-DOS) registers must be written to (located at F4F8h to F4FFh and F8F8h to F8FFh, respectively). The CPU must be in the expanded I/O mode to access the non-DOS registers.

Table 5. Hamilton Hallmark H4T Memory and I/O Address Map

	Memory or I/O	Starting Address	Ending Address	Wait States	Chip Select
Keypad	I/O	060h	064h	0	GCS0
LCD Module	I/O	310h	31Fh	31	GCS1
PCMCIA READ/WR#	I/O	320h	32Fh	4	GCS2
ADLATCH PCMCIA	I/O	330h	—	0	GCS3
COM1 HP Barcode	I/O	2F8h	2FFh	Polled	Internal to E3X
COM2 RF Module	I/O	3F8h	3FFh	Polled	Internal to E3X
SRAM Data	Memory	00	1FFFFh	0	0
SRAM Stack	Memory	20000h	3FFFFh	0	0
EPROM	Memory	80000h	FFFFFh	2	2

Code Fragment 11

```
;**********
; Initialize Asynchronous Serial Port 0
; internal serial ports are ins8250/NS16450 compatible
;disable interrupts: reset Interrupt Enable Register bits = 00h
;will be using a polled mode

    SetE3XRegByte    IER0D0S,    00h          ;IER0D0S EQU 03F9H

;set communications rate to 9600 baud assuming 20MHZ input clock rate
;Note! The baud rate input clock must equal 16X the final baud rate. The
;input to the baud rate generator on the E3X is equal to
;the processor input clock freqency divided by four,
;or in this example, 5MHZ.
;
;need to produce internal clock rate = 9600 x 16 ==> 153.6khz
;so divisor = 5Mhz / (x) = 153.6khz ==> 32.55
;(will use 33 and accept 1.5% error)
;
;must set DLAB bit high in the Line Control Register in order to access the
;Programmable Baud Rate Generator Divisor latches

    SetE3XRegByte    LCR0D0S,    10000000b    ;addr = 03fbh

;write Programmable Baud Rate Generator.
;set divisor latch register (HI) then divisor latch register (L0)

    SetE3XRegByte    DLH0D0S     00h          ;addr = 03F9h
    SetE3XRegByte    DLL0D0S     33d          ;addr = 03F8h

;Line Control Register
;reset DLAB bit and specify format of communications exchange
;8 bit data   no parity   1 stop bit

    SetE3XRegByte    LCR0D0S,    00000011b    ;addr = 03fbh

;set Modem Control Register bits / turn on DTR and RTS

    SetE3XRegByte    MCR0D0S,    00h          ;addr = 03fch
```

9.0 SUMMARY

The H4T Reference Design has been discussed regarding both hardware and software implementation details. Since the H4T is a DOS compatible handheld terminal, users can easily develop application software on any PC with off-the-shelf tools. The Intel386 EX processor is a highly integrated embedded CPU with the key peripheral components onboard to build a cost effective, yet compact system for portable applications. Other key technologies, such as the Intel Boot Block Flash Memory, Complex PLD use, PCMCIA slot, and power management have also been discussed. Handheld terminal architectures and markets are well served by embedded Intel Architecture processors that can provide for low cost development, fast time to market, proven support infrastructure, and long lifecycles.

The H4T has been built, debugged, and tested as a working design. This reference design was developed by Hamilton Hallmark (an Avnet Company) with Intel support.

10.0 REFERENCES

Intel386™ EX Embedded Microprocessor Reference Manual Order No. 272485-000

Seiko L2014 LCD Data Sheet

National Semiconductor NS16450 Data Sheet

Microsoft MASM 6.11 User's Manual

11.0 FILES AVAILABLE ON THE FLOPPY DISK IN THE H4T REFERENCE DESIGN KIT

H4T.ASM	Boot Source Code
EQU.INC	Equate Include File, used w/H4T.ASM
PROC.INC	Procedure Include File, used w/H4T.ASM
MACRO.INC	Macro Include File, used w/H4T.ASM
H4T.ABL	Abel Source Code for Lattice 1024 CPLD
BOM.DOC	H4T Bill Of Materials
H4T.ZIP	Orcad Schematic Files, Libraries, and Gerber Plot Files