A Software Driver for the **HPC Universal Peripheral Interface Port**

ABSTRACT

This application note covers the use of the National Semiconductor HPC46083 High-Performance microController as an intelligent Peripheral Interface and Interrupt controller for another "Host" CPU, using its 8-bit or 16-bit parallel UPI (Universal Peripheral Interface) Port. Included in the discussion is the source text of an HPC driver program, which can be tailored as an "executive" for a wide variety of HPC tasks. A simple application is built from this software, which interfaces a National NS32CG16 CPU to a typical front panel (LED indicators, LCD alphanumeric display, pushbuttons and beeper).

1.0 INTRODUCTION

The National Semiconductor HPC family of microcontrollers includes as a feature the ability to be slaved to another "Host" processor over that processor's memory bus. This feature, called the Universal Peripheral Interface (or UPI) Port, allows:

1. Transfer of either 8-bit or 16-bit data in a single bus transaction.

National Semiconductor Application Note 550 **Brian Marley** April 1992



2. Polling to determine the status of the port from either side (Ready for Write/Ready for Read), and

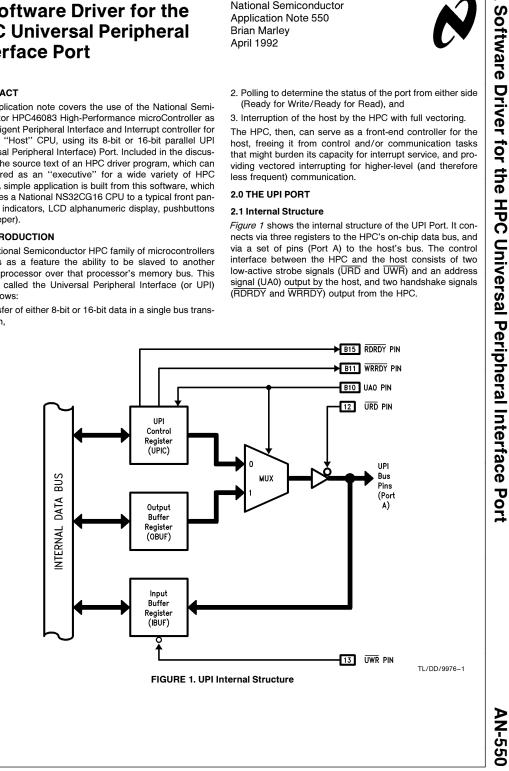
3. Interruption of the host by the HPC with full vectoring.

The HPC, then, can serve as a front-end controller for the host, freeing it from control and/or communication tasks that might burden its capacity for interrupt service, and providing vectored interrupting for higher-level (and therefore less frequent) communication.

2.0 THE UPI PORT

2.1 Internal Structure

Figure 1 shows the internal structure of the UPI Port. It connects via three registers to the HPC's on-chip data bus, and via a set of pins (Port A) to the host's bus. The control interface between the HPC and the host consists of two low-active strobe signals (URD and UWR) and an address signal (UA0) output by the host, and two handshake signals (RDRDY and WRRDY) output from the HPC.



RRD-B30M105/Printed in U. S. A

The UPI Port may be configured either as a 16-bit bus (using all of Port A: pins A0–A15) or as an 8-bit bus (pins A0–A7), allowing pins A8–A15 to be used as general-purpose bit-programmable I/O pins. This selection is made by HPC firmware.

2.2 Basic Operations

Three types of operation may be performed over the UPI Port:

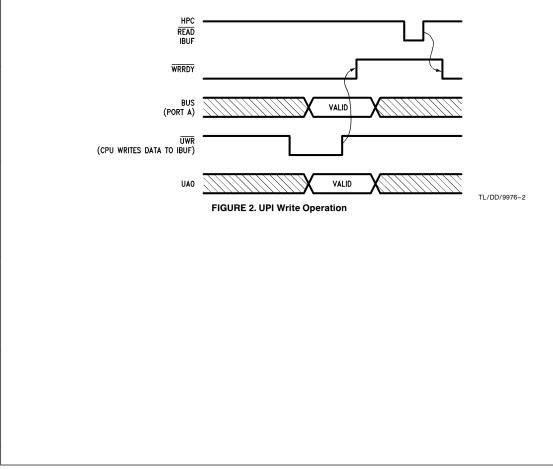
- 1. Transfer of a byte or word of data from the host to the HPC's IBUF register. This is called a "UPI Write" operation.
- Transfer of a byte or word of data from the HPC's OBUF register to the host. This is called a "UPI Read" operation.
- Polling by the host to determine whether the HPC is ready for the next UPI Write or UPI Read operation. This involves the host reading the UPIC (UPI Control) register, which contains the states of the WRRDY and RDRDY pins as two of its bits.

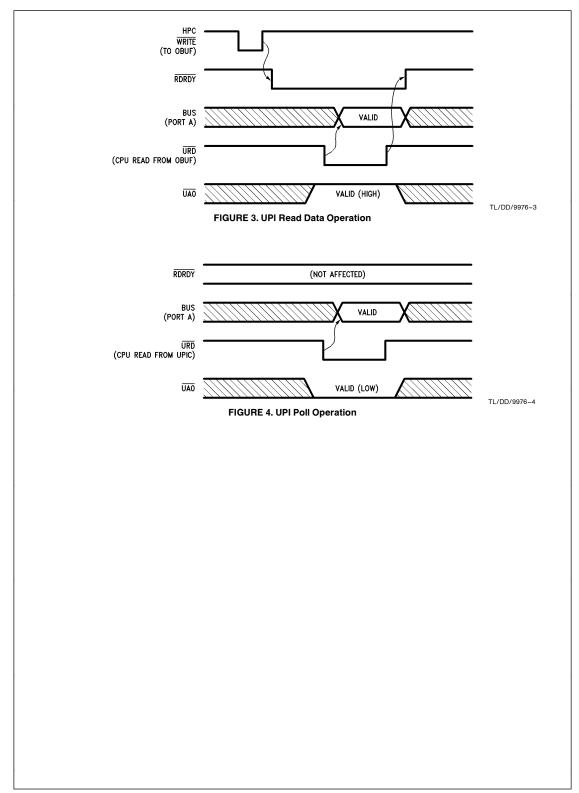
As shown in *Figure 2*, whenever the host writes to the HPC (by pulsing the $\overline{\rm UWR}$ signal low) data is latched into the HPC's IBUF register. At this time also, the value on the UA0 pin is latched into the UPIC (UPI Control) register, allowing

HPC firmware to route the data just written. (For example, this bit can be used by the HPC firmware to distinguish between commands and data written to it.) The rising edge of UWR is detected by an edge-trigger circuit on-chip, which may be used to trigger an interrupt or for polling, to alert the HPC firmware to the presence of new data. The WRRDY handshake signal, normally low, goes high until the HPC firmware has sampled the data written to it (by reading internally from the IBUF register).

Figure 3 shows the sequence of events in reading data from the HPC. The transfer starts when the HPC writes a value to the internal OBUF register. The $\overline{\text{RDRDY}}$ handshake signal, normally high, goes low to indicate that data is present for the host. (This pin can be used to interrupt the host as well.) By pulsing the $\overline{\text{URD}}$ pin low while holding the UA0 pin to a "1", the host reads the contents of the OBUF register, and the $\overline{\text{RDRDY}}$ pin goes back high.

The polling operation (*Figure 4*) allows the host to monitor the RDRDY and WRRDY conditions as data bits, by pulsing the URD pin low with a "0" held on the UA0 pin. This effectively reads from the UPIC register; the WRRDY condition appears on bit 0 (the least-significant bit), and the RDRDY condition appears on bit 1 (the next most significant bit). Polling in this manner does not affect the state of the RDRDY bit.





2.3 Typical Hardware Configurations

Typical connections between the host and the HPC are shown in *Figures 5* through *7*.

2.3.1 Polled Synchronization

In the simplest case (*Figure 5*), the WRRDY and RDRDY signals are not used, and the host synchronizes itself with the HPC strictly by polling the UPIC register for the Read Ready and Write Ready conditions. The only additional logic always required is a pair of OR gates to activate URD and UWR only when the HPC is selected by the host's address decoder. Depending on the host, it may also be necessary to add WAIT states, as is often required in peripheral interfaces to match the bus timing characteristics of the two ends.

Sophisticated synchronization schemes are not available using this simple an interface, but it does save the HPC $\overline{\text{RDRDY}}$ and $\overline{\text{WRRDY}}$ pins for any other general-purpose I/O functions.

2.3.2 Interrupt-Driven Synchronization

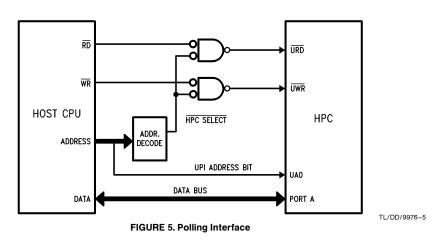
Assuming that the host has interrupt control capability, the circuit above can be enhanced to implement an interruptdriven synchronization scheme, as shown in *Figure 6*. A falling edge on either RDRDY or WRRDY will trigger an interrupt to the host, informing it when the HPC becomes ready for either direction of data transfer. No additional logic is required (except for possible buffering or inversion), but only dedication of the WRRDY and/or RDRDY pins for the interrupt function. It is not necessary for both RDRDY and WRRDY conditions to trigger interrupts; one can be polled and the other interrupt-driven, as dictated by the require ments of the system and the structure of the host and HPC software. Also, depending on the host, it is often possible for the HPC itself to provide interrupt vectoring, thus eliminating the need for an external interrupt controller entirely. The approach taken in the driver program, described below, implements the HPC as the interrupt controller, with interrupts asserted only by the RDRDY pin.

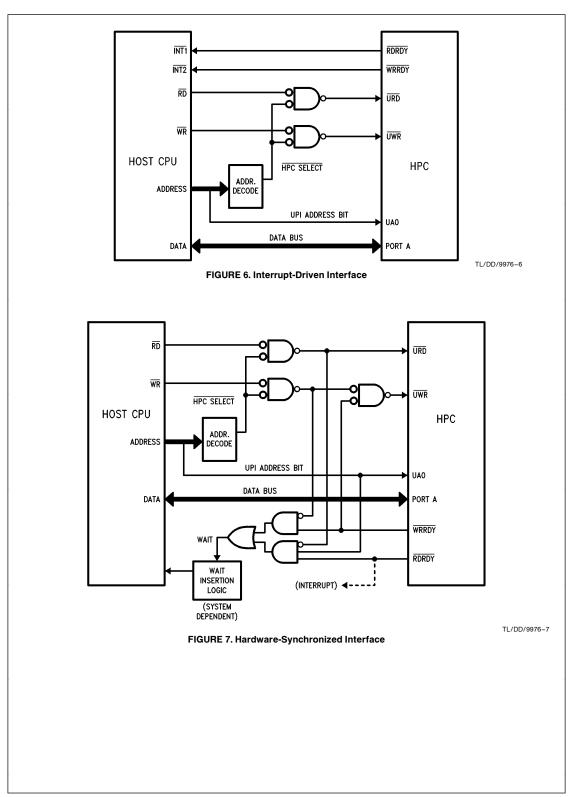
2.3.3 Hardware Synchronization

Figure 7 shows the connections required to implement hardware synchronization between the host and the HPC. In this scheme, there is no host software involved in synchronizing with the HPC; if the host attempts a UPI transfer for which the HPC is not prepared, the host is held in "Wait states" until the HPC is ready. Note that the UPIC register is an exception; Wait states are not to be inserted when the CPU polls the UPI port's status (UA0 = 0).

The main advantage of this scheme is speed: the CPU and HPC transfer data as fast as they can both run the transfer loop. (One will generally find that the HPC stays ahead of the CPU; the CPU tends to be in the critical path due to more complex buffer management algorithms.) The main disadvantage is that if the HPC is allowed to be interrupted in the middle of the transfer, the CPU is not free to do any-thing else at all, including servicing its own interrupts.

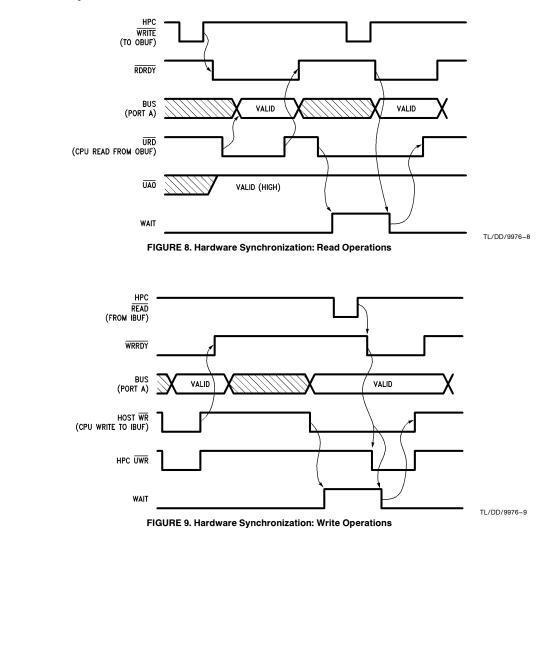
In addition to the logic to detect when to hold the host (at the bottom of the figure), additional gating is required on the \overline{UWR} signal, to prevent it from being asserted until the \overline{WRRDY} signal is active. This is required because the IBUF register of the HPC is a fall-through latch, and its contents would be lost if \overline{UWR} were allowed to go active too soon.





Figures 8 and *9* illustrate the timing involved in hardware synchronization. *Figure 8* shows the host attempting two UPI Read accesses in quick succession; the second Read access is held pending until the HPC has supplied the data. *Figure 9* shows the host attempting two UPI Write accesses in quick succession; it is held in Wait states (with the UWR signal suppressed) until the HPC has emptied the first value from the IBUF register.

This scheme and the interrupt-driven scheme above are not mutually exclusive; as shown in *Figure 6*, one might tie RDRDY or WRRDY, or both, to CPU interrupts. The application hardware described implements both schemes, leaving CPU software the option of using hardware synchronization or not. The driver program in the HPC operates the same, independent of the option used.



3.0 A UPI DRIVER AND SAMPLE APPLICATION

The circuit and program described below implement an interface between the HPC and a National microprocessor, the NS32CG16, as the host CPU. The UPI port is configured to be 8 bits wide. The hardware supports both interrupt-driven (RDRDY only) and hardware synchronization, as well as polling.

In order to demonstrate some real commands to support, a set of simple interfaces is attached to the HPC, typical of a front panel.

- Up to 8 pushbuttons
- Up to 8 LED indicators
- A 16-character alphanumeric LCD display
- A speaker for "beeps" on alert conditions or input errors
 A real-time clock interrupt function, giving the CPU the
- means to measure time intervals accurately.

This application by itself is admittedly not enough to justify the presence of an HPC in a system, but it is a simple application, and we expect that this will often be part of the HPC's job. For a much more comprehensive application, which includes this one as a subset, see the next application note in this series: "The HPC as a Front-End Processor".

We will describe in this section a specific set of hardware and software, and a UPI command and response protocol to make these interfaces play.

3.1 UPI Port Connections to NS32CG16

The attached schematic shows the HPC UPI port as it has been used a real application. On Sheet 1, a block diagram is given, showing the components involved. The CPU is an NS32CG16 microprocessor, running at a 15 MHz clock rate (crystal frequency 30 MHz). The HPC component is the HPC46083, running at a crystal frequency of 19.6608 MHz.

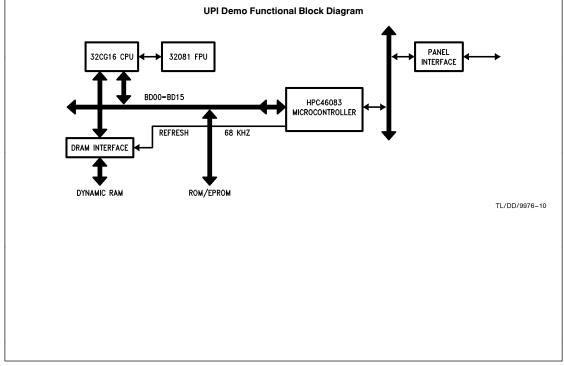
It would be unrealistic to present only the UPI interface section, since tradeoffs and implementation considerations abound when dealing with fast processors and large addressing spaces. For this reason, we include on sheets 5, 6 and 7 the circuitry involved in NS32CG16 address decoding and dynamic RAM control.

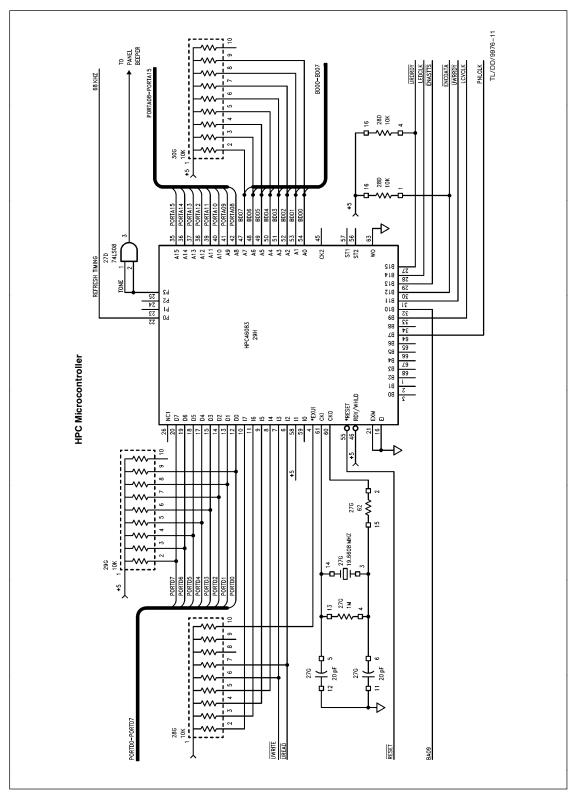
The UREAD and UWRITE UPI strobes are generated for the HPC in area B1 of Sheet 6. In addition, the latched CPU address bit BA09 is used as the UA0 addressing bit.

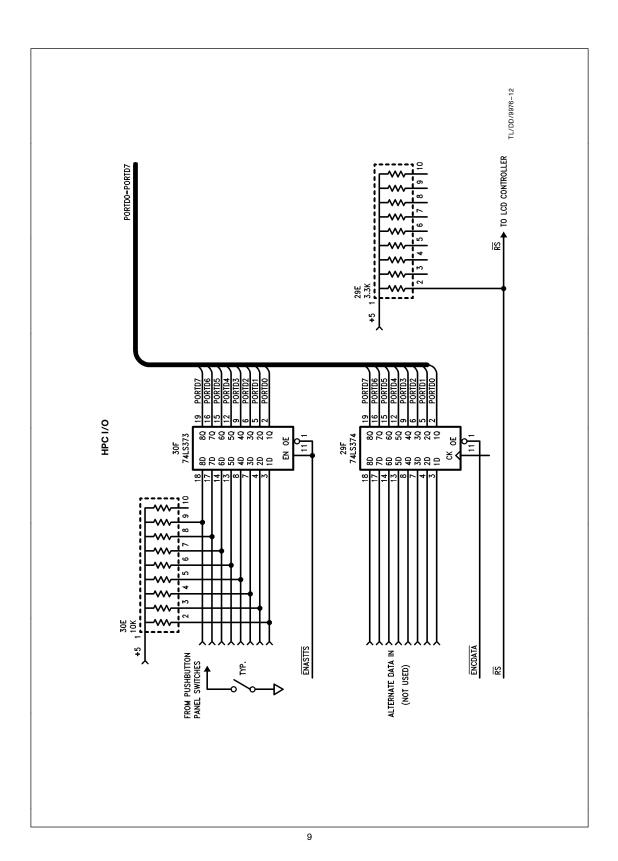
Hardware and Interrupt synchronization are accomplished as follows. On Sheet 6, area D8, the HPC signals URDRDY and UWRRDY enter a synchronizer, and emerge as URDRDYS and UWRRDYS. The URDRDYS signal goes to the CPU as its Maskable Interrupt signal (Sheet 5, area C8). After gating, which yields URDRDYSQ and UWRRDYSQ, they enter the PAL16L8 in area C7 of Sheet 6. This PAL's relevant outputs are WAIT1 and WAIT2, which go to the CPU for Wait State generation, and ACWAIT, which also goes to the CPU (as CWAIT) after passing through the PAL20R8 device in area D4 of Sheet 6.

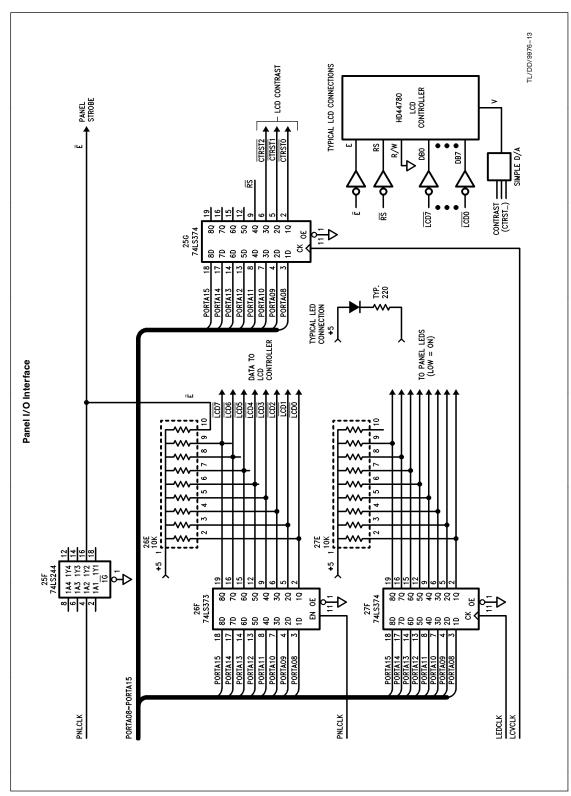
In addition, the HPC provides from Timer T4 a square wave at approximately 68 kHz, which triggers refreshes of dynamic RAM. The signal involved is called "68 kHz", and goes from the HPC on Sheet 4, area D1, to Sheet 6, area D8. Note that the detector in area D7 is held on at Reset, to preserve RAM contents by continuous refreshing while the HPC is being reset.

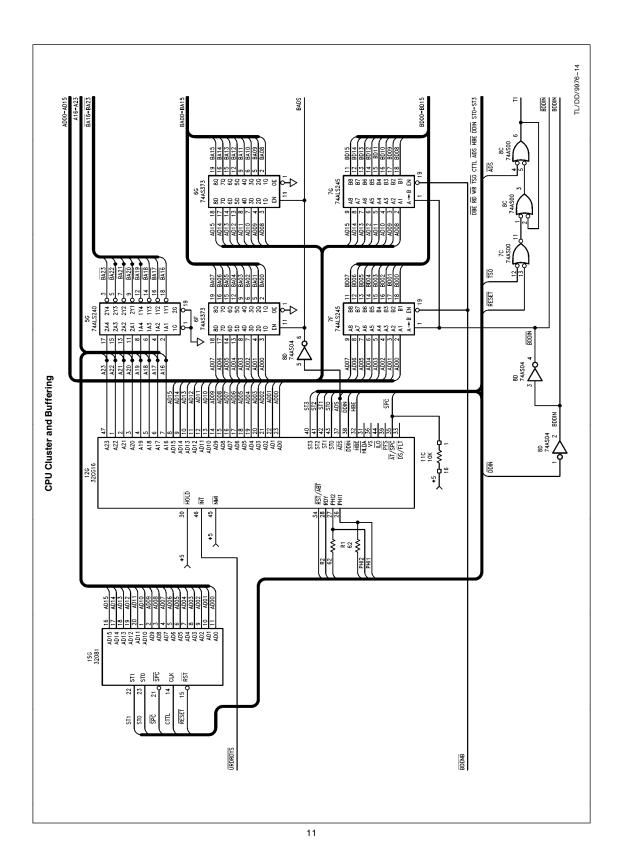
3.1.1 Schematic

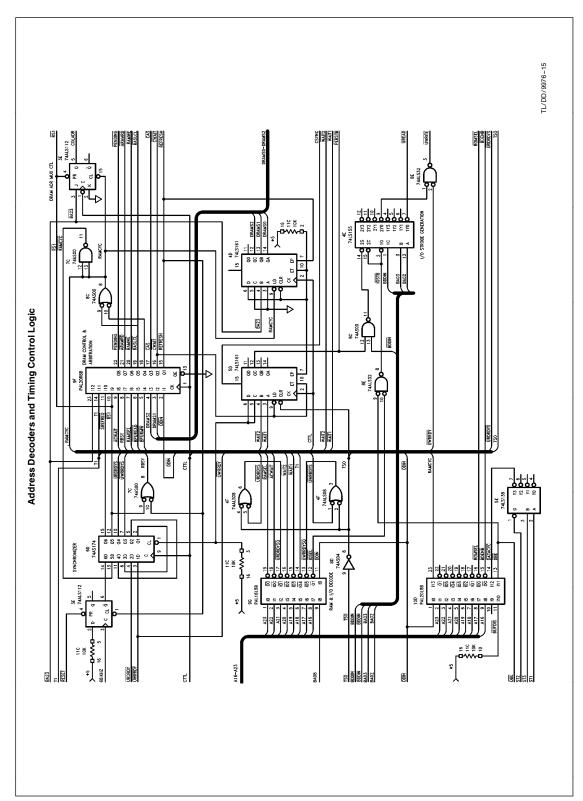


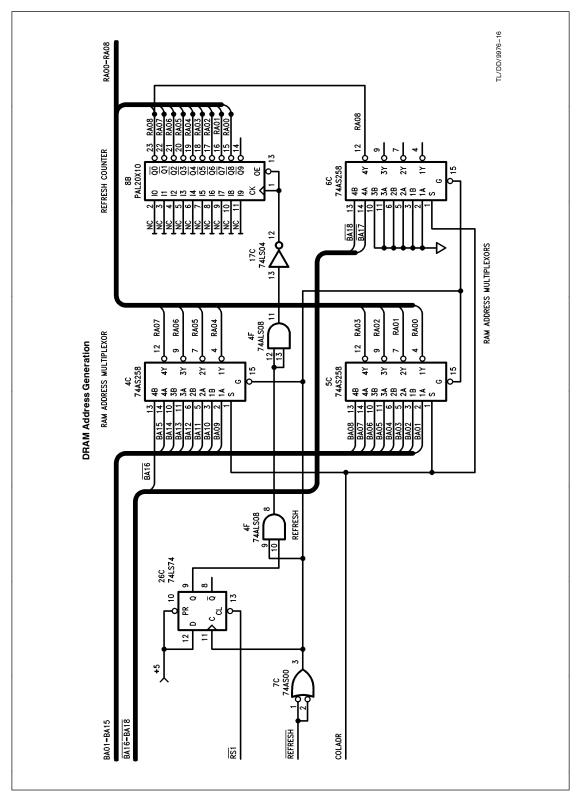












3.1.2 PAL	- Equations Schematic Sheet 7, Area 3D	
Name	REFRESH.PLD;	
Partno	XXXXX;	
Date	05/19/87;	
Revision	1A;	
Designer	FOX;	
Company	NSC;	
Assembly	Х7А;	
Location	8B;	
Device	p20x10;	

/*		4
•	: 9 BIT REFRESH COUNTER	4
/*		1 /c

•	le Target Device Types: PAL20X10	Ȣ
•	***************************************	***1
/** Inputs	•	
Pin l	= !refresh ;/* refresh pulse	2
/** Output	s **/	
	23]= [ra08] ;/* ram refresh address	*
Pin 14	= !refron ;/* refresh enabled output	*
	ations and Intermediate Variable definitions **/	
\$define	#	
<pre>!ra2.d = !ra3.d = !ra4.d = !ra5.d = !ra6.d = !ra7.d = !ra8.d = refron.d=</pre>	!ra3 \$ ra0 & ra1 & ra2; !ra4 \$ ra0 & ra1 & ra2 & ra3; !ra5 \$ ra0 & ra1 & ra2 & ra3 & ra4; !ra6 \$ ra0 & ra1 & ra2 & ra3 & ra4 & ra5;	
	14	

Г

	_				
Name		AM.PLD;			
Partno		XXXX;			
Date		7/25/87;			
Revision		A;			
Designer		0X;			
Company Assembly		SC;			
Location		7A;			
Device		F;			
	-	20r8;	ale ale ale ale ale ale a	******	de ale ale ale ale ale ale ale ale a
/*				***************************************	*********
	CONTE		E EMW	BPU CYCLE, SEPARATE BUSES	26
		States of		-	3)
•		ert rsl	leiaui		, x)
-			*****	*****	
				pes: PAL20R8B	s)
		0		рег• типсолор	
/** Inp	outs	**/			
n	1	= cttl	;	/* clock input	2
		= !ddin	-	/* data direction in signal	2)
'in		= .uuin = dramsl	-	/* DRAM state counter, bit 1	2)
	-	= drams1	-	/* DRAM state counter, bit 2	1
		= !bpurmw	,	/* BPU read modify write cycle	1
		= !bpuread	,	/* BPU source read (comb.)	2
		= !ramsel		/* Any RAM address decode	2)
		= busy		/* DRAM busy indication (rsl refresh)	2)
		= lacwait		/* Advanced CWAIT from ROM, or I/O	2
	-	= !rsl	-	/* ram cycle delayed by one Tstate	2
		= !srefreq		/* Refresh Request	s)
		= tl	-	/*Processor Il state	3)
	23	_ 01 !a23	-	/* Address 23	s)
			,	,	
	puts	-			
		= !refresh	,	/* refresh cycle	4
		= !cwait	-	/* 32C201 cwait	4
		= !cas		/* CAS, local & cartridge	4
		= !rascart	-	/* RAS for DRAM cartridge	4
		= !raslcl	;	/* RAS for local DRAM	1
	20	= !ramwe	;	/* DRAM Write enable	4
		= !aramrd		/* DRAM read	¢
		= !pending		/* DRAM cycle requested, but ctl busy	ų
				art, raslcl, ramwe, aramrd, pending] = 2;	
				ate Variable Definitions **/	
	-	= [pending			
define			-	t sequencer idle */	
define			-	t sequencer waiting for busy DRAM */	
define	GEXTW	t 1	/* wai	t sequencer waiting for cycle extension */	

```
Schematc Sheet 6, Area 5D (Continued)
field ctl = [refresh,cas,raslcl,rascart];
$define idle 00
$define cras
               01
$define crascas 05
$define casend 04
$define lras 02
$define lrascas 06
$define refadr 08
$define refras Ob
$define | #
field drscount = [drams2..drams1];
/** Logic Equations **/
                  = ramsel & !a23;
= ramsel & a23;
       lcl_sel
       cart_sel
       lclread
                     = !a23 & ddin;
                     = !a23 & !ddin;
       lclwrite
       holdoff
                     rsl;
                     = refresh | holdoff; (generated externally)
/*
       busy
       cart_start = cart_sel & (t1 | pending) & !holdoff;
       local_start = lcl_sel & (tl | pending) & !holdoff;
       ram_start = cart_start | local_start;
                     = drscount: [6..7] & ramwe;
       drrco
sequence waitseq {
      acwait & ramsel are mutually exclusive conditions */
/*
present widle if (ramsel | bpurmw & bpuread) & busy & tl next busywt;
               if acwait | (ramsel & !busy & tl & !bpurmw)
                      next cextwt;
                default next widle;
present busywt if busy
                                                    next busywt;
                if !busy & (bpurmw)
                                                    next widle;
               if !busy & !(bpurmw)
                                                    next cextwt;
present cextwt if ramsel & drscount: [0..1] | acwait next cextwt;
                default next widle;
}
sequence ctl {
present idle
               if cart_start
                                                    next cras;
                if local_start
                                                    next lras;
                                                   next refadr;
                if !ram_start & srefreq
                default next idle:
               if !rsl
present cras
                                                    next cras;
                if rsl
                                                     next crascas;
present crascas if (!bpurmw & drscount: [4..7]) | (bpurmw & drrco)
                                                    next casend;
                default next crascas;
present lras
                                                     next lrascas;
```

*/

```
Schematc Sheet 6, Area 5D (Continued)
present lrascas if (!bpurmw & drscount: [4..7]) | (bpurmw & drrco)
                                                     next casend;
                default next lrascas;
present casend if srefreq
                                                     next refadr;
                if !srefreq
                                                     next idle;
present refadr if srefreq
                                                     next refadr;
                if !srefreq & !rsl
                                                    next refras;
                if !srefreq & rsl
                                                    next idle;
present refras if ramwe
                                                    next refadr;
                default next refras;
}
/* remember ramwe & aramrd are delayed by one t-state */
ramwe.d = !refresh & (bpurmw & drscount: [6..7] & !ramwe
                        | !bpurmw & !ddin & (ram_start | ctl: cras
                                | (cart_sel & drscount: [0..3]) | ctl:lras)
                )
       ctl:refras & rsl & !ramwe;
aramrd.d = (bpurmw & drscount: [0..3] | !bpurmw & ddin)
       & (ctl:cras | ctl:crascas | ctl:lras | ctl:lrascas);
```

```
Schematic Sheet 6, Area 7C
Name
         DCD1.PLD;
Date
         07/03/87;
Revision
        lA;
        FOX;
Designer
Company
         NSC:
Assembly
         X7A:
Location 9G;
Device
         p1618:
/*
                                                                         */
/* DECODE 1: I/O DECODE, PROM & HPC I/F WAIT CONTROL
                                                                         */
/* 6/3: two waits for hpc write
                                                                         */
/* 6/4: 1 wait min. for ALL i/o, including HPC
                                                                         */
/* 6/4: 3 wait min. for i/o
                                                                         */
/*
                                                                         */
/* Allowable Target Device Types: PAL16L8B
                                                                         */
/** Inputs **/
      [1..8] = [a23..16] ;/* high order address bus
Pin
                                                                         */
Pin
         9
                = ba8
                            ;/* address bit 8
                                                                         */
                            ;/* cpu ddin/
        11
                = !ddin
Pin
                                                                         */
                = !uwrrdys ;/* (HPC) UWRRDY/, synchronized
Pin
         13
                                                                         */
                            ;/* Tl state of CPU
Pin
         14
                = tl
                                                                         */
Pin
         17
                = !urdrdys
                            ;/* (HPC) URDRDY/, synchronized
                                                                         */
/** Outputs **/
                           ;/* I/O select decode
Pin 12
              = !iosel
                                                                         */
Pin
        15
               = !waitlo
                           ;/* WAIT1 output
                                                                         */
Pin
              = !wait2o
                           ;/* WAIT2 output
        16
                                                                         */
Pin
        18
              = !acwait
                           ;/* Advance CWAIT for RAM ctl
                                                                         */
                            ;/* DRAM address decode
Pin
         19
               = !ramsel
                                                                         */
/** Declarations and Intermediate Variable Definitions **/
$define | #
                           ;/* address field
field address
               = [a23..16]
                                                                         */
               = [acwait,wait2o,waitlo]; /* wait value field
field waitv
                                                                         */
                "b'000
$define nowaits
               ("b'100 & tl)
$define waitly
/* note use of \# in next 3 defines because $define not nestable
                                                                         */
                            ("b'l01 & ("b'011 # "b'l00 & tl))
$define wait2v
$define wait3v
                            ("b'll0 & ("b'0ll # "b'l00 & tl))
$define wait4v
                            ("b'lll & ("b'0ll # "b'100 & tl))
$define cwaitonly "b'100
/** Logic Equations **/
                = address: [0780000..07fffff] | address: [0800000..0bfffff];
ramsel
                = address: [0fd0000..0ffffff] & !ba8;
iosel
                = wait3v & address: [0000000..00fffff] /* main rom, 3 waits */
waitv
                | wait4v & address: [0200000..05fffff] /* font rom, 4 waits */
                | wait3v & address: [0fd0000..0ffffff] & !ba8 /* i/o, 1 wait */
                cwaitonly & address: Off0000 & !ba8 &
                            (!urdrdys & ddin | !uwrrdys & !ddin);
```

Name				
11 CHILD	DCD2.PLD;			
Partno	XXXXX;			
Date	07/27/87;			
Revision	1C;			
Designer	FOX;			
Company	NSC;			
Assembly	X7A;			
Location	10D;			
Device	p2018;			
/********	*****	* * * * * * * * * * * * *	* * * * * * * * * * * * * * * * * * * *	****/
/*				*/
/* DECODE 2	e: ROM DEC	ODE, BUFFER	CONTROL, BPU DECODE	
/* 5/24: ir	ncluded en	bpu in bpucy	vc generation	*/
/* 5/28: ad	lded bpucy	c to rdenb		*/
/* 5/31: ad	lded fcxxx	x to bdenb		*/
/* 6/23: ad	lded buffe	r disable te	erm for SPLICE	*/
/* 7/25: re	configure	d for bpurmw	/ & bpuread	*/
	-	-	ubpu \geq enablebpu (for master enb)	*/
/********	*****	*****	* * * * * * * * * * * * * * * * * * * *	****/
/* Allowabl	e Target	Device Types	S: PAL20B	*/
/*********	*****	*****	* * * * * * * * * * * * * * * * * * * *	****/
/** Inputs	**/			
Pin	1 =	!ddin	;/* ddin/ from cpu	*/
Pin	= [29]=	[a2316]	;/* high order address bus	*/
Pin	10 =	!enablebpu	;/* BPU enable, static bit	*/
Pin	11 =	!bufdis	;/* buffer disable	*/
Pin	13 =	!dbe	;/* dbe/ from tcu	*/
Pin	14 =	!datacyc	;/* data cycle status decode	*/
Pin	23 =	ramcyc	;/* ram cycle in progress	*/
/** Outputs	**/			
Pin	15 =	!bdenb	;/* BD bus enable	*/
Pin	16 =	!romsel	;/* Main rom select	*/
Pin	17 =	!romcart	;/* rom cartridge select	*/
Pin	18 =	!bpurmw	;/* BPU read modify write	*/
Pin	19 =	!bpuread	;/* BPU read cycle (comb.)	*/
Pin	20 =	!vramsel	;/* video ram select	*/
Pin	21 =	rdbufin	;/* RAM data bus direction (in)	*/
Pin	22 =	!rdenb	;/* RAM data bus enable	*/
/** Declara	tions and	Intermediat	e Variable Definitions **/	
field addres	-	[a07]0]	. It address field	* /
	-	[a2316]	;/* address field	*/
romspace			00000005fffff];	
ramspace		-	07800000bfffff];	
stack	=	address: [()780000078ffff];	
\$define	#			
	min b_ddi	n = 0;		
/** Logic H	lquations	**/		
romsel = add	lress: [00	0000000fff	ff]; /* main rom	*/
	-		fff]; /* font rom	*/
	-			

```
Schematic Sheet 6, Area 7A (Continued)
vramsel = address: [0f00000..0f0ffff];
                                                             /* video ram (scan buffer)
                                                                                             */
/*
/*
            bpucyc & b_ddin are D latches implemented in the PAL
/*
/*
            basic d latch equation (w/o set or clear) is:
/*
                       Q = (G \& D) | (!G \& Q) | (D \& Q)
/*
/*
            The b_ddin latch is fall through while ramcyc not asserted,
/*
            latched while ramcyc is asserted, therefore, for both latches:
*/
                       = !ramevc:
            g
/*
/*
            The bpurmw latch d input is ""bpurange'', defined as:
*/
                                                             /* rom
            bpurange= address: [0000000..05fffff]
                                                                                             */
                      address: [0790000..0bfffff];
                                                            /* dram, less stack
                                                                                             */
/*
            This ""d'' input would use too many terms. The bpucyc output,
/*
            however, need only be latched when it is asserted, as this is
/*
            the situation that can allow the cpu and ram control to
/*
            not be synchronized. This simplification allows the simplification
/*
            of the latch to:
/*
                       Q = D \mid (!G \& Q)
*/
            bpurmw = enablebpu & (!ddin & bpurange & datacyc | (!g & bpurmw));
            bpuread = enablebpu & ddin & bpurange & datacyc;
/*
            rdenb enables cpu access to the ram data bus
*/
            rdenb
                       = dbe & bufdis &
                       ( !bpurmw & bpuread & romspace
                                                             /* buffer must be off for bpu
                                                             /* but on for source in rom
                                                                                             */
                        /* no DRAM or bpu control writes are permitted
                                                                                             */
                        /* while in inner loop of bitblt
                                                                                             */
                        /* (within interrupt ok due to vector read!)
                        ramspace
                        address: [Ofe0000..0feffff]);
                                                            /* i/o access to bpu
                                                                                             */
            !rdbufin
                      = (ramspace | address: [0fe0000..0feffff]) & !ddin
                       romspace & bpuread;
                                                             /* any rom
            bdenb
                       = dbe & !bufdis & (romspace
                                                                                             */
                                address: [0f00000..0f0ffff] /* scan buffer
                                                                                             */
                                address: [0fd0000..0fdffff] /* cmnd/status
                                                                                             */
                                | address: [Off0000..0ffffff] /* non-bpu i/o
                                                                                             */
```

3.2 Application Connections

The connections made to the HPC are shown in schematic sheets 2 through 4.

3.2.1 LCD Data

An 8-bit parallel interface connects the upper half of Port A, through buffers and latches on Sheet 4, to a Hitachi HD44780 alphanumeric LCD display controller. The signals in our application are inverted with respect to the HD44780 documentation, due to the nature of the front panel module we used.

Sending data from the HPC to the LCD display involves the following procedure:

- Setup the RS signal: 1 for a command, 0 for data. This is done by setting up LCD Contrast status on the high-order byte of Port A (pins A8–A15), with the desired RS state on pin A11, then pulsing the signal LCVCLK (pin B9) high, the low.
- 2. Setup the panel data on HPC pins A8-A15.
- 3. Set the PNLCLK signal (pin B7) low for 1.2 μ s, then high. This clocks the data into the LCD display controller. Note that the latch in area B6 of Sheet 4 is effectively serving only as a buffer; the PNLCLK Enable signal, being normally high, allows data to fall through whenever it changes when used as described here.
- 4. Since the handshaking capability of the HD44780 is not being used here, it is necessary for the HPC to use an internal timer to determine when the controller is ready after sending a command or data. The delay time is either 120 μ s or 4.9 ms, depending on the type of command sent.

3.2.2. LCD Contrast (LCD Voltage)

A three-bit value is presented for LCD contrast on signals CTRST0 through CTRST2. A value of 000 is highest contrast, and 111 is lowest contrast. To change the contrast, the value is placed on HPC pins A8 (LSB), A9 and A10 (MSB), the LCVCLK (pin B9) is pulsed high, then low.

Note that some other bits within this latch have other functions: bit 3 (from HPC pin A11) is the \overline{RS} signal to the LCD controller, and bit 7 (from pin A15) is used by the HPC firmware as a Fatal Error flag. These bits must be setup correctly whenever the LCD Contrast latch is written to.

3.2.3 LEDs

Up to 8 LED indicators may be connected, through the latch in area A6 of Sheet 4, to the upper byte of Port A. The LED's are assumed to be connected already to their own current-limiting resistors.

The desired data is setup on Port A pins A8–A15, then a pulse is presented on the LEDCLK signal (pin B14); high and then low. Data is presented in complemented form by the HPC (0 = on, 1 = 0 ff). Any or all (or none) of the latch bits may be connected to drive LEDs.

3.2.4 Speaker (Beeper)

A tone is produced on a speaker by enabling Port P pin P3 as the Timer T7 output, and running Timer T7 so as to produce a 3 kHz square wave. Since timer outputs toggle on underflows, this corresponds to a timer underflow rate of 6 kHz. The tone signal is shown is area D1 of Sheet 2.

3.2.5 Pushbutton Switches

Up to eight pushbuttons may be connected to the HPC's Port D pins, through the buffer in area D6 of Sheet 3. Each

pushbutton is assumed to be an SPST switch, shorting to ground when depressed. The pull-up resistors present a "1" level otherwise. The HPC must de-bounce the inputs in its firmware before issuing them to the CPU.

The pushbuttons are examined every 10 ms, by setting the ENASTTS signal (pin B13) low while ensuring that ENCDATA (pin B12) is high. This presents the switch outputs onto Port D. Unused bits should be pulled high to avoid triggering spurious pushbutton events.

3.3 Protocol Between CPU and HPC

The scheme supported by the UPI Driver program is asynchronous full-duplex communication with CPU. That is, either side is allowed to speak at any time. To avoid confusion, however, any message is restricted to send data in only one direction: in sequences initiated by the CPU ("Command" sequences), only the CPU talks, and in sequences initiated by the HPC ("Interrupt" sequences), only the HPC talks. Thus, a Command sequence and an Interrupt sequence can be in progress simultaneously without confusion.

Acknowledgement of a Command or an Interrupt sequency is possible; a Command can trigger an acknowledgement Interrupt sequence, and an Interrupt sequence can result in a subsequent Command sequence. The critical distinction, though, is that the acknowledgement need not come immediately. If, for example, the HPC is already in the process of sending an Interrupt message, and receives a Command, it will complete the current Interrupt sequence before acknowledging the Command with a new Interrupt.

Command sequences (from the CPU to the HPC) consist of a one-byte command code, followed by any argument values necessary to complete the command. Each byte written to the HPC triggers an internal interrupt (I3); the HPC buffers up these bytes until a full command has been received, then acts on it in the last byte's interrupt service routine. Commands taking a significant amount of processing time can be scheduled within the HPC using interrupts, either from external events or from one of the HPC's eight timers; each interrupt triggering the next step of the command.

Interrupt sequences (from the HPC to the CPU) operate similarly, but with a small difference. Only the first byte presented by the HPC causes an interrupt to the CPU: this byte is the interrupt vector value, which triggers the interrupt (through the RDRDY pin) and selects the CPU's service routine. The CPU remains in its interrupt service routine until the transfer of data associated with that interrupt event is finished, then returns to its previous task. This is not to say that the CPU must keep all other interrupts disabled during an Interrupt sequence, but only that no other interrupt occurring during this time may cause the CPU to read from the HPC, or to terminate reading, until the current Interrupt seguence is complete. With the NS32C016 processor as host, the main challenge is to keep the Interrupt Acknowledge bus cycles from other interrupts, which appear as Read cycles, from causing URD pulses to the HPC. It is possible to distinguish a Non-Maskable Interrupt from a Maskable Interrupt by the address asserted by the CPU in acknowledging the interrupt, and in a larger kind of system containing an NS32202 Interrupt Control Unit, the NS32000 Cascaded Interrupt feature can be used to prevent unwanted reads from the HPC from occurring as a result of other Maskable interrupts as well. In our application hardware, the only type of extraneous interrupt occurring is the Non-Maskable Interrupt; address decoding logic isolates the HPC's UPI port from these.

and any argumen FFFE00. The CPU FD0000 to determ byte, or it can simp held in Wait states ed, the CPU may	Amand code) is sent to address FFFC00, Int bytes are then written to address U may poll the UPIC register at address ine when the HPC can receive the next by attempt to write, in which case it will be until the HPC can receive it. Unless not- send commands continuously, without ledgement interrupts from previous com- This command has two functions. The first INITIALIZE command after a hard- ware reset (or RESET command) en- ables the IRTC and IBUTTON-DATA interrupts. The INITIALIZE command may be re-issued by the CPU to either start or stop the IRTC interrupts. There is one argument:	A5 RESET-HPC Resets the HPC if it is written to ad- dress FFFC00. It may be written at any time that the UPI port is ready for in- put; it will automatically cancel any partially-entered command. The CPU's Maskable Interrupt must be disabled before issuing this command. After issuing this command. After issuing this command. After issuing this command, the CPU should first poll the UPIC register at address FD0000 to see that the HPC has input the command (the least-sig- nificant bit [Write Ready] is zero). It must then wait for at least 25 μs, then read a byte from address FFFE00. The HPC now begins its internal re-initiali- zation. The CPU must wait for at least 80 μs to allow the HPC to re-initialize the UPI port. Since part of the RESET procedure causes Ports A and B to
01 SET- CONTRAST	RTC-Interval: One-byte value. If zero, IRTC interrupts are disabled. Other- wise, the IRTC interrupts occur at the interval specified (in units of 10 ms per count). The single argument is a 3-bit number specifying a contrast level for the LCD panel (0 is least contrast, 7 is highest	float briefly (this includes the CPU's Maskable Interrupt input pin), the CPU should keep its maskable interrupt dis- able during this time. It also must not enter a command byte during this time because the byte may be lost. 3.5 Interrupts The HPC interrupts the CPU, and provides the following val-
02 SEND-LCD	contrast). There is no response inter- rupt. Does not require INITIALIZE command first. This writes a string of up to 8 bytes to	ues as the interrupt vectors for the CPU hardware. The CPU then reads data from the HPC at address FFFE00. All data provided by the HPC must be read by the CPU before re- turning from the interrupt service routine, otherwise the HPC
	the LCD panel. Arguments are: flags: a single byte, containing the RS bit associated with each byte of data. The first byte's RS value is in the least- significant bit of the FLAGS byte. #bytes: The number of bytes to be	would either hang or generate a false interrupt. The CPU may poll the UPIC register at address FD0000 to determine when each data byte is ready, or it may simply attempt to read from address FFFE00, and it will be held in Wait states until the data is provided by the HPC. Note: All CPU interrupt service routines, including the NMI interrupt routines,
	written to the LCD display.	must return using the "RETT 0" instruction. Do NOT use "RETI". 00-0F (Reserved for CPU internal traps and the NMI inter-
	byte[1]—byte[#bytes]: The data bytes themselves. The HPC determines the proper delay timing required for command bytes (RS = 0) from their encodings. This is either 4.9 ms or 120 μ s. The response from the HPC is the !ACK-SEND-LCD interrupt, and this command must not be repeated until	rupt.) 11 !RTC Real-Time Clock Interrupt. No data returned. Enabled by INITIALIZE command if interval value supplied is non-zero. Note: this version of HPC firmware issues a non-fatal !DIAG in- terrupt if the CPU fails to service each !RTC interrupt before the next one becomes pending.
03 SEND-LED	the interrupt is received. This com- mand does not require an INITIALIZE command first. The single argument is a byte contain-	17 IACK-SEND-LCD This is the response to the SEND- LCD command, to acknowledge that data has all been written to Panel LCD display. No other data is provid-
VO SEND-LED	ing a "1" in each position for which an LED should be lit. There is no response interrupt, and	ed with this interrupt. Always en- abled, but occurs only in response to a SEND-LCD command.
04 BEEP	this command does not require the INITIALIZE command first. No arguments. This beeps the panel for approximately one second. No re- sponse interrupt. If a new BEEP com- mand is issued during the beep, no er- ror occurs (the buzzer tone is extend- ed to one second beyond the most re-	18 IBUTTON-DATA Pushbutton status has changed: one or more buttons have been either pressed or released. The new status of the switches is reported in a data byte, encoded as follows: Any pushbutton that is depressed is presented as a "1". All other bit posi- tions, including unused positions, are
	cent command). Does not require INI- TIALIZE command first.	zeroes. The pushbuttons are de- bounced before being reported to

the CPU. This interrupt is enabled by the first INITIALIZE command after a reset.
1D !DIAG Diagnostic Interrupt. This interrupt is used to report failure conditions and CPU command errors. There are five data bytes passed by this interrupt:
Severity Error Code
Data in Error (passed, but contents not defined)
Current Command (passed, but con- tents not defined) Command Status (passed, but con- tents not defined)
The Severity byte contains one bit for
each severity level, as follows:
N (Note): least severe. The CPU missed an event; currently only the IRTC interrupt will cause this.
C (Command): medium severity. Not currently implemented. Any com- mand error is now treated as a FA- TAL error (below).
F (Fatal): highest severity: the HPC has recognized a non-recoverable error. It must be reset before the CPU may re-enable its Maskable In- terrupt. In this case, the remaining
data bytes may be read by the CPU, but they will all contain the value 1D (hexadecimal). The CPU must issue a RESET command, or wait for a hardware reset. See below for the procedure for FATAL error recovery.
The Error Code byte contains, for non-FATAL errors, a more specific indication of the error condition:
RTC (Reserved for COMMAND)
RTC = Real-Time Clock overrun: CPU did not acknowledge the RTC interrupt before two had occurred.
The other bits are reserved for de- tails of Command errors, and are not implemented at this time.
The remaining 3 bytes are not yet de- fined, but are intended to provide de- tails of the HPC's status when an ille- gal command is received.
Note: Except in the FATAL case, all 5 bytes provided by the HPC <i>must</i> be read by the CPU, regardless of the specific cause of the error.
Fatal Error Recovery: When the HPC signals a !DIAG error with FATAL severity, the CPU may use the following procedure to recov- er:
1. Write the RESET command (A5 hex) to the HPC at address FFFC00.

- 2. By inspecting the UPIC register at address FD0000, wait for the HPC to read the command (the *WRRDY bit will go low).
- 3. Wait an additional 25 μ s.
- 4. Read from address FFFE00. This will clear the OBUF register and reset the Read Ready status of the UPI port. The HPC will guarantee that a byte of data is present; it is not necessary to poll the UPIC register. This step is necessary because only a hardware reset will clear the Read Ready indication otherwise (HPC firmware cannot clear it).
- 5. Wait at least 80 μ s. This gives the HPC enough time to re-initialize the UPI port.
- 6. After Step 5 has been completed, the CPU may re-enable the Maskable Interrupt and start issuing commands. Since the HPC is still performing initialization, however, the first command may sit in the HPI IBUF register for a few milliseconds before the HPC starts to process it.

4.0 SOURCE LISTINGS AND COMMENTARY

4.1 HPC Firmware Guide

Refer to this section for help in following the flow of the HPC firmware in the listing below. Positions in the code are referenced by assembly language labels rather than by page or line numbers.

The firmware for the HPC is almost completely interruptdriven. The main program's role is to poll mailboxes that are maintained by the interrupt service routines, and to send an interrupt to the CPU whenever an HPC interrupt routine requests one in its mailbox.

On reset, the HPC firmware begins at the label "start". However, the first routine appearing in ROM is the Fatal Error routine. This was done for ease of breakpointing, to keep this routine at a constant address as changes were made elsewhere in the firmware.

4.1.1 Fatal Error Routine

At the beginning of the ROM is a routine (label "hangup") that is called when a fatal error is detected by the HPC. This routine is usually called as a subroutine (although it never returns). It disables HPC internal interrupts, and then sets bit 7 of the LCD Contrast Latch as a trigger for a logic analyzer, MOLE or ISE system.

Its next action is to display its subroutine return address in hexadecimal on the LCD panel. This address shows where the error was detected. The HPC then enters an infinite loop, which continuously presents the !DIAG interrupt. It may be terminated either by a hardware reset or by sending the RESET command from the CPU. On receiving the RE-SET command, the HPC jumps to label "xreset", which is within the command processing routine. The "xreset" routine waits for the CPU to read from the UPI port, then clears a set of registers to simulate a hardware reset and jumps to the start of the program.

4.1.2 Initialization

On receiving a Reset signal, the HPC begins execution at the label "start". A required part of any application is to load the PSW register, to select the desired number of Wait states (without this step, the Reset default is 4 Wait states, which is safe but usually unnecessary).

Other initializations here are application-dependent, and so they relate to our application system and front-panel operations.

At label "srfsh", the program starts the Refresh clock pulses running for the dynamic RAM on our application hardware, from HPC pin P0 (controlled by Timer T4). For debugging purposes, a circuit within the RAM controller section performs continuous refreshes during Reset pulses, so data in dynamic RAM is never lost unless power is removed.

At "supi", the UPI port is initialized for transfers between the HPC and the CPU.

At label "sram", all RAM within the HPC is initialized to zero. This is done for debugging purposes, to help ensure that programming errors involving uninitialized data will have more consistent symptoms.

At "sskint", the stack pointer is initialized to point to the upper bank of on-chip RAM (at address 01C0). The address of the fatal error routine "hangup" is then pushed, so that it will be called if the stack underflows. This is not necessary in all applications, since the Stack Pointer starts at address 0002, but for our purposes it was more convenient to relocate it.

At "tminit", the timers T1–T3 are stopped and any interrupts pending from timers T0–T3 are cleared.

In addition, some miscellaneous port initializations are performed here. The upper byte of Port A is set as an output port (for data going to the LCD and LED displays), and the Port B pins which select pushbutton data are initialized.

At "sled", the LED control signals are initialized, and all LED indicators on the panel are turned off.

At "stmrs", all timers are loaded with their initial values, and timers T5-T7 are stopped and any interrupts pending from them are cleared. (Timer T4 keeps running for dynamic RAM refresh.)

At "sled", the panel LCD display is initialized to a default contrast level of 5, then commands are sent to initialize it to 8-bit, 2-line mode, with the cursor visible and moving to the right by default. This section calls a subroutine "wrpnl", located at the end of the program, which simply writes the character in the accumulator out to the LCD display and waits for approximately 10 ms. Note that if the CPU fails to initialize the LCD display further, a single cursor (underscore) character is all that appears: a recognizable symptom of a CPU problem.

The program now continues to label "minit", which performs some variable initializations which are necessary for operation of the UPI Driver itself (as opposed to the application). This much must always be present, but any other initializations required by the application should appear here as well. For our front-panel application, there are no such initializations required. At label "runsys", the necessary interrupts are enabled (from the timers, and from pin I3, which is the UPI port interrupt from the CPU), and the program exits to the Main Program loop at label "mainlp".

4.1.3 Main Program (UPI Output to CPU)

The Main Program is the portion of the UPI Driver that runs with interrupts enabled. It consists of a scanning loop at label "mainlp", calling a set of subroutines (explained below). It is responsible for interrupting the CPU and passing data to it. The HPC is allowed to write data to the CPU only after interrupting it. The main loop scans a bit-mapped variable in on-chip RAM that is set up by interrupt service routines (a word called "alert") to determine whether any conditions exist that should cause an interrupt to the CPU.

The "alert" word contains one bit for each interrupt that the HPC can generate. If a bit is set (by an interrupt service routine), the Main Program jumps to an appropriate subroutine to notify the CPU. Each subroutine first checks whether the UPI interface's OBUF register is empty, and if not, it waits (by calling the subroutine "rdwait"). It then writes the 32000 interrupt vector number to the OBUF register. This has the effect of interrupting the CPU (Because the pin URDRDY goes low), and the CPU hardware reads the vector from the OBUF register. If there is more information to give to the CPU, the HPC places it, one byte at a time, into the OBUF register, waiting each time for OBUF to be emptied by the CPU. This technique assumes that the CPU remains in the interrupt service routine until all data has been transferred. If the CPU were to return from interrupt service too early, the next byte of data given to it would cause another interrupt, with the data value taken as the vector number. (Note, however, that a Non-Maskable interrupt is allowed. It simply delays the process of reading data from the HPC. Since the HPC is running its main program at this point, with its internal interrupts still enabled, it is not stalled by this situation.)

Subroutines called from the Main Program loop are:

- sndrtc: sends a Real-Time Clock interrupt to the CPU. No data is transferred; only the interrupt vector.
- sndlak: interrupts the CPU to acknowledge that a string of data (from a SEND-LCD command) has been written to the LCD display. No data is transferred for this interrupt.
- sndbtn: interrupts the CPU to inform it that a pushbutton has been pressed or released. A data byte is transferred from variable "swlsnt", which shows the new states of all the pushbuttons.
- sndiag: interrupts the CPU to inform it of a !DIAG interrupt condition, when it is of NOTE severity. (Other !DIAG conditions are handled at label "hangup".)

4.1.4 Interrupt Service Routines

All of the remaining routines are entered by the occurrence of an interrupt.

4.1.4.1 UPI Port Input from CPU (Interrupt I3)

This interrupt service routine, at label "upiwr", accepts commands from the CPU. Each byte of a command triggers an interrupt on the I3 pin. When the last byte is received, the command is processed before the I3 interrupt routine returns. The HPC is therefore immediately ready to start collecting another command. Any command that involves waiting is only initiated before the I3 routine returns, and interrupts are set up to activate more processing when the time is right. Therefore, this interrupt service routine returns promptly, even for time-consuming commands.

At any time, the "upiwr" routine may be in one of the following states:

- 1. Waiting for the first byte of a command. In this state, the variable "curcmd" (Current Command) has its top bit ("cmdemp") set, meaning that it is empty. When a byte is received from the CPU in this state, this routine jumps to the label "firstc". The byte is placed in the "curcmd" byte (clearing the top bit), and then a multi-way branch (jidw) is performed, whose destination depends on the contents of the byte. The possible destinations have labels starting with the letters "fc". If the command has only one byte (for example, the command BEEP), it is processed immediately in the "fc" sequence, and the "curcmd" variable is set empty again. If, however, the command is longer than one byte, its "fc" routine will place a value into the variable "numexp", which gives the number of additional bytes that are expected for this command, and then will return from the interrupt. Note that the "curcmd" byte now appears to be full, because its top bit is no longer set.
- 2. Collecting bytes of a command. The code that is relevant in this state is between the labels "upiwr" and "lastc". This state is in effect while the "cmdemp" bit of "curcmd" is zero and the "numexp" variable is non-zero. Each I3 interrupt causes the routine to place the command byte into a buffer ("cpubuf", with pointer variable "cpuad"), decrement the "numexp" variable, and return if the result is non-zero. If the result is zero, then the routine has collected an entire command, and it goes to the label "lastc", and enters state (3) below.
- 3. In this state, the requested number of bytes has been collected, and this usually means that the entire command, except for the first byte, is in the "cpubuf" area of RAM. The code for this state is at label "lastc". First, the "curcmd" byte is checked to see whether "extended collection" is being performed (bit 6 set: see below). If not, the "curcmd" byte is set empty. A multiway branch is then performed (jidw), which transfers control depending on the command byte in "curcmd". All routines that are destinations of this branch start with the letters "lc". The "Ic" routine for each command uses the data in "cpubuf" to process the current command. In some cases, this processing is completed very quickly. For example, at label "lcsled", a value is simply transferred from "cpubuf" to a latch that drives the LEDs on the front panel, and this interrupt service routine returns. But a more complex command can move data out of "cpubuf" to other variables in RAM, and start a timer to sequence the process of executing the command.

In some commands (for example, SEND-LCD), state (3) above is entered twice. This is called "extended collection", and occurs when a command has variable length. State (3) is entered once to collect enough information to determine the exact length of the command. It then sets up the "numexp" variable again, re-entering state (2) to collect the second time, it processes the command. A bit in the "curcmd" variable (bit 6, called "getcnt") is set in state (1), which indicates that another collection will be performed, and prevents state (3) from setting the "curcmd" byte empty the first time it is entered.

Command Processing Routines			
INITIALIZE	13 interrupt labels:	State 1 = fcinit	State 3 = Icinit
SET-CONTRAST	l3 interrupt labels: At label ''lcslcv'' (Set LCD Volt latch is loaded from the value s		State 3 = Icslcv
SEND-LCD	bytes are requested for collect meaning that these are not the (jumping to label "lcslc1"), the # bytes value supplied by the O requested, this time with the "C collected, control is transferred data bytes for the panel are un LCD string buffer "lcdbuf". The and the number of bytes to be "lcdsct". Timer T6 is now start bytes from the LCD string buffe On occurrence of each T6 inte written to the LCD display. Dep and the value sent to the panel triggers the next transfer. Whe Timer T6 has provided the prop	State 1 = fcslcd inded collection" feature. At labe ion, but the "getcnt" bit of "curc last bytes of the command. At length of the instruction is deter PU, and a second collection of getcnt" bit off. When the last byt d to the label "lcslcd", then to "I loaded from the CPU buffer are e flag (RS) bits are loaded into v sent to the LCD display is place ed, to provide scheduling interna- er to the LCD display. rrupt (labels "t6int" and "t6nxtc bending on the state of the RS fl I, T6 may run for either 120 µs con the last character has been tra- ber delay after it, the bit "alcdak ain program to send an IACK-St	cmd'' is set, label "lcslcd'' rmined from the bytes is the has been cslc2". Here, the ariable "lcdsfg", d into variable upts for writing the "'), one byte is ag for that byte, or 4900 μ s before it ansferred, and " is set in the

SEND-LED	13 interrupt labels:	State 1 = fcsled	State 3 = Icsled
	At label "lcsled", the byte	provided by the CPU is written to	the LED latch.
BEEP	13 interrupt labels:	State 1 = fcbeep	State 3 = (none)
	At label "fcbeep", Port P p	oin P3 is enabled to toggle on eac	h underflow of Timer T7,
	which has been initialized	at the beginning of the program (I	abel "stmrs") to
	underflow at a rate of 6 kH	z. Pin P3, then, presents a 3 kHz	square wave to the panel
	buzzer. To time out the du	ration of the beep tone, interrupts	from Timer T0 are
	enabled, which then occu	once every 53 ms. The variable '	"beepct" is set up with
	the number of T0 interrupt	s to accept, and is decremented	on each T0 interrupt.
	When it has been decrem	ented to zero (meaning that one s	econd has elapsed), pin
	P3 is reset to a constant z	ero to turn off the tone.	

4.1.4.2 Background Timer (T1) Task

The Timer T1 interrupt service routine represents a task that is not triggered directly by CPU commands. Its functions are to interrupt the CPU periodically for the Real-Time Clock function, and to present the !BUTTON-DATA interrupt whenever the pushbutton inputs change state.

Timer T1 is loaded with a constant interval value which is used to interrupt the HPC at 10 ms intervals. When the Timer T1 interrupt occurs (labels 'tmrint'', to "(11poll'', to "(11int''), then if the real-time interrupt is enabled, the variable "rtccnt" is decremented to determine whether an IRTC interrupt should be issued to the CPU. If so, the bit "artc" in the "alert" word is set, requesting the main program to issue the interrupt. The main program, at label "sndrtc", actually interrupts the CPU. No other data is passed to the CPU with the interrupt.

At label "kbdchk" the panel pushbutton switches are also sampled. If the pattern matches the last sample taken (saved in variable "swlast") then it is considered to be sta-

4.2 HPC Firmware Listing

NSC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987)

ble, and it is then compared to the last switch pattern sent to the CPU (in variable "swlsnt"). If the new pattern differs, then it is placed in "swlsnt", and the bit "abutton" in variable "alert" is set, requesting the main program to send a IBUTTON-DATA interrupt. The main program, at label "sndbth", triggers the interrupt and passes the new pattern to the CPU from variable "swlsnt".

4.1.4.3 Timer T6 Interrupt

Because the LCD controller's command acknowledgement capability was not used in our application, Timer T6 is used to time out the LCD controller's processing times. See the description of the SEND-LCD command above.

4.1.4.4 Timer T0 Interrupt

The interrupt service routine for Timer T0 (labels "tmrint", to "t0poll", to "t0int") is used simply to provide timing for the duration of the speaker tone. The interrupt is enabled in response to the BEEP command from the CPU, and is disabled on occurrence of the interrupt. It provides an interval of approximately one second.

25-Feb-88 10:05 PAGE 1

1 2 3 4 5 6 7 8 9 19 11 11 12 13 14 15 16 17 18 19 29	.title HPCUPI,'UPI PORT INTERFACE DEMO' Demo program for HPC46083 UPI Port: Demonstrates use of the HPC as an interface between an NS32C016 CPU and some typical front-panel types of devices: LED indicators (up to 8) Pushbuttons (up to 8) LCD alphanumeric display controller (Hitachi HD44780) Speaker for error beeps Also generates Real-Time Clock interrupts at a selectable rate. Generates !DIAG interrupt on errors; severity code of NOTE (e.g. real-time event lost), or FATAL (e.g. bad command). Recovery from fatal errors provided by RESET command.	
		TL/DD/9976-17

HPCUPT

PI PORT INTERFACE DEM eclarations: Registe			25-Feb-88 19: PAGE
21	.form	'Declarations: Register Addresses'	
22			
23 99C9 24 99C8	psw = al =	x'CØ:w ; PSW register x'C8:b ; Low byte of Accumulator.	
25 0009	ah =	x'C9:b ; High byte of Accumulator.	
26 ØØCC	bl ≠	x'CC:b ; Low byte of Register B.	
27 ØØCD 28 ØØCE	bh = xl =	x'CD:b ; High byte of Register B. x'CE:b ; Low byte of Register X.	
29 ØØCF	xh =	x'CF:b ; High byte of Register X.	
30			
31 0000 32 0002	enir = irpd =	x'DØ:b x'D2:b	
33 ØØD4	ircd =	x'04:b	
34 0006	sio =	x'D6:b	
35 ØØD8 36 ØØEØ	porti = obuf =	x'D8:b x'EØ:b ; (Low byte of PORTA.)	
37 ØØE1	portah =	x'E1:b ; High byte of PORTA.	
38 ØØE2	portb =	x'E2:w	
39 00E2 40 00E3	portbl = portbh =	x'E2:b ; Low byte of PORTB. x'E3:b ; High byte of PORTB.	
41 ØØE6	upic ≖	x'E6:b	
42 ØØFØ	ibuf =	x'FB:b ; (Low byte of DIRA.)	
43 00F1 44 00F2	dirah = dirb =	x'F1:b ; High byte of DIRA. x'F2:w	
45 ØØF2	dirbl =	x'F2:b ; Low byte of DIRB.	
46 ØØF3	dirbh =	x'F3:b ; High byte of DIRB.	
47 ØØF4 48 ØØF4	bfun = bfunl =	x'F4:w x'F4:b ; Low byte of BFUN.	
49 ØØF5	bfunh =	x'f5:b ; High byte of BFUN.	
50			
51 Ø1Ø4 52 Ø12Ø	portd = enu =	x'9194:b	
53 0122	enu = enui =	x'0120:b x'0122:b	
54 Ø124	rbuf =	x'9124:b	
55 0126	tbuf =	x'9126:b	
56 Ø128 57	enur =	x'9128:b	
58 Ø14Ø	t4 =	x'0140:w	
59 0142	r4 =	x'0142:w	
60 0144 61 0146	t5 = r5 =	x'9144;w x'9146;w	
62 9148	t6 =	x'0148:w	
63 Ø14A	r6 =	x'014A:w	
64 Ø14C 65 Ø14E	t7 = r7 =	x*Ø14C:w x*Ø14E:w	
66 9159	pwmode =	x · Ø142 : W x · Ø159 : W	
67 Ø15Ø	pwmdit =	x'Ø15Ø:b ; Low byte of PWMODE.	
68 9151	pwmdh =	x'0151:b ; High byte of PWMODE.	
69 Ø152 70 Ø152	portp = portpi =	x10152:w x10152:b ; Low byte of PORTP.	
		· · · · · · · · · · · · · · · · · · ·	
			TL/DD/9976
	taSite (Sep 14 14:30 198	37) HPCUPI	25-řeb-88 19:
PI PORT INTERFACE DE eclarations: Regist	MO		PAGE
eclarations: Regist	MO		
eclarations: Regist 71 Ø153	MO er Addresses portph =	x'9153:b ; High byte of PORTP.	
eclarations: Regist 71 Ø153 72 Ø15C	MO er Addresses	x'Ø153:b ; High byte of PORTP. x'Ø15C:b	
eclarations: Regist 71 Ø153	MO er Addresses portph =		
eclarations: Regist 71 0153 72 015C 73 74 0182 75 0184	HO er Addresses portph = eicon = t1 = r1 =	x'915C:b x'9182:w x'9184:w	
eclarations: Regist 71 Ø153 72 Ø15C 73 74 Ø182 75 Ø184 76 Ø186	M0 er Addresses portph = eicon = t1 = r1 = r2 =	x*915C:b x*9182:w x*9184:w x*9186:w	
eclarations: Regist 71 0153 72 015C 73 74 0182 75 0184	M0 er Addresses portph = eicon = t1 = r1 = r2 = t2 =	x*915C:b x*9182:w x*9184:w x*9186:w x*9186:w	
eclarations: Regist 71 Ø153 72 Ø15C 73 74 Ø182 75 Ø184 76 Ø186 77 Ø186 78 Ø186 78 Ø186 78 Ø186	MO er Addresses eicon = t1 = r1 = r2 = t2 = r3 = t3 =	x*915C:b x*9182:w x*9184:w x*9185:w x*9188:w x*9188:w x*9184:w x*9184:w	
eclarations: Regist 71 Ø153 72 Ø15C 73 74 Ø182 75 Ø184 76 Ø184 76 Ø188 77 Ø188 78 Ø184 79 Ø182 80 Ø18E	MO er Addresses portph = eicon = t1 = r1 = r2 = t2 = r3 = t3 = t3 =	x*915C:b x*9182:w x*9186:w x*9186:w x*9186:w x*9186:w x*9180:w x*9180:w	
eclarations: Regist 71 Ø153 72 Ø15C 73 74 Ø182 75 Ø184 76 Ø186 77 Ø186 77 Ø188 78 Ø184 79 Ø186 80 Ø18E 81 Ø18E	MO er Addresses portph = eicon = t1 = r2 = t2 = r3 = t3 = divby = divby =	x'915C:b x'9182:w x'9184:w x'9183:w x'9183:w x'9183:w x'918C:w x'918E:b ; Low byte of DIVBY.	
eclarations: Regist 71 0153 72 015C 73 74 0182 75 0184 76 0186 77 0188 77 0188 78 0184 79 0184 80 0185 81 0185 82 0185 83 0190	MO er Addresses portph = eicon = t1 = r2 = t2 = t3 = t3 = divbyl = divbyl = divbyl = divbyl =	x'\$15C:b x'\$182:w x'\$186:w x'\$188:w x'\$188:w x'\$182:c x'\$182:b x'\$182:b x'\$182:b x'\$187:b x'\$187:b x'\$187:b x'\$187:b x'\$198:w	
eclarations: Regist 71 0153 72 0150 73 0150 75 0184 75 0184 76 0188 78 0184 78 0188 81 0185 81 0185 82 0185 83 0190	MO er Addresses portph = eicon = r1 = r2 = t2 = t3 = divby = divbyt = divbyt = tmmode = tmmode =	x'\$15C:b x'\$182:w x'\$186:w x'\$186:w x'\$188:w x'\$182:w x'\$182:b x'\$18E:b x'\$18E:b x'\$18E:b x'\$18F:b x'\$18F:b x'\$198:b x'\$199:b x'\$197:b x'\$100:b x'\$10:	
eclarations: Regist 71 Ø153 72 Ø15C 73 75 Ø182 75 Ø184 76 Ø184 76 Ø184 77 Ø188 77 Ø188 78 Ø184 80 Ø186 81 Ø186 82 Ø18F 83 Ø190 84 Ø190 85 Ø191	MO er Addresses portph = eicon = t1 = r2 = t2 = t3 = t3 = t3 = divbyl = divbyl = divbyl = timmode = tmmcd =	x'\$15C:b x'\$182:w x'\$186:w x'\$186:w x'\$188:w x'\$182:w x'\$182:w x'\$182:b x'\$182:b x'\$182:b x'\$182:b x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$193:b x'\$10:b x'\$	
eclarations: Regist 71 Ø153 72 Ø15C 73 75 Ø182 75 Ø184 76 Ø184 76 Ø184 78 Ø184 79 Ø186 81 Ø186 81 Ø186 82 Ø187 83 Ø198 84 Ø199 85 Ø191 86 Ø192	MO er Addresses portph = eicon = r1 = r2 = t2 = t3 = divby = divbyt = divbyt = tmmode = tmmode =	x'\$15C:b x'\$182:w x'\$186:w x'\$186:w x'\$188:w x'\$182:w x'\$182:b x'\$18E:b x'\$18E:b x'\$18E:b x'\$18F:b x'\$18F:b x'\$198:b x'\$199:b x'\$197:b x'\$100:b x'\$10:	
eclarations: Regist 71 0153 72 015C 73 75 0184 75 0184 76 0186 77 0188 78 0184 78 0186 89 0186 81 0186 82 0186 83 0190 85 0191 85 0192	MO er Addresses portph = eicon = t1 = r2 = t2 = t3 = t3 = t3 = divbyl = divbyl = divbyl = timmode = tmmcd =	x'\$15C:b x'\$182:w x'\$186:w x'\$186:w x'\$188:w x'\$182:w x'\$182:w x'\$182:b x'\$182:b x'\$182:b x'\$182:b x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$193:b x'\$10:b x'\$	PAGE
eclarations: Regist 71 Ø153 72 Ø15C 73 75 Ø182 75 Ø184 76 Ø184 76 Ø184 78 Ø184 79 Ø186 81 Ø186 81 Ø186 82 Ø187 83 Ø198 84 Ø199 85 Ø191 86 Ø192	MO er Addresses portph = eicon = t1 = r2 = t2 = t3 = t3 = t3 = divbyl = divbyl = divbyl = timmode = tmmcd =	x'\$15C:b x'\$182:w x'\$186:w x'\$186:w x'\$188:w x'\$182:w x'\$182:w x'\$182:b x'\$182:b x'\$182:b x'\$182:b x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$193:b x'\$10:b x'\$	PAGE
eclarations: Regist 71 Ø153 72 Ø15C 73 75 Ø182 75 Ø184 76 Ø184 76 Ø184 78 Ø184 79 Ø186 81 Ø186 81 Ø186 82 Ø187 83 Ø198 84 Ø199 85 Ø191 86 Ø192	MO er Addresses portph = eicon = t1 = r2 = t2 = t3 = t3 = t3 = divbyl = divbyl = divbyl = timmode = tmmcd =	x'\$15C:b x'\$182:w x'\$186:w x'\$186:w x'\$188:w x'\$182:w x'\$182:w x'\$182:b x'\$182:b x'\$182:b x'\$182:b x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$193:b x'\$10:b x'\$	
eclarations: Regist 71 Ø153 72 Ø15C 73 75 Ø182 75 Ø184 76 Ø184 76 Ø184 78 Ø184 79 Ø186 81 Ø186 81 Ø186 82 Ø187 83 Ø198 84 Ø199 85 Ø191 86 Ø192	MO er Addresses portph = eicon = t1 = r2 = t2 = t3 = t3 = t3 = divbyl = divbyl = divbyl = timmode = tmmcd =	x'\$15C:b x'\$182:w x'\$186:w x'\$186:w x'\$188:w x'\$182:w x'\$182:w x'\$182:b x'\$182:b x'\$182:b x'\$182:b x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$193:b x'\$10:b x'\$	PAGE
eclarations: Regist 71 Ø153 72 Ø15C 73 75 Ø182 75 Ø184 76 Ø184 76 Ø184 78 Ø184 79 Ø186 81 Ø186 81 Ø186 82 Ø187 83 Ø198 84 Ø199 85 Ø191 86 Ø192	MO er Addresses portph = eicon = t1 = r2 = t2 = t3 = t3 = t3 = divbyl = divbyl = divbyl = timmode = tmmcd =	x'\$15C:b x'\$182:w x'\$186:w x'\$186:w x'\$188:w x'\$182:w x'\$182:w x'\$182:b x'\$182:b x'\$182:b x'\$182:b x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$193:b x'\$10:b x'\$	PAGE
eclarations: Regist 71 Ø153 72 Ø15C 73 75 Ø182 75 Ø184 76 Ø184 76 Ø184 78 Ø184 79 Ø186 81 Ø186 81 Ø186 82 Ø187 83 Ø198 84 Ø199 85 Ø191 86 Ø192	MO er Addresses portph = eicon = t1 = r2 = t2 = t3 = t3 = t3 = divbyl = divbyl = divbyl = timmode = tmmcd =	x'\$15C:b x'\$182:w x'\$186:w x'\$186:w x'\$188:w x'\$182:w x'\$182:w x'\$182:b x'\$182:b x'\$182:b x'\$182:b x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$193:b x'\$10:b x'\$	PAGE
eclarations: Regist 71 Ø153 72 Ø15C 73 75 Ø182 75 Ø184 76 Ø184 76 Ø184 78 Ø184 79 Ø186 81 Ø186 81 Ø186 82 Ø187 83 Ø198 84 Ø199 85 Ø191 86 Ø192	MO er Addresses portph = eicon = t1 = r2 = t2 = t3 = t3 = t3 = divbyl = divbyl = divbyl = timmode = tmmcd =	x'\$15C:b x'\$182:w x'\$186:w x'\$186:w x'\$188:w x'\$182:w x'\$182:w x'\$182:b x'\$182:b x'\$182:b x'\$182:b x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$193:b x'\$10:b x'\$	PAGE
eclarations: Regist 71 Ø153 72 Ø15C 73 75 Ø182 75 Ø184 76 Ø184 76 Ø184 78 Ø184 79 Ø186 81 Ø186 81 Ø186 82 Ø187 83 Ø198 84 Ø199 85 Ø191 86 Ø192	MO er Addresses portph = eicon = t1 = r2 = t2 = t3 = t3 = t3 = divbyl = divbyl = divbyl = timmode = tmmcd =	x'\$15C:b x'\$182:w x'\$186:w x'\$186:w x'\$188:w x'\$182:w x'\$182:w x'\$182:b x'\$182:b x'\$182:b x'\$182:b x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$193:b x'\$10:b x'\$	PAGE
eclarations: Regist 71 Ø153 72 Ø15C 73 75 Ø182 75 Ø184 76 Ø184 76 Ø184 78 Ø184 79 Ø186 81 Ø186 81 Ø186 82 Ø187 83 Ø198 84 Ø199 85 Ø191 86 Ø192	MO er Addresses portph = eicon = t1 = r2 = t2 = t3 = t3 = t3 = divbyl = divbyl = divbyl = timmode = tmmcd =	x'\$15C:b x'\$182:w x'\$186:w x'\$186:w x'\$188:w x'\$182:w x'\$182:w x'\$182:b x'\$182:b x'\$182:b x'\$182:b x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$193:b x'\$10:b x'\$	PAGE
eclarations: Regist 71 Ø153 72 Ø15C 73 75 Ø182 75 Ø184 76 Ø184 76 Ø184 78 Ø184 79 Ø186 81 Ø186 81 Ø186 82 Ø187 83 Ø198 84 Ø199 85 Ø191 86 Ø192	MO er Addresses portph = eicon = t1 = r2 = t2 = t3 = t3 = t3 = divbyl = divbyl = divbyl = timmode = tmmcd =	x'\$15C:b x'\$182:w x'\$186:w x'\$186:w x'\$188:w x'\$182:w x'\$182:w x'\$182:b x'\$182:b x'\$182:b x'\$182:b x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$193:b x'\$10:b x'\$	PAGE
eclarations: Regist 71 Ø153 72 Ø15C 73 75 Ø182 75 Ø184 76 Ø184 76 Ø184 78 Ø184 79 Ø186 81 Ø186 81 Ø186 82 Ø187 83 Ø198 84 Ø199 85 Ø191 86 Ø192	MO er Addresses portph = eicon = t1 = r2 = t2 = t3 = t3 = t3 = divbyl = divbyl = divbyl = timmode = tmmcd =	x'\$15C:b x'\$182:w x'\$186:w x'\$186:w x'\$188:w x'\$182:w x'\$182:w x'\$182:b x'\$182:b x'\$182:b x'\$182:b x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$193:b x'\$10:b x'\$	PAGE
eclarations: Regist 71 Ø153 72 Ø15C 73 75 Ø182 75 Ø184 76 Ø184 76 Ø184 78 Ø184 79 Ø186 81 Ø186 81 Ø186 82 Ø187 83 Ø198 84 Ø199 85 Ø191 86 Ø192	MO er Addresses portph = eicon = t1 = r2 = t2 = t3 = t3 = t3 = divbyl = divbyl = divbyl = timmode = tmmcd =	x'\$15C:b x'\$182:w x'\$186:w x'\$186:w x'\$188:w x'\$182:w x'\$182:w x'\$182:b x'\$182:b x'\$182:b x'\$182:b x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$193:b x'\$10:b x'\$	PAGE
eclarations: Regist 71 Ø153 72 Ø15C 73 75 Ø182 75 Ø184 76 Ø184 76 Ø184 78 Ø184 79 Ø186 81 Ø186 81 Ø186 82 Ø187 83 Ø198 84 Ø199 85 Ø191 86 Ø192	MO er Addresses portph = eicon = t1 = r2 = t2 = t3 = t3 = t3 = divbyl = divbyl = divbyl = timmode = tmmcd =	x'\$15C:b x'\$182:w x'\$186:w x'\$186:w x'\$188:w x'\$182:w x'\$182:w x'\$182:b x'\$182:b x'\$182:b x'\$182:b x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$193:b x'\$10:b x'\$	PAGE
eclarations: Regist 71 Ø153 72 Ø15C 73 75 Ø182 75 Ø184 76 Ø184 76 Ø184 78 Ø184 79 Ø186 81 Ø186 81 Ø186 82 Ø187 83 Ø198 84 Ø199 85 Ø191 86 Ø192	MO er Addresses portph = eicon = t1 = r2 = t2 = t3 = t3 = t3 = divbyl = divbyl = divbyl = timmode = tmmcd =	x'\$15C:b x'\$182:w x'\$186:w x'\$186:w x'\$188:w x'\$182:w x'\$182:w x'\$182:b x'\$182:b x'\$182:b x'\$182:b x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$193:b x'\$10:b x'\$	PAGE
clarations: Regist 71 0153 72 015C 73 75 0182 75 0184 76 0186 77 0188 78 0184 78 0186 80 0186 81 0186 81 0186 83 0190 85 0191 85 0191 86 0192 87	MO er Addresses portph = eicon = t1 = r2 = t2 = t3 = t3 = t3 = divbyl = divbyl = divbyl = timmode = tmmcd =	x'\$15C:b x'\$182:w x'\$186:w x'\$186:w x'\$188:w x'\$182:w x'\$182:w x'\$182:b x'\$182:b x'\$182:b x'\$182:b x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$192:w x'\$193:b x'\$10:b x'\$	PAGE

SC ASMHPC, Ver D1-Betas PI PORT INTERFACE DEMO eclarations: Register		HPCUP1	25-řeb-88 19:9 PAGE
89		'Declarations: Register Bit Positions'	
90 91	• Nome Do	sition Perister(s)	
92	Name Pos	sition Register(s)	
93	,		
94 0000	gie =	Ø ; enir	
95 0002	i2 =	2 ; enir, irpd, ircd	
96 0003	i3 =	3 ; enir, irpd, ircd	
97 8004	i4 =	4 ; enir, irpd, ircd	
98 9995	tmrs =	5 ; enir, irpd	
99 0006 100 0007	uart = ei =	o ; enir, irpd	
1, 01	ei =	7 ; enir, irpd	
102 0001	uwmode =	1 ; ircd	
103 0000	uwdone =	Ø ; irpd	
104		, ,	
105 0000	tbmt =	Ø ; enu	
106 0001	rbfl =	1 ; enu	
107 0004	b8or9 =	4 ; enu	
108 0005	xbit9 =	5 ; enu	
109 0002	wakeup ≠	2 ; enur	
110 0003	rbit9 =	3 ; enur	
111 0006	frmerr =	6 ; enur	
112 0007	doeerr = eti =	7 ; enur	
113 0000 114 0001	eti = eri =	9 ; enui 1 ; enui	
114 0001 115 0002	eri = xtclk =	1 ; enui 2 ; enui	
116 0003	xrclk =	3; enui	
117 0007	b2stp =	3 ; enui 7 ; enui	
118		,	
119 0000	wrrdy =	Ø ;upic	
120 0001	rdrdy =	1 ; upic	
121 0002	laø =	2 ; upic	
122 0003	upien =	3 ; upic	
123 0004	b8or16 =	4 ; upic	
124			
125 0000	tøtie =	Ø ; tmmdl	
126 0001	tøpnd =	1 ; tmmdl	
127 0003	tØack =	3 ; tmmdl	
128 0004	t1tie =	4 ; tmmdl	
129 0005	t1pnd =	5 tmmdl	
130 0006	t1stp =	6 ; tmmdl	
131 0007	t1ack =	7 ; tmmdl	
132 0000	t2tie =	Ø; tmmdh	
133 0001	t2pnd =	1 ; tmmdh	
134 0002	t2stp =	2 ; tmmdh	
135 0003	t2ack =	3 ; tmmdh	
136 0004	t3tie =	4 ; tmmdh	
137 0005 138 0006	t3pnd = t3stp =	5 trinndh 6 tinnndh	
130 0000	ussip =	6 ; tmmdh	
			TL/DD/9976-
			12/20/99/0-

UPI PORT INTERFACE DEM		1987)	HPCUP1	25-Feb-88 10:0 PAGE
Declarations: Registe	t3ack =	7	; tmmch	
140	LJACK -	'	; cmuch	
141 0000	t4tie =	ø	; pwmdl	
142 0001	t4pnd =	1	; pwmdl	
143 0002 144 0003	t4stp = t4ack =	23	; pwmdl ; pwmdl	
145 0004	t5tie =	4	; pumul	
146 0005	t5pnd =	5	; pwmdl	
147 0006	t5stp =	6	; pwmdl	
148 0007 149 0000	t5ack = t6tie =	7 Ø	; pwmdl ; pwmdh	
150 0001	tópnd =	1	; pwmdh	
151 0002	tőstp =	23	; pwmdh	
152 0003	t6ack =	3	; pwmdh	
153 0004 154 0005	t7tie = t7pnd =	4 5	; pwmdh ; pwmdh	
155 0006	t7stp =	6	; pwmdh ; pwmdh	
156 0007	t7ack =	7	pwmdh	
157	A (•		
158 ØØØØ 159 ØØØ3	t4out = t4tfn =	Ø 3	; portpl ; portpl	
160 0004	tSout =	4	; portpl	
161 0007	t5tfn =	7	; portpl	
162 0000	t6out =	9	; portph	
163 0003 164 0004	t6tfn = t7out =	3 4	; portph	
165 0007	t7tfn =	7	; portph ; portph	
166				
167 0000	eipol =	9	; eicon	
168 ØØØ1 169 ØØØ2	eimode = eiack =	1	; eicon	
170	erack -	2	; eicon	
171 0005	so =	5	; portbl, dirbl, bfunl	
172 0006	sk =	6	; portbl, dirbl, bfunl ; portbl, dirbl, bfunl ; portbl, dirbl	
173 ØØØ7 174	pnlclk =	7	; portbl, dirbl	
175 0001	lcvclk =	1	; portbh, dirbh	
176	; ua@ would	lbe 2	, but requires no setup.	
177 0003	uwrrdy =	3	; portbh, dirbh, bfunh	
178 0004	cdata = astts =	4	; portbh (enables non-pushbutton data to Port	: D).
179 0005 180 0006	ledclk =	5	; portbh (enables pushbutton data to Port D). ; portbh, dirbh	
181 0007	urdrdy =	7	; portbh, dirbh, bfunh	
182				
183	; CO	ISTANTS		
184 185 ØØ11	xon= x*	1 • 10	I character: Control-Q	
186 0013	xoff= x'	3 ; xo	f character: Control-S	
187				
188				
				TL/DD/9976-

pace Declarations			
189 190 0000	.form .sect	'Space Declarations' DSECT,BASE,REL ; Basepage RAM variables (addresses ØØØØ-ØØBF)	
191 192	; WORD-ALIGNED		
193 0000	dummy: .dsw 1	; x'00,01 ; Destroyed on reset (address 0). upicsv,dummy ; Temporary image of UPIC register.	
194 0000 195 0002	.set	; Alert status bits to main program:	
196		; generate interrupts to CPU.	
197 0003 198 0004	.set	alerth,alert+1:b ; Declare top byte of ALERT word.	
199 0006	cpubuf: .dsw 4	; Current address within CPU command buffer. ; Buffer for accepting command parameters from CPU.	
200 000E 201	lcdsix: .dsw 1	; Pointer into LCD character string buffer.	
202	;BYTE-ALIGNED		
203 0010 204 0011	curcmd: .dsb 1 numexp: .dsb 1	; Current command byte from CPU being processed. ; Number of parameter bytes expected before command processing	
205		; begins.	
206 0012 207	lcvs: .dsb 1	; Image of LCD Voltage (Contrast) latch setting; needed with	
208 0013	lcdfgs: .dsb 1	; LCD RS (PAUXØ) signal coming from this latch. ; Holds flag bits for characters sent to Panel LCD display.	
209 0014 210 0015	lodefar deb 1	; Number of characters to be sent to LCD display. ; Flag bits associated with characters in LCD String Buffer.	
211 0016	lcdsct: .dsb 1	; Counter for characters being sent to LCD display from String	
212 213 0017		; Buffer.	
214 0018	swisnt: .dsb 1	; Last-sampled switch values. ; Last switch values sent to CPU.	
215 ØØ19 216 ØØ1A	beepct: .dsb 1	; Beep duration count. Counts occurrences of TØ interrupt.	
217 ØØ1B	rtccnt: .dsb 1	; Real-Time Clock Interval (units of 10 milliseconds). ; Real-Time Clock Current Count (units of 10 milliseconds).	
218 001C 219 001D	rtevs: .dsb 1 dsevc: .dsb 1	; Events to check for on Timer I1 interrupts.	
220 001E	derrc: .dsb 1	; Diagnostic Interrupt: Severity Code. ; Diagnostic Interrupt: Error Code.	
221 001F 222 0020		; Diagnostic Interrupt: Error Byte.	
223 0021		; Diagnostic Interrupt: Current Command. ; Diagnostic Interrupt: Qualifier (Command Status).	
224		· · · · · · · · · · · · · · · · · · ·	
225 226	; BIT POS	ITIONS	
227			
228 229	. AIFRT	status word (low-order byte) bits:	
230			
231 0000 232 0001	abutton = artc =	9 ; Pushbutton switch state change. 1 ; Real-Time Interrupt detected.	
233 0002	adiag =	2 ; Diagnostic interrupt.	
234 0003 235	alcdak = ; (Other bits n	3 ; LCD Panel Write Acknowledge.	
236	-		
237 238	; ALERT	status word (high-order byte, named alerth) bits:	
			TL/DD/9976
	atita (tan 1/ 1/ 70 1007)		75 F.L 00 40
SC ASMHPC, Ver DI-Bet PI PORT INTERFACE DEM pace Declarations	aSite (Sep 14 14:30 1987) O) HPCUPI	25-Feb-88 10:05 PAGE
239	; (Other bits r	not defined.)	
240			
241 242	; CURCH	10 byte: Current CPU command. The lower 5 bits contain the	
243	1	command code. The upper two bits contain	
244 245 ØØØ7	cmdemp= 7	further information about command collection: ; Bit 7 (MSB) of curcmd = 1 means that no command is being	
246		; processed and curcmd byte is "empty".	
247 ØØØ6 248	getcnt= 6	Bit 6 of curcmd = 1 means that the count is being received for a variable-length command.	
249			
250 251	; LCVS	byte: LCD Voltage (Contrast) Latch memory image. Contains voltage value in its least-significant 3 bits,	
252	;	RS signal to LCD controller in bit 3, and debugging	
253 254 ØØØ3	; pnlrs= 3	information in its top 4 bits.	
255	pricto- 0	; Bit 3 is (inverted) RS signal to panel.	
256 257	. PTEN	byte: Events to check for at 10-millisecond intervals.	
258		(T1 Underflows)	
250	rtcenb= 🖗	; 1 = Real-Time Clock interrupts enabled to CPU.	
259 0000			
259 ØØØØ 260 261		STACK,RAM16,REL ; On-chip RAM in addresses Ø1CØ-Ø1FF. 16 ; Space for 8 words beyond	
259 ØØØØ 260 261 262 ØØØØ	.sect stackb: .dsw		
259 8888 268 261 262 8888 263 8888 263 8888 264	stackb: .dsw	; interrupt context.	
259 0000 260 261 262 0000 263 0000 264 265 0020	stackb: .dsw avail: .dsw	; interrupt context. 12 ; Spare portion of this space.	
259 8888 268 261 262 8888 263 8888 263 8888 264	stackb: .dsw	; interrupt context.	
259 0000 260 261 262 0000 263 0000 264 265 0020 266 0038	stackb: .dsw avail: .dsw	; interrupt context. 12 ; Spare portion of this space.	TL/DD/9976
259 0000 260 261 262 0000 263 0000 264 265 0020 266 0038	stackb: .dsw avail: .dsw	; interrupt context. 12 ; Spare portion of this space.	TL/DD/9976
259 0000 260 261 262 0000 263 0000 264 265 0020 266 0038	stackb: .dsw avail: .dsw	; interrupt context. 12 ; Spare portion of this space.	TL/DD/9976

72 73 0000						
71 72 73 0000			.form .sect	'Code Section' CSECT,ROM16,REL	; Code space. (On-chip ROM)	
73 0000			; Decla	arations of subro	utines called by one-byte JSRP instruction.	
			.spt	rdwait	; Waits for CPU to read a value from UPI port.	
74 0000 75			.spt	wrpnl	; Writes to LCD panel (for initialization only).	
76 77			; Progi ; erro	ram starts at lab or handler, locat	el "start" on reset. This routine is the fatal ad here for convenience in setting breakpoint.	
78 79 0000 960018		hangup:		gie,enir	; Fatal error: signal it and halt.	
80 0003 961201 81 82 0001 011201			sbit	7,lcvs	; Signal error on most-significant bit of ; LCD Contrast Latch.	
82 0006 961208 83 0009 8c12e	R		sbit ld	pnlrs,lcvs portah,lcvs	; Select command mode for LCD controller. ; Place error on Port A for latch.	
84 ØØØC 96E3Ø9 85 ØØØF 96E319			sbit rbit	lcvclk,portbh lcvclk,portbh	; Clock LCD Contrast Latch high, ; then low to load it.	
86 ØØ12 B6Ø151	ØA		sbit	t6stp,pwmdh	;	
87 0016 860151 88 001a 40	10		rbit nop	tótie,pwm.dh	; Set up Timer T6 for non-interrupt use.	
89 ØØ18 B6Ø151	19		rbit	t6pnd,pwmdh	; Clear Pending bit.	
90 001F 3F00 91 0021 870000	C4 R		pop ld	Ø.w sp,#stackb	; Get error address from stack. ; In case of stack underflow, re-initialize SP.	
92 0025 9001 93 0027 2E	R		ld	A,#x*91 wrpnl	; Clear LCD panel.	
94 0028 961210	R		jsrl rbit	phirs, lovs	; Set up panel for data.	
95 0028 8C12E 96 002E 96E309			ld sbit	portah,lcvs lcvclk,portbh	; Place error on Port A for latch. ; Clock LCD Contrast Latch high,	
97 ØØ31 96E319			rbit	lcvclk,portbh	; then low to load it.	
98 ØØ34 88Ø1 99 ØØ36 38			ld	A,1.b	; Process first character of return address.	
ØØ ØØ37 99ØF			swap and	A,#x'ØF		
01 0039 A6007/ 02 003E 2E	.C888 R R		ld jsrl	A,hextab[A].b wrpnl	; Display it on LCD panel.	
Ø3 ØØ3F 88Ø1	ĸ		ĺd	A,1.b	; Process second character of return address.	
04 0041 990F 05 0043 A6007/	C888 R		and ld	A,#x'ØF A,hextab[A].b		
Ø6 ØØ48 2E	R		jsrl	wrpnl	; Display it on LCD panel.	
07 0049 8800 08 0048 38			ld swap	A,Ø.b A	; Process third character of return address.	
Ø9 ØØ4C 99ØF			and	A,#x'ØF		
10 004E A6007A 11 0053 2E	C888 R R		ld jsrl	A,hextab[A].b wrpnl	; Display it on LCD panel.	
12 0054 8800			ld	A,Ø.b	; Process last character of return address.	
13 0056 990F 14 0058 A6007A			and ld	A,#x'ØF A,hextab[A].b		
15 ØØ5D 2E 16	R		jsrl	wrpnl	; Display it on LCD panel.	
17 ØØ5E 96E611		hgupi:	ifbit	rdrdy,upic	; Check to see if OBUF register is full.	
						TL/DD/997

de Se	T IŃI	ERFACE DEMO	e (Sep 14 14:	30 1987;) Н	PCUPI	25-feb-88 1Ø:Ø PAGE
318	0061	971DEØ		lď	obuf,#vdiag	; If not, fill it with !DIAG vector	
319						; continuously.	
	0067 0067	96D213 41		ifbit jp	i3,irpd hgupi1	; Check for UPI data ready.	
322				jp	hgupi		
323	aako	82A5FØDC	hgupi1:	ifea	ibuf,#x'A5	; Check for RESET command.	
325	006D	41		jp	hgrst	, check for Reper connorme	
	996E	47 96E612	hgrst:	jp ifbit	hgupi2 laØ,upic		
328	ØØ72	43	ligi st.	jp	hgupi2		
329 (330	ØØ73	B4927A		jmpl	xreset	; If so, then go reset the HPC.	
331						; This is part of the outer loop, waiting for	
332	0074	97F7D2	hgupi2:	1.4	irpd,#x'F7	; the RESET command.	
334			ngopiz.	jp	hgupi	; Clear the UWR detector, ; and keep looking. This is an	
335 336						; infinite loop until RESET is seen.	
337	007A	3Ø	hextab:	.byte	ני,י2י,י1י,יפי	','4','5','6','7'	
	007B 007C						
6	9970	33					
	007E 007F						
	9989						
	9981			hu ch a	191 101 141 15		
338	0082 0083			.byte	·o·,·y·,·A·,·B	','C','D','E','F'	
6	0084	41					
	9985 9986						
ŝ	0087	44					
	0088 0089						
339							
							TL/DD/9976
POR	T IÑ	Ver D1-BetaSit TERFACE DEMO	e (Sep 14 14:	30 1987) 	IPCUPI	25-Feb-88 10: PAGE
rowar	e in	itialization					
34Ø 341				.form	'Hardware Init	tialization'	
341 342	ØØ8A	97 9 8CØ	start:		'Hardware Init psw.b,#x'08	tialization' ; Set one WAIT state.	
341 342 343		9798c9	start: srfsh:		psw.b,#x'Ø8	; Set one WAIT state.	
341 342 343 344 345	ØØ8D			ld	psw.b,#x'98 ; Star ; as	; Set one WAIT state. *t dynamic RAM refreshing, quickly as possible.	
341 342 343 344 345	ØØ8D	9708c0 86015208			psw.b,#x'Ø8 ; Star	; Set one WAIT state. t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh	
341 342 343 344 345 346 347 348	ØØ8d ØØ8d			ld	psw.b,#x'98 ; Star ; as	; Set one WAIT state. t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; immediately. ; Stop timer T4 to	
341 342 343 344 345 346 346 347 348 349	ØØ8d ØØ8d ØØ91	B6Ø152Ø8		ld sbit	psw.b,#x'Ø8 ; Star ; as t4out,portpl	; Set one WAIT state. •t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; immediately.	
341 342 343 344 345 346 347 348 349 350 351	ØØ8d ØØ8d ØØ91 ØØ95 ØØ9a	B6Ø152Ø8 B6Ø15ØØA 83Ø8Ø14ØAB B6Ø15Ø1A		ld sbit sbit Id rbit	psw.b,#x'Ø8 ; Star ; as t4out,portpl t4stp,pwmdl t4.w,#8 t4stp,pwmdl	; Set one WAIT state. t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; immediately. ; Stop timer T4 to ; allow loading, ; then load it. ; Start timer T4.	
341 342 343 344 345 346 347 348 349 350 351 352	0080 0080 0091 0095 009A 009E	86015208 8601500A 83080140AB 8601501A 8601501A 8601520B		ld sbit sbit Id	psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4.w,#8 t4stp,pwmdl t4tfn,portpl	; Set one WAIT state. :t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; immediately. ; Stop timer T4 to ; allow loading, ; then load it. ; Start timer T4. ; Enable pulses out.	
341 342 343 344 345 346 347 348 349 350 351 352 353 354	0080 0091 0091 0095 009A 009E 00A2	B6Ø152Ø8 B6Ø15ØØA 83Ø8Ø14ØAB B6Ø15Ø1A	srfsh:	ld sbit sbit Id rbit sbit	psw.b,#x'Ø8 ; Star ; as t4out,portpl t4stp,pwmdl t4.w,#8 t4stp,pwmdl	; Set one WAIT state. :t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; immediately. ; Stop timer T4 to ; allow loading, ; then load it. ; Start timer T4. ; Enable pulses out. ; Load R4.	
341 342 343 344 345 346 347 348 347 350 351 352 353 354 355	0080 0080 0091 0095 0094 0095 0094 0095 0095 0095 0095	86915298 8691599A 83989149A8 8691591A 86915298 83989142A8		ld sbit sbit ld rbit sbit ld	psw.b,#x'98 ; star ; as t4out,portpl t4stp,pwmdl t4.w,#8 t4stp,pwmdl t4tfn,portpl r4.w,#8	; Set one WAIT state. t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; immediately. ; Stop timer T4 to ; allow loading, ; then load it. ; Start timer T4. ; Enable pulses out. ; Load R4. ; Set up UP1 port.	
341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 355 356 357	0080 0080 0091 0095 0094 0095 0094 0095 0095 0095 0095	86015208 8601500A 83080140AB 8601501A 8601501A 8601520B	srfsh:	ld sbit sbit Id rbit sbit	psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4.w,#8 t4stp,pwmdl t4tfn,portpl	; Set one WAIT state. :t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; immediately. ; Stop timer T4 to ; allow loading, ; then load it. ; Start timer T4. ; Enable pulses out. ; Load R4.	
341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 355 356 357 358	0080 0080 0091 0095 0094 0095 0094 0095 0042 0047 0047	B6Ø152Ø8 B6Ø15ØØA 83Ø8Ø14ØAB B6Ø15Ø1A B6Ø152Ø8 83Ø8Ø142AB 9718E6	srfsh:	ld sbit sbit ld rbit sbit ld	psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4stp,pwmdl t4stp,pwmdl t4stp,pwmdl t4tfn,portpl r4.w,#8 upic,#x'18	; Set one WAIT state. t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; inmediately. ; Stop timer T4 to ; allow loading, ; then load it. ; Start timer T4. ; Enable pulses out. ; Load R4. ; Set up UPI port. ; 8-Bit UPI Mode ; enabled.	
341 342 343 344 345 346 347 348 350 351 352 353 354 355 355 355 355 355 355 357 358 359 360	0080 0091 0095 0094 0095 0094 0095 0094 0097 0047 0047	86015208 8601500A 83080140A8 8601501A 86015208 83080142A8 9718E6 96F508 96F508	srfsh:	ld sbit sbit ld rbit sbit ld ld sbit	psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4stp,pwmdl t4stp,pwmdl t4tfn,portpl r4.w,#8 upic,#x'18 uwrrdy,bfumh uwrrdy,dirbh	; Set one WAIT state. t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; inmediately. ; Stop timer T4 to ; allow loading, ; then load it. ; Start timer T4. ; Enable pulses out. ; Load R4. ; Set up UPI port. ; 8-Bit UPI Mode ; enabled. ; Enable UWRRDY/ out.	
341 342 343 344 345 346 347 348 359 351 352 353 354 355 355 355 355 355 356 357 358 359 369 361	0080 0091 0095 0094 0095 0094 0095 0094 0097 0047 0047	86015208 8601500A 83080140A8 8601550A 86015208 83080142A8 9718E6 965508	srfsh:	ld sbit sbit ld rbit sbit ld ld	psw.b,#x'98 ; star ; as t4out,portpl t4stp,pwmdl t4.w,#8 t4stp,portpl r4.w,#8 upic,#x'18 uwrrdy,bfumh	<pre>; Set one WAIT state. t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; inmediately. Stop timer T4 to ; allow loading, ; then load it. ; Enable pulses out. ; Load R4. ; Set up UPI port. ; 8-Bit UPI Mode ; enabled. ; Enable UWRRDY/ out. ; Empty 1BUF register,</pre>	
341 342 344 344 345 346 347 348 359 351 352 353 354 355 355 355 355 355 355 355 355	0080 0091 0095 0094 0095 0094 0095 0094 0094 0094	86015208 8601500A 83080140A8 8601501A 86015208 83080142A8 9718E6 9718E6 96F508 88F0	srfsh:	ld sbit sbit ld ld sbit sbit ld	psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4.w,#8 t4stp,pwmdl t4tfn,portpl r4.w,#8 upic,#x'18 uwrrdy,bfunh uwrrdy,bfunh uwrrdy,bfunh	; Set one WAIT state. t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; inmediately. Stop timer T4 to allow loading, ; then load it. ; Start timer T4. ; Enable pulses out. ; Load R4. ; Set up UPI port. ; 8-Bit UPI Mode ; enabled. ; Enable UWRRDY/ out. ; Empty 18UF register, ; in case of false trigger.	
341 342 343 344 345 346 348 349 350 351 352 353 354 355 355 355 355 355 355 355 355	8080 0091 0091 0094 0094 0094 0094 0094 009	B6015208 B601500A 83080140AB B601501A B6015208 83080142AB 9718E6 96F508 88F0 96F50F	srfsh:	ld sbit ld rbit sbit ld sbit sbit ld sbit	psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4stp,pwmdl t4stp,portpl r4.w,#8 upic,#x'18 uwrrdy,bfunh uwrrdy,dirbh A,ibuf urdry,bfunh	<pre>; Set one WAIT state. t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; inmediately. Stop timer T4 to ; allow loading, ; then load it. ; Enable pulses out. ; Load R4. ; Set up UPI port. ; 8-Bit UPI Mode ; enabled. ; Enable UWRRDY/ out. ; Empty 1BUF register,</pre>	
341 342 343 344 345 346 357 350 351 352 353 354 355 355 355 355 355 355 355 355	8080 0091 0091 0094 0094 0094 0094 0094 009	86015208 8601500A 83080140A8 8601501A 86015208 83080142A8 9718E6 9718E6 96F508 88F0	srfsh:	ld sbit sbit ld ld sbit sbit ld	psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4.w,#8 t4stp,pwmdl t4tfn,portpl r4.w,#8 upic,#x'18 uwrrdy,bfunh uwrrdy,bfunh uwrrdy,bfunh	; Set one WAIT state. :t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; immediately. Stop timer T4 to ; allow loading, ; then load it. ; Start timer T4. ; Enable pulses out. ; Load R4. ; Set up UPI port. ; 8-Bit UPI Mode ; enabled. ; Enable UWRRDY/ out. ; Enable UWRRDY/ out. ; Enable URDRDY/ out.	
341 342 343 344 345 346 350 351 352 353 354 355 354 355 354 355 356 357 358 357 358 357 358 357 358 356 357 358 364 365 364 365 364 365 364 365 366 367 367 358 367 357 358 357 357 358 357 357 358 357 357 358 357 357 357 357 357 357 357 357 357 357	8080 0080 0091 0094 0094 0094 0094 0094 0044 004	B6Ø152Ø8 B6Ø15ØØA 83Ø8Ø14ØAB B6Ø152Ø8 83Ø8Ø142AB 9718E6 96F5ØB 88FØ 96F5ØF 96F5ØF	srfsh:	ld sbit ld rbit sbit ld ld sbit ld sbit sbit sbit	psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4.w,#8 t4stp,pwmdl t4fn,portpl r4.w,#8 upic,#x'18 uwrrdy,bfunh A,ibuf urrdy,bfunh urdrdy,bfunh	<pre>; Set one WAIT state. t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; inmediately. ; Stop timer T4 to allou loading, ; then load it. ; Start timer T4. ; Enable pulses out. ; Load R4. ; Set up UPI port. ; 8-Bit UPI Mode ; enabled. ; Enable UWRRDY/ out. ; Empty IBUF register, ; in case of false trigger. ; Enable URDRDY/ out. ; Set up UREAD/ interrupt.</pre>	
341 342 343 344 345 346 347 358 359 359 355 355 355 355 355 355 355 355	8980 9980 9991 9994 9994 9994 9994 9994 9994 999	B6015208 B601500A 83080140AB B601501A B6015208 83080142AB 9718E6 96F508 88F0 96F50F	srfsh:	ld sbit ld rbit sbit ld sbit sbit ld sbit	psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4.w,#8 t4stp,pwmdl t4tfn,portpl r4.w,#8 upic,#x'18 uwrrdy,bfunh uwrrdy,bfunh uwrdy,bfunh urdrdy,bfunh i2, ircd	; Set one WAIT state. :t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; immediately. Stop timer T4 to ; allow loading, ; then load it. ; Start timer T4. ; Enable pulses out. ; Load R4. ; Set up UPI port. ; 8-Bit UPI Mode ; enabled. ; Enable UWRRDY/ out. ; Enable UWRRDY/ out. ; Enable URDRDY/ out.	
341 342 343 344 345 346 351 352 353 355 355 355 355 355 355 355 355	8980 9980 9991 9994 9994 9994 9994 9994 9994 999	B6015208 B601500A 83080140AB B601501A B601520B 83080142AB 9718E6 961500B 961500B 88F0 96530B 88F0 96550F 96550F	srfsh:	ld sbit sbit ld rbit sbit ld sbit sbit sbit sbit	psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4.w,#8 t4stp,pwmdl t4fn,portpl r4.w,#8 upic,#x'18 uwrrdy,bfunh A,ibuf urrdy,bfunh urdrdy,bfunh	<pre>; Set one WAIT state. :t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; inmediately. Stop timer T4 to ; allow loading, ; then load it. ; Enable pulses out. ; Load R4. ; Set up UPI port. ; 8-Bit UPI Mode ; enabled. ; Enable UWRRDY/ out. ; Enable UWRRDY/ out. ; Enable URDRDY/ out. ; Set up UREAD/ interrupt. ; Detects rising edges.</pre>	
341 342 343 344 345 346 347 350 351 352 353 354 355 355 355 355 355 355 355 355	8980 9980 9991 9994 9994 9994 9994 9994 9994 999	86015208 8601500A 83080140A8 8601501A 8601501A 86015208 83080142A8 9718E6 9718E6 96F508 96F508 96F508 96F50F 96F50F 96F50F 96F50F	srfsh:	ld sbit sbit ld sbit sbit ld sbit sbit sbit sbit ld	psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4.w,#8 t4stp,pwmdl t4tfn,portpl r4.w,#8 upic,#x'18 uwrrdy,dirbh A,ibuf urdrdy,bfunh urdrdy,bfunh urdrdy,dirbh i2,ircd inpd,#x'FB	<pre>; Set one WAIT state. t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; inmediately. Stop timer T4 to ; allow loading, ; then load it. ; Enable pulses out. ; Load R4. ; Set up UPI port. 8-Bit UPI Mode ; enabled. ; Enable UWRRDY/ out. ; Enable UWRRDY/ out. ; Enable UWRRDY/ out. ; Enable URDRDY/ out. ; Set up UREAD/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRITE/ interrupt.</pre>	
341 342 343 344 345 346 347 350 353 353 353 353 353 354 355 355 355 355	00000000000000000000000000000000000000	86015208 8601500A 83080140A8 86015208 83080142A8 9718E6 96F508 88F0 96F50F 96F50F 96F50F 96F50F 96F50F	srfsh:	ld sbit sbit ld sbit ld sbit sbit ld sbit sbit sbit sbit	<pre>psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4stp,pwmdl t4stp,pwmdl t4tfn,portpl r4.w,#8 upic,#x'18 uwrrdy,dirbh A,ibuf urdrdy,dirbh i2,ircd irpd,#x'FB i3,ircd</pre>	<pre>; Set one WAIT state. t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; inmediately. ; Stop timer T4 to ; allow loading, ; then load it. ; Start timer T4. ; Enable pulses out. ; Load R4. ; Set up UPI port. ; 8-Bit UPI Mode ; enabled. ; Enable UWRRDY/ out. ; Enable UWRRDY/ out. ; Enable UWRRDY/ out. ; Enable URDRDY/ out. ; Set up UREAD/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRIE/ interrupt. ; Detects rising edges.</pre>	
341 342 343 344 345 346 347 350 351 352 354 359 355 356 355 356 357 358 359 361 363 364 365 366 367 368 366 367 371 372 373 372 373 373	00000000000000000000000000000000000000	86015208 8601500A 83080140A8 8601501A 8601501A 86015208 83080142A8 9718E6 9718E6 96F508 96F508 96F508 96F50F 96F50F 96F50F 96F50F	srfsh:	ld sbit sbit ld sbit sbit ld sbit sbit sbit sbit ld	psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4.w,#8 t4stp,pwmdl t4tfn,portpl r4.w,#8 upic,#x'18 uwrrdy,dirbh A,ibuf urdrdy,bfunh urdrdy,bfunh urdrdy,dirbh i2,ircd inpd,#x'FB	<pre>; Set one WAIT state. t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; inmediately. Stop timer T4 to ; allow loading, ; then load it. ; Enable pulses out. ; Load R4. ; Set up UPI port. 8-Bit UPI Mode ; enabled. ; Enable UWRRDY/ out. ; Enable UWRRDY/ out. ; Enable UWRRDY/ out. ; Enable URDRDY/ out. ; Set up UREAD/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRITE/ interrupt.</pre>	
341 342 343 344 345 346 347 359 351 352 354 355 355 355 355 355 355 355 355 355	00000000000000000000000000000000000000	86015208 8601500A 83080140A8 86015208 83080142A8 9718E6 96F508 88F0 96F50F 96F50F 96F50F 96F50F 96F50F	srfsh: supi:	ld sbit sbit ld sbit ld sbit sbit ld sbit sbit sbit sbit	psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4.u,#8 t4stp,pwmdl t4tfn,portpl r4.w,#8 upic,#x'18 uwrrdy,dirbh A,ibuf urdrdy,dirbh i2,ircd irpd,#x'FB i3,ircd irpd,#x'F7	<pre>; Set one WAIT state. t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; inmediately. ; Stop timer T4 to allow loading, ; then load it. ; Start timer T4. ; Start timer T4. ; Start timer T4. ; Set up UPI port. ; 8-Bit UPI Mode ; enabled UWRRDY/ out. ; Empty 1BUF register, ; in case of false trigger. ; Enable UWRRDY/ out. ; Set up UREAD/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Clear any false interrupt ; Detects rising interrupt ; Detects rising edges. ; Clear any false interrupt ; Clear any false interrupt ; due to mode change.</pre>	
3411 342 343 344 345 345 345 347 348 359 351 352 353 355 355 355 355 355 355 355 355	8980 9980 9991 9094 9094 9094 9094 9094 9094 909	86015208 83080140A8 8601501A 86015208 83080142A8 9718E6 961508 88F0 96F508 88F0 96F50F 96F50F 96F50F 96640A 97FBD2	srfsh:	ld sbit sbit ld sbit ld sbit sbit ld sbit sbit sbit sbit	psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4.w,#8 t4stp,pwmdl t4tfn,portpl r4.w,#8 upic,#x'18 uwrrdy,bfunh uwrrdy,bfunh uwrdy,bfunh urdrdy,bfunh i2,ircd irpd,#x'FB i3,ircd irpd,#x'F7 ; Clee	<pre>; Set one WAIT state. t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; inmediately. ; Stop timer T4 to allow loading, ; then load it. ; Start timer T4. ; Start timer T4. ; Enable pulses out. ; Load R4. ; Set up UPI port. ; 8-Bit UPI Mode ; enabled. ; Enable UWRRDY/ out. ; Empty 1BUF register, ; in case of false trigger. ; Enable UWRRDY/ out. ; Set up UREAD/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; Detects rising edges. ; Clear any false interrupt</pre>	
341 342 343 344 345 346 347 351 353 354 355 355 355 355 355 355 355 355	8080 0080 00991 0095 0099 0096 0004 0004 0004 0004 0008 0008 0008 0008 0008 0008 0008 0008 0008 0008 0008 0008 0008 0008 0008 0008 0008 0008 0008 0009 0008 0009 0000 0009 0000 0000 0000 0000 0000 0000 0000	B6Ø152Ø8 B6Ø15ØØA 83Ø8Ø14ØAB B6Ø15Ø1A B6Ø152Ø8 83Ø8Ø142AB 9718E6 96F5ØB 96F5ØF 96F5ØF 96F5ØF 96F5ØF 96F6ØP 97FBD2	srfsh: supi:	ld sbit sbit ld sbit sbit sbit sbit sbit sbit ld sbit ld sbit ld	psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4.w,#8 t4stp,pwmdl t4.fn,portpl r4.w,#8 upic,#x'18 uwrrdy,dirbh A,ibuf urdrdy,bfunh urdrdy,bfunh urdrdy,dirbh i2,ircd irpd,#x'FB i3,ircd ; Clea BK,#x'9089,#x'	<pre>; Set one WAIT state. t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; inmediately. ; Stop timer T4 to ; allow loading, ; then load it. ; Start timer T4. ; Enable pulses out. ; Load R4. ; Set up UPI port. ; 8-Bit UPI Mode ; enabled. ; Enable UWRRDY/ out. ; Enable UWRRDY/ out. ; Enable UWRRDY/ out. ; Enable URDRDY/ out. ; Set up UREAD/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRITE/ interrupt ; due to mode change. ; Clear any false interrupt ; due to mode change. ar all RAM locations. r Basepage bank:</pre>	
341 342 343 344 345 345 345 345 357 358 359 359 359 359 359 359 359 359 364 355 355 355 355 355 355 355 355 355 35	ØØ80 ØØ80 ØØ80 ØØ91 ØØ95 ØØ95 ØØ96 ØØ97 ØØ80 ØØ97 ØØ80 ØØ80	B6Ø152Ø8 B6Ø15ØØA 83Ø8Ø14ØAB B6Ø15Ø1A B6Ø15ZØB 83Ø8Ø142AB 9718E6 96F5ØB 96F5ØF 96F5ØF 96F5ØF 96F5ØF 96F5ØF 96F64ØA 97FBD2 9604ØB 97F7D2 80ØØBE	srfsh: supi:	ld sbit sbit tbit sbit ld sbit sbit ld sbit sbit ld sbit ld ld ld ld	<pre>psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4stp,pwmdl t4stp,pwmdl t4tfn,portpl r4.w,#8 upic,#x'18 uwrrdy,bfunh uurrdy,dirbh A,ibuf urdrdy,dirbh i2,ircd irpd,#x'FB i3,ircd irpd,#x'F7 ; Cleas BK,#x'90909,#x'A</pre>	<pre>; Set one WAIT state. t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; inmediately. ; Stop timer T4 to ; allow loading, ; then load it. ; Start timer T4. ; Enable pulses out. ; Load R4. ; Set up UPI port. ; 8-Bit UPI Mode ; enabled. ; Enable UWRRDY/ out. ; Enable UWRRDY/ out. ; Enable UWRRDY/ out. ; Enable URDRDY/ out. ; Set up UREAD/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRITE/ interrupt ; due to mode change. ; Clear any false interrupt ; due to mode change. ar all RAM locations. r Basepage bank:</pre>	
341 342 343 345 345 350 350 350 350 355 355 355 355 355 35	8080 0080 00991 0095 0099 0096 0004 0004 0004 0004 0008 0008 0008 0008 0008 0008 0008 0008 0008 0008 0008 0008 0008 0008 0008 0008 0008 0008 0008 0009 0008 0009 0000 0009 0000 0000 0000 0000 0000 0000 0000	B6Ø152Ø8 B6Ø15ØØA 83Ø8Ø14ØAB B6Ø15Ø1A B6Ø152Ø8 83Ø8Ø142AB 9718E6 96F5ØF 96F5ØF 96F5ØF 96F3ØF 96F3ØF 97FBD2 96D4ØA 97FFD2 80ØØBE ØB	srfsh: supi:	ld sbit sbit ld sbit sbit sbit sbit sbit sbit ld sbit ld sbit ld	psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4.w,#8 t4stp,pwmdl t4.fn,portpl r4.w,#8 upic,#x'18 uwrrdy,dirbh A,ibuf urdrdy,bfunh urdrdy,bfunh urdrdy,dirbh i2,ircd irpd,#x'FB i3,ircd ; Clea BK,#x'9089,#x'	<pre>; Set one WAIT state. t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; inmediately. ; Stop timer T4 to ; allow loading, ; then load it. ; Start timer T4. ; Enable pulses out. ; Load R4. ; Set up UPI port. ; 8-Bit UPI Mode ; enabled. ; Enable UWRRDY/ out. ; Enable UWRRDY/ out. ; Enable UWRRDY/ out. ; Enable URDRDY/ out. ; Set up UREAD/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRITE/ interrupt ; due to mode change. ; Clear any false interrupt ; due to mode change. ar all RAM locations. r Basepage bank:</pre>	
341 342 343 344 345 346 357 358 357 359 360 351 362 353 364 363 364 363 364 363 364 363 364 363 364 365 370 377 378 377 378 377 378 389 365 377 377 378 389 365 377 378 377 378 377 378 377 378 377 378 377 378 377 377	8980 9980 9991 9995 9994 9994 9994 9994 9994 9998 9988 998	B6Ø152Ø8 B6Ø15ØØA 83Ø8Ø14ØAB B6Ø15Ø1A B6Ø152Ø8 83Ø8Ø142AB 9718E6 96F5ØF 96F5ØF 96F5ØF 96F3ØF 96F3ØF 97FBD2 96D4ØA 97FFD2 80ØØBE ØB	srfsh: supi:	ld sbit sbit ld sbit ld sbit ld sbit ld sbit ld sbit ld sbit ld ld sbit ld	psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4stp,pwmdl t4stp,pwmdl t4stp,pwmdl t4tfn,portpl r4.w,#8 upic,#x'18 uurrdy,bfunh uurrdy,bfunh urdrdy,bfunh urdrdy,bfunh i2,ircd irpd,#x'FB i3,ircd irpd,#x'FB i3,ircd rpd,#x'FB i3,ircd rpd,#x'FB	<pre>; Set one WAIT state. t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; inmediately. ; Stop timer T4 to ; allow loading, ; then load it. ; Start timer T4. ; Enable pulses out. ; Load R4. ; Set up UPI port. ; 8-Bit UPI Mode ; enabled. ; Enable UWRRDY/ out. ; Enable UWRRDY/ out. ; Enable UWRRDY/ out. ; Enable URDRDY/ out. ; Set up UREAD/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRITE/ interrupt ; due to mode change. ; Clear any false interrupt ; due to mode change. ar all RAM locations. r Basepage bank:</pre>	
3412 3423 3434 345 345 356 357 3552 3553 3554 3555 3557 3569 3561 3565 3569 370 3773 3563 3564 3663 3774 3775 3776 3779 381 3823 3744 3775 3778 3779 381 3823 3833 3845 3774 3775 3778 3779 3813 3823 3774 3775 3778 3779 3813 3823 3774 3775 3778 3779 3777 3778 3779 3777 3778 3779 3777 3778 3779 3777 3778 3777 3777 3778 3777 3778 3777 3778 3777 3778 3777 3777 3777 3778 3777 3777 3777 3777 3777 3777 3777 3777 3777 3778 3777 3777 3777 3777 3777 37777 3778 3777 3777 3778 3778 37	ØØ80 ØØ80 ØØ91 ØØ95 ØØ90 ØØ95 ØØ90 ØØ90 ØØ00 ØØ00	B6Ø152Ø8 B6Ø15ØØA 83Ø8Ø14ØAB B6Ø15Ø1A B6Ø152Ø8 83Ø8Ø142AB 9718E6 96F5ØB 96F5ØB 96F5ØF 96F3ØF 96F3ØF 96F4ØA 97FBD2 96644ØA 97FFD2 80ØØ8E E1 62	srfsh: supi: sram: sraml1:	ld sbit sbit ld sbit sbit ld sbit sbit ld sbit ld sbit ld ld sbit ld ld sbit ld	psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4stp,fstp,dstp,dstp,dstp,dstp,dstp,dstp,dstp,d	<pre>; Set one WAIT state. ; t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; inmediately. ; Stop timer 14 to ; allow loading, ; then load it. ; Start timer 14. ; Enable pulses out. ; Load R4. ; Set up UPI port. ; 8-Bit UPI Mode ; enabled. ; Enable UWRDY/ out. ; Enable UWRDY/ out. ; Enable UWRDY/ out. ; Enable URDRDY/ out. ; Set up UPEAD/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRITE/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRITE/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRITE/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRITE/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRITE/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ar all RAM locations. puble ; Establish loop base and limit.</pre>	
341 342 343 344 345 345 356 357 355 355 355 355 355 356 356 356 356 356	\$\$\mathcal{P}\$\$ \$\$\$\mathcal{P}\$\$ \$\$\$\$\$ \$	B6Ø152Ø8 B6Ø15ØØA 83Ø8Ø14ØAB B6Ø15ØJA B6Ø15ØJA B6Ø15ØBA B6Ø15ØBA B6Ø152ØB B718E6 9718E6 96F5ØB 96F5ØF 96F5ØF 96F4ØBA 97FBD2 9604ØB 97F7D2 80ØØBE 90 81	srfsh: supi:	ld sbit sbit ld sbit sbit ld sbit sbit ld sbit ld sbit ld ld sbit ld ld sbit ld	psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4stp,dstp,dstp,dstp,dstp,dstp,dstp,dstp,d	<pre>; Set one WAIT state. ; t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; inmediately. ; Stop timer 14 to ; allow loading, ; then load it. ; Start timer 14. ; Enable pulses out. ; Load R4. ; Set up UPI port. ; 8-Bit UPI Mode ; enabled. ; Enable UWRDY/ out. ; Enable UWRDY/ out. ; Enable UWRDY/ out. ; Enable URDRDY/ out. ; Set up UPEAD/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRITE/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRITE/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRITE/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRITE/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRITE/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ar all RAM locations. puble ; Establish loop base and limit.</pre>	
341 342 343 344 345 345 356 357 355 355 355 355 355 356 356 356 356 356	ØØ80 ØØ80 ØØ80 ØØ91 ØØ95 ØØ95 ØØ80 ØØ95 ØØ80 ØØ80	B6Ø152Ø8 B6Ø15ØØA 83Ø8Ø14ØAB B6Ø15ØJA B6Ø15ØJA B6Ø15ØBA B6Ø15ØBA B6Ø152ØB B718E6 9718E6 96F5ØB 96F5ØF 96F5ØF 96F4ØBA 97FBD2 9604ØB 97F7D2 80ØØBE 90 81	srfsh: supi: sram: sraml1:	ld sbit ld rbit sbit ld sbit ld sbit sbit sbit ld sbit ld ld sbit ld ld sbit ld ld sbit ld	psw.b,#x'98 ; Star ; as t4out,portpl t4stp,pwmdl t4.w,#8 t4stp,pwmdl t4tfn,portpl r4.w,#8 upic,#x'18 uwrrdy,dirbh A,ibuf urdrdy,bfunh urdrdy,dirbh i2,ircd irpd,#x'FB i3,ircd irpd,#x'F7 ; Clee BK,#x'9009,#x'A A, [B+].w sraml1 BK,#x'91C9,#x'A	<pre>; Set one WAIT state. ; t dynamic RAM refreshing, quickly as possible. ; Trigger first refresh ; inmediately. ; Stop timer 14 to ; allow loading, ; then load it. ; Start timer 14. ; Enable pulses out. ; Load R4. ; Set up UPI port. ; 8-Bit UPI Mode ; enabled. ; Enable UWRDY/ out. ; Enable UWRDY/ out. ; Enable UWRDY/ out. ; Enable URDRDY/ out. ; Set up UPEAD/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRITE/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRITE/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRITE/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRITE/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ; Set up UWRITE/ interrupt. ; Detects rising edges. ; Clear any false interrupt ; due to mode change. ar all RAM locations. puble ; Establish loop base and limit.</pre>	

PORT INT	Ver D1-BetaSit ERFACE DEMO tialization	e (Sep	o 14 14:	50 1987)	HPC	UPI	25-Feb-88 10: PAGE
uware ini 390 0002			sskint:		: Set un	Stack and remove	
391					; indiv	idual interrupt enables.	
392 ØØD2 393		R		ld	••	; Move stack to high ; bank of on-chip RAM.	
394 ØØD6 395	87 00000000 0AB	R		ld	stackb.w,#hangup	; Safeguard against ; stack underflow.	
396 ØØDC	970000			ld	enir,#x'00	; Disable interrupts	
397 398						; individually.	
399 ØØDF	83Ø8Ø1928B		tminit:		tØcon,#x'Ø8		
400 00E4 401 00FA	8744400190AB 8355018EAB			ldi ldi	tmmode,#x'4440/ divby,#x'00055	; Stop timers T1, T2, T3. ; Timers T2 and T3 set to	
402						; clock externally.	
403 00EF 404	87CCC80190AB			ld	tmmode,#x'CCC8	; Clear and disable timer ; TØ-T3 interrupts.	
405	070004				Alash Harr		
406 00F5 407 00F8				ld sbit	dirah,#x'FF astts,portbh	; Initialize Port A upper byte for output. ; Enable and de-assert ENASIIS/ signal	
408 00FB				sbit		; (enables pushbutton data to Port D).	
409 00FE 410 0101				sbit sbit	cdata,portbh cdata,dírbh	; Enable and de-assert ENCDATA/ signal. ; (enables other data to Port D).	
411 412 Ø104			sled:	ld		; Set up to turn off LED's.	
413 Ø107	96E31E		steu.	rbit	ledclk,portbh	; Start with LEDCLK low,	
414 010A 415 0100	96F3ØE 96F3ØF			sbit sbit		; (enable output), ; then high,	
416 0110				rbit		; then low again.	
417 418 Ø113			stmrs:			; Set up remaining timers.	
419						; (T1-T3 already stopped	
420 421						; and pending bits cleared ; at tminit above, as	
422						; part of MICROWIRE init.)	
423 424 Ø113	872FFFØ182AB			ld	t1,#12287	; T1 runs at 10-millisecond real-time interval.	
425 Ø119	872FFFØ184AB			ld	r1,#12287	· Times semine should and interact	
426 427						; Timer remains stopped, and interrupt ; disabled, until INITIALIZE command.	
428	87444 99 159AB			ld			
30 0125				nop	pwmode,#x'444ø	; Stop timers T4-T7. ; Wait for valid PND	
31 0126	40 87ccc80150AB			nop	numeda #v10008	; bits. ; Clear and disable	
433	OICCOPTIPAB			ld	pwmode,#x'CCC8	; interrupts from all	
434 435						; PWM timers.	
436 Ø12D	87FFFFØ14AAB			lđi	r6,#x'FFFF	; No modulus for LCD Display Ready timer.	
437 438 0133	83CCØ14CAB			ld	t7,#204	; Set 17 to underflow at 6 KHz rate	
	83CCØ14EAB			ld	r7,#204	; (= 3 KHz at pin).	
							TL/DD/997

	INT	Ver D1-BetaSite ERFACE DEMO tialization	(Se	p 14 14:	30 1987) +	IPCUP I	25-Feb-88 10:0 PAGE 1
	13D	B601531F B601511E			rbit rbit	t7tfn,portph t7stp,pwmdh	; Disable beep tone to panel speaker. ; Start T7 running.	
443				-1		0.4		
444 91 445	145			slcd:			up LCD display. Jires use of timer T6, so	
446						; app	wears after timer initialization.	
447 448						: Firs	t, set up LCD contrast.	
449 01	145	97ØA12	R		ld	lcvs,#x'ØA	; Initialize memory image of LCD Voltage	
450 451 01	48	8C12E1	R		lď	portah,lcvs	; latch, containing RS (PAUXØ) bit also. ; Arbitrary initial contrast level of 5,	
452							; and RS/ (PAUXØ/) is high (="command").	
453 Ø1 454 Ø1					rbit sbit	lcvclk,portbh lcvclk,dirbh	; Start with LCVCLK low, ; (enable output)	
455 Ø1	151	96E3Ø9			sbit	lcvclk,portbh	; then high,	
456 Ø1 457	174	ADEDIA			rbit	lcvclk,portbh	; then low to get it into LCV latch.	
458 459 Ø1	57 (04E 20E			sbit	; Init	ialize PNLCLK (Panel "E" signal).	
469 91					sbit	phicik, dirbl	; Start with PNLCLK high ; (enable output).	
461 462								
463						; exe	for worst-case command cution time (4.9 ms, twice), in case	
464 465						; a p	wanel command was triggered while CLK was floating.	
466 81		B6Ø151ØB			sbit	t6ack,pwmdh	; Clear T6 PND bit.	
		8732C80148AB 8601511A			ld rbit	t6,#13000	; Set T6 to twice 4.9 milliseconds.	
469 Ø1		B6915111		lcdlp1:		t6stp,pwmdh t6pnd,pwmdh	; Start timer T6. ; Wait for T6 PND bit	
470 471 01	6F -	41			jp	lcdgo1	; to be set.	
472 Ø1	70 (65			jp	lcdip1		
		B601510A B601510B		lcdgo1:	sbit sbit	tóstp,pwmdh tóack,pwmdh	; Stop timer T6. ; Clear T6 PND bit.	
475							•	
476 477							t Panel controller (per Hitachi HD44780 er's Manual).	
478								
479 48ø							el RS signal was set LCD Contrast initialization above,	
481						; so	no change needed here to	
482 483						; fla	g these as commands.)	
484 Ø1 485 Ø1					ld	A,#x 38	; Send "8-Bit Mode, 2 Lines" command: one;	
486 01			R		jsrl ld	wrpnl A,#x'38	; two;	
487 Ø1 488 Ø1			R		jsrl	wrpnl		
489 01	81	2E	R		ld jsrl	A,#x'38 wrpnl	; three;	
								TL/DD/9976
				- 4/ 4/.	74 4007			25-Feb-88 10:0
		Ver D1-BetaSite	(Se	p 14 14:	26 1201	, ,	IPCUPI	
I PORT	INT							PAGE
I PORT rdware	INT Ini	tialization			L.A.		. four times	
I PORT rdware 499 91 491 91	1NT Ini 182 184	tialization 9038 2E	R		ld jsrl	A,#x'38 wrpnl	; four times.	
I PORT rdware 499 91 491 91 492 91	1NT Ini 182 184 185	tialization 9038 2E 9008			jsrl ld	A,#x'38 wrpni A,#x'98	; four times. ; Disable display.	
I PORT rdware 499 91 491 91 492 91 493 91 494 91	1NT Ini 182 184 185 187 188	tialization 9038 2E 9008 2E 9001	R R		jsrl	A,#x'38 wrpnl		
I PORT rdware 490 01 491 01 492 01 493 01 493 01 494 01 495 01	1NT Ini 182 184 185 187 188	tialization 9038 2E 9008 2E 9001			jsrl ld jsrl	A,#x'38 wrpnl A,#x'98 wrpnl	; Disable display.	
I PORT rdware 490 01 491 01 492 01 493 01 494 01 495 01 496 497	1NT Ini 182 184 185 187 188	tialization 9038 2E 9008 2E 9001	R		jsrl ld jsrl ld	A,#x'38 wrpnl A,#x'98 wrpnl A,#x'91 wrpnl	; Disable display.	
I PORT rdware 490 01 491 01 492 01 493 01 494 01 495 01 496 497 498	1NT Ini 182 184 185 187 188 188	tialization 9038 2E 9008 2E 9001 2E	R		jsrl ld jsrl ld	A,#x'38 wrpnl A,#x'98 wrpnl A,#x'91 wrpnl ; Init	; Disable display. ; Clear display RAM. ;al default mode settings.	
I PORT rdware 490 01 491 01 492 01 493 01 493 01 495 01 496 497 498 499 01 500 01	1NT Ini 182 184 185 187 188 188 188	tialization 9038 2E 9008 2E 9001 2E 9006 2E	R		jsrl ld jsrl ld jsrl ld jsrl	A,#x138 wrpnl A,#x198 wrpnl A,#x191 wrpnl ; Init A,#x196 wrpnl	; Disable display. ; Clear display RAM. ial default mode settings. ; Set mode to move cursor to the right, no ; automatic shifting of display.	
I PORT rdware 499 91 491 91 492 91 493 91 494 91 495 91 496 497 498 499 91 599 91 599 91	1NT Ini 182 184 185 187 188 188 188 188 188	tialization 9038 2E 9008 2E 9001 2E 9006 2E 9006 2E 9006	R R R		jsrl ld jsrl ld jsrl ld jsrl ld	A,#x'38 wrpnl A,#x'98 wrpnl A,#x'91 wrpnl ; Init A,#x'96 wrpnl A,#x'96	; Disable display. ; Clear display RAM. ;ial default mode settings. ; Set mode to move cursor to the right, no	
I PORT rdware 499 01 491 01 492 01 493 01 493 01 495 01 496 497 498 499 01 500 01 502 01 502 01 502 01	1NT Ini 182 184 185 187 188 188 188 188 188	tialization 9038 2E 9008 2E 9001 2E 9006 2E 9006 2E 9006	R R		jsrl ld jsrl ld jsrl ld jsrl	A,#x138 wrpnl A,#x198 wrpnl A,#x191 wrpnl ; Init A,#x196 wrpnl	; Disable display. ; Clear display RAM. ial default mode settings. ; Set mode to move cursor to the right, no ; automatic shifting of display.	
I PORT rdware 490 01 491 01 492 01 493 01 494 01 496 01 496 497 498 497 498 497 500 01 501 01 503 01 503 504	1NT Ini 182 184 185 187 188 188 188 188 188	tialization 9038 2E 9008 2E 9001 2E 9006 2E 9006 2E 9006	R R R		jsrl ld jsrl ld jsrl ld jsrl jsrl	A,#x"38 wrpnl A,#x"98 wrpnl A,#x"91 ; Init A,#x"96 wrpnl A,#x"9E wrpnl	; Disable display. ; Clear display RAM. ial default mode settings. ; Set mode to move cursor to the right, no ; automatic shifting of display.	
I PORT rdware 499 01 491 01 492 01 493 01 493 01 495 01 496 497 498 499 01 500 01 502 01 502 01 502 01	1NT Ini 182 184 185 187 188 188 188 188 188	tialization 9038 2E 9008 2E 9001 2E 9006 2E 9006 2E 9006	R R R	;	jsrl ld jsrl ld jsrl ld jsrl jsrl	A,#x"38 wrpnl A,#x"98 wrpnl A,#x"91 ; Init A,#x"96 wrpnl A,#x"9E wrpnl	; Disable display. ; Clear display RAM. ; al default mode settings. ; Set mode to move cursor to the right, no ; automatic shifting of display. ; Enable display: non-blinking cursor mode.	PAGE 2
I PORT rdware 490 01 491 01 492 01 493 01 494 01 496 01 496 497 498 497 498 497 500 01 501 01 503 01 503 504	1NT Ini 182 184 185 187 188 188 188 188 188	tialization 9038 2E 9008 2E 9001 2E 9006 2E 9006 2E 9006	R R R	;	jsrl ld jsrl ld jsrl ld jsrl jsrl	A,#x'38 wrpnl A,#x'98 wrpnl A,#x'91 ; Init A,#x'96 wrpnl A,#x'9E wrpnl	; Disable display. ; Clear display RAM. ; al default mode settings. ; Set mode to move cursor to the right, no ; automatic shifting of display. ; Enable display: non-blinking cursor mode.	PAGE 2
I PORT rdware 490 01 491 01 492 01 493 01 494 01 496 01 496 497 498 497 498 497 500 01 501 01 503 01 503 504	1NT Ini 182 184 185 187 188 188 188 188 188	tialization 9038 2E 9008 2E 9001 2E 9006 2E 9006 2E 9006	R R R	;	jsrl ld jsrl ld jsrl ld jsrl jsrl	A,#x'38 wrpnl A,#x'98 wrpnl A,#x'91 ; Init A,#x'96 wrpnl A,#x'9E wrpnl	; Disable display. ; Clear display RAM. ; al default mode settings. ; Set mode to move cursor to the right, no ; automatic shifting of display. ; Enable display: non-blinking cursor mode.	PAGE 2
I PORT rdware 490 01 491 01 492 01 493 01 494 01 496 01 496 497 498 497 498 497 500 01 501 01 503 01 503 504	1NT Ini 182 184 185 187 188 188 188 188 188	tialization 9038 2E 9008 2E 9001 2E 9006 2E 9006 2E 9006	R R R	;	jsrl ld jsrl ld jsrl ld jsrl jsrl	A,#x'38 wrpnl A,#x'98 wrpnl A,#x'91 ; Init A,#x'96 wrpnl A,#x'9E wrpnl	; Disable display. ; Clear display RAM. ; al default mode settings. ; Set mode to move cursor to the right, no ; automatic shifting of display. ; Enable display: non-blinking cursor mode.	PAGE 2
I PORT rdware 490 01 491 01 492 01 493 01 494 01 496 01 496 497 498 497 498 497 500 01 501 01 503 01 503 504	1NT Ini 182 184 185 187 188 188 188 188 188	tialization 9038 2E 9008 2E 9001 2E 9006 2E 9006 2E 9006	R R R	;	jsrl ld jsrl ld jsrl ld jsrl jsrl	A,#x'38 wrpnl A,#x'98 wrpnl A,#x'91 ; Init A,#x'96 wrpnl A,#x'9E wrpnl	; Disable display. ; Clear display RAM. ; al default mode settings. ; Set mode to move cursor to the right, no ; automatic shifting of display. ; Enable display: non-blinking cursor mode.	PAGE 2
I PORT rdware 490 01 491 01 492 01 493 01 494 01 496 01 496 497 498 497 498 497 500 01 501 01 503 01 503 504	1NT Ini 182 184 185 187 188 188 188 188 188	tialization 9038 2E 9008 2E 9001 2E 9006 2E 9006 2E 9006	R R R	;	jsrl ld jsrl ld jsrl ld jsrl jsrl	A,#x'38 wrpnl A,#x'98 wrpnl A,#x'91 ; Init A,#x'96 wrpnl A,#x'9E wrpnl	; Disable display. ; Clear display RAM. ; al default mode settings. ; Set mode to move cursor to the right, no ; automatic shifting of display. ; Enable display: non-blinking cursor mode.	PAGE 2
I PORT rdware 490 01 491 01 492 01 493 01 494 01 496 01 496 497 498 497 498 497 500 01 501 01 503 01 503 504	1NT Ini 182 184 185 187 188 188 188 188 188	tialization 9038 2E 9008 2E 9001 2E 9006 2E 9006 2E 9006	R R R	;	jsrl ld jsrl ld jsrl ld jsrl jsrl	A,#x'38 wrpnl A,#x'98 wrpnl A,#x'91 ; Init A,#x'96 wrpnl A,#x'9E wrpnl	; Disable display. ; Clear display RAM. ; al default mode settings. ; Set mode to move cursor to the right, no ; automatic shifting of display. ; Enable display: non-blinking cursor mode.	PAGE 2
I PORT rdware 490 01 491 01 492 01 492 01 493 01 494 01 496 497 498 497 498 497 500 01 500 01 502 01 503 504	1NT Ini 182 184 185 187 188 188 188 188 188	tialization 9038 2E 9008 2E 9001 2E 9006 2E 9006 2E 9006	R R R	;	jsrl ld jsrl ld jsrl ld jsrl jsrl	A,#x'38 wrpnl A,#x'98 wrpnl A,#x'91 ; Init A,#x'96 wrpnl A,#x'9E wrpnl	; Disable display. ; Clear display RAM. ; al default mode settings. ; Set mode to move cursor to the right, no ; automatic shifting of display. ; Enable display: non-blinking cursor mode.	PAGE 2
I PORT rdware 490 01 491 01 492 01 492 01 493 01 494 01 496 497 498 497 498 497 500 01 500 01 502 01 503 504	1NT Ini 182 184 185 187 188 188 188 188 188	tialization 9038 2E 9008 2E 9001 2E 9006 2E 9006 2E 9006	R R R	;	jsrl ld jsrl ld jsrl ld jsrl jsrl	A,#x'38 wrpnl A,#x'98 wrpnl A,#x'91 ; Init A,#x'96 wrpnl A,#x'9E wrpnl	; Disable display. ; Clear display RAM. ; al default mode settings. ; Set mode to move cursor to the right, no ; automatic shifting of display. ; Enable display: non-blinking cursor mode.	PAGE 2
I PORT rdware 490 01 491 01 492 01 492 01 493 01 494 01 496 497 498 497 498 497 500 01 500 01 502 01 503 504	1NT Ini 182 184 185 187 188 188 188 188 188	tialization 9038 2E 9008 2E 9001 2E 9006 2E 9006 2E 9006	R R R	;	jsrl ld jsrl ld jsrl ld jsrl jsrl	A,#x'38 wrpnl A,#x'98 wrpnl A,#x'91 ; Init A,#x'96 wrpnl A,#x'9E wrpnl	; Disable display. ; Clear display RAM. ; al default mode settings. ; Set mode to move cursor to the right, no ; automatic shifting of display. ; Enable display: non-blinking cursor mode.	PAGE 2
I PORT rdware 490 01 491 01 492 01 492 01 493 01 494 01 496 497 498 497 498 497 500 01 500 01 502 01 503 504	1NT Ini 182 184 185 187 188 188 188 188 188	tialization 9038 2E 9008 2E 9001 2E 9006 2E 9006 2E 9006	R R R	;	jsrl ld jsrl ld jsrl ld jsrl jsrl	A,#x'38 wrpnl A,#x'98 wrpnl A,#x'91 ; Init A,#x'96 wrpnl A,#x'9E wrpnl	; Disable display. ; Clear display RAM. ; al default mode settings. ; Set mode to move cursor to the right, no ; automatic shifting of display. ; Enable display: non-blinking cursor mode.	
I PORT rdware 490 01 491 01 492 01 492 01 493 01 494 01 496 497 498 497 498 497 500 01 500 01 502 01 503 504	1NT Ini 182 184 185 187 188 188 188 188 188	tialization 9038 2E 9008 2E 9001 2E 9006 2E 9006 2E 9006	R R R	;	jsrl ld jsrl ld jsrl ld jsrl jsrl	A,#x'38 wrpnl A,#x'98 wrpnl A,#x'91 ; Init A,#x'96 wrpnl A,#x'9E wrpnl	; Disable display. ; Clear display RAM. ; al default mode settings. ; Set mode to move cursor to the right, no ; automatic shifting of display. ; Enable display: non-blinking cursor mode.	PAGE 1
I PORT rdware 490 01 491 01 492 01 492 01 493 01 494 01 496 497 498 497 498 497 500 01 500 01 502 01 503 504	1NT Ini 182 184 185 187 188 188 188 188 188	tialization 9038 2E 9008 2E 9001 2E 9006 2E 9006 2E 9006	R R R	;	jsrl ld jsrl ld jsrl ld jsrl jsrl	A,#x'38 wrpnl A,#x'98 wrpnl A,#x'91 ; Init A,#x'96 wrpnl A,#x'9E wrpnl	; Disable display. ; Clear display RAM. ; al default mode settings. ; Set mode to move cursor to the right, no ; automatic shifting of display. ; Enable display: non-blinking cursor mode.	PAGE 1

Schemer, Ver D1-BataSite (Sep 14.14:39 1987) HPCUP1 25:-:0.83 18:g PAGE 1 Sin Scan Loop .form 'Main Scan Loop'	in Pro		ERFACE DEMO 1 Initializatio	n					
<pre>set approx nint: : : : : : : : : : : : : : : : : : :</pre>						.form	'Main Progra	n Initialization'	
<pre>j mer-only initializations. j Dece-only initializations. j Uree-only initialization to bit set means 'more'. j Uree-only initialization to be set means 'more'. j Uree-only Uree-only</pre>	508	A1 01			minit·				
512 2010 278014 1010 2700014 1010 2700014 10100014 1010 2700014 1010 27000000000000000000000000000000000	510	w 1 7 1					; One	ce-only initializations.	
355	512 Ø	0194	B7000604	R		ld	cpuad,#cpubu	f ; Set CPU command index to beginning of buffer.	
519 519 519 529 529 529 529 529 529 529 529 529 52	515 516 517		,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	ĸ		iu.	Ari	pitrary set of initialization values for variables, n effect until receipt of the first INITIALIZE	
323 3097 runsys: ; Enable interrupts, start timers and go to main loop. 324 307 960900 sbit timer, min : Enable timer interrupts. (Done here in all Gu certain commands without an interrupt. (Done here) 327 81A2 960980 sbit is, min : Enable CPU Command interrupt. 328 309 sbit is, min : Enable CPU Command interrupt. 329 309 sbit is, min : Enable CPU Command interrupt. 320 309 .form 'Main Scan Loop' : Command interrupt. 333 309 .form 'Main Scan Loop' : Command interrupt. 333 309 .form 'Main Scan Loop' : Command interrupt. 333 309 .form 'Main Scan Loop' : Command interrupt. 333 339 .form : Command interrupt. : Comma	519	1100	P700000			1.4			
523 prof 9600000 sbit ters,enir : Enable terr interrupts. (Dore here in the state interrupt.) 525 prof 96000000000000000000000000000000000000	521		87999992			lu		, , -	
525 527 9142 90089 it is and is and is it is and it is an it is a	523		0/2.002			- 1. 7 4			
528 BMAS 960088 sbit gie,enir ; Enable interrupt system. 539 TLUDD/9076 539 TOT INTERACE DFM 531 .form 'Main Scan Loop' 532 .form 'Main Scan Loop' 533 .form 'Main Scan Loop' 534 .form 'Main Scan Loop' 535 .form 'Main Scan Loop' 537 .form 'Main Scan Loop' 538 .form 'Main Scan Loop' 539 .form 'Main Scan Loop' 541 .form An	525 526						tmrs,entr	; to allow certain commands without an ; INITIALIZE command first.)	
SC ASMHPC, Ver D1-BetaSite (Sep 14 14:39 1987) HPCUP1 25:-/-o-88 19:8 ST ASMHPC, Ver D1-BetaSite (Sep 14 14:39 1987) HPCUP1 25:-/-o-88 19:8 ST ASMHPC, Ver D1-BetaSite (Sep 14 14:39 1987) FRei Time Clock vector number. Status ST ASMHPC, Ver D1-BetaSite (Sep 14 14:39 1987) Interpretation Status Status ST ASMHPC, Ver D1-BetaSite (Sep 14 14:39 1987) Interpretation Status Status ST ASMHPC, Ver D1-BetaSite (Sep 14 14:39 1987) Interpretation Status Status Status ST ASMHPC, Ver D1-BetaSite (Sep 14 14:39 1987) Interpretation Status Status<	528 Ø 529								
PAGE 1 PAGE 1 331 .form 'Main Scan Loop' 332 .form 'Main Scan Loop' 333 ; Declarations 334 yrtc = x'11 ; Real-Time Clock vector number. 335 B#11 vrtc = x'11 ; Real-Time Clock vector number. 336 B#11 vrtc = x'11 ; Real-Time Clock vector number. 337 B#10 vdtag = x'10 ; Diagnostic Interrupt. 338 B#11 vrtc = x'11 ; Real-Time Clock vector number. 339 B#10 vdtag = x'10 ; Diagnostic Interrupt. 340 y unexpected interrupt. Sectors for unimplemented or ; unexpected. 341 ; Level Ø is Reast, provided by assembler. Sectors in never expected. 342 ; Level Ø is Reast, provided by assembler. Sectors in never expected. 354 ; Diagnostic Interrupt Vector: never expected. Sectors in never expected. 355 Ø#1A8 mainlp: Sectors in never expected. 355 Ø#1A8 Seg#2#27C Sectors in never expected. 356 Ø#1A8 iffit antc,alert.b ; Check for B1C interrupt request. Sectors interrupt. 356 Ø#1A8 iffit antout,alert.b ; Check for Diagnostic Interrupt.<									TL/DD/9976-
533 .form 'Main Scan Loop' 533 ; Declarations 533 ; Declarations 535 9811 vtcd # *17 ; Acknowledge finished writing to LCD panel. 535 9817 vtcdak # *17 ; Acknowledge finished writing to LCD panel. 537 9810 vdcdak # *17 ; Acknowledge finished writing to LCD panel. 537 9810 vdcdag = *110 ; Diagnostic Interrupt. 544 ; Error Vectors for unimplemented or ; unexpected Interrupts. 545 ; Level # is Reset, provided by assembler. .form 'Hains Zang / Juney / Ju	I POR	T INT	TERFACE DEMO	te (Se	p 14 14:	30 1987;	ŀ	HPCUPI	25-2
<pre>533 ; Declarations 534 0011 vrtc = x'11 ; Real-Time Clock vector number. 535 0017 vbutton = x'16 ; Achanoledge finished writing to LDD panel. 536 0010 vbutton = x'16 ; Achanoledge finished writing to LDD panel. 537 0010 vbutton = x'16 ; Achanoledge finished writing to LDD panel. 538 0010 vdiag = x'10 ; Diagnostic Interrupt. 549 10 vdiag = x'10 ; Diagnostic Interrupt. 540 FFC 0000 R ; Interrupt vectors for unimplemented or 541 ; Level Ø is Reset, provided by assembler. 545 ; Level Ø is Reset, provided by assembler. 546 FFC 0000 R . ipt 1, hangup ; UPI READ READY: never expected. 547 FFFA 0000 R . ipt 2, hangup ; UPI READ READY: never expected. 548 FFC 0000 R . ipt 2, hangup ; UPI READ READY: never expected. 549 FFC 0000 R . ipt 7, hangup ; UPI READ READY: never expected. 549 FFC 0000 R . ipt 7, hangup ; UPI READ READY: never expected. 559 01A8 meintp: 555 01A8 20002FC R chkalt: ifeq alert.u, #x'00 ; Check for alert conditions. 558 01A0 000211 R ifbit alcdak,alert.b ; Check for alert conditions. 559 01A0 000211 R ifbit alcdak,alert.b ; Check for a pushbutton change. 550 01B3 0000211 R ifbit alcdak,alert.b ; Check for a pushbutton change. 560 01B2 000213 R ifbit alcdak,alert.b ; Check for a pushbutton change. 560 01B2 000212 R ifbit alcdak,alert.b ; Check for a pushbutton change. 561 01B2 000212 R ifbit alcdak,alert.b ; Check for a pushbutton change. 567 01BC 000212 R ifbit aliga, elert.b ; Check for a pushbutton change. 567 01BC 000212 R ifbit aliga, elert.b ; Check for a pushbutton change. 567 01BC 000212 R ifbit aliga, elert.b ; Check for a pushbutton change. 578 01C1 70 jmpl chkalt ; No "responses" defined yet; just close loop. 577 1122 00017 Main: Send Real-Time Clock Interrupt. 576 01C2 sondrt: 577 0122 000219 R isrl sndlag ; If so, then send interrupt and data. 579 01C2 00219 R isrl sndlag ; If so, then send interrupt and data. 570 01C2 sondrt: 577 01C2 000219 R isrl antc, alert.b ; Check therput! 577 01C2 000219 R isrl antc, alert.b ; Check therput! 578 10 cont interr</pre>	531					.form	'Main Scan L	oop'	
535 0011 vrc = x'11 ; Real-Time Clock vector number. 537 0017 vlcdk = x'17; Acknowledge finished writing to LCD panel. 537 0017 vlcdk = x'18; Pushbutton status change: a button pressed or ; released. 539 0010 vdiag = x'10; Diagnostic Interrupt. 540 ; Furo Vectors for unimplemented or ; unexpected interrupts. 541 ; Level Ø is Beset, provided by assembler. 544 ; Level Ø is Beset, provided by assembler. 545 ; Level Ø is Beset, provided by assembler. 546 ; Dirte GabBB 547 ; FFA 0000 547 ; Dirte GabBB 547 ; Dirte GabBB 548 ; Dirte GabBB 549 ; Level Ø 547 ; FFA 0000 548 ; Dirte GabBB 549 ; Dirte GabBB 547 ; Dirte GabBB 548 ; Dirte GabBB 549 ; Dirte GabBB 549 ; Dirte GabBB 540 ; Dirte GabBB 54118 maintp: 555 jlA8 556 jlA8 557 jlA8 20002rC <td< td=""><td>533</td><td></td><td></td><td></td><td>;</td><td>Declara</td><td>tions</td><td></td><td></td></td<>	533				;	Declara	tions		
536 Ø817 victak = x'17 ; Acknowledge finished writing to LCD panel. 537 Ø816 vbuttan = x'16; Fushbutton status change: a buttan pressed or ; released. 538 gøld vdiag = x'10; j Diagnostic Interrupt. 541 : Error Vectors for unimplemented or ; unexpected interrupts. 542 : level Ø is Reset, provided by assembler. 543 : ipt 1,hnoupp : Wil: never expected. 544 : ipt 1,hnoupp : Val Interrupt Vector: never expected. 545 : ipt 1,hnoupp : Val Interrupt Vector: never expected. 546 : ipt 7,hangup ; UAI Interrupt Vector: never expected. 557 Ø1A8 mainlp: 558 Ø1A8 620902FC R chkalt: ifeq alert.w.#x'#0 ; Check for alert conditions. 558 Ø1A8 620902FC R ifbit artc.alert.b ; Check for RTC interrupt request. 559 Ø1A8 620902FC R ifbit adcak.alert.b ; Check for a pushbutton change. 558 Ø1A8 620902FC R ifbit adiag.alert.b ; Check for a pushbutton change. 558 Ø1A8 620902FC R ifbit adiag.alert.b ; Check for a pushbutton change. 559 Ø1A8 620902FC R ifbit adiag.alert.b ; Check for a pushbutton change. 558 Ø1A8 620902FC R ifbit adiag.alert.b ; Check for a pushbutton change to the CPU.	535							al-Time Clock vector number.	
533 9910 vdiag ; released. 539 9910 vdiag x '10 ; Diagnostic Interrupt. 549 549 ; Error Vactors for Unimplemented or ; unexpected interrupts. 544 545 ; level Ø is Reset, provided by assembler. 545 546 547 ; level Ø is Reset, provided by assembler. 546 547 ; level Ø is Reset, provided by assembler. 547 ; level Ø is Reset, provided by assembler. 547 ; level Ø is Reset, provided by assembler. 547 ; level Ø 547 ; level Ø 547 ; level Ø 547 ; level Ø 548 ; ipt (,hangup) ; UHI READ READ: never expected. 549 ; ipt (,hangup) ; UART Interrupt Vector: never expected. 549 ; ipt (,hangup) ; UART Interrupt Vector: never expected. 559 ; if so, then send Real-Time Clock interrupt. 555 gilA0 060211 R ifbit astc,alert.b ; Check for RIC interrupt request. ; 559 gis7 softer adiag,alert.b ; Check for Diagnostic Interrupt. 564 jis7 069210 R ifbit adiag,alert.b ; Check for Diagnostic Interrupt. 566 jis7 softer softer bis0 ji	536	0017			vlcdak		x'17 Ac	knowledge finished writing to LCD panel.	
549 ; Error Vectors for unimplemented or 541 ; Urexel Ø is Reset, provided by assembler. 545 ; Ievel Ø is Reset, provided by assembler. 546 ; Pit I, hangup ; PHI: nevre expected. 547 FFFA 0000 Rpit 4, hangup ; UPI EAD READY: never expected. 548 FFFA 0000 Rpit 4, hangup ; UAR Interrupt Vector: never expected. 549 FFF2 0000 Rpit 4, hangup ; UAR Interrupt Vector: never expected. 550 FFF2 0000 Rpit 4, hangup ; EI Interrupt Vector: never expected. 551 ipt 7, hangup ; EI Interrupt Vector: never expected. 552 Ø1AB mainlp: 553 BIAB 220002FC R chkalt: if eq alert.w.#x/90 ; Check for alert conditions. 554 jpt rist sndite ; if so, then send Real-Time Clock interrupt. 555 Ø1BB 2000213 R ifbit alcdak,alert.b ; Check for LCD Panel write done. 564 Ø1BE 060213 R ifbit alcdak,alert.b ; Check for Jamsend Lock knowledge interrupt. 565 Ø1BB 20002 R ifbit adiag,alert.b ; Check for Jamsend Lock interrupt. 566 Ø1BE 060212 R ifbit adiag,alert.b ; Check for Diagnostic Interrupt. 566 Ø1BE 5023 jard data : jarl sndiag ; If so, then send interrupt and data. 566 Ø1BE 5023 jard is neight sndiag, ifbit and isg,alert.b ; Check for Diagnostic Interrupt and data. <td>538 539</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>; r</td> <td>eleased.</td> <td></td>	538 539						; r	eleased.	
<pre>543 ; unexpected interrupts. 546 545 ; level Ø is Reset, provided by assembler. 547 FFFA ØØØØ R .ipt 2, hangup ; UPI READ REAV: never expected. 547 FFFA ØØØØ R .ipt 2, hangup ; UPI READ REAV: never expected. 549 FFF2 ØØØØ R .ipt 6, hangup ; UART Interrupt Vector: never expected. 540 FFF2 ØØØØ R .ipt 7, hangup ; UART Interrupt Vector: never expected. 550 FFF2 ØØØØ R .ipt 7, hangup ; UART Interrupt Vector: never expected. 551 frag ØØØ R .ipt 7, hangup ; UART Interrupt Vector: never expected. 552 Ø1A8 mainlp: 553 Ø1A8 & 2000@2FC R chkalt: if eq alert.w.#x'ØØ ; Check for alert conditions. 555 Ø1A8 & 2000@2FC R chkalt: if eq alert.w.#x'ØØ ; Check for RTC interrupt request. 556 Ø1A8 & 64200 557 Ø1B0 962211 R ifbit artc, alert.b ; Check for RTC interrupt request. 558 Ø1B0 962213 R ifbit alcdak, alert.b ; Check for CLO Panel write done. 556 Ø1B3 Ø10 508218 R ifbit abutton, alert.b ; Check for a pushuton change. 556 Ø1B3 Ø16 jsrl sodtak ; If so, then send Real-Time Clock interrupt. 566 Ø1B7 968218 R ifbit abutton, alert.b ; Check for a pushuton change. 556 Ø1B7 968218 R ifbit adiag, alert.b ; Check for a pushuton change. 567 Ø1BC 968212 R ifbit adiag, alert.b ; Check for a pushuton change. 567 Ø1BC 968212 R ifbit adiag, alert.b ; Check for a pushuton change. 567 Ø1BC 968212 R ifbit adiag, alert.b ; Check for a pushuton change. 567 Ø1BC 968212 R ifbit adiag, alert.b ; Check for Diagnostic Interrupt. 568 Ø1B3 Ø16 jsrl sodiag ; If so, then send interrupt and data. 570 Ø1C1 70 jmpl chkalt ; No "responses" defined yet; just close loop. 571 TU/DD/9976 572 .reb-88 19:9 573 .rot atta transfer; just trigger interrupt and continue. 575 ø1C2 968219 R jsrl rot, alert.b ; Clear ALERT bit. 576 Ø1C2 sodtrc: 577 Ø1C2 968219 R jsrl rotwit ; joad Real-Time Clock Interrupt' 576 Ø1C5 ZF R jsrl rotwit ; Clear ALERT bit. 578 Ø1C5 ZF R jsrl rotwit ; Load Real-Time Clock vector into OBUF for CPU. 578 Ø1C5 ZF R jsrl rotwit ; Load Real-Time and continue. 578 Ø1C5 ZF R jsrl rotwit ; Load Real-Time and loop. 578 Ø1C5 ZF R jsrl ret ; Load Real</pre>	541								
545 ; level Ø is Reset, provided by assembler. 546 FFFC 0000 R .ipt 1,hangup; VMI: never expected. 547 FFFA 0000 R .ipt 2,hangup; VMI: never expected. 547 FFF2 0000 R .ipt 4,hangup; VMI: never expected. 548 FFF6 0000 R .ipt 7,hangup; VMI: never expected. 559 FFF2 0000 R .ipt 7,hangup; VARI Interrupt Vector: never expected. 550 FFF2 0000 R .ipt 7,hangup; VARI Interrupt Vector: never expected. 551 FF2 0000 R .ipt 7,hangup; VARI Interrupt Vector: never expected. 552 01A0 mainlp: .ipt 7,hangup; VARI Interrupt Vector: never expected. 553 0180 000000 mainlp: .ipt 7,hangup; VARI Interrupt Vector: never expected. 554 01A0 000000 mainlp: .ipt 7,hangup; VARI Interrupt Vector: never expected. 555 01A0 0000000 mainlp: .ipt 7,hangup; VARI Interrupt Vector: never expected. 555 01A0 0000000000000000000000000000000	543								
547 FFFA 0000 R .ipt 2,hangup ; UPI READ REAUX: never expected. 546 FFFA 0000 R .ipt 6,hangup ; UART Interrupt Vector: never expected. 559 FFF2 0000 R .ipt 6,hangup ; UART Interrupt Vector: never expected. 551 FF2 0000 R .ipt 7,hangup ; EI Interrupt Vector: never expected. 551 FF2 0000 R .ipt 7,hangup ; EI Interrupt Vector: never expected. 552 01A8 mainlp: .ipt 7,hangup ; EI Interrupt Vector: never expected. 553 01AC mainlp: .ipt 7,hangup ; EI Interrupt Vector: never expected. 553 01AC mainlp: .ipt .ipt 7,hangup ; EI Interrupt Vector: never expected. 553 01A2 064011 R alert.w,#x*00 ; Check for alert conditions. .ipt 554 01A2 0640213 R ifbit alcdak,alert.b ; Check for Linerrupt request. .ipt 564 01B7 960210 R ifbit alcdak,alert.b ; Check for a pushbuton change. .ipt 566 01B7 960212 R ifbit adiagalert.b ; Check for balonstic Interrupt. .ipt 566 01B7 9823	545	FFFC	0000	P	;		Ø is Reset	, provided by assembler.	
549 FFE2 0000 R .ipt 6,hangup ; UART Interrupt Vector: never expected. 551 FFE2 0000 R .ipt 7,hangup ; El Interrupt Vector: never expected. 551 552 01A8 meinlp: .ipt 7,hangup ; El Interrupt Vector: never expected. 552 01A8 020002FC R chkalt: ifeq alert.w.#x'00 ; Check for alert conditions. 553 01A8 020002FC R chkalt ; If none, keep looping. 555 01A8 02002FC R chkalt ; If so, then send Real-Time Clock interrupt. 560 01B2 0602210 R ifbit alcdak,alert.b ; Check for LDD Panel write done. 562 01B3 3010 jsrl sndtar ; If so, then send Real-Time Clock interrupt. 563 01B2 0602210 R ifbit abutton,alert.b ; Check for a pushbutton change. 564 01B3 90212 R ifbit adiag,alert.b ; Check for biagnostic Interrupt. 566 01B4 90212 R ifbit adiag,alert.b <td>547</td> <td>FFFA</td> <td>0000</td> <td>R</td> <td></td> <td>. ipt</td> <td>2,hangup</td> <td>; UPI READ READY: never expected.</td> <td></td>	547	FFFA	0000	R		. ipt	2,hangup	; UPI READ READY: never expected.	
558 FFF9 9889 R .ipt 7,hangup ; El Interrupt Vector: never expected. 551 9188 mainlp: .ipt 7,hangup ; El Interrupt Vector: never expected. 554 555 9188 820902FC R chkalt: ifeq alert.w,#x'90 ; Check for alert conditions. 556 9180 644 ; jp chkalt ; If none, keep looping. 557 9188 820902FC R chkalt ; If none, keep looping. 557 9189 910 sindt: ; Check for Linterrupt request. ist is indrice. 558 9189 96211 R ifbit alcdak,alert.b ; Check for a pushbutton change. 564 9187 96219 R ifbit abuton,alert.b ; Check for a pushbutton change. 565 9188 3916 jsrl sndtak ; If so, then report the change to the CPU. 566 9187 96219 R ifbit abuton,alert.b ; Check for a pushbutton change. 566 9187 3923 jsrl sndtar ; If so, then send interrupt. <	549	FFF2	8888	R		.ipt	6,hangup	; UART Interrupt Vector: never expected.	
552 P1A8 mainlp: 553 mainlp: mainlp: 554 p chkalt: ifeq alert.w,#x'99 ; Check for alert conditions. 556 P1A8 829992FC R chkalt: ; If none, keep looping. 557 P1A8 829992FC R ifbit alert.w,#x'99 ; Check for alert conditions. 557 P1A8 20902FC R ifbit and continuerupt. ; If none, keep looping. 557 P1A8 20902FC R ifbit and continuerupt. ; If so, then send Real-Time Clock interrupt. 569 P1B5 3913 jsrl sndlak ; If so, then send LCD Acknowledge interrupt. 564 P1B7 048219 R ifbit abuton, alert.b ; Check for a pushbutton change. 564 P1B7 048210 R ifbit adiag, alert.b ; Check for Diagnostic Interrupt. 566 P1B8 3923 jsrl sndiag ; If so, then send interrupt and data. 579 910C 79 jmpl chkalt ; No "responses" defined yet; just close loop. 571 PORT INTERFACE DEMO in:: Send Real-Time Clock Interrupt PAGE 1 </td <td>55Ø</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>; EI Interrupt Vector: never expected.</td> <td></td>	55Ø							; EI Interrupt Vector: never expected.	
555 91AC 64 R chkalt: ifeq jp alert.w,#x'90 ; Check for alert conditions. ; If none, keep looping. 556 91AC 64 ; if none, keep looping. ; if none, keep looping. 557 558 91B9 3010 ; if if it artc.alert.b ; Check for RTC interrupt request. ; if so, then send Real-Time Clock interrupt. 560 91B2 960213 R ifbit alcdak,alert.b ; Check for LCD Panel write done. ; of so, then send LCD Acknowledge interrupt. 563 564 91B7 960210 R ifbit abutton,alert.b; Check for a pushbutton change. ; of so, then report the change to the CPU. 564 91B7 960212 R ifbit adiag,alert.b ; Check for Diagnostic Interrupt. 566 91B7 3023 jsrl sndiag ; If so, then send interrupt and data. 567 91BC 960212 R ifbit adiag,alert.b ; Check for Diagnostic Interrupt. 568 91BF 3023 jsrl sndiag ; If so, then send interrupt and data. 569 91BC 960212 R ifbit and ag,alert.b ; Check for Diagnostic Interrupt. 578 91BC 91C1 79 jmpl chkalt ; No "responses" defined yet; just close loop. 571 100 send Real-Ti	552	Ø1A8			mainlp:				
557	554 555			R	chkalt:				
568 161 1812 968213 R ibit alcdak,alert.b ; Check for LCD Panel write done. 563 9185 3913 jsrl sndlak ; If so, then send LCD Acknowledge interrupt. 564 9187 969218 R ifbit abutton,alert.b ; Check for a pushbutton change. 564 9184 3916 jsrl sndbtn ; If so, then report the change to the CPU. 566 9186 96212 R ifbit adiag,alert.b ; Check for Diagnostic Interrupt. 568 9186 5923 jsrl sndiag ; If so, then send interrupt and data. 569 918C 769 91C1 79 jmpl chkalt ; No "responses" defined yet; just close loop. 571 91C1 79 jmpl chkalt ; No "responses" defined yet; just close loop. 572 .form 'Main: Send Real-Time Clock Interrupt 25-feb-88 19:9 574 91C2 .form 'Main: Send Real-Time Clock Interrupt' 575 575 91C2 .form 'Main: Send Real-Time Clock Interrupt 575	557 558	Ø1AD	960211	R		ifbit	artc,alert.b	; Check for RTC interrupt request.	
562 ØH85 3ØH3 jsrl sndlak ; If so, then send LCD Acknowledge interrupt. 563 ØH87 96Ø2HØ R ifbit abutton,alert.b; ; Check for a pushbutton change. 565 ØH87 96Ø2HØ R ifbit abutton,alert.b; ; Check for a pushbutton change. 566 ØH87 96Ø2HØ R ifbit adiag,alert.b; ; Check for Diagnostic Interrupt. 566 ØH87 3Ø25 jsrl sndtag; ; If so, then send interrupt and data. 567 JBE0 96Ø2H2 R ifbit adiag,alert.b; ; Check for Diagnostic Interrupt. 568 JBE7 Spit sndtag; ; If so, then send interrupt and data. 569 JBE0 96Ø2H2 R ifbit shdiag; ; If so, then send interrupt and data. 569 Jmpl chkalt ; No "responses" defined yet; just close loop. 571 Jmpl chkalt ; No "responses" defined yet; just close loop. 572 Jmpl chkalt ; No "responses" defined yet; just close loop. 573 Jorn "Main: Send Real-Time Clock Interrupt' 25-feb-88 10:9 573 Jorn <td>56Ø 561</td> <td>Ø1B2</td> <td>960213</td> <td>R</td> <td></td> <td>-</td> <td></td> <td>.b ; Check for LCD Panel write done.</td> <td></td>	56Ø 561	Ø1B2	960213	R		-		.b ; Check for LCD Panel write done.	
565 #IBA 3016 jsrl sndbtn ; If so, then report the change to the CPU. 567 #IBC 960212 R ifbit adiag,alert.b ; Check for Diagnostic Interrupt. 568 #IBF 3025 jsrl sndiag ; If so, then send interrupt and data. 569 #IBF 3025 jsrl sndiag ; If so, then send interrupt and data. 569 #IC1 79 jmpl chkalt ; No "responses" defined yet; just close loop. 571 #IC ASMHPC, Ver DI-BetaSite (Sep 14 14:30 1987) HPCUPI 25-fab-88 10:0 10 PORT INTERFACE DEMO 25-fab-88 10:0 PAGE 1 in: Send Real-Time Clock Interrupt 25-fab-88 10:0 PAGE 1 572 .form 'Main: Send Real-Time Clock Interrupt' 575 577 #IC2 960219 R rbit artc,alert.b ; Clear ALERT bit. 576 #IC2 960219 R rbit artc,alert.b ; Clear ALERT bit. 576 #IC2 960219 R rbit artc,alert.b ; Clear ALERT bit. 576 #IC2 960219 R rbit artc,alert.b ; Clear ALERT bit. 576 #IC2 960219	562 563	Ø1B5	3013	D		jsrl	sndlak	; If so, then send LCD Acknowledge interrupt.	
568 #IBF 3@23 jsrl sndiag ; If so, then send interrupt and data. 569 jmpl chkalt ; No "responses" defined yet; just close loop. 571 jmpl chkalt ; No "responses" defined yet; just close loop. TL/DD/9976 S74 jmpl chkalt ; No "responses" defined yet; just close loop. TL/DD/9976 S74 25-feb-88 10:0 PAGE 1 S77 S74 ; No data transfer; just trigger interrupt and continue. S75 S74 ; No data transfer; just trigger interrupt and continue. S75 S76 #1C2 96@219 R rbit S78 #1C2 96@219 R rbit rcheck that	565 566	Ø18A	3016			jsrl	sndbtn	; If so, then report the change to the CPU.	
570 ØIC1 79 jmpl chkalt ; No "responses" defined yet; just close loop. 571 TL/DD/9976 TL/DD/9976 C ASMHPC, Ver DI-BetaSite (Sep 14 14:30 1987) HPCUPI 25-feb-88 10:0 1 PORT INTERFACE DEMO 10 PORT INTERFACE DEMO Send Real-Time Clock Interrupt 572 .form 'Main: Send Real-Time Clock Interrupt' 575 576 0:02 Soft colspan="2">clear ALERT bit. 576 0:02 Soft colspan="2">Clear ALERT bit. 576 0:02 Soft colspan="2">Soft colspan="2" <tr< td=""><td>568 569</td><td>Ø1BF</td><td>3023</td><td>R</td><td></td><td></td><td></td><td></td><td></td></tr<>	568 569	Ø1BF	3023	R					
SC ASMHPC, Ver D1-BetaSite (Sep 14 14:30 1987) HPCUP1 25-feb-88 10:0 11 PORT INTERFACE DEMO PAGE 1 11:n:send Real-Time Clock Interrupt PAGE 1 572 .form 'Main: Send Real-Time Clock Interrupt' 573 .form 'Main: Send Real-Time Clock Interrupt' 574 ; No data transfer; just trigger interrupt and continue. 575 .form 'Main: Send Real-Time Clock Interrupt' 576 .form 'Main: Send Real-Time Clock Interrupt and continue. 575 .form 'Main: Send Real-Time Clock Interrupt and continue. 576 .form 'Main: Send Real-Time Clock Interrupt and continue. 577 .form 'Main: Send Real-Time Clock Interrupt and continue. 577 .form 'Joint artc,alert.b ; Clear ALERT bit. 578 .gliC2 Sep219 R 580 .jorn rdwait ; Check that UPI interface is ready. 580 .jorn cloop until it is. 580 .jorn cloop until it is. 581 .jorn cloop into 08UF for CPU. 582 .jorn cloop. 581 .jorn cloop.	570	Ø1C1	79			jmpl	chkalt	; No "responses" defined yet; just close loop.	
P1 PORT INTERFACE DEMO PAGE 1 ain: Send Real-Time Clock Interrupt 572 573 .form 'Main: Send Real-Time Clock Interrupt' 574 ; No data transfer; just trigger interrupt and continue. 575 .form 'Main: Send Real-Time Clock Interrupt' 576 g1C2 577 Ø1C2 sndrtc: 577 Ø1C2 Sendrtc: 578 Ø1C2 rbit artc,alert.b ; Clear ALERT bit. 578 Ø1C5 2F R jsrl rdwait ; Check that UPI interface is ready. 580 ; If not, loop until it is. 580 91C6 9711EØ 181 Ø1C6 9711EØ Id obuf,#vrtc ; Load Real-Time Clock vector into OBUF for CPU.									TL/DD/9976-
<pre>sin: Send Real-Time Clock Interrupt fin: Send Real-Time Clock Price Real-Time Clock Price fin: Send Real-Time Clock Price Real-Time Clock Price fin: Send Real-Time Clock Price Real-Time Clock Price fin: Send Real-Time</pre>	POR POR	T IN	TERFACE DEMO		-	30 1987:)	HPCUP I	25-5eb-88 10:0 PAGE 10
573 ; No data transfer; just trigger interrupt and continue. 575 ; Sno data transfer; just trigger interrupt and continue. 575 sndrtc: 576 plC2 577 ØlC2 960219 R 578 ØlC5 2F R 580 ; Check that UP1 interface is ready. 580 ; If not, loop until it is. 581 ØlC6 9711EØ Id obuf,#vrtc 582 ØlC9 3C ret ; Return to main loop.		Send	Real-Time Clo	ck Int	errupt	form	Main: Com	Peal-Time Clock Interrunt!	
575 576 Ø1C2 sndrtc: 576 Ø1C2 sndrtc: ; Clear ALERT bit. 577 Ø1C2 96Ø219 R rbit artc,alert.b 578 Ø1C5 2F R jsrl rdwait 579 ; Check that UP1 interface is ready. 579 ; If not, loop until it is. 580 ; B1C6 9711EØ ld 581 Ø1C9 3C ret ; Return to main loop.	573				: No da				
577 Ø1C2 96Ø219 R rbit artc,elert.b ; Clear ALERT bit. 578 Ø1C5 ZF R jsrl rdwait ; Check that UP1 interface is ready. 579 ; If not, loop until it is. ; 580 ; If not, loop until it is. 580 jsrl obuf,#vrtc ; Load Real-Time Clock vector into OBUF for CPU. 582 Ø1C9 3C ret	575	a107					just ti	ageerrape end contentider	
579 ; If not, loop until it is. 580 ; Load Real-Time Clock vector into OBUF for CPU. 581 Ø1C6 9711EØ ld obuf,#vrtc ; Load Real-Time Clock vector into OBUF for CPU. 582 Ø1C9 3C ret ; Return to main loop.	577	Ø1C2			snartc:				
580 581 Ø1C6 9711EØ ld obuf,#vrtc ; Load Real-Time Clock vector into OBUF for CPU. 582 Ø1C9 3C ret ; Return to main loop.		Ø1C5	2F	R		jsrl	rdwait		
582 Ø1C9 3C ret ; Return to main loop.	58Ø	81 C4	971150			Id	obut thete		
							Jour,#vrtC		

584								
585					.form		Write Acknowledge Interrupt'	
586 587				; No da	ta trans	sfer; just trigge	r interrupt and continue.	
588 589	Ø1CA Ø1CA Ø1CD	96Ø21B	R	sndlak:	rbit jsrl	rdwait	; Clear ALERT bit. ; Check that UPI interface is ready. ; If not, loop until it is.	
	Ø1CE Ø1D1	9717EØ 3C			ld ret	obuf,#vlcdak	; Load LCD-Acknowledge vector into OBUF for CPU. ; Return to main loop.	
								TL/DD/997
UPI POF	RT IŃ	TERFACE DE	etaSite (S EMO on Status		30 1987;) HPC	UP I	25-Feb-88 10:0 PAGE
596 597					.form	'Main: Send Pus	hbutton Status to CPU'	
598	Ø1D2 Ø1D2	2F	R	sndbtn:	jsrl	rdwait	; Check that UP1 interface is ready. ; If not, loop until it is.	
601 602	Ø1D3	9718EØ			ld		; Load BUTTON-DATA vector into OBUF for CPU.	
603	Ø1D6		R		jsrl	rdwait	; Check that UPI interface is ready. ; If not, loop until it is.	
607 608 609	Ø1DA Ø1DD	960918 8C18E9 969218 960998 3C	R R		rbit ld rbit sbit ret	obuf,swlsnt abutton,alert.b gie,enir	; *** Begin Indivisible Sequence *** ; Load Pushbutton Data Byte into OBUF for CPU. ; Clear ALERT bit. ; *** End Indivisible Sequence *** ; Return to main loop.	
012								TL/DD/997

Г

613					form	Main: Send D	impostic Interrupt to CPU	
614					.form	Main: Send D	iagnostic Interrupt to CPU'	
615 Ø 616 Ø	1E4 164	25	R	sndiag:	jsrl	rdwait	; Wait for UPI interface ready.	
617 Ø	1E5	971DEØ			ld	obuf,#vdiag	; Load vector into OBUF for CPU.	
618 0			R		jsrl	rdwait	; Wait for UPI interface ready.	
		96DØ18 8C1DEØ	R		rbit ld	gie,enir obuf,dsevc	; *** Begin Indivisible Sequence *** ; Transfer Severity Code.	
621 Ø	1EF	97001D	R		ld	dsevc,#Ø	; Clear it.	
622 B		881E 97001E	R		ld Id	A,derrc derrc,#0	; Get Error Code. ; Clear it.	
624 Ø	1F7	96Ø21A	R		rbit	adiag,alert.b	; Clear ALERT bit.	
625 👂	1FA	960008			sbit	gie,enir	<pre>*** End Indivisible Sequence ***</pre>	
626 Ø 627 Ø			R		jsrl st	rdwait A,obuf	; Wait for UPI interface ready. ; Transfer Error Code.	
628 02			R		jsrl	rdwait	; Wait for UPI interface ready.	
629 63Ø						; Kennen ; COMP	ining bytes will have meaning only for mand errors.	
631 02		8C1FEØ	R		ld	obuf,dbyte	; Transfer Byte Received.	
632 Ø2 633 Ø2	204 205	2F 8C2ØEØ	R		jsrl ld	rdwait obuf,dccmd	; Wait for UPI interface ready. ; Transfer Current Command.	
634 Ø2	208	2F	R		jsrl	rdwait	; Wait for UPI interface ready.	
635 Ø2 636 Ø2		8C21EØ 3C	R		ld ret	obuf,dqual	; Transfer Command Count.	
637	C pro	36			ret		; Return to main program loop.	
								TL/DD/997
		Ver D1-BetaS ERFACE DEMO	ite (Se	p 14 14:	30 1987) H	PCUPI	25-705-88 10:0 PAGE 2
IPI (13)	Int	errupt: Dat	a from	CPU				
638 639					.form	'UPI (I3) Inter	rrupt: Data from CPU'	
6410 FI	FF8	ØDØ2	R		.ipt	3,upiwr	; Declare upiwr as vector for Interrupt 3.	
641 642 Ø	7 9 0			upiwr:			; Write Strobe received from CPU.	
643 0	2ØD			upini.	push	A	; write strobe received from CPU. ; Save Context	
644 0					push	psw		
645 646 Ø	211	8CE600	R		ld	upicsv.b,upic	; Save UPIC register image for LAØ bit test.	
647								
648 Ø 649 Ø		961Ø17 9400	R		ifbit jmpl	cmdemp,curcmd firstc	; If expecting first byte of a command, ; then go process it as such.	
65Ø					յութե	THEL	; then go process it as such.	
651 02	219	88FØ			lđ	A, ibuf	; If not, input it for entry into cpubuf.	
652 653 Ø2	21B	9CA5			ifeq	A,#x'A5	; Check for RESET command.	
654 02	21D	46			jp	lcrst		
655 Ø2 656	216	960012	R		ifbit	laØ,upicsv.b	; Check for command argument written to proper ; address.	
657 Ø					jp .	lcord	; If so, go process as a normal argument.	
658 Ø2 659	222	3622			jsrl	hangup	; If not, process as a FATAL error, generating ; !DIAG interrupt.	
660								
661 Ø2 662 Ø2		96E612		lcrst:		laø,upic	; Continue checking for a RESET command.	
663 Ø	228	94C6			jp jmpl	lcord xreset	; If so, go reset the HPC.	
664				I ada				
665 Ø	22A	ADØ48E	к	lcord:	x	A, [cpuad].b	; If not, place it in next available cpubuf ; entry.	
667 Ø			R		inc	cpuad	, entry.	
668 Ø2		8A11 B4Ø1ØF	R		decsz jmpl	numexp upwret	; If not final byte of command, then return.	
670			_					
671 Ø2		881Ø 96C816	R	lastc:	ld ifbit	A,curcmd getcnt,A.b	; Else, process current command. ; Check if extended collection is being made.	
673 02	239	47			jp	lastc1	; If not, then:	
		96100F	R R		sbit	cmdemp, curcmd	; Set command slot available again.	
676		B7ØØØ6 Ø 4	ĸ		ld	cpuad,#cpubuf	; Reset CPU buffer pointer to beginning.	
677 02				lastc1:	-1-1	A,#x'1F	; Mask off flag bits.	
678 Ø2 Ø2	245				SIL	•	; scale by two, and then	
679 Ø2	245				.odd			
680 02 681	245	EC			jidw		; jump based on command value:	
682 Ø2				lastab:		lcinit	; θ = INITIALIZE command.	
683 Ø2					.ptw .ptw	lcslcv lcslcd	; 1 ≈ SET-CONTRAST command. ; 2 = SEND-LCD command.	
684 Ø2 685 Ø2	24C	8000			.ptw .ptw	lcsled	; 5 = SEND-LED command.	
686 Ø2	24E	F300			.ptw	illc	; (BEEP command has only one byte. Error.)	
								TL/DD/997

		terrupt: Data	from		30 1987) H			25-deb-88 19:95 PAGE 21
687									
688 689						; Proce	ess INITI	IALIZE Command.	
690		97Ø11C	R	lcinit:	ld	rtevs,#x'Ø1	; Enabl	le only Real-Time Clock interrupts, but	
		820006DC	R		ifeq	cpubuf.b,#Ø	; disa	able them again if	
		961C18 8CØ61A	R		rbit ld	rtcenb,rtevs		command argument is zero. argument into Real-Time	
694								ck interval.	
695 696	Ø25D	809618	R		ld	rtccnt,cpubuf.	Put a	argument into Real-Time	
	Ø26Ø	B601900C			sbit	titie,tmmdl		ck count. .e Timer T1 interrupt, if not already	
698							; enab	oled.	
699 700	Ø264	B6Ø19Ø1E			rbit	tistp,tmmdl	; Start	timer, if not already running.	
701	Ø268	B7000002	R		ld	alert.w,#Ø	: Set r	no events pending.	
702		970017							
		970018	R		ld ld	swlast,#0 swisnt,#0		up initial switch values. • current and last sent)	
705							,		
706 707	9272	94CF			jmpl	upwret	; Retur	'n.	
708									
709						; Proce	ss SET-C	CONTRAST Command.	
710 711	0274	8896	R	lcslcv:	Id	A,cpubuf.b	· Load	LCD Voltage latch (Contrast) from byte	
712								plied by CPU.	
713	Ø276	Ø1			comp	A	; (3-bi	t value is in complemented form.)	
715	ø279	99ø7 82f812D9	R		and and	A,#x'07 lcvs,#x'F8	; USE O	wly lower three bits. field in memory image.	
716	Ø27D	80C812DA	R		00	lcvs,A.b	; Merge	e new field into image.	
		8C12E1 96E3Ø9	R		ld sbit	portah,lcvs lcvclk,portbh	; Place	on Port A (input to latch).	
719	Ø287	96E319			rbit	lcvclk,portbh	; CLOCK	taten.	
	Ø28A	9487			jmpl	upwret			
721 722									
723						; Proce	ss SEND-	LCD Command.	
724	a28c	961016	р	loglade		astant summed		Oberth from filmet an event of the state	
726			ĸ	lcsicd:	jmpl	getcnt,curcmd lcslc1		; Check for first or second collection ; phase.	
727					1			, pluse.	
728 (729	0291			lcslc2:				d phase: begins execution of the LCD mand.	
	0291	A1060038AB	R		ld	lcdbuf.w,cpubuf		; Copy CPU buffer to LCD string buffer.	
		A108003AAB	R		ld	lcdbuf+2.w.cpub	uf+2.w	, .,,	
		A10A003CAB A10C003EAB	R		ld ld	<pre>lcdbuf+4.w,cpub lcdbuf+6.w,cpub</pre>			
734 (8C1416	Ř		id	lcdsct,lcdnum	u1+0.w	; Move number of characters to string	
735 736 (0.240	9014	R		inc	lcdsct		; count byte	
150,	PERO	0,10	r.		III.	luser		; (incremented by one because of	
									TL/DD/9976
PI POR	T IÑ	Ver D1-BetaSit TERFACE DEMO terrupt: Data			30 1987;) ні	CUPI		TL/DD/9976 25-Feb-88 10:05 PAGE 22
PI POR PI (13 737	T IÑ	TERFACE DEMO			30 1987;) н	PCUPI	; extra interrupt occurring after	25-Feb-88 10:05
PI POR PI (13 737 738	T IÑ) In	TERFACE DEMO terrupt: Data	from				PCUP I	; last character has been sent).	25-Feb-88 10:05
PI POR PI (13 737 738 739 749	02AA	TERFACE DEMO			3Ø 1987; ld ld) Hi lcdsix,#lcdbuf lcdsfg,lcdfgs	PCUP I		25-Feb-88 10:05
PI POR PI (13 737 738 739 749 741	Ø2AA Ø2AE	TERFACE DEMO terrupt: Data B700380E 8C1315	from R		ld ld	lcdsix,#lcdbuf lcdsfg,lcdfgs	CUPI	; last character has been sent). ; Set string pointer to first byte. ; Move flag bits to string location.	25-Feb-88 10:05
PI POR PI (13 737 738 739 749 741 742	02AA 02AA 02AE 02B1	TERFACE DEMO terrupt: Data B700380E	from R		ld	lcdsix,#lcdbuf lcdsfg,lcdfgs r6,#x'FFFF	CUP I	; last character has been sent). ; Set string pointer to first byte. ; Move flag bits to string location. ; Set up R6 and T6 to trigger string ; transfer.	25-Feb-88 10:05
PI POR PI (13 737 738 739 749 749 741 742 743 744	02AA 02AA 02AE 02B1 02B7 02BC	TERFACE DEMO terrupt: Data B700380E 8C1315 87FFFF014AAB 83000148AB B6015108	from R		ld ld ld sbit	lcdsix,#lcdbuf lcdsfg,lcdfgs r6,#9 t6;#9 t6tie,pwmdh	PCUP I	; last character has been sent). ; Set string pointer to first byte. ; Move flag bits to string location. ; Set up R6 and T6 to trigger string ; transfer. ; Enable timer T6 interrupt.	25-Feb-88 10:05
PI POR PI (13 737 738 739 749 741 742 743 744 745	02AA 02AA 02AE 02B1 02B7 02BC	TERFACE DEMO terrupt: Data B700380E 8C1315 87FFFF014AAB 83000148AB	from R		ld ld ld	Lcdsix,#lcdbuf lcdsfg,lcdfgs r6,#x'FFFF t6,#0	PCUPT	 ; last character has been sent). ; Set string pointer to first byte. ; Nove flag bits to string location. ; Set up R6 and T6 to trigger string ; transfer. ; Enable timer T6 interrupt. ; Start timer to trigger (immediate) 	25-Feb-88 10:05
PI POR PI (13 737 738 739 749 741 742 743 744 745 746	02AA 02AA 02AE 02B1 02B7 02BC 02C0	TERFACE DEMO terrupt: Data B700380E 8C1315 87FFFF014AAB 83000148AB B6015108	from R		ld ld ld sbit	lcdsix,#lcdbuf lcdsfg,lcdfgs r6,#9 t6;#9 t6tie,pwmdh	PCUPI	; last character has been sent). ; Set string pointer to first byte. ; Move flag bits to string location. ; Set up R6 and T6 to trigger string ; transfer. ; Enable timer T6 interrupt.	25-Feb-88 10:05
PI POR PI (13 737 738 739 749 741 742 743 744 745 746 747 748	02AA 02AA 02AE 02B1 02B7 02BC 02C0 02C4	TERFACE DEMO terrupt: Data 8700380E 8C1315 87FFFF014AAB 83000148AB 86015108 8601511A	from R	CPU	ld ld ld sbit rbit jmpl	lcdsix,#lcdbuf lcdsfg,lcdfgs r6,#2/FFF t6,#0 t6tie,pwmdh t6stp,pwmdh		 last character has been sent). Set string pointer to first byte. Move flag bits to string location. Set up R6 and T6 to trigger string transfer. Enable timer T6 interrupt. Start timer to trigger (immediate) interrupt from timer T6. 	25-Feb-88 10:05
PI POR PI (13 737 738 739 749 741 742 743 744 745 745 745 747 748 749	02AA 02AA 02AE 02B1 02B7 02BC 02C0 02C4	TERFACE DEMO terrupt: Data 8700380E 8C1315 87FFFF014AAB 83000148AB 86015108 8601511A	from R		ld ld ld sbit rbit jmpl	lcdsix,#lcdbuf lcdsfg,lcdfgs r6,#2/FFF t6,#0 t6tie,pwmdh t6stp,pwmdh	; First	 last character has been sent). Set string pointer to first byte. Move flag bits to string location. Set up R6 and T6 to trigger string transfer. Enable timer T6 interrupt. Start timer to trigger (immediate) interrupt from timer T6. 	25-Feb-88 10:05
PI POR PI (13 737 738 739 749 741 742 744 745 744 745 746 747 748 749	02AA 02AA 02AE 02B1 02B7 02BC 02C0 02C4 02C6	TERFACE DEMO terrupt: Data 8700380E 8C1315 87FFFF014AAB 83000148AB 86015108 8601511A	from R	CPU	ld ld ld sbit rbit jmpl	lcdsix,#lcdbuf lcdsfg,lcdfgs r6,#2/FFF t6,#0 t6tie,pwmdh t6stp,pwmdh	; First ; mor	<pre>; last character has been sent). ; Set string pointer to first byte. ; Move flag bits to string location. ; Set up R6 and T6 to trigger string ; transfer. ; Enable timer T6 interrupt. ; Start timer to trigger (immediate) ; interrupt from timer T6. t phase: Prepare to collect up to 8 re bytes of command. ; Get flag bits supplied by CPU.</pre>	25-Feb-88 10:05
PI POR PI (13 737 738 739 749 749 742 743 744 745 746 746 747 748 749 759 759 759	02AA 02AA 02AE 02B1 02B7 02BC 02C0 02C6 02C6 02C6	TERFACE DEMO terrupt: Data 8790380E 8C1315 87FFFF014AAB 83000148AB B6015108 B601511A 947D	from R R	CPU	ld ld ld sbit rbit jmpl	lcdsix,#lcdbuf lcdsfg,lcdfgs r6,#2,FFFF t6,#9 t6tie,pwmdh t6stp,pwmdh upwret	; First ; mor	<pre>; last character has been sent). ; Set string pointer to first byte. ; Wove flag bits to string location. ; Set up R6 and T6 to trigger string ; transfer. ; Enable timer T6 interrupt. ; Start timer T6 interrupt. ; Start timer to trigger (immediate) ; interrupt from timer T6. tphase: Prepare to collect up to 8 e bytes of command.</pre>	25-Feb-88 10:05
PI POR PI (13 737 738 739 749 741 742 743 744 745 746 747 746 747 748 759 751 753	02AA 02AA 02AE 02B1 02B7 02BC 02C6 02C6 02C6 02C6 02C6	TERFACE DEMO terrupt: Data 8700380E 8C1315 87FFFF014AAB 83000148AB 86015108 8601511A 947D 8C0613 8C0613 8C0714	from R R R	CPU	ld ld sbit rbit jmpl ld	Lodsix,#Lodbuf Lodsfg,Lodfgs r6,#x'FFFF t6,#9 t6tie,pwmdh t6stp,pwmdh upwret Lodfgs,cpubuf.t Lodfgs,cpubuf.t	;First ; mor).b	<pre>; last character has been sent). ; Set string pointer to first byte. ; Move flag bits to string location. ; Set up R6 and T6 to trigger string ; transfer. ; Enable timer T6 interrupt. ; Start timer to trigger (immediate) ; interrupt from timer T6. t phase: Prepare to collect up to 8 re bytes of command. ; Get flag bits supplied by CPU. ; Get character count from CPU.</pre>	25-Feb-88 10:05
PI POR PI (13 737 738 739 740 741 742 743 744 745 744 745 746 746 747 759 752 753 755	02AA 02AA 02AE 02B1 02B7 02BC 02C6 02C6 02C6 02C6 02C6	TERFACE DEMO terrupt: Data 8700380E 8C1315 87FFFF014AAB 83000148AB 86015188 8601511A 947D 8C0613	from R R	CPU	ld ld sbit rbit jmpl	Lodsix,#Lodbuf Lodsfg,Lodfgs r6,#x'FFFF t6,#9 t6tie,pwmdh t6stp,pwmdh upwret Lodfgs,cpubuf,L	;First ; mor).b	<pre>; last character has been sent). ; Set string pointer to first byte. ; Move flag bits to string location. ; Set up R6 and T6 to trigger string ; transfer. ; Enable timer T6 interrupt. ; Start timer to trigger (immediate) ; interrupt from timer T6. tphase: Prepare to collect up to 8 re bytes of command. ; Get flag bits supplied by CPU. ; Get character court from CPU. ; Request another collection of ; data from the CPU (the string of ; data from the CPU (the string of</pre>	25-Feb-88 10:05
PI POR PI (13 737 738 739 749 741 743 744 745 746 747 748 747 748 759 751 755 755	11 IN 10	TERFACE DEMO terrupt: Data 8700380E 801315 87FFFF014AAB 83000148AB B6015108 B6015108 B601511A 947D 800613 800714 801411	from R R R	CPU	ld ld ld sbit rbit jmpl ld ld	Lodsix,#Lodbuf Lodsfg,Lodfgs r6,#x:FFF t6,#0 t6tie,pwmdh t6tie,pwmdh upwret Lodfgs,cpubuf.t Lodnum,cpubuf+ numexp,Lodnum.t	;First ; mor).b	<pre>; last character has been sent). ; Set string pointer to first byte. ; Move flag bits to string location. ; Set up R6 and T6 to trigger string ; transfer. ; Enable timer T6 interrupt. ; Start timer to trigger (immediate) ; interrupt from timer T6. t phase: Prepare to collect up to 8 re bytes of command. ; Get flag bits supplied by CPU. ; Get character count from CPU. ; Request another collection of ; data from the CPU (the string of ; data for the panel).</pre>	25-Feb-88 19:95 PAGE 22
PI POR PI (13 737 7389 749 741 743 744 744 744 744 744 746 747 746 752 753 755 755 755 756 758	T IN 02AA 02AE 02B1 02B7 02CC 02CC 02CC 02CC 02CC 02CC	TERFACE DEMO terrupt: Data 8700380E 8C1315 87FFFF014AAB 83000148AB B601510A 9601510A 947D 8C00613 8C00714 8C1411 B70000604	from R R R	CPU	ld ld sbit rbit jmpl ld ld ld	Lcdsix,#Lcdbuf Lcdsfg,Lcdfgs r6,#x'FFFF t6,#0 t6tie,pwmdh t6tie,pwmdh upwret Lcdfgs,cpubuf.t Lcdnum,cpubuf+ numexp,Lcdnum.t	;First ; mor).b	<pre>; last character has been sent). ; Set string pointer to first byte. ; Move flag bits to string location. ; Set up R6 and T6 to trigger string ; transfer. ; Enable timer T6 interrupt. ; Start timer to trigger (immediate) ; interrupt from timer T6. t phase: Prepare to collect up to 8 re bytes of command. ; Get flag bits supplied by CPU. ; Get character count from CPU. ; Get character collection of ; data from the CPU (the string of ; data from the CPU (the string of ; data for the panel). ; Reset CPU collection pointer to start ; of command buffer.</pre>	25-Feb-88 19:95 PAGE 22
PI POR PI (13 737 738 739 741 743 744 745 744 745 744 745 751 755 755 755 755 755 755 755 755 75	T IN 02AA 02AE 02B1 02B7 02CC 02CC 02CC 02CC 02CC 02CC	TERFACE DEMO terrupt: Data 8700380E 801315 87FFFF014AAB 83000148AB B6015108 B6015108 B601511A 947D 800613 800714 801411	from R R R	CPU	ld ld ld sbit rbit jmpl ld ld	Lodsix,#Lodbuf Lodsfg,Lodfgs r6,#x:FFF t6,#0 t6tie,pwmdh t6tie,pwmdh upwret Lodfgs,cpubuf.t Lodnum,cpubuf+ numexp,Lodnum.t	;First ; mor).b	<pre>; last character has been sent). ; Set string pointer to first byte. ; Wove flag bits to string location. ; Set up R6 and T6 to trigger string ; transfer. ; Enable timer T6 interrupt. ; Start timer to trigger (immediate) ; interrupt from timer T6. tphase: Prepare to collect up to 8 re bytes of command. ; Get flag bits supplied by CPU. ; Get character count from CPU. ; Get character count from CPU. ; Request another collection of ; data from the CPU (the string of ; data from the CPU (the string of ; data from the CPU (the string of ; data from the collection pointer to start ; of command buffer. ; Declare that it will be the final</pre>	25-Feb-88 19:95 PAGE 22
PI POR PI (13 737 7389 749 7412 7445 7445 7445 7445 7445 7445 7445 7445 7551 7556 7557 7558 7569 769	11 IN 10	TERFACE DEMO terrupt: Data 8700380E 8C1315 87FFFF014AAB 83000148AB 86015108 8601511A 947D 8C0613 8C0714 8C1411 87000604 96101E	from R R R R R R	CPU	ld ld sbit rbit jmpl ld ld ld	Lodsix,#Lodbuf Lodsfg,Lodfgs r6,#x'FFFF t6,#0 t6tie,pwmdh upwret Lodfgs,cpubuf.t Lodfgs,cpubuf.t Lodnum,cpubuf+' numexp,Lodnum.t cpuad,#cpubuf getont,curcmd	;First ; mor).b	<pre>; last character has been sent). ; Set string pointer to first byte. ; Move flag bits to string location. ; Set up R6 and T6 to trigger string ; transfer. ; Enable timer T6 interrupt. ; Start timer to trigger (immediate) ; interrupt from timer T6. t phase: Prepare to collect up to 8 re bytes of command. ; Get flag bits supplied by CPU. ; Get character count from CPU. ; Get character collection of ; data from the CPU (the string of ; data from the CPU (the string of ; data for the panel). ; Reset CPU collection pointer to start ; of command buffer.</pre>	25-Feb-88 19:95 PAGE 22
PI POR 737 738 740 740 742 743 742 743 744 745 744 745 744 745 744 745 744 745 754 751 752 754 755 757 758 757 758 769 761 762	11 IN 10	TERFACE DEMO terrupt: Data 8700380E 8C1315 87FFFF014AAB 83000148AB B601510A 9601510A 947D 8C00613 8C00714 8C1411 B70000604	from R R R R R R	CPU	ld ld sbit rbit jmpl ld ld ld	Lcdsix,#Lcdbuf Lcdsfg,Lcdfgs r6,#x'FFFF t6,#0 t6tie,pwmdh t6tie,pwmdh upwret Lcdfgs,cpubuf.t Lcdnum,cpubuf+ numexp,Lcdnum.t	;First ; mor).b	<pre>; last character has been sent). ; Set string pointer to first byte. ; Wove flag bits to string location. ; Set up R6 and T6 to trigger string ; transfer. ; Enable timer T6 interrupt. ; Start timer to trigger (immediate) ; interrupt from timer T6. tphase: Prepare to collect up to 8 re bytes of command. ; Get flag bits supplied by CPU. ; Get character count from CPU. ; Get character count from CPU. ; Request another collection of ; data from the CPU (the string of ; data from the CPU (the string of ; data from the CPU (the string of ; data from the collection pointer to start ; of command buffer. ; Declare that it will be the final</pre>	25-Feb-88 19:95 PAGE 22
PI POR PI (13 737 738 739 740 741 742 743 744 745 744 745 744 745 744 745 744 745 754 759 759 755 756 757 758 759 757 758 759 761 762 763	11 IN 10	TERFACE DEMO terrupt: Data 8700380E 8C1315 87FFFF014AAB 83000148AB 86015108 8601511A 947D 8C0613 8C0714 8C1411 87000604 96101E	from R R R R R R	CPU	ld ld sbit rbit jmpl ld ld ld	Lcdsix,#Lcdbuf Lcdsfg,Lcdfgs r6,#%'FFFF t6,#% t6stp,pwmdh upwret Lcdfgs,cpubuf,t Lcdnum,cpubuf+ numexp,Lcdnum.t cpuad,#cpubuf getcnt,curcmd upwret	; First ; mor .b	<pre>; last character has been sent). ; Set string pointer to first byte. ; Wove flag bits to string location. ; Set up R6 and T6 to trigger string ; transfer. ; Enable timer T6 interrupt. ; Start timer to trigger (immediate) ; interrupt from timer T6. t phase: Prepare to collect up to 8 e bytes of command. ; Get flag bits supplied by CPU. ; Get character count from CPU. ; Get character count from CPU. ; Request another collection of ; data for the DFU (the string of ; deta for the panel). ; Reset CPU collection pointer to start ; Of command buffer. ; Declare that it will be the final ; collection.</pre>	25-Feb-88 19:95 PAGE 22
PI POR 738 739 740 741 742 741 742 745 744 745 744 745 744 745 744 745 744 745 754 747 755 757 758 757 758 757 758 759 760 757 759 760 759 763 763 764	11 IN 10	TERFACE DEMO terrupt: Data 8700380E 8C1315 87FFFF014AAB 83000148AB 86015108 8601511A 947D 8C0613 8C0714 8C1411 87000604 96101E	from R R R R R R	CPU	ld ld sbit rbit jmpl ld ld ld	Lcdsix,#Lcdbuf Lcdsfg,Lcdfgs r6,#%'FFFF t6,#% t6stp,pwmdh upwret Lcdfgs,cpubuf,t Lcdnum,cpubuf+ numexp,Lcdnum.t cpuad,#cpubuf getcnt,curcmd upwret	; First ; mor .b	<pre>; last character has been sent). ; Set string pointer to first byte. ; Wove flag bits to string location. ; Set up R6 and T6 to trigger string ; transfer. ; Enable timer T6 interrupt. ; Start timer to trigger (immediate) ; interrupt from timer T6. tphase: Prepare to collect up to 8 re bytes of command. ; Get flag bits supplied by CPU. ; Get character count from CPU. ; Get character count from CPU. ; Request another collection of ; data from the CPU (the string of ; data from the CPU (the string of ; data from the CPU (the string of ; data from the collection pointer to start ; of command buffer. ; Declare that it will be the final</pre>	25-Feb-88 19:95 PAGE 22
IPI POR 737 738 738 738 739 741 741 742 743 744 744 745 750 756 755 755 758 759 760 763 764 763 764 763 764 763 764 765 763 764 765 765	11 IN 10 10 10 10 10 10 10 10 10 10	TERFACE DEMO terrupt: Data 8700380E 8C1315 87FFFF014AAB 83000148AB 86015188 8601511A 947D 8C0613 8C0714 8C1411 87000604 96101E 9468	from R R R R R R R R R	CPU	ld ld sbit rbit jmpl ld ld ld ld ld	Lcdsix,#Lcdbuf Lcdsfg,Lcdfgs r6,#%'FFFF t6,#% t6stp,pwmdh upwret Lcdfgs,cpubuf,t Lcdnum,cpubuf+ numexp,Lcdnum.t cpuad,#cpubuf getcnt,curcmd upwret	; First ; mor .b ess SEND ; Load	<pre>; last character has been sent). ; Set string pointer to first byte. ; Move flag bits to string location. ; Set up R6 and T6 to trigger string ; transfer. ; Enable timer T6 interrupt. ; Start timer to trigger (immediate) ; interrupt from timer T6. tphase: Prepare to collect up to 8 re bytes of command. ; Get flag bits supplied by CPU. ; Get character count from CPU. ; Request another collection of ; data from the CPU (the string of ; data from the interval). ; Declare that it will be the final ; collection. -LED Command. LED latch from byte supplied by CPU.</pre>	25-Feb-88 19:95 PAGE 22
IPI POR 737 7 738 738 739 738 739 738 739 741 742 743 743 744 744 743 745 746 747 743 744 744 745 746 747 748 748 749 751 755 756 757 758 755 756 767 768 769 760 761 764 765 764 765 764 767	TIN 11 12 <td>TERFACE DEMO terrupt: Data 8700380E 8201315 87FFFF014AAB 83000148AB 8601510A 947D 8C0613 8C0714 8C1411 87000604 96101E 9468</td> <td>from R R R R R R R R R</td> <td>CPU</td> <td>ld ld sbit rbit jmpl ld ld ld ld ld ld ld ld ld ld ld ld ld</td> <td>Lcdsix,#Lcdbuf Lcdsfg,Lcdfgs r6,#x'FFFF t6,#0 t6tie,pwmdh t6stp,pwmdh upwret Lcdfgs,cpubuf.t Lcdnum,cpubuf+ numexp,Lcdnum.t cpusd,#cpubuf getcnt,curcmd upwret ; Proce A,cpubuf.b</td> <td>; First ; mor </td> <td><pre>; last character has been sent). ; Set string pointer to first byte. ; Nove flag bits to string location. ; Set up R6 and T6 to trigger string ; transfer. ; Enable timer T6 interrupt. ; Start timer to trigger (immediate) ; interrupt from timer T6. t phase: Prepare to collect up to 8 e bytes of command. ; Get flag bits supplied by CPU. ; Get character count from CPU. ; Get character count from CPU. ; Get character count from cPU. ; Request another collection of ; data from the CPU (the string of ; data from the CPU (the string of ; data for the panel). ; Reset CPU collection pointer to start ; of command buffer. ; Declare that it will be the final ; collection. LED Latch from byte supplied by CPU. a goest to LED's in complemented form.)</pre></td> <td>25-Feb-88 19:95 PAGE 22</td>	TERFACE DEMO terrupt: Data 8700380E 8201315 87FFFF014AAB 83000148AB 8601510A 947D 8C0613 8C0714 8C1411 87000604 96101E 9468	from R R R R R R R R R	CPU	ld ld sbit rbit jmpl ld ld ld ld ld ld ld ld ld ld ld ld ld	Lcdsix,#Lcdbuf Lcdsfg,Lcdfgs r6,#x'FFFF t6,#0 t6tie,pwmdh t6stp,pwmdh upwret Lcdfgs,cpubuf.t Lcdnum,cpubuf+ numexp,Lcdnum.t cpusd,#cpubuf getcnt,curcmd upwret ; Proce A,cpubuf.b	; First ; mor 	<pre>; last character has been sent). ; Set string pointer to first byte. ; Nove flag bits to string location. ; Set up R6 and T6 to trigger string ; transfer. ; Enable timer T6 interrupt. ; Start timer to trigger (immediate) ; interrupt from timer T6. t phase: Prepare to collect up to 8 e bytes of command. ; Get flag bits supplied by CPU. ; Get character count from CPU. ; Get character count from CPU. ; Get character count from cPU. ; Request another collection of ; data from the CPU (the string of ; data from the CPU (the string of ; data for the panel). ; Reset CPU collection pointer to start ; of command buffer. ; Declare that it will be the final ; collection. LED Latch from byte supplied by CPU. a goest to LED's in complemented form.)</pre>	25-Feb-88 19:95 PAGE 22
IPI POR 737 737 738 738 738 738 738 738 738 738 738 742 743 742 743 744 745 745 746 759 759 759 759 755 756 757 758 759 769 762 757 764 765 765 765 765 765 765 765 765 765 765 765 765 765 765 765 765 765 765 765 765 766 765 766 765 768	T IN 02AA 02A 02B1 02B2 02C4 02CC 02C4 02CC 02C4 02CC 02C5 02CC 02C6 02CC 02C7 02CC 02C7 02CC 02C7 02CC 02C7 02CC 02C7 02CC 02C7 02CC	TERFACE DEMO terrupt: Data 8700380E 8C1315 87FFFF014AAB 83000148AB 8601511A 947D 8C0613 8C0714 8C1411 87000604 96101E 9468 8806 01 8806	from R R R R R R R R R	CPU	ld ld sbit rbit jmpl ld ld ld ld ld ld ld ld ld somp st	Lodsix,#Lodbuf Lodsfg,Lodfgs r6,#x'FFF t6,#9 t6tie,pwmdh upwret Lodfgs,cpubuf.t Lodfgs,cpubuf.t Lodnum,cpubuf+ numexp,Lodnum.t cpuad,#cpubuf getont,curomd upwret ; Proce A,cpubuf.b A A,portah	; First ; mor .b ; Load ; Clata ; Plata	<pre>; last character has been sent). ; Set string pointer to first byte. ; Wove flag bits to string location. ; Set up R6 and T6 to trigger string ; transfer. ; Enable timer T6 interrupt. ; Start timer to trigger (immediate) ; interrupt from timer T6. tphase: Prepare to collect up to 8 re bytes of command. ; Get flag bits supplied by CPU. ; Get character count from CPU. ; Request another collection of ; data from the CPU (the string of ; data from the ISU (the string of ; data from the isupplied by CPU. a goes to LED's in complemented form.) e new value on Port A (input to latch).</pre>	25-Feb-88 19:95 PAGE 22
PI POR PI (13 738 738 738 738 738 738 738 738 738 73	T IN 02AA 02AA 02BA 02CA 02CCF 02CCF 02CF 02CCF 02CCF 02CCF 02CD8 02DA 02D8 02DB 02CD8 02CP	TERFACE DEMO terrupt: Data 8700380F 8C1315 87FFFF014AAB 83000148AB B6015188 B601511A 947D 8C0613 8C0714 8C1411 B7000604 96101E 946B 8806 01 8851 96E31E	from R R R R R R R R R	CPU	ld ld sbit rbit jmpl ld ld ld ld rbit sbit sbit	Lcdsix,#Lcdbuf Lcdsfg,Lcdfgs r6,#x'FFFF t6,#0 t6tie,pwmdh upwret Lcdfgs,cpubuf.t Lcdfgs,cpubuf.t Lcdnum,cpubuf+ numexp,Lcdnum.t cpuad,#cpubuf getcnt,curcmd upwret ; Proce A,cpubuf.b A A,portah Ledclk,portbh	; First ; mor .b ; Load ; Clata ; Plata	<pre>; last character has been sent). ; Set string pointer to first byte. ; Nove flag bits to string location. ; Set up R6 and T6 to trigger string ; transfer. ; Enable timer T6 interrupt. ; Start timer to trigger (immediate) ; interrupt from timer T6. t phase: Prepare to collect up to 8 e bytes of command. ; Get flag bits supplied by CPU. ; Get character count from CPU. ; Get character count from CPU. ; Get character count from cPU. ; Request another collection of ; data from the CPU (the string of ; data from the CPU (the string of ; data for the panel). ; Reset CPU collection pointer to start ; of command buffer. ; Declare that it will be the final ; collection. LED Latch from byte supplied by CPU. a goest to LED's in complemented form.)</pre>	25-Feb-88 19:95 PAGE 22
PI POR PI (13 738 738 738 738 738 738 738 738 738 73	T IN 02AA 02AA 02BA 02CA 02CCF 02CCF 02CF 02CCF 02CCF 02CCF 02CD8 02DA 02D8 02DB 02CD8 02CP	TERFACE DEMO terrupt: Data 8700380E 8201315 87FFFF914AAB 83000148AB 8601511A 947D 820013 820714 820411 87000604 96101E 9468 8806 01 8806 01 8805 91 8805	from R R R R R R R R R	CPU	ld ld sbit rbit jmpl ld ld ld ld ld ld sbit	Lcdsix,#Lcdbuf Lcdsfg,Lcdfgs r6,#x'FFFF t6,#0 t6tie,pumdh upwret Lcdfgs,cpubuf.t Lcdnum,cpubuf+' numexp,Lcdnum.t cpuad,#cpubuf getcnt,curcmd upwret ; Proce A,cpubuf.b A A,portah Ledclk,portbh	; First ; mor .b ; Load ; Clata ; Plata	<pre>; last character has been sent). ; Set string pointer to first byte. ; Wove flag bits to string location. ; Set up R6 and T6 to trigger string ; transfer. ; Enable timer T6 interrupt. ; Start timer to trigger (immediate) ; interrupt from timer T6. tphase: Prepare to collect up to 8 re bytes of command. ; Get flag bits supplied by CPU. ; Get character count from CPU. ; Request another collection of ; data from the CPU (the string of ; data from the ISU (the string of ; data from the isupplied by CPU. a goes to LED's in complemented form.) e new value on Port A (input to latch).</pre>	25-Feb-88 19:95 PAGE 22

		Ver D1-BetaSite IERFACE DEMO of First Byte of) H		25-Feb-88 10:05 PAGE 23
774					.form	'Processing of	First Byte of Command (Code)'	
775 776 777 778 779 789						; Longer comma	mmands are processed in this section. ands are scheduled for collection of bytes, and are processed in routines	
781 782 783	Ø2E5 Ø2E7 Ø2EA	960012	R	firstc:	ifbit	A,ibuf laØ,upicsv.b	; Get command from UPI port. ; Check for out-of-sequence condition ; (argument instead of command).	
785	PCLA	JOEX			jsrl	hangup	; If so, process as a FATAL error (previous ; command was too short).	
786 787						; Proce	essing of RESET command.	
790	Ø2EC Ø2EE Ø2EF	41			ifeq jp jp	A,#x'A5 xreset fcord	; Check for RESET command.	
793 794							; This code is entered whenever a RESET ; command is received.	
795	Ø2FØ	971DEØ		xreset:	ld	abud thudian		
797 798	Ø2F3 Ø2F4	2F	R		isrl Id	obuf,#vdiag rdwait A,#Ø	; Present dummy value for CPU, ; (in case a value was already in OBUF), ; and wait for it to be read by CPU. ; Initialize registers.	
801 802	02F6 02F8 02FA 02FC	ABFØ ABF2			st st st st	A,upic A,ibuf.w A,dirb	; (Actually all of DIRA.)	
8ø4 8ø5 8ø6	Ø2FE Ø3ØØ Ø3Ø4	88D4 869152AB ABC4			st st st	A,bfun A,ircd A,portp A,sp	; Then, through RESET vector,	
897 898	Ø3Ø6 Ø3Ø8	ABCØ 3C			st ret	A, psw	; jump to start of program.	
809 810						· Hore	process an ordinary command (not RESET).	
811				(I		; nere,	process an ordinary command (not RESET).	
813 814	Ø3Ø9 Ø3Ø9 Ø3Øb Ø3Ød	9D11		fcord:	and ifgt jmpl	A,#x'1F A,#x'11 illc	; Use only least-significant 5 bits. ; Check for command out of range.	
	Ø3ØF		R		st	A, curcmd	; Save as current command.	
818	Ø311 Ø312 Ø313				shl .odd	A	; Scale by two, and then	
	Ø313	EC			jidw		; jump based on command value:	
	Ø314	ØA09		firstab:	.ptw	fcinit	; Ø = INITIALIZE command.	
								TL/DD/9976
		Ver D1-BetaSite	(Se	- 1/ 1/.	70 1007		PCUPI	25-Feb-88 10:05 PAGE 24
/*1 PUH	(10	ERFACE DEMO				,		
Process	sing					,		
Process 823	sing (Ø316	FERFACE DEMO of First Byte of 9099			de) .ptw	fcslcv	; 1 = SET-CONTRAST command. ; 2 = SEND-LCD command.	
823 824 825	9316 9318 9318 9318	FERFACE DEMO of First Byte of ØDØØ ØFØØ 1400			de) .ptw .ptw .ptw	fcslcv fcslcd fcsled	; 2 = SEND-LCD command. ; 3 = SEND-LED command.	
823 824 824 825 826 827	9316 9318 9318 9318 9310	TERFACE DEMO of First Byte of ØDØØ ØFØØ 1400 1600	Com	mand (Co	de) .ptw .ptw .ptw .ptw	fcslcv fcslcd fcsled fcbeep	; 2 = SEND-LCD command. ; 3 = SEND-LED command. ; 4 = BEEP command.	
823 824 825 826 826 827 828 829	9316 9318 9318 931A 931C 931C	TERFACE DEMO of first Byte of ØDØØ ØFØØ 1400 1600 970111	Com		de) .ptw .ptw .ptw .ptw ld	fcslcv fcslcd fcsled fcbeep numexp,#1	; 2 = SEND-LCD command. ; 3 = SEND-LED command. ; 4 = BEEP command. ; First byte of INITIALIZE command. ; Expects 1 more byte (RTC interval).	
823 824 824 825 826 826 827 828 829 830	9316 9318 9318 9318 9310	TERFACE DEMO of first Byte of ØDØØ ØFØØ 1400 1600 970111	Com	mand (Co	de) .ptw .ptw .ptw .ptw	fcslcv fcslcd fcsled fcbeep	; 2 = SEND-LCD command. ; 3 = SEND-LED command. ; 4 = BEEP command. ; First byte of INITIALIZE command. ; Expects 1 more byte (RTC interval). ; Return.	
823 824 825 826 827 828 829 839 839 831 832 833 834	9316 9318 9318 931A 931C 931C 931E 9321	IERFACE DEMO of First Byte of BFAD BFAD 1400 1600 9791111 9420 970111	Com R	mand (Co	de) .ptw .ptw .ptw .ptw ld jmpl	fcslcv fcslcd fcsled fcbeep numexp,#1	; 2 = SEND-LCD command. ; 3 = SEND-LED command. ; 4 = BEEP command. ; First byte of INITIALIZE command. ; Expects 1 more byte (RTC interval).	
823 824 825 826 827 828 829 831 832 833 834 835 836 837 838	9316 9316 9318 9318 9310 9310 9316 9321 9323 9326 9327 932A	IERFACE DEMO of First Byte of BFAD BFAD 1400 1600 9791111 9420 970111	Com R	mand (Co	de) .ptw .ptw .ptw .ptw ld jmpl ld	fcslcv fcslcd fcsled fcbeep numexp,#1 upwret numexp,#1	 ; 2 = SEND-LCD command. ; 3 = SEND-LCD command. ; 4 = BEEP command. ; 4 = BEEP command. ; Expects 1 more byte (RTC interval). ; Return. ; First byte of SET-CONTRAST command. ; Set up to expect one more byte. ; First byte of SEND-LCD command. ; Set up to expect one more byte. ; Note extended collection mode in Current 	
823 824 825 826 827 828 827 828 839 839 831 834 833 834 835 835 836 837 838 839	9316 9316 9318 931A 931C 931C 9321 9323 9326 9327 932A	TERFACE DEMO of First Byte of prop prop 1400 970111 9420 970111 55 970211 96100E	Com R R	fcinit: fcslcv:	de) .ptw .ptw .ptw .ptw ld jmpl ld jmpl	fcslcv fcslcd fcbeep numexp,#1 upwret numexp,#1 upwret numexp,#2	<pre>; 2 = SEND-LCD command. ; 3 = SEND-LCD command. ; 4 = BEEP command. ; First byte of INITIALIZE command. ; Expects 1 more byte (RTC interval). ; Return. ; First byte of SET-CONTRAST command. ; Set up to expect one more byte.</pre>	
823 824 825 826 826 828 828 839 839 839 832 833 832 833 834 835 836 837 838 838 834 836 837 838 834 834 834 834 834 834 834 834 834	sing (9316 9318 9317 9310 9316 9318 9321 9323 9326 9327 932A 9320 9322 9322	IERFACE DEMO of First Byte of pOD0 pF00 1400 970111 9420 970111 5C 970211 96100E 55 970111	R R R	fcinit: fcslcv:	de) .ptw .ptw .ptw .ptw .ptw ld jmpl ld sbit jmpl ld	fcslcv fcslcd fcsled fcbeep numexp,#1 upwret numexp,#1 getcnt,curcmd upwret numexp,#1	 ; 2 = SEND-LCD command. ; 3 = SEND-LCD command. ; 4 = BEEP command. ; 4 = BEEP command. ; Expects 1 more byte (RTC interval). ; Return. ; First byte of SET-CONTRAST command. ; Set up to expect one more byte. ; First byte of SEND-LCD command. ; Set up to expect one more byte. ; Note extended collection mode in Current 	
823 824 825 826 826 827 828 829 839 831 831 833 834 835 834 835 834 835 834 835 834 834 835 834 834 835 834 834 835 834 834 835 834 834 834 834 835 834 834 834 834 834 834 834 834 834 834	sing (9316 9318 9318 9318 9318 9318 9318 9310 9311 9322 9324 9322 9324 9322 932 93	FERFACE DEMO of First Byte of pOpp pFop 1600 970111 9420 970111 5C 970211 96100E 55 970111 51 96100F	R R R R	fcinit: fcslcv: fcslcd:	de) .ptw .ptw .ptw .ptw ld jmpl ld sbit jmpl ld jmpl	fcslcv fcslcd fcsled fcbeep numexp,#1 upwret numexp,#1 upwret getcnt,curcmd upwret numexp,#1 upwret cmdemp,curcmd	<pre>; 2 = SEND-LCD command. ; 3 = SEND-LCD command. ; 4 = BEEP command. ; 4 = BEEP command. ; Expects 1 more byte (RTC interval). ; Return. ; First byte of SET-CONTRAST command. ; Set up to expect one more byte. ; First byte of SEND-LCD command. ; Set up to expect one more byte. ; Note extended collection mode in Current ; Command byte. ; First byte of SEND-LED command. ; Send to LED's: Set up to expect one more byte. ; Process one-byte BEEP command. ; No arguments; set CURCMD byte empty.</pre>	
823 3824 825 826 827 828 826 827 828 826 827 828 839 831 832 838 831 832 838 837 838 839 834 835 836 837 838 849 843 3844 835 836 847 848 843 845 846 847 848 845 846 847 848 849 859	sing (9316 9318 9318 9318 9318 9316 9321 9322 9326 9330 9330 9339 9330 93000 93000 93000 93000 93000 93000 9300000 930000	IERFACE DEMO of First Byte of BPG0 BF00 1400 979111 9420 970111 5C 970211 96100E 55 970111 51	R R R R	fcinit: fcslcv: fcslcd: fcsled:	de) .ptw .ptw .ptw .ptw ld jmpl ld sbit jmpl ld jmpl	fcslcv fcslcd fcbeep numexp,#1 upwret numexp,#1 getcnt,curcmd upwret numexp,#1 upwret	<pre>; 2 = SEND-LCD command. ; 3 = SEND-LCD command. ; 4 = BEEP command. ; First byte of INITIALIZE command. ; Expects 1 more byte (RTC interval). ; Return. ; First byte of SET-CONTRAST command. ; Set up to expect one more byte. ; First byte of SEND-LCD command. ; Set up to expect one more byte. ; Note extended collection mode in Current ; Command byte. ; First byte of SEND-LED command. ; Send to LED's: Set up to expect one more byte. ; Process one-byte BEEP command. ; No arguments; set CURCMD byte empty. ; Enable beep tone to panel speaker. ; Enable Timer TØ interrupt. ; Initialize duration count (approximately</pre>	
823 3824 825 826 827 828 826 827 828 826 827 828 826 827 828 839 839 834 835 836 837 838 836 837 838 836 837 838 836 837 838 836 837 838 836 837 838 836 837 838 836 837 838 836 837 838 836 837 838 836 837 838 836 837 838 836 837 838 836 837 838 836 837 838 836 837 838 838	sing (9316 (9318 (9328 (9328 (9328 (9328 (9328 (9339 (9339 (9339 (9339 (9339 (9339 (9349 (9349 (9316 (9328 (9338 (9348 (FERFACE DEMO of First Byte of prop prop 1400 970111 9420 970111 9420 970111 55 970211 96100E 55 970211 96100E 55 970111 51	R R R R R R	fcinit: fcslcv: fcslcd: fcsled:	de) .ptw .ptw .ptw .ptw ld jmpl ld sbit sbit sbit sbit	fcslcv fcslcd fcsled fcbeep numexp,#1 upwret numexp,#1 upwret numexp,#2 getcnt,curcmd upwret numexp,#1 upwret cmdemp,curcmd t7tfn,portph	<pre>; 2 = SEND-LCD command. ; 3 = SEND-LCD command. ; 4 = BEEP command. ; First byte of INITIALIZE command. ; Expects 1 more byte (RTC interval). ; Return. ; First byte of SET-CONTRAST command. ; Set up to expect one more byte. ; First byte of SEND-LCD command. ; Set up to expect one more byte. ; Note extended collection mode in Current ; Command byte. ; First byte of SEND-LED command. ; Send to LED's: Set up to expect one more byte. ; Process one-byte BEEP command. ; No arguments; set CURCMD byte empty. ; Enable beep tone to panel speaker. ; Enable Imer TB interrupt.</pre>	
823 3824 824 825 826 826 826 826 827 828 829 828 839 831 832 838 839 831 832 838 835 836 837 838 839 837 838 839 849 847 848 843 8443 845 846 847 848 846 847 848 845 846 851 852 853 854	sing (9316 (9318 (9317 (9310 (9310 (9310 (9312 (9322 (9322 (9322 (9322 (9322 (9322 (9320 (9332 (933 (9	FERFACE DEMO of First Byte of pogg grap 1400 9700111 9420 9700111 9420 9700111 95100E 55 9700111 51 96100E 55 9700111 51 96100F 86001550F 86001550F 86001550F 86001550F 8600159088 9771319	R R R R R R	fcinit: fcslcv: fcslcd: fcsled: fcbeep:	de) -ptw -ptw -ptw -ptw ld jmpl ld sbit sbit sbit sbit sbit sbit	fcslcv fcslcd fcsled fcbeep numexp,#1 upwret numexp,#1 upwret numexp,#2 getent,curemd upwret numexp,#1 upwret cmdemp,curemd t7tfn,portph t#tie,tmmdl beepct,#19 upwret	<pre>; 2 = SEND-LCD command. ; 3 = SEND-LCD command. ; 4 = BEEP command. ; First byte of INITIALIZE command. ; Expects 1 more byte (RTC interval). ; Return. ; First byte of SEIT-CONTRAST command. ; Set up to expect one more byte. ; First byte of SEND-LCD command. ; Set up to expect one more byte. ; Note extended collection mode in Current ; Command byte. ; First byte of SEND-LED command. ; Send to LED's: Set up to expect one more byte. ; Process one-byte BEEP command. ; No arguments; set CURCMD byte empty. ; Enable beep tone to panel speaker. ; Initialize duration count (approximately ; 1 second, in units of Timer TØ overflows).</pre>	
823 824 824 825 826 826 827 826 827 826 839 831 834 839 834 835 835 837 838 834 834 835 837 838 834 849 842 843 845 844 845 846 847 859 859 852 855 854 855 855 855 855 855 855 855 855	sing 0316 0318 0317 0318 0317 0318 0317 0323 0322 0322 0322 0322 0322 0322 0322 0322 0322 0322 0323 0322 0323 0322 0 02 02 02 02 02 02 02 02 0	FERFACE DEMO of First Byte of prop prop 1400 970111 9420 970111 9420 970111 55 970211 96100E 55 970211 96100E 55 970111 51	R R R R R R	fcinit: fcslcv: fcslcd: fcsled:	de) .ptw .ptw .ptw .ptw .ptw .ptw .ptw .ptw	fcslcv fcslcd fcsled fcbeep numexp,#1 upwret numexp,#1 upwret numexp,#2 getcnt,curcmd upwret numexp,#1 upwret cmdemp,curcmd t7tfn,portph t@tiet,mmdl beepct,#19	<pre>; 2 = SEND-LCD command. ; 3 = SEND-LED command. ; 4 = BEEP command. ; First byte of INITIALIZE command. ; Expects 1 more byte (RTC interval). ; Return. ; First byte of SEID-LCD command. ; Set up to expect one more byte. ; First byte of SEND-LCD command. ; Set up to expect one more byte. ; Note extended collection mode in Current ; Command byte. ; First byte of SEND-LED command. ; Send to LED's: Set up to expect one more byte. ; Process one-byte BEEP command. ; No arguments; set CURCMD byte empty. ; Enable beep tone to panel speaker. ; Initialize duration count (approximately ; I second, in units of Timer IØ overflows). ; Process illegal command codes.</pre>	
823 824 825 826 826 827 828 827 828 827 828 831 835 837 838 835 835 836 837 836 837 836 837 836 837 836 837 836 837 836 837 836 837 836 837 836 837 836 837 836 837 836 837 836 837 836 837 837 838 844 847 838 844 847 848 847 848 848 847 848 848 84	sing 9316 9318 9318 9318 9312 9321 9322 9322 9322 9322 9322 9322 9322 9322 9322 9322 9324 9332 9339 9339 9349 9341 9343	FERFACE DEMO of First Byte of pOD0 pF00 1600 970111 9420 970111 9420 970111 5C 970211 96100E 55 970111 51 96100E 55 970111 51 96100F 86019008 971319 42 3741	R R R R R R	fcinit: fcslcv: fcslcd: fcsled: fcbeep:	de) -ptw -ptw -ptw -ptw ld jmpl ld sbit sbit sbit sbit sbit sbit sbit	fcslcv fcslcd fcsled fcbeep numexp,#1 upwret numexp,#1 upwret numexp,#2 getcnt,curcmd upwret numexp,#1 upwret cmdemp,curcmd t7tfn,portph t8tie,tmmdl beepct,#19 upwret hangup	<pre>; 2 = SEND-LCD command. ; 3 = SEND-LCD command. ; 4 = BEEP command. ; First byte of INITIALIZE command. ; Expects 1 more byte (RTC interval). ; Return. ; First byte of SEIT-CONTRAST command. ; Set up to expect one more byte. ; First byte of SEND-LCD command. ; Set up to expect one more byte. ; Note extended collection mode in Current ; Command byte. ; First byte of SEND-LED command. ; Send to LED's: Set up to expect one more byte. ; Process one-byte BEEP command. ; No arguments; set CURCMD byte empty. ; Enable beep tone to panel speaker. ; Initialize duration count (approximately ; 1 second, in units of Timer TØ overflows).</pre>	
**************************************	sing 9316 9318 9318 9317 9317 9317 9317 9321 9322 9327 9328 9327 9328 9330 9339 9349	TERFACE DEMO of First Byte of pOpp pFop 1499 979111 9429 979111 9429 979111 5C 979211 96199E 55 979111 51 96199F B601539F B601539F B601539F 86019908 971319 42 3741 3FCP 3FCB	R R R R R R	mend (Co fcinit: fcslcv: fcslcd: fcsled: fcbeep: illc:	de) -ptw 	fcslcv fcslcd fcsled fcbeep numexp,#1 upwret numexp,#1 upwret numexp,#2 getent,curemd upwret numexp,#1 upwret cmdemp,curemd t7tfn,portph t#tie,tmmdl beepct,#19 upwret	<pre>; 2 = SEND-LED command. ; 3 = SEND-LED command. ; 4 = BEEP command. ; First byte of INITIALIZE command. ; Expects 1 more byte (RIC interval). ; Return. ; First byte of SET-CONTRAST command. ; Set up to expect one more byte. ; First byte of SEND-LED command. ; Set up to expect one more byte. ; Note extended collection mode in Current ; Command byte. ; First byte of SEND-LED command. ; Send to LED's: Set up to expect one more byte. ; Process one-byte BEEP command. ; Not arguments; set CURCMD byte empty. ; Enable beep tone to panel speaker. ; Enable Timer TØ interrupt. ; Initialize duration count (approximately ; 1 second, in units of Timer TØ overflows). ; Process illegal command codes. ; Return from UPI Write interrupt.</pre>	

863	errupt Handler			.form	'Timer Interrup	ht Handler'	
864	F4 48Ø3	R		.ipt	5,tmrint	; Declare entry point for Timer Interrupt.	
866		•	tmrint:		A	; Save context.	
868 Ø3	48 AFC8 4A AFCC 4C AFC9		1001101	push push push	A B psw	;	
	4 E B6019015 52 54		t1poll:	ifbit jmpl	t1pnd,tmmdl t1int	; Poll for Timer T1 interrupt (Real-Time Clock). ; If set, go service it.	
874 Ø3	53 B6015111 57 944C		t6poll:	ifbit jmpl	t6pnd,pwmdh t6int	; Poll for Timer T6 interrupt (LCD Panel Timing ; Interrupt).	
877 Ø3 878 Ø3 879 Ø3 880			tØpoll:	ifbit jp jp	tøpnd,tmmdl tøpdg tønotp	; Poll for Timer TØ interrupt (Beep Duration). ; If set, check the Enable bit; TØ is not ; always enabled to interrupt, but it runs ; continuously.	
	5F B6Ø19Ø1Ø 63 9488		tøpdg:	ifbit jmpl	tØtie,tmmdl tØint	; If enable is also set, then go service TØ.	
884 Ø3 885			t₿notp:	J		; (This label is deliberately here.)	
	65 3765		noint:	jsrl	hangup	; Error: no legal timer interrupt pending.	
							TL/DD/997
PORT	C, Ver D1-Beta INTERFACE DEMO Interrupt Serv	t i		30 1987)) HI	PCUP1	25-Feb-88 19 PAGE
889				.form	'Timer T1 Inter	rrupt Service Routine'	
	57 B601900F		tlint:	sbit	t1ack,tmmdl	: Acknowledge I1 interrupt.	
893 Ø3		R		ifbit jp	rtcenb,rtevs t1int1	; Check if RTC interrupts are enabled.	
896 Ø3	70 8A1B	R	t1int1:	jmpl decsz jmpl	kbdchk rtccnt kbdchk	; If not, then go check other events. ; Decrement interval value. ; If interval has not elapsed, then go check	
	73 8C1A1B 76 960211	R		ld ifbit	rtcont, rtcivl	; for other events. ; Reload counter value for next interval.	
700 Ø3	79 44			ifbit jp	artc,alert.b t1rerr	; Check if CPU has received previous interrupt ; request; report error if not.	
7Ø2 Ø3		R		sbit jp	artc,alert.b kbdchk	; Set Real-Time Interrupt request to main ; program.	
704 031	7E 961DØ8 31 961EØF	R	tirerr:	sbit	Ø,dsevc 7,derrc	; Signal NOTE severity. ; Signal multiple-RTC error.	
906	34 96020A	R		sbit	adiag,alert.b	; Request !DIAG interrupt from main program.	
907 031 908 031	37 37 96E31D		kbdchk:	rbit	astts,portbh	; Check keyboard switches. ; Enable pushbutton data to Port D.	
909 031	3A B6010488 3E 96E30D			ld sbit	A, portd astts, portbh	: Sample pushbutton switches. ; Disable pushbutton data to Port D.	
911 Ø39	91 9BFF			хог	A,#x'FF	; Complement low-order 8 bits of A.	
913 Ø39	93 8E17 95 9617DC	R		x ifeq	A,swlast A,swlast	; Exchange with last sample. ; Check if the data is stable (same as last	
914 915 Ø39				jp	kbint1	; sample).	
916 Ø39 917				jmpl	tmochk	; If not, go check other events (if any).	
	PA 9618DC	R	kbint1:	ifeq	A,swlsnt	; Check if the data differs from the last ; pattern sent to the CPU.	
920 039 921	2045			jmpl	tmochk	; If not, go check other events (if any).	
22 939 23 93/ 24	PE 8818 NØ 960208	R R		st sbit	A,swlsnt abutton,alert.b	; Place new pattern in "last sent" location.); Request "BUTION-DATA" interrupt to CPU.	
925 926 Ø31	13		tmochk:				
27 28	2.0/50					y other RTC events here. ***	
29 Ø3/ 230 231	3 9459			jmpl	tmrret	; Return from Timer T1 interrupt.	
							TL/DD/997

	TERFACE DEMO terrupt Service		14 14: De	319 1987.) HP		25-Feb-88 1Ø: PAGE
932		Nouci		.form	'Timer 16 Inter	rupt Service Routine'	
933 934					• Timer	T6 interrupt routine: sends characters from	
935					; LCD	String Buffer to the panel.	
	8691519A 8691519B		t6int:	sbit sbit	tóstp,pwmdh tóack,pwmdh	; Stop timer T6. ; Acknowledge T6 interrupt.	
939 Ø3AD 940 Ø3AF 941		R		decsz jmpl	lcdsct tónxtc	; Decrement LCD character count. ; If not done, go send another character.	
942 Ø3BØ	96ø2øb	R		sbit	alcdak,alert.b	; If done, request main program to send LCD	
943 944 Ø383	9449			jmpi	tmrret	; Acknowledge interrupt to CPU.	
945 946 Ø385		R	tónxtc:		A,lcdsfg	; Get flags byte (for panel RS signal).	
947 Ø387 948 Ø388	C7 8815	R		shr st	A A,lcdsfg	; Shift right, LSB into carry. ; Store shifted value back.	
949 Ø3BA	96120B	R		sbit	pnlrs, levs	; Determine proper state for RS signal from	
950 03BD 951 03BE	96121B	R		ifc rbit	pnlrs,lcvs	; current character's flag (= flag inverted).	
952 Ø3C1	8C12E1	R		ld sbit	portah,lcvs lcvclk,portbh	; Send new RS value to LCD Voltage (LCV) latch. ; Clock the latch. RS signal is now valid.	
953 Ø3C4 954 Ø3C7				rbit	lcvclk,portbh	, clock the laten. Is signal is now varia.	
955 956 ø3ca		R		ld	A,[lcdsix].b	; Get next LCD character from string buffer.	
957 Ø3CD 958 Ø3CF		R		inc comp	lcdsix A	; Increment character pointer. ; Complement character, then	
959 Ø3DØ	8BE1			st	A, portah	; place it on Port A for LCD display.	
960 03D2 961 03D5				rbit sbit	pnicik,portbi pnicik,portbi	; Clock it into panel.	
962 Ø308 963	Ø 1			comp	A	; Restore A to uncomplemented form for ; test performed below.	
964 965 Ø3D9	8394Ø148AB			ld	t6,#148	; Set up normal delay time in timer Tó	
966 967 Ø3DE				ifgt	A,#x'Ø3	; (120 microseconds). ; Check whether the longer delay	
968 Ø3EØ	47			jp	tónxt2	; (4.9 milliseconds) is necessary.	
969 970 03E1 971 03E2	96 871 7869148 AB			ifnc ld	t6,#6022	; This happens if RS=0 and the byte sent to ; the panel is a value of hex 03 or less. ; If so, change timer to 4.9 milliseconds.	
972	B6Ø1511A		tónxt2:		tóstp,pwmdh tmrret	; Start Timer Tó to time out the character. ; Return from the interrupt.	
975							
975 976							TL/DD/9976
976 ASMHPC,	Ver D1-BetaSite	e (Sep	14 14::	30 1987)) HPI	CUPI	25-Feb-88 10:
976 ASMHPC, PORT INT	Ver D1-BetaSite ERFACE DEMO errupt Service			30 1987)) HPI	CUPI	25-Feb-88 10:
976 ASMHPC, PORT INT Pr TØ Int 977	ERFACE DEMO			30 1987) .form		CUPI rupt Service Routine'	TL/DD/9976 25-Feb-88 10: PAGE
ASMHPC, PORT INT PT TØ INT 277 278	ERFACE DEMO	Routi	ne		'Timer TØ Inter	rupt Service Routine'	25-Feb-88 10:
976 ASMHPC, PORT INT PT 977 978 979 93ED 80	ERFACE DEMO	Routi			'Timer TØ Inter ; Count ; to zu	rupt Service Routine' duration of beep tone. Restore beep signal ero and re-enable switch sampling interrupt	25-Feb-88 10:
976 ASMHPC, PORT INT F TØ INT 77 78 779 Ø3ED 881 881 882 Ø3ED	ERFACE DEMO errupt Service B601900B	Routin	ne	.form	'Timer TØ Inter ; Count ; to zu ; when tØack,tmmdl	rupt Service Routine' duration of beep tone. Restore beep signal ero and re-enable switch sampling interrupt done. ; Acknowledge interrupt from Timer TØ.	25-Feb-88 10:
976 PORT INT PORT INT PORT INT PT 10 INT 778 979 03ED 881 881 882 03ED 883 03F1	ERFACE DEMO errupt Service B601900B 8A19	Routi	ne	.form sbit decsz	'Timer TØ Inter ; Count ; to zu ; when tØack,tmmdl beepct	rupt Service Routine' duration of beep tone. Restore beep signal ero and re-enable switch sampling interrupt done. ; Acknowledge interrupt from Timer TØ. ; Check whether beep time has finished.	25-Feb-88 10:
ASMHPC, PORT INT PT 10 Int 777 788 79 03ED 780 881 882 03ED 883 03F1 884 03F3 885 03F4	ERFACE DEMO errupt Service B601900B 8A19 4A	Routin	ne	.form	'Timer TØ Inter ; Count ; to zu ; when tØack,tmmdl	rupt Service Routine ¹ duration of beep tone. Restore beep signal ero and re-enable switch sampling interrupt done. ; Acknowledge interrupt from Timer TØ. ; Check whether beep time has finished. ; No: return from interrupt. ; Yes: disable Timer TØ interrupts and	25-Feb-88 10:
976 ASMHPC, PORT INT PTØ Int P77 P80 P81 P82 Ø36 Ø36 P35 Ø36 P35 Ø36 P36 P36 P36 P36 P36 P36 P36 P36 P36 P	ERFACE DEMO errupt Service B601900B 8A19 4A	Routin	ne	.form sbit decsz jmpl	'Timer TØ Inter ; Count ; to z ; when tØack,tmmdl beepct tmrret	rupt Service Routine' duration of beep tone. Restore beep signal ero and re-enable switch sampling interrupt done. ; Acknowledge interrupt from Timer TØ. ; Check whether beep time has finished. ; No: return from interrupt.	25-Feb-88 10:
976 ASMHPC, PORT INT PT TØ INT 777 778 779 820 830 832 9350 840 855 9354 855 9354 885 9354 885 9354 885 9358	ERFACE DEMO errupt Service B6019008 8A19 66019018 830F015309	Routin	ne	.form sbit decsz jmpl rbit	'Timer TØ Inter ; Count ; to z; ; when tØack,tmmdl beepct tmrret tØtie,tmmdl	rupt Service Routine' duration of beep tone. Restore beep signal ero and re-enable switch sampling interrupt done. ; Acknowledge interrupt from Timer TØ. ; Check whether beep time has finished. ; No: return from interrupt. ; Yes: disable Timer TØ interrupts and ; continue.	25-Feb-88 10:
976 ASMHPC, PORT INT ;r TØ Int ;77 %80 %81 %82 %82 %83 %83 %85 %85 %85 %85 %85 %85 %85 %85 %85 %90 %90 %90 %90 %90 %90 %90 %90 %90 %90	ERFACE DEMO errupt Service B601900B 8A19 4A B6019018 830F0153D9 40	Routin R	ne tØint∶	.form sbit decsz jmpl rbit and jmpl	'Timer TØ Inter ; Count ; to z ; when tØack,tmmdl beepct tmrret tØtie,tmmdl portph,#x'ØF tmrret ; Common	rupt Service Routine ¹ duration of beep tone. Restore beep signal ero and re-enable switch sampling interrupt done. ; Acknowledge interrupt from Timer TØ. ; Check whether beep time has finished. ; No: return from interrupt. ; Yes: disable Timer TØ interrupts and ; continue. ; Disable speaker output. ; Return from interrupt. n return for timer interrupt service routines.	25-Feb-88 10:
ASHHPC, PORT INT PORT INT PORT INT PORT INT PORT INT PORT INT PORT PORT PORT PORT PORT PORT PORT POR	ERFACE DEMO errupt Service 86019008 8A19 4A 86019018 830F015309 40 3FC0	Routin R	ne	.form sbit decsz jmpl rbit and jmpl	'Timer TØ Inter ; Count ; to z. ; when beect tmrret tØtie,tmmdl portph,#x'ØF tmrret	rupt Service Routine' duration of beep tone. Restore beep signal ero and re-enable switch sampling interrupt done. ; Acknowledge interrupt from Timer TØ. ; Check whether beep time has finished. ; No: return from interrupt. ; Yes: disable Timer TØ interrupts and ; continue. ; Disable speaker output. ; Return from interrupt.	25-Feb-88 10:
976 ASMHPC, PORT IMI PT TØ Int PT7 P08 P08 P08 P08 P09 P09 P09 P09 P09 P09 P09 P09 P09 P09	ERFACE DEMO errupt Service 86019008 8019 40 83019018 830F015309 40 3FC0 3FC0 3FC0 3FC8	Routin R	ne tØint∶	.form sbit decsz jmpl rbit and jmpl pop pop	'Timer TØ Inter ; Count ; to zı ; when beepct tmrret tØtie,tmmdl portph,#x'ØF tmrret ; Common psw	rupt Service Routine ¹ duration of beep tone. Restore beep signal ero and re-enable switch sampling interrupt done. ; Acknowledge interrupt from Timer TØ. ; Check whether beep time has finished. ; No: return from interrupt. ; Yes: disable Timer TØ interrupts and ; continue. ; Disable speaker output. ; Return from interrupt. n return for timer interrupt service routines.	25-Feb-88 10:
976 ASMHPC, PORT IMI PT TØ Int 777 778 779 Ø3ED 779 880 883 Ø3F1 884 Ø3F5 785 Ø3F8 886 Ø3F0 886 Ø3F0 886 Ø3F0 888 Ø3F0 899 991 Ø3F8 899 993 Ø402 995	ERFACE DEMO errupt Service 86019008 8019 40 83019018 830F015309 40 3FC0 3FC0 3FC0 3FC8	Routin R	ne tØint∶	.form sbit decsz jmpl rbit and jmpl pop	'Timer TØ Inter ; Count ; to z; ; when tBack,tmmdl beepct tmrret t@fie,tmmdl portph,#x'ØF tmrret ; Commou psw B	rupt Service Routine ¹ duration of beep tone. Restore beep signal ero and re-enable switch sampling interrupt done. ; Acknowledge interrupt from Timer TØ. ; Check whether beep time has finished. ; No: return from interrupt. ; Yes: disable Timer TØ interrupts and ; continue. ; Disable speaker output. ; Return from interrupt. n return for timer interrupt service routines.	25-Feb-88 10:
976 ASMHPC, PORT INT FT 70 INT FT 70 182 183 183 183 184 184 185 185 185 185 185 185 185 185 185 185	ERFACE DEMO errupt Service 86019008 8019 40 83019018 830F015309 40 3FC0 3FC0 3FC0 3FC8	Routin R	ne tØint∶	.form sbit decsz jmpl rbit and jmpl pop pop	'Timer TØ Inter ; Count ; to z; ; when tBack,tmmdl beepct tmrret t@fie,tmmdl portph,#x'ØF tmrret ; Commou psw B	rupt Service Routine ¹ duration of beep tone. Restore beep signal ero and re-enable switch sampling interrupt done. ; Acknowledge interrupt from Timer TØ. ; Check whether beep time has finished. ; No: return from interrupt. ; Yes: disable Timer TØ interrupts and ; continue. ; Disable speaker output. ; Return from interrupt. n return for timer interrupt service routines.	25-Feb-88 10: PAGE
976 ASMHPC, PORT IMI PT TØ Int 777 778 779 Ø3ED 779 880 883 Ø3F1 884 Ø3F5 785 Ø3F8 886 Ø3F0 886 Ø3F0 886 Ø3F0 888 Ø3F0 899 991 Ø3F8 899 993 Ø402 995	ERFACE DEMO errupt Service 86019008 8019 40 83019018 830F015309 40 3FC0 3FC0 3FC0 3FC8	Routin R	ne tØint∶	.form sbit decsz jmpl rbit and jmpl pop pop	'Timer TØ Inter ; Count ; to z; ; when tBack,tmmdl beepct tmrret t@fie,tmmdl portph,#x'ØF tmrret ; Commou psw B	rupt Service Routine ¹ duration of beep tone. Restore beep signal ero and re-enable switch sampling interrupt done. ; Acknowledge interrupt from Timer TØ. ; Check whether beep time has finished. ; No: return from interrupt. ; Yes: disable Timer TØ interrupts and ; continue. ; Disable speaker output. ; Return from interrupt. n return for timer interrupt service routines.	25-Feb-88 10: PAGE
ASMHPC, PORT INT PT TØ Int PT77 778 779 Ø3ED 777 880 881 882 Ø3ED 883 Ø3F1 884 Ø3F3 886 Ø3F3 886 Ø3F4 886 Ø3F4 886 Ø3F4 886 Ø3F4 897 998 999 993 Ø4Ø2 999 993 Ø4Ø2 995 995 995 995 995 995 995 995 995 99	ERFACE DEMO errupt Service B6019008 8A19 4A B6019018 830F0153D9 40 3FC0 3FC0 3FC0 3FC0 3FC0 3FC0 3FC0 3FC	Routir R e (Sep	ne tØint: tmrret: 14 14:3	.form sbit decsz jmpl rbit and jmpl pop pop reti	'Timer TØ Inter ; Count ; to z ; when tØack,tmmdl beepct tmrret tØtie,tmmdl portph,#x'ØF tmrret ; Common psw B A	rupt Service Routine ¹ duration of beep tone. Restore beep signal ero and re-enable switch sampling interrupt done. ; Acknowledge interrupt from Timer TØ. ; Check whether beep time has finished. ; No: return from interrupt. ; Yes: disable Timer TØ interrupts and ; continue. ; Disable speaker output. ; Return from interrupt. n return for timer interrupt service routines.	25-Feb-88 10: PAGE TL/DD/9976 25-Feb-88 10:
ASMHPC, PORT INT Pr TØ Int Pr TØ Int Pr78 Pr79 Ø3ED Pr80 Pr83 Ø3F1 Pr84 Ø3F3 Pr85 Ø3F4 Pr88 Ø3F0 Pr98 Pr98 Pr98 Pr98 Pr98 Pr98 Pr98 Pr98	ERFACE DEMO errupt Service B601900B 8A19 4A 86019018 830F0153D9 40 3FC0 3FC0 3FC0 3FC0 3FC 3FC0 3FC 3FC0 3FC 3FC0 3FC	Routir R e (Sep	ne tØint: tmrret: 14 14:3	.form sbit decsz jmpl rbit and jmpl pop pop pop reti	'Timer TØ Inter ; Count ; to z ; when tBack,tmmdl beepct tmrret tBite,tmmdl portph,#x'ØF tmrret ; Commod B A	rupt Service Routine ¹ duration of beep tone. Restore beep signal ero and re-enable switch sampling interrupt done. ; Acknowledge interrupt from Timer TØ. ; Check whether beep time has finished. ; No: return from interrupt. ; Yes: disable Timer TØ interrupts and ; continue. Disable speaker output. ; Return from interrupt. n return for timer interrupt service routines. ; Restore context.	25-Feb-88 19: PAGE TL/DD/9976 25-Feb-88 19:
ASMHPC, PORT INT Pr TØ Int Pr TØ Int Pr78 80 820 0353 841 850 0354 883 0351 884 0353 885 0354 886 0355 886 0355 886 0355 886 0355 886 0355 8970 9378 886 0355 8970 9376 9376 9376 9376 9376 9376 9376 9376	ERFACE DEMO errupt Service B6019008 8A19 4A B6019018 830F0153D9 40 3FC0 3FC0 3FC0 3FC0 3FC0 3FC0 3FC0 3FC	Routir R E (Sep E Empt)	ne tØint: tmrret: 14 14:2	.form sbit decsz jmpl rbit and jmpl pop pop pop reti .form	'Timer TØ Inter ; Count ; to z ; when tBack, trmdl beepct tmrret tmrret ; Common psw B A 'Subroutine to N	rupt Service Routine' duration of beep tone. Restore beep signal ero and re-enable switch sampling interrupt done. ; Acknowledge interrupt from Timer TØ. ; Check whether beep time has finished. ; No: return from interrupt. ; Yes: disable Timer TØ interrupts and ; continue. Disable speaker output. ; Return from interrupt. n return for timer interrupt service routines. ; Restore context.	25-Feb-88 10:
976 ASMHPC, PORT INT PT TØ Int PT7 P08 80 81 82 83 83 84 83 83 85 83 85 83 85 83 85 83 85 83 85 83 85 83 85 83 85 83 85 83 85 83 85 85 85 85 85 85 85 85 85 85	ERFACE DEMO errupt Service B6019008 8A19 4A B6019018 830F0153D9 40 3FC0 3FC0 3FC0 3FC0 3FC0 3FC0 3FC0 3FC	Routir R e (Sep	ne tØint: tmrret: 14 14:2	.form sbit decsz jmpl rbit and jmpl pop pop pop reti .form	'Timer TØ Inter ; Count ; to z ; when beepct tmrret tØtie,tmmdl portph,#x'ØF tmrret ; Common psw B A HPG 'Subroutine to M subroutine: wait	rupt Service Routine ¹ duration of beep tone. Restore beep signal ero and re-enable switch sampling interrupt done. ; Acknowledge interrupt from Timer TØ. ; Check whether beep time has finished. ; No: return from interrupt. ; Yes: disable Timer TØ interrupts and ; continue. Disable speaker output. ; Return from interrupt. n return for timer interrupt service routines. ; Restore context.	25-Feb-88 19: PAGE TL/DD/9976 25-Feb-88 19:
ASHHPC, PORT INT PORT PORT INT PORT PORT PORT PORT PORT PORT PORT PORT	ERFACE DEMO errupt Service 86019008 8019 40 83019018 830F0153D9 40 3FCC 3FCC 3FCC 3FCC 3FCC 3FCC 3FCC 3FC	Routir R E (Sep E Empt)	ne tØint: tmrret: 14 14:2	.form sbit decsz jmpl rbit and jmpl pop pop reti 50 1987) .form RDWAIT	'Timer TØ Inter ; Count ; to z ; when tØack,tmmdl beepct tmrret tØtie,tmmdl portph,#x'ØF tmrret ; Commou psw B A 'Subroutine to W subroutine: wait UPI	rupt Service Routine' duration of beep tone. Restore beep signal ero and re-enable switch sampling interrupt done. ; Acknowledge interrupt from Timer TØ. ; Check whether beep time has finished. ; No: return from interrupt. ; Yes: disable Timer TØ interrupts and ; continue. ; Disable speaker output. ; Return from interrupt. n return for timer interrupt service routines. ; Restore context.	25-Feb-88 19: PAGE TL/DD/9976 25-Feb-88 19:
ASMHPC, PORT INT: PORT INT: PO	ERFACE DEMO errupt Service B601900B 8A19 4A B6019018 830F0153D9 40 3FCD 3FCD 3FCC 3FCC 3FCC 3FCC 3FCC 3FCC	Routir R E (Sep E Empt)	ne tØint: tmrret: 14 14:2	.form sbit decsz jmpl rbit and jmpl pop pop pop reti .form RDWAIT ifbit ret	'Timer TØ Inter ; Count ; to z ; when tBack, tmmdl beepct tmrret tBite, tmmdl portph,#x'ØF tmrret ; Common psw B A HPU 'Subroutine to N subroutine: wait UPI rdrdy, upic	rupt Service Routine' duration of beep tone. Restore beep signal ero and re-enable switch sampling interrupt done. ; Acknowledge interrupt from Timer TØ. ; Check whether beep time has finished. ; No: return from interrupt. ; Yes: disable Timer TØ interrupts and ; continue. ; Disable speaker output. ; Return from interrupt. n return for timer interrupt service routines. ; Restore context. CUPI Wait for OBUF Empty' ts until the CPU has read a byte from the	25-Feb-88 19: PAGE TL/DD/9976 25-Feb-88 19:
ASMHPC, PORT INT: PORT INT: PO	ERFACE DEMO errupt Service B601900B 8A19 4A B6019018 830F0153D9 40 3FCD 3FCD 3FCC 3FCC 3FCC 3FCC 3FCC 3FCC	Routir R E (Sep E Empt)	ne tØint: tmrret: 14 14:2	.form sbit decs2 jmpl pop pop pop reti 39 1987) .form RDWAIIT ifbit	'Timer TØ Inter ; Count ; to z ; when tØack,tmmdl beepct tmrret tØtie,tmmdl portph,#x'ØF tmrret ; Commou psw B A 'Subroutine to W subroutine: wait UPI	rupt Service Routine' duration of beep tone. Restore beep signal ero and re-enable switch sampling interrupt done. ; Acknowledge interrupt from Timer TØ. ; Check whether beep time has finished. ; No: return from interrupt. ; Yes: disable Timer TØ interrupts and ; continue. ; Disable speaker output. ; Return from interrupt. n return for timer interrupt service routines. ; Restore context.	25-Feb-88 19: PAGE TL/DD/9976 25-Feb-88 19:

	DEMO outine					PAGE 30
1007			.form	'Write to Panel	Subroutine'	
1008 1009					Panel subroutine.	
1010				; Used	only at initialization or to report a	
1011					protocol error, since it performs	
1012				; the t	iming delay using timer T6 without interrupts.	
1013					nel RS signal must be set up previously in the	
1014				;	V latch by the calling routine.)	
1015 1016 040a 01		wrpnl:	comp	A	; Complement value for bus.	
1017 0408 8BE1		a pire.	st	A,portah	Put value on panel bus.	
1018 040D 96E21F			rbit	phicik, portbl	: Set Panel Clock low.	
1019 0410 96E20F			sbit	pnlclk,portbl	; then high again;	
1020 1021					; pulse width approx. ; 1.2 microsec.	
1022					, 1.2 milliosce.	
1023				; Wait	for another	
1024					milliseconds (twice).	
1025 0413 8732080			ld rbit	t6,#13000	; Twice 4.9 milliseconds. ; Start timer T6.	
1026 0419 8601511 1027 0410 8601511	1	wrplp:	ifbit	tóstp,pwmdh tópnd,pwmdh	; Wait for PND to be set.	
1928 9421 41		m , p(p,	jp	wrpgo	, water for the to be set	
1029 0422 65			JP.	wrplp		
1030 0423 B601510		wrpgo:	sbit	tóstp,pwmdh	; Stop timer T6.	
1031 0427 B601510	5		sbit ret	t6ack,pwmdh	; Clear T6 PND bit. ; Return from subroutine.	
1032 0428 3C 1033						
1034		; END O	F PROGRA	M: RESET VECTOR	R SET TO LABEL "start".	
1ø35			an-1	start		
1036 042C			.end	5(8)(
						TL/DD/9976-
SC ASMHPC, Ver D1-	atabit- 10-	n 16 17 ·	1007		CUPI	25-Feb-88 10:0
PI PORT INTERFACE	DEMO	p 14 14:	1707)	ne.	COPI	PAGE 3
ite to Panel Subr	outine					
	s Null s Null					
	s Null s Byte					
	s Byte					
alcdak 0003 Ab	s Null					
	Word BASE					
	L Byte BASE	=				
irte 0001 Ab istts 0005 Ab	s Null s Null					
	Word RAM1	16				
2stp 0007 Ab	s Null					
80r16 0004 Ab						
80r9 0004 Ab		-				
xeepct βiβ19 Re ⊳fun βiβF4 Ab	l Byte BASE s Word	-				
	s Byte					
	s Byte					
h ØØCD Ab	s Byte					
	Byte					
data 0004 Ab hkalt 01A8 Re	s Null L Null ROM1	16				
	s Null ROM					
puad 0004 Re		E				
pubuf 0006 Re	Word BASE					
urcmd 0010 Re						
lbyte ØØ1F Re Iccmd ØØ2Ø Re						
iccma 1919219 Re Ierrc 19191E Re						
	s Byte	-				
lirb ØØF2 Ab	s Word					
	s Byte					
	s Byte s Word					
livby Ø18E Ab livbyh Ø18F Ab	s word s Byte					
livbyl Ø18E Ab						
loeerr ØØØ7 Ab	s Núll	_				
kqual 0021 Re						
isevc øø1D Re Jummy øøøø Re						
auniny popopo ke ≘i ØØØ7 Ab		-				
iack 0002 Ab	s Null					
	s Byte					
imode ØØØ1 Ab	s Null					
	s Null S Byte					
	s byte s Byte					
enir ØØDØ Ab	s Byte					
enir 0,000 Ab enu 0,120 Ab						
enir 0000 Ab enu 0120 Ab enui 0122 Ab enur 0128 Ab						
enir 0000 Ab enu 0120 Ab enui 0122 Ab enur 0128 Ab eni 0001 Ab	s Núll					
enir 0000 Ab enu 0120 Ab enui 0122 Ab enur 0128 Ab	s Nüll					
nir 0000 Ab mu 0120 Ab mui 0122 Ab mur 0128 Ab mi 0001 Ab	s Núll					TL/DD/9976

	Panel 9				
cbeep cinit cord	0332 031E 0309	Rei Rei Rei	Null Null Null	ROM16 ROM16 ROM16	
cslcd	Ø327 Ø323	Rel	Null Null	ROM16 ROM16	
csled	Ø32E	Rel	Null	ROM16 ROM16 ROM16	
irstc	02E5 0006	Rel	Null Null	ROM16	
etcnt	0006 0000	Abs Abs			
angup extab	0000 007a	Rel	Null Byte	ROM16 ROM16	
grst gupi	006F 005E	Rel Rel	Null Null	ROM16 ROM16	
gupi1 gupi2	0069 0076	Rel Rel	Null	ROM16 ROM16	
2 3 4	0002 0003 0004	Abs	Null Null Null		
or buf llc	00F0 0341	Abs Rel	Byte Null	ROM16	
rcđ rpđ	0004 0002	Abs Abs	Byte	50114 (
bdchk bint1 a₿	0387 039a 0002	Rel Rel Abs	Null Null Null	ROM16 ROM16	
astab astc	Ø246 Ø234	Rel Rel	Null	ROM16 ROM16	
astc1 cdbuf	0241 0038	Rel Rel	Null	ROM16	
cdfgs cdgo1	0013 0171	Rel Rel	Byte Null	BASE ROM16	
cdlp1 cdnum	Ø168 ØØ14	Rel Rel	Null Byte	ROM16 BASE	
cdsct cdsfg	0016 0015	Rel Rel	Byte Byte	BASE BASE	
cdsix cinit	000E 0250	Rel Rel	Word Null	BASE ROM16	
cord crst	022A 0224	Rel Rel	Null Null	ROM16 ROM16 ROM16	
cslc1 cslc2 cslcd	02C6 0291 028C	Rel Rel Rel	Null Null Null	ROM16 ROM16 ROM16	
cslcv csled	Ø274 Ø2D8	Rel Rel	Null	ROM16 ROM16	
cvclk cvs	0001 0012	Abs Rel	Null Byte		
edclk ainlp	0006 01a8	Abs Rel	Null Null	ROM16	
					TL/DD/9976-

<pre>minit #919 ket Null R0416 numexp #911 Ret Byte Byse philt #84 Byte Byse philt #84 Byte Byse philt #84 Byte philt #84 Byte</pre>	inite to Par	Ver D1-Be TERFACE DE nel Subrou				PAGE
rtcivl ØØØA Abs Null rtcivl ØØ1A Rel Byte BASE runsys Ø17 Rel Byte BASE runsys Ø17 Rel Byte BASE runsys Ø17 Rel Null ROM16 sio ØØ06 Abs Byte BASE sk ØØ06 Abs Null ROM16 slcd Ø145 Rel Null ROM16 sndbtn Ø1D2 Rel Null ROM16 sndiag Ø124 Rel Null ROM16 sndiak Ø1C4 Rel Null ROM16 sndrtc Ø127 Rel Null ROM16 sram1 ØØC7 Rel Null ROM16	noint 00 polici 00 polici 00 polici 00 portah	365 Rel 9f1 Rel 9f2 Abs 9f15 Abs 9f2 Abs 9f15 Abs 9f15 Abs 9f15 Abs 9f15 Abs 9f16 Abs 9f17 Abs 9f18 Abs 9f19 Abs 9f11 Abs 9f15 Abs 9f16 Abs 9f17 Abs 9f18 Abs 9f14 Abs 9f14 Abs 9f14 Abs 9f14 Abs 9f14 Abs 9f15 Abs	Null Byte Byte Null Byte Byte Byte Byte Byte Byte Byte Byte	ROM16 BASE		
sio 9006 Abs Byte sk 9006 Abs Null slcd 9145 Ret Null ROM16 srdbtr 9102 Ret Null ROM16 sndbtr 9102 Ret Null ROM16 sndbtr 9102 Ret Null ROM16 sndbta 9102 Ret Null ROM16 sndtra 9124 Ret Null ROM16 sndtra 9124 Ret Null ROM16 sndtra 9124 Ret Null ROM16 sndtra 9024 Ret Null ROM16 sram 9024 Ret Null ROM16 sram11 9027 Ret Null ROM16 srtsh 9808 Ret Null ROM16 sskint 9802 Ret Null ROM16	rtcenb Ø rtcenb Ø rtcivl Ø rtevs Ø	0/18 Rel 0/00 Abs 0/1A Rel 0/1C Rel	Byte Null Byte Byte	BASE BASE BASE		
sndiag Ø1E4 Rel Null ROM16 sndiak Ø1CA Rel Null ROM16 sndrtc Ø1C2 Rel Null ROM16 sram Ø0C4 Rel Null ROM16 sram10 Ø0C7 Rel Null ROM16 sram12 ØØCF Rel Null ROM16 srfsh ØØC8 Rel Null ROM16 sskint ØØD2 Rel Null ROM16	sio Ø sk Ø slcd Ø sled Ø	006 Abs 006 Abs 145 Rel 104 Rel	Byte Null Null Null	ROM16 ROM16		
sram 00C4 Rel Null ROM16 sraml1 00C7 Rel Null ROM16 sraml2 00CF Rel Null ROM16 srfsh 0080 Rel Null ROM16 sskint 0002 Rel Null ROM16	sndiag Ø sndlak Ø sndrtc Ø	1E4 Rel 1CA Rel 1C2 Rel	Null Null Null	ROM16 ROM16		
	sram Ø sraml1 Ø sraml2 Ø	ØC4 Rel ØC7 Rel	Null Null Null	ROM16 ROM16 ROM16		
			NULL			
			Null	RUMIS		TL/DD/9976
			Null	RUMIG		TL/DD/9976
			Null	KUM IG		TL/DD/9976
			Null	KUM IG		TL/DD/9976
			Null	KUM IG		TL/DD/9976
			Null	KUM IG		TL/DD/9976
			Mull	KUM IG		TL/DD/9976
			Mull	KUM IG		TL/DD/9976

rite to	Panel	Subrou	utine	e (Sep 14 14:30 1987)	PAGE 3
stackb start	0000 008a	Rel Rel	Word Null	RAM16 ROM16	
stmrs supi	Ø113 ØØA7	Rel Rel	Null Null	ROM16 ROM16	
swlast swlsnt	0017 0018	Rel Rel	Byte Byte	BASE	
tØack	0003	Abs	Null	BASE	
tØcon tØint	Ø192 Ø3ED	Abs Rel	Byte Null	ROM16	
tØnotp tØpdg	Ø365 Ø35 F	Rel Rel	Null Null	ROM16 ROM16	
tøpnd tøpoll	0001 0359	Abs Rel		ROM16	
tØtie t1	8889 8182	Abs Abs	Word		
t1ack t1int	0007 0367	Abs Rel	Null	ROM16	
t1int1 t1pnd	0370 0005	Rei Abs	Null Null	ROM16	
tipoll tirerr	Ø34E Ø37E	Rel Rel	Null Null	ROM16 ROM16	
tlstp tltie	0006 0004	Abs Abs	Null Null		
t2 t2ack	Ø188 ØØØ3	Abs Abs	Word Null		
t2pnd t2stp	0001 0002	Abs Abs	Null Null		
t2tie t3	0000 018C	Abs Abs	Null Word		
t3ack t3pnd	0007 0005	Abs Abs	Null Null		
t3stp t3tie	0006 0004	Abs Abs	Null		
4 4ack	0140 0003	Abs Abs	Word		
4out 4pnd	0000 0001	Abs	Null		
4stp 4tfn	0002 0003	Abs	Null Null		
4tie	0000	Abs	Null		
5 5ack	0144 0007	Abs Abs	Word Null		
5out 5pnd	0004 0005	Abs Abs	Null Null		
5stp 5tfn	0006 0007	Abs Abs	Null Null		
5tie 6	0004 0148	Abs Abs	Word		
:6ack :6int	ØØØ3 Ø3A5	Abs Rel	Null Null	ROM16	
					TL/DD/9976-

	Panel S	Subrou	tine				
t6nxt2	Ø3E8		Null				
t6nxtc t6out	Ø385 ØØØØ	Rel Abs	Null Null	ROM16			
t6pnd	9991	Abs	Null				
t6poll	Ø353	Rel	Null	ROM16			
t6stp t6tfn	0002 0003	Abs Abs	Null Null				
tót ie	6666	Abs	Null				
t7	Ø14C	Abs	Word				
t7ack	0007 000/	Abs	Null				
t7out t7pnd	0004 0005	Abs Abs	Null Null				
t7stp	0006	Abs	Null				
t7tfn	0007	Abs	Null				
t7tie tbmt	0004 0000	Abs Abs	Null Null				
tbuf	Ø126	Abs	Byte				
tminit	ØØD F	Rel	Null	ROM16			
tmmdh	0191	Abs	Byte				
tmmdi tmmode	Ø19Ø Ø19Ø	Abs Abs	Byte Word				
tmochk	Ø3A3	Rel	Null	ROM16			
tmrint	Ø348	Rei	Null	ROM16			
tmrret tmrs	03FE 0005	Rel Abs	Null Null	ROM16			
uart	9996	Abs	Null				
upic	00E6	Abs	Byte				
upicsv	9000 0007	Rel Abs	Word Null	BASE			
upien upiwr	0003 0200	Rel	Null	ROM16			
upwret	0343	Rei	Null	ROM16			
urdrdy	9997 0007	Abs	Null				
uwdone uwmode	0000 0001	Abs Abs	Null Null				
uwrrdy	6663	Abs	Null				
vbutton	ØØ18	Abs	Null				
vdiag	991D	Abs	Null				
vlcdak vrtc	0017 0011	Abs Abs	Null Null				
wakeup	0002	Abs	Null				
wrpgo	0423	Rel	Null	ROM16			
wrplp	041D 040a	Rel Rei	Null	ROM16 ROM16			
wrpnl wrrdy	0000	Abs	Null	KUMIO			
xbit9	0005	Abs	Null				
xh xl	00CF 00CE	Abs Abs	Byte Byte				
xoff	9913	Abs	Null				
xon xrclk	0011 0003	Abs Abs	Null Null				
							TL/DD/9976-
NSC ASMH UPI PORT Write to	INTERF	ACE DE	MO	e (Sep 14	14:30 1987)	HPCUPI	25-Feb-88 19:0 PAGE 3
UPI PORT	INTERF	ACE DE Subrou Rel	MO	-	14:30 1987)	HPCUP I	
UPI PORT Write to xreset	DANERF Panel 92F0 9992	ACE DE Subrou Rel Abs	MO utine Null	ROM16	14:30 1987)	HPCUP I	
UPI PORT Write to xreset xtclk	DANERF Panel 92F0 9992	ACE DE Subrou Rel Abs	MO utine Null Null	ROM16	14:30 1987)	HPCUP I	
UPI PORT Write to xreset xtclk	DANERF Panel 92F0 9992	ACE DE Subrou Rel Abs	MO utine Null Null	ROM16	14:30 1987)	HPCUP I	PAGE 3
UPI PORT Write to xreset xtclk	DANERF Panel 92F0 9992	ACE DE Subrou Rel Abs	MO utine Null Null	ROM16	14:30 1987)	HPCUP I	PAGE :
UPI PORT Write to xreset xtclk	DANERF Panel 92F0 9992	ACE DE Subrou Rel Abs	MO utine Null Null	ROM16	14:30 1987)	HPCUP I	PAGE 3
UPI PORT Write to xreset xtclk	DANERF Panel 92F0 9992	ACE DE Subrou Rel Abs	MO utine Null Null	ROM16	14:30 1987)	HPCUP I	PAGE 3
UPI PORT Write to xreset xtclk	DANERF Panel 92F0 9992	ACE DE Subrou Rel Abs	MO utine Null Null	ROM16	14:30 1987)	HPCUP I	PAGE 3
UPI PORT Write to xreset xtclk	DANERF Panel 92F0 9992	ACE DE Subrou Rel Abs	MO utine Null Null	ROM16	14:30 1987)	HPCUP I	PAGE 3
UPI PORT Write to xreset xtclk	DANERF Panel 92F0 9992	ACE DE Subrou Rel Abs	MO utine Null Null	ROM16	14:30 1987)	HPCUP I	PAGE 3
UPI PORT Write to xreset xtclk	DANERF Panel 92F0 9992	ACE DE Subrou Rel Abs	MO utine Null Null	ROM16	14:30 1987)	HPCUP I	PAGE 3

Γ

4.3 Two Demo Programs (NS32CG16 Source Code)

The following two programs run on the NS32CG16 CPU, and exercise the functions implemented in the HPC firmware.

One thing to note in this software is that the interrupt service routines are not written as such; they are simple subroutines called by the actual service routines, which are contained within a modified version of the MON16 monitor program. The reasons for modifying MON16 were two-fold:

- 1. There is no RAM in the application system within the first 64k of the addressing space. The presence of RAM there is necessary for MON16 to support custom interrupt handlers without internal modification.
- The HPC requires use of the "RETT 0" instruction, rather than "RETI", to return from maskable as well as nonmaskable interrupts.

Given these two constraints, it was considered most useful to modify MON16 to contain a set of interrupt service routines, which would then use a set of addresses in RAM (a table at address "vex") to call custom interrupt servers as standard subroutines. An interrupt service routine calls its custom subroutine after saving the dedicated registers and the general registers, R0, R1 and R2 on the stack.

The symbol "vex" is defined externally, and must be declared to match the address used by the modified MON16. Details of the modified MON16 are available from National Semiconductor Corporation, Microprocessor Applications Group or the Microcontroller Applications Group, phone (408) 721-5000. These modifications are also a standard part of the MONCG monitor program for the NS32CG016 microprocessor.

4.3.1 Panel Exerciser Program

This program for the NS32CG16 CPU exercises several functions of a panel consisting of the following:

- A two-line (8 chars. per line) LCD panel, arranged horizontally into a single 16-line display.
- A speaker, activated by the BEEP command.
- Six pushbuttons, which are presented by the !BUTTON-DATA interrupt to the CPU as follows:
 Keyboard Status Bute

		Re	yboard	Jiaiu	SDyle		
0	PB6	PB5	PB4	0	PB2	PB1	PB0
			المراجع المراجع				

• Five LED's, activated in the SEND-LED command by the following bits:

LED Control Byte

— — LD5 LD4 LD3 LD2 LD1	_
-------------------------	---

The intended layout for the front panel is as shown below. (Please pardon the apparently haphazard assignment of the pushbuttons and LED's; this was dictated by the nature of the module we used for developing this application.)

						F	ront Pa	anel La	yout							
Cursor Addr. \rightarrow	00	01	02	03	04	05	06	07	40	41	42	43	44	45	46	47
LCD's:	*			*			*			*			*			*
LED's:	LD2			LD3			LD1			LD5			LD4			(Beep)
PB's:	PB5]		PB1			PB0			PB4			PB6]		PB2

The locations shown with asterisks on the LCD panel above will display an asterisk character while the corresponding pushbutton below it is depressed. (The number above each LCD location indicates its cursor address in hexadecimal.) Each time a pushbutton (except PB2) is pressed, the corresponding LED indicator above it is toggled. Rather than toggling an LED, PB2 causes a BEEP command to be issued. The program starts up the panel with the LCD display blank, and LED's LD1 and LD2 on.

1 2			# Front	Panel Exerciser Prog	ram.	
3 4 5 6			# "vex" # "vex" #	contains absolute ad +4 starts list of mas first is interrupt	dress of NMI service r kable interrupt routin Øx10.	outine entry point. e entry points;
7 8 9 10 11 12			# Note: # # #	Before running, mak Also, all unused in	hat it is running in S e sure to set PSR to Ø kterrupts automatically wint should be set the	200 hex. branch to label
13 14 15 16 17 18			.globl .globl .globl .globl .globl	start,main rtcint lcdint swint badint		
19 20 21 22			.set .set .set	hpcctrl,ØxFFFCØØ hpcdata,ØxFFFEØØ hpcpoll,ØxFDØØØØ	# HPC Control/Stat # HPC Data I/O loc # HPC Poll address	ation.
23 24 25 26 27 28 29			.set .set .set .set .set .set	INIT,0x0 SET_CONT,0x1 SEND_LCD,0x2 SEND_LED,0x3 BEEP,0x4 RESET_HPC,0xA5		
30 31 32		start:		# Fi	ill interrupt vector lo	ocations.
33 34	TØØØØØØØØ	67ddc000 025a0000	addr	badint, vex	# Interrupt NMI.	(Unimplemented)
35	TØØØØØØØa	0000 67ddc000 0250000	addr	badint,vex+4	# Interrupt Øx1Ø.	(Unimplemented)
36	TØØØØØØ14	0004 67ddc000 021c0000	addr	rtcint,vex+8	<pre># Interrupt Øx11.</pre>	Real-Time Clock.
37	TØØØØØØ1e	Ø23cØØØØ	addr	badint, vex+12	# Interrupt Øx12.	(Unimplemented)
38	TØØØØØØ28	000c 67ddc000 02320000	addr	badint,vex+16	# Interrupt Øx13.	(Unimplemented)
39	TØØØØØØ32	02280000	addr	badint,vex+20	# Interrupt Øx14.	(Unimplemented)
40	T0000003c	Ø21eØØØØ	addr	badint,vex+24	# Interrupt Øx15.	(Unimplemented)
41	TØØØØØ46	0018 67ddc000	addr	badint,vex+28	# Interrupt Øx16.	(Unimplemented)

43 44 45 46 47 48 49 59 51 52 53 54 55 56 57 58 59 60 61 62 65 66	TØØØØØØØ50 TØØØØØØ53 TØØØØØØ64 TØØØØØØ64 TØØØØØØ65 TØØØØØØ82 TØØØØØØØ82 TØØØØØØØ82 TØØØØØØØ82 TØØØØØØØ82 TØØØØØØØ82 TØØØØØØØ82 TØØØØØØØ82 TØØØØØØ82 TØØØØØØ82 TØØØØØØ82 TØØØØØØ82 TØØØØØØ82 TØØØØØØØ82 TØØØØØØØ82 TØØØØØØ82 TØØØØØØ82 TØØØØØØ82 TØØØØØØ82 TØØØØØØ82 TØØØØØØ82 TØØØØØ82 TØØØØØ82 TØØØØØ82 TØØØØØ82 TØØØØ82 TØØØØØ82 TØØØØØ82 TØØØØ82 TØØØØØ82 TØØØØØ82 TØØØØ908 TØØØØØ82 TØØØØ82 TØØØ908 TØØØ908 TØØØ908 TØØØØ908 TØØØ908 TØØØ908	01-6809090 9020 67-ddc9090 9022 67-ddc9090 9022 67-ddc9090 9028 67-ddc9090 9028 67-ddc9090 9020 67-ddc9090 9030 67-ddc9090 9034 67-ddc9090 9035 67-ddc9090 9035 67-ddc9090 9035 67-ddc9090 9036 67-ddc9090 9037 67-ddc9090 9038 57-ddc9090 9038 57-ddc9090 9038 57-ddc9090 9038 57-ddc9090 9038 57-ddc9090 9038 57-ddc9090 9038 57-ddc9090 9049 9058 57-ddc9090 9049 9058 57-ddc9090 9049 9058 57-ddc9090 9058 57-ddc900 90		addr addr addr addr addr addr addr addr	<pre>lcdint,vex+3 swint,vex+34 badint,vex+4 badint,vex+4 badint,vex+4 badint,vex+4 badint,vex+5 badint,vex+6 badint,vex+6 badint,vex+6 badint,vex+7 \$INIT,hpcctr \$Ø,hpcdata \$SEND_LED,hp \$ØxØ6,hpcdat \$\$XP06,leds</pre>	5 9 9 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	# Inter # Inter # Inter # Inter # Inter # Inter # Inter # Inter TIALIZE co value: f	rupt Øx18. rupt Øx19. rupt Øx1A. rupt Øx1B. rupt Øx1C. rupt Øx1C. rupt Øx1E. rupt Øx1F. rupt Øx21. mmand. eature dis	(Unimplemented) (Unimplemented) (Unimplemented) Diagnostic: stop. (Unimplemented) (Unimplemented) (Unimplemented) (Unimplemented)
43 44 45 46 47 48 49 59 51 52 53 54 55 56 57 58 59 60 61 62 65	T PPPPPPPP T PPPPPPP	67ddc9090 91260909 9029 67ddc9090 9024 67ddc9090 9028 67ddc9090 9028 67ddc9090 9028 67ddc9090 9029 67ddc9090 9030 67ddc9090 91280909 9030 67ddc9090 91280909 9033 67ddc9090 91260909 9036 67ddc9090 91260909 9036 67ddc9090 9036 67ddc9090 9036 67ddc9090 9036 67ddc9090 9036 67ddc9090 9049 81260909 9049 81260909 9049 81260909 9049 81260909 9049 81260909 9049 81260909 9049 81260909 9049 81260909 9049 81260909 9049 81260909 9049 81260909 9049 81260909 9049 81260909 9049 81260909 9049 81260909 9049 81260909 9049 81260000 81260000 81260000 8126000000000000000000000000000000000000		addr addr addr addr addr addr addr addr	swint, vex+36 badint, vex+4 badint, vex+4 badint, vex+4 badint, vex+5 badint, vex+5 badint, vex+6 badint, vex+6 badint, vex+6 badint, vex+7 \$INIT, hpcctrr \$Ø, hpcdata \$SEND_LED, hp	5 9 9 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	# Inter # Inter # Inter # Inter # Inter # Inter # Inter # Inter TIALIZE co value: f	rupt Øx18. rupt Øx19. rupt Øx1A. rupt Øx1B. rupt Øx1C. rupt Øx1C. rupt Øx1E. rupt Øx1F. rupt Øx21. mmand. eature dis	 Pushbutton event. (Unimplemented) (Unimplemented) (Unimplemented) (Unimplemented) Diagnostic: stop. (Unimplemented) (Unimplemented) (Unimplemented) (Unimplemented) (Unimplemented) (Unimplemented) (Unimplemented) (Unimplemented)
44 45 46 47 48 49 59 51 52 53 55 55 55 55 56 57 58 59 60 61 62 62 65 65	TØPØPØPØP64 TØPØPØPØ64 TØPØPØPØP82 TØPØPØPØP86 TØPØPØPØP86 TØPØPØPØP86 TØPØPØPØP86 TØPØPØPØP86 TØPØPØPØP86 TØPØPØPØP86 TØPØPØPØP86 TØPØPØPØP65 TØPØPØPØP63	67ddc000 01ea0000 0022 67ddc000 01ea000 0028 67ddc000 01ec0000 01ec0000 01ec0000 01ec0000 01ec0000 01ec0000 01a0000 01a0000 01a0000 01a0000 01a0000 01a0000 01a0000 01a0000 01a0000 01a0000 01a0000 01a0000 01a0000 01a0000 01a0000 01a0000 0543500 543500c0 fffe00 5435000 fffe00 543500c0 fffe00 543500c0 fffe00 543500c0 fffe00 543500c0 fffe00 543500c0 fffe00 543500c0 fffe00 543500c0 fffe00 543500c0 fffe00 543500c0 fffe00 543500c0 fffe00 543500c0 fffe00 5435000 fffe00 543500c0 fffe00 543500c0 fffe00 543500c0 fffe00 543500c0 fffe00 543500c0 fffe00 5435000 fffe00 5435000 fffe00 5435000 fffe00 5435000 fffe00 5435000 fffe00 5435000 fffe00 5435000 fffe00 5435000 fffe00 5435000 fffe00 5435000 fffe00 5435000 fffe00 5435000 fffe00 5435000 fffe00 5435000 fffe00 5435000 fffe00 5435000 fffe00 fffe00 5435000 fffe00 5435000 fffe00 5435000 fffe00 5435000 fffe00 5435000 fffe00 5435000 fffe00 fffe00 5435000 fffe00 5435000 fffe00 5435000 fffe00 5435000 fffe00 5435000 fffe00		addr addr addr addr addr addr addr addr	badint, vex+4 badint, vex+4 badint, vex+4 badint, vex+5 badint, vex+6 badint, vex+6 badint, vex+6 badint, vex+6 badint, vex+7 \$INIT, hpcctrr \$Ø, hpcdata \$SEND_LED, hp	99 4 88 52 56 54 58 54 58 54 54 58 54 54 54 54 54 54 54 54 54 54 55 54 54	# Inter # Inter # Inter # Inter # Inter # Inter # Inter TIALIZE co value: f	rupt Øx19. rupt Øx1A. rupt Øx1B. rupt Øx1C. rupt Øx1D. rupt Øx1E. rupt Øx1F. rupt Øx29. rupt Øx21. mmand. eature dis	<pre>(Unimplemented) (Unimplemented) (Unimplemented) (Unimplemented) Diagnostic: stop. (Unimplemented) (Unimplemented) (Unimplemented) (Unimplemented)</pre>
45 46 47 48 49 50 51 52 53 55 56 57 58 59 60 61 62 63 64 65	ТРРРРРРРС ТРРРРРРРС ТРРРРРРРС ТРРРРРРРС ТРРРРРРРР ТРРРРРРРС ТРРРРРРРС ТРРРРРРРС ТРРРРРРРС ТРРРРРРС ТРРРРРРС ТРРРРРРС ТРРРРРРС	0024 67ddc000 01f60000 01c0000 01c00000 01c00000 01c00000 01c00000 01c00000 01c00000 01c00000 01c00000 01c000000 0004 01c00000 01c00000 01c000000 00000000000 0004 000000000000000000000000000000000000		addr addr addr addr addr addr addr addr	badint,vex+4 badint,vex+4 badint,vex+5 badint,vex+5 badint,vex+6 badint,vex+6 badint,vex+6 badint,vex+7 \$INIT,hpcctr \$0,hpcdata \$SEND_LED,hp	4 8 52 66 54 58 72 *1 # INIT # RTC bectrl	# Inter # Inter # Inter # Inter # Inter # Inter # Inter TIALIZE co value: f	rupt Øx1A. rupt Øx1B. rupt Øx1C. rupt Øx1D. rupt Øx1E. rupt Øx1F. rupt Øx2Ø. rupt Øx21. mmand. eature dis	(Unimplemented) (Unimplemented) (Unimplemented) Diagnostic: stop. (Unimplemented) (Unimplemented) (Unimplemented) (Unimplemented)
46 47 48 49 50 51 52 53 54 55 55 56 57 58 59 60 61 62 63 64 65	ТФРФФФФ78 ТФФФФФФ82 ТФФФФФФ8с ТФФФФФФ96 ТФФФФФФ96 ТФФФФФФ64 ТФФФФФФ65 ТФФФФФФ65 ТФФФФФФ63	0028 67ddc000 01cc0000 01cc0000 01cc0000 01c20000 01d80000 0044 01d80000 01d80000 01d80000 01d80000 01d80000 01d80000 01d800000 01d8000000 01d80000000000000000000000000000000000		addr addr addr addr addr addr addr movb movb movb	badint,vex+4 badint,vex+5 badint,vex+5 badint,vex+6 badint,vex+6 badint,vex+7 \$INIT,hpcctr \$Ø,hpcdata \$SEND_LED,hp	88 52 56 59 54 58 58 58 52 51 <i>#</i> INIT <i>#</i> RTC 500000000	# Inter # Inter # Inter # Inter # Inter # Inter TIALIZE co value: f	rupt Øx18. rupt Øx10. rupt Øx10. rupt Øx1E. rupt Øx1F. rupt Øx20. rupt Øx21. mmand. eature dis	(Unimplemented) (Unimplemented) Diagnostic: stop. (Unimplemented) (Unimplemented) (Unimplemented) (Unimplemented)
47 48 49 59 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65	TØØØØØØ82 TØØØØØØ8c TØØØØØØ8c TØØØØØØ8a TØØØØØØ8a TØØØØØØ8be TØØØØØØbe TØØØØØØ65 TØØØØØØcc TØØØØØØca	002c 67ddc0000 91220000 9030 67ddc0000 91320 67ddc0000 91220000 9122000 9120000 9122000 91220000 91220000 91220000 91220000 91220000 912200000 912200000 912200000 912200000000000000000000000000000000000		addr addr addr addr addr addr movb movb movb	badint,vex+5 badint,vex+5 badint,vex+6 badint,vex+6 badint,vex+7 \$INIT,hpcctr \$0,hpcdata \$SEND_LED,hp	52 56 50 54 58 72 51 # INIT # RTC 500000000	# Inter # Inter # Inter # Inter # Inter TIALIZE co value: f	rupt Øx1C. rupt Øx1D. rupt Øx1E. rupt Øx1F. rupt Øx20. rupt Øx21. mmand. eature dis	(Unimplemented) Diagnostic: stop. (Unimplemented) (Unimplemented) (Unimplemented) (Unimplemented)
48 49 59 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65	ТФРФФФФВс ТФРФФФФФ ТФРФФФФФа ТФРФФФФФа ТФРФФФФФс ТФРФФФФСс ТФРФФФФС3	0030 67-04-000 91-0500 01-0500 01-050000 01-050000 01-050000 01-05000		addr addr addr addr addr movb movb movb	badint,vex+5 badint,vex+6 badint,vex+6 badint,vex+6 badint,vex+7 \$INIT,hpcctr \$Ø,hpcdata \$SEND_LED,hp	56 59 54 58 72 *L # INIT # RTC bectrl	# Inter # Inter # Inter # Inter # Inter TIALIZE co value: f	rupt Øx1D. rupt Øx1E. rupt Øx1F. rupt Øx2Ø. rupt Øx21. mmand. eature dis	Diagnostic: stop. (Unimplemented) (Unimplemented) (Unimplemented) (Unimplemented)
49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 64 65 66	ТФФФФФФФ ТФФФФФФФ ТФФФФФФФ ТФФФФФФФ ТФФФФФФ	003.4 67ddc000 91ce0000 01ce0000 03c 67ddc000 04000 093c 67ddc000 01ce000 01ce000 01ce000 01ce000 01c0000 01c0000 01c0000 01c0000 01c0000 01c00000 01c00000 01c00000 01c00000 01c00000 01c000000 01c000000 01c000000000 000000000000000000000000000000000000		addr addr addr addr movb movb movb	<pre>badint,vex+d badint,vex+d badint,vex+d badint,vex+f \$INIT,hpcctr \$0,hpcdata \$SEND_LED,hp \$0x06,hpcdata</pre>	50 54 58 72 51 # INIT # RTC 500000001	# Inter # Inter # Inter # Inter TIALIZE co value: f	rupt Øx1E. rupt Øx1F. rupt Øx2Ø. rupt Øx21. mmand. eature dis	(Unimplemented) (Unimplemented) (Unimplemented) (Unimplemented)
59 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66	Т Ø Ø Ø Ø Ø Ø Ø Ø Т Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø	0038 67ddc0000 01c40000 07ddc0000 01ba00000 00400 67ddc0000 01b000000 00400 00400 0048 54a500c0 fffc00 54a500c0 fffc00 54a503c0 fffc00 54a503c0 fffc00 54a503c0 fffc00 54a503c0 fffc00 54a503c0 fffc00 54a503c0 fffc00 54a503c0 fffc00 ffffc00 ffffc00 ffffc00 ffffffff		addr addr addr movb movb movb	badint,vex+d badint,vex+d badint,vex+7 \$INIT,hpcctr \$Ø,hpcdata \$SEND_LED,hp \$ØxØ6,hpcdat	54 58 72 *I # INIT # RTC bectrl	# Inter # Inter # Inter TIALIZE co value: f	rupt Øx1F. rupt Øx2Ø. rupt Øx21. mmand. eature dis	(Unimplemented) (Unimplemented) (Unimplemented)
59 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66	Т Ø Ø Ø Ø Ø Ø Ø Ø Т Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø Ø	01-640000 093c 67cdc000 01ba0000 01ba0000 01b0000 0040 01b0000 0044 67cdc0000 01a60000 0048 54a500c0 fffc00 54a500c0 fffc00 54a503c0 fffc00 fffc00 54a503c0 fffc00 54a5		addr addr addr movb movb movb	badint,vex+d badint,vex+d badint,vex+7 \$INIT,hpcctr \$Ø,hpcdata \$SEND_LED,hp \$ØxØ6,hpcdat	54 58 72 *I # INIT # RTC bectrl	# Inter # Inter # Inter TIALIZE co value: f	rupt Øx1F. rupt Øx2Ø. rupt Øx21. mmand. eature dis	(Unimplemented) (Unimplemented) (Unimplemented)
51 52 53 54 55 56 57 58 59 60 61 62 63 64 64 65 66	Т <i>ФФФФФФ</i> Т <i>ФФФФФФ</i> Т <i>ФФФФФФ</i> Т <i>ФФФФФФ</i> Т <i>ФФФФФ</i> Т <i>ФФФФФ</i> Т <i>ФФФФФ</i> Т <i>ФФФФФ</i> 3	0150000 0040 0755000 0755000 0040 0040 004		addr addr movb movb movb	badint,vex+d badint,vex+d \$INIT,hpcctr \$Ø,hpcdata \$SEND_LED,hp \$ØxØ6,hpcdat	58 72 1 # INIT # RTC Sectrl	# Inter # Inter TIALIZE co value: f	rupt Øx20. rupt Øx21. mmand. eature dis	(Unimplemented) (Unimplemented) abled.
52 53 54 55 56 57 58 59 60 61 62 63 64 65	Т00000006 Т00000065 Т000000c5 Т000000cc Т000000c3	015000000 0044 67446000 01460000 0048 54450060 fffc00 54450060 fffc00 5445060 fffc00 54450660 000145		addr movb movb movb	badint,vex+i \$INIT,hpcctr \$Ø,hpcdata \$SEND_LED,hp \$ØxØ6,hpcdat	2 t # INIT # RTC pectrl	# Inter TIALIZE co value: f	rupt Øx21. mmand. eature dis	(Unimplemented) abled.
53 54 55 56 57 58 59 60 61 62 63 64 63 64 65 66	TØØØØØØbe TØØØØØØc5 TØØØØØØcc TØØØØØØd3	67645000 01460000 9048 544500c0 fffc00 544500c0 fffe00 544503c0 fffe00 544506c0 fffe00 c44606c0 p00145		movb movb movb movb	\$INIT,hpcctr \$Ø,hpcdata \$SEND_LED,hp \$ØxØ6,hpcdat	l # INIT # RTC pectrl	TIALIZE co value: f	mmand. eature dis	abled.
54 55 57 58 59 60 61 62 63 64 65 66	TØØØØØØØC5 TØØØØØØØCC TØØØØØØd3	54a500c0 fffc00 54a500c0 fffe00 54a503c0 fffc00 54a506c0 fffe00 d4a606c0 000145		movb movb	\$0,hpcdata \$SEND_LED,hp \$0x06,hpcdat	# RTC	value: f	eature dis	
56 57 58 59 60 61 62 63 64 65 66	TØØØØØØCC TØØØØØØd3	543500c0 fffe00 543503c0 fffc00 543506c0 fffe00 d43606c0 000145		movb movb	\$SEND_LED,hp \$ØxØ6,hpcdat	occtrl			
57 58 59 60 61 62 63 64 65 65 66	TØØØØØØd3	54a503c0 fffc00 54a506c0 fffe00 d4a606c0 000145		movb	\$ØxØ6,hpcdat		# Initi	alize LEDs	to normal state.
59 60 61 62 63 64 65 66		54a506c0 fffe00 d4a606c0 000145				а			
60 61 62 63 64 65 66	TØØØØØØda	d4a606c0 000145		movb	\$ØxØ6,leds				
61 62 63 64 65 66						# Save	e in memor	y image.	
64 65 66	TØØØØØØe1	7da30800	run:	bispsru	w \$0x800	# Enat	ble interr	upts from	HPC.
66			main:		# Ma	in program	m starts h	еге.	
	1000000e5	5cd8c000		movqb	\$Ø,lcdflg		waiting f		
									TL/DD/9

		Ø139					
68 69	T000000eb	54a5Ø2cØ		movb	\$SEND_LCD, hpcctr	l # Turn off LCD cursor and clear p	banel.
70	T000000f2	fffcØØ 54a5ØØcØ		movb	\$Ø,hpcdata		
71	T000000f9	fffeØØ 54a5Ø2cØ		movb	\$2,hpcdata		
72	TØØØØØ1ØØ	fffeØØ 54a5ØccØ		movb	\$ØxØC,hpcdata		
73	TØØØØØ1Ø7	fffeØØ 54a5Ø1cØ		movb	\$1,hpcdata		
74		fffeØØ					
75	TØØØØØ1Øe	f4a600c0 000110	l1:	tbitb		# Wait for panel available.	
76 77	100000115	9a79		bfc	11		
78	100000117	5cd8c000 0104		movqb	\$Ø,kbdflg		
79 8Ø							
81 82			kbdlp:				
83	T0000011d	f4a6ØØcØ ØØØØfe	12:	tbitb	., .	# Wait for keyboard data.	
84 85	TØØØØØ124	9a79		bfc	12	4 Country and underse	
86 87	TØØØØØ126 TØØØØØ12a	7da10800 14d8c000		bicpsrw movb	\$Øx8ØØ kbdnew,rØ	# Sample, and update semaphores.	
88	TØØØØØ13Ø	ØØf2 54d8cØØØ		movb	kbdold,r1		
89	TØØØØØ136	ØØed d4decØØØ ØØe6cØØØ ØØe7		movb	kbdnew,kbdold		
9Ø	TØØØØØ14Ø	pper 5cd8c000 00db		movqb	\$Ø,kbdflg		
91 92	TØØØØØ146	7da30800		bispsrw	\$Ø×8ØØ		
92 93 94	TØØØØØ14a	5f1Ø		movqd	\$Ø,r2	# Initialize offset pointer in r2.	
95 96 97	TØØØØØ14c TØØØØØ14e TØØØØØ15Ø	7800 1c08 1a10		xorb cmpqb bne		# Generate map of differing bits. # Check that a change actually occurred.	
98 99	TØØØØØ152	54a5Ø3cØ		movb	\$SEND_LED, hpcctr	l # If not, error is shown by turning on	
100	TØØØØØ159	fffcØØ 54a52ØcØ fffeØØ		movb	\$0x20,hpcdata	# ALARM LED.	
101 102			lcdlp:				
1Ø3 1Ø4 105	T00000160	6e84Ø8 8abfba	(cutp:	ffsb bfs	r1,r2 kbdlp	<pre># Find first differing bit. # If none, go wait for another keyboard @</pre>	event.
105 106	TØØØØØ163 TØØØØØ166	8abtba 4e481Ø		cbitb	r2,r1	# If none, go wart for another keyboard (# Clear difference flag.	
							TL/DD/9976-

107 108	TØØØØØ169	5cd8c000 00b5		movqb	\$Ø,lcdflg	# Do LCD command: first clear Acknowledge fl	ag.
109 110	TØØØØØ16f	74a500c0 fd0000	l3:	tbitb	\$0,hpcpoll		
111 112	TØØØØØ176 Tøøøøø178	8a79 54a502c0 fffc00		bfs movb	l3 \$SEND_LCD,hpcc	trl # Start command to display new bit state.	
113 114	TØØØØØ17f	74a500c0 fd0000	l4:	tbitb	\$0,hpcpoll		
115 116	TØØØØØ186 TØØØØØ188	8a79 54a502c0 fffe00		bfs movb	l4 \$2,hpcdata	# Flags: One command followed by one data.	
117 118	TØØØØØ18f	74a500c0 fd0000	l5:	tbitb	\$Ø,hpcpoll		
119 12Ø	TØØØØØ196 Tøøøøø198	8a79 54a5Ø2cØ fffeØØ		bfs movb	l5 \$2,hpcdata	# Two data bytes follow.	
121 122	T0000019f	74a500c0 fd0000	l6:	tbitb	\$Ø,hpcpoll		
123 124	TØØØØØ1a6 TØØØØØ1a8	8a79 54e5dacØ ØØØØ78cØ fffeØØ		bfs movb	l6 lcdloc[r2:b],h	pcdata # Send cursor position byte.	
125 126	TØØØØØ163	74a500c0 fd0000	ι7:	tbitb	\$Ø,hpcpoll		
127 128 129 130	T000001ba T000001bc T000001be T000001c0	8a79 3410 8a0c 54a520c0		bfs tbitb bfs movb	l7 r2,rØ l8 \$Øx2Ø,hpcdata	# If new bit is zero, send blank.	
131	TØØØØØ1c7	fffeØØ ea8Ø46		br	lout		
132 133	TØØØØØ1ca	54a52acØ fffeØØ	l8:	movb	\$Øx2A,hpcdata	# If bit is one, send asterisk instead,	
134 135	T000001d1	74a500c0 fd0000	19:	tbitb	\$Ø,hpcpoll		
136 137 138 139	T000001d8 T000001da T000001dd	8a79 34a002 9a0b		bfs tbitb bfc	l9 \$2,rØ l1Ø	# and if the key is MENU,	
140	T000001df	54a504c0 fffc00		movb	\$BEEP,hpcctrl	# then beep,	
141 142	TØØØØØ1e6	ea27		br	lout		
143	TØØØØØ1e8	54a503c0 fffc00	l10:	movb	\$SEND_LED, hpcc		
144	T000001ef	f8e6dacØ ØØØØ39cØ		xorb	ledloc[r2:b],l		
						TL/I	DD/99

145	TØØØØØ1fa		L11:	tbitb	\$Ø,hpcpoll	
146 147		54ddc000		bfs movb	l 11 l eds, hpcdata	
148		ØØ1ccØff feØØ				
149		f4a600c0 000011	lout:	tbitb	\$Ø,lcdflg	# Wait for LCD Acknowledge interrupt.
15Ø 151				bfc	lout	
152 153 154		eabf4a		br	lcdlp	# Go check for any more differing bits.
155 156		1200		ret	Ø # Er	d of main program.
157 158 159			mainda	t:	# Data for M	ain Program.
160 161 162	TØØØØØ215 TØØØØØ21c TØØØØØ21d	00 00	kbdnew kbdold	: .byte : .byte : .byte	Ø #Ne Ø #Sa	yboard data ready. w keyboard data (from interrupt service). ved (previous) keyboard states.
163 164 165	TØØØØØ21f		leds:	: .byte .byte		D display ready. D states.
166	TØØØØØ22Ø	c18Øc481		: .byte		C7,0x81,0xC1,0x80,0xC4,0x81
167 168		02080000 20041000	ledloc	: .byte	ØxØ2,ØxØ8,Ø>	0,0×0,0×20,0×04,0×10,0×0
169 170 171 172 173 174 175				#Invo # sa # Be	ked by ROM int wed, but no EM cause ROM moni	Service Routines. errupt service. Registers RØR2 are already TER instruction has been performed yet. tor returns using "EETI", we must bypass it tly with "RETT Ø".
176 177 178			rtcint	:	# Interrupt	Øx11. Real-Time Clock.
179 18Ø	TØØØØØ23Ø	ea2a		br	badint	# UNEXPECTED (bypass code below) # Interrupt return procedure:
181 182 183	TØØØØØ232 TØØØØØ234 TØØØØØ236	72eØ		cmpqd restor rett	\$0,tos e [r0,r1,r2] 0	 # Discard return address to monitor. # Restore registers saved by monitor. # Return from interrupt directly.
184 185 186		dcd8ffff	lcdint	: movqb	<pre># Interrupt \$1,lcdflg</pre>	Øx17. LCD data written. # Flag that interrupt has occurred.
187 188 189 190 190	TØØØØØ23e TØØØØØ24Ø TØØØØØ242	72eØ		cmpqd restor rett	\$0,tos e [r0,r1,r2] 0	<pre># Interrupt return procedure: # Discard return address to monitor. # Restore registers saved by monitor. # Return from interrupt directly.</pre>
.,,						TL/C
GNX Sei	ries32000 CC	OFF ASSEMBLEF	R Version	2.5 6/6	5/88 Pag	2: 6
192 193	TØØØØØ244	dcd8ffff ffd7	swint:	movqb	<pre># Interrupt Ø \$1,kbdflg</pre>	<pre>(18. Pushbutton event. # Flag that interrupt has occurred.</pre>
194	TØØØØØ24a	d4aecØff feØØffff ffd2		movb	hpcdata,kbdne	# Save new keyboard state.
197 198		1fb8 72eØ		restore	\$0,tos [r0,r1,r2] 0	<pre># Interrupt return procedure: # Discard return address to monitor. # Restore registers saved by monitor. # Return from interrupt directly.</pre>
199 200 201			badint:		# Trap for un	mplemented interrupts. PLACE BREAKPOINT HERE.
202 203 204	TØØØØØ25a TØØØØØ25c TØØØØØ25e	72eØ			\$Ø,tos [rØ,r1,r2] Ø	<pre># Interrupt return procedure: # Discard return address to monitor. # Restore registers saved by monitor. # Return from interrupt directly.</pre>
						TL/C

1 2 3 4 5 6 7 7 8 9 10 11 12 13 14 15 16 17 18			# # "vex" # "vex" #	contains absolute +4 starts list of mas first is interrupt This code assumes t Before running, mak Also, all unused ir	skable interrupt routi :Øx10. :hat it is running in ke sure to set PSR to	e routine entry point. ne entry points; Supervisor Mode. 0200 hex.
4 5 7 8 9 10 11 12 13 15 16 17 18			# "vex" # # Note: #	+4 starts list of mas first is interrupt This code assumes t Before running, mak Also, all unused ir	skable interrupt routi :Øx10. :hat it is running in ke sure to set PSR to	ne entry points; Supervisor Mode. Ø200 hex.
8 9 10 11 12 13 14 15 16 17 18			# Note: # #	This code assumes t Before running, mak Also, all unused ir	that it is running in the sure to set PSR to	0200 hex.
11 12 13 14 15 16 17 18			#	Also, all unused in	te sure to set PSR to	0200 hex.
14 15 16 17 18				"badint"; a breakp	wint should be set th	
17 18			globl	start,main		
	•		.globi .globi .globi	rtcint lcdint swint		
19 20 21	1		.globl .set	badint hpcctrl,ØxFFFCØØ	# HPC Control/Sta	tus I/O location.
22 23			.set .set	hpcdata,ØxFFFEØØ hpcpoll,ØxFDØØØØ	# HPC Data I/O lo # HPC Poll addres	cation.
24 25 26			.set .set .set	INIT,ØXØ SET_CONT,ØX1 SEND_LCD,ØX2		
27 28 29	, 3		.set .set .set	SEND_LED,Øx3 BEEP,Øx4 RESET_HPC,ØxA5		
3Ø 31		sta	art:			
32 33 34		67ddc000	addr	# Fi badint,vex	ill interrupt vector l # Interrupt NMI.	(Unimplemented)
35	T0000000a	924e9999 9999 67ddc999	addr	badint,vex+4	# Interrupt Øx10.	(Unimplemented)
36	TØØØØØ014	02440000 0004 67ddc000 02040000	addr	rtcint,vex+8	# Interrupt Øx11.	Real-Time Clock.
37	TØØØØØ01e	0008 67ddc000 02300000	addr	badint,vex+12	# Interrupt Øx12.	(Unimplemented)
38	TØØØØØ28	000c 67ddc000 02260000	addr	badint,vex+16	# Interrupt Øx13.	(Unimplemented)
39	TØØØØØØ32	0010 67ddc000 021c0000	addr	badint,vex+20	# Interrupt Øx14.	(Unimplemented)
40	T000003c	0014 67ddc000 02120000	addr	badint,vex+24	# Interrupt Øx15.	(Unimplemented)
41	TØØØØØØ46	0018 67ddc000	addr	badint,vex+28	# Interrupt Øx16.	(Unimplemented)
						TL/DD/9976-

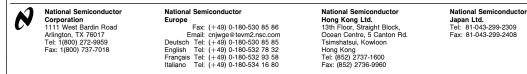
		02080000			
4	2 100000050	001c 67ddc000 01e40000	addr	lcdint,vex+32	# Interrupt Øx17. LCD data written.
4	3 TØØØØØØ5a	0020 67ddc000 01e80000	addr	swint,vex+36	<pre># Interrupt Øx18. Pushbutton event.</pre>
4.	4 100000064	0024 67ddc000 01ea0000	addr	badint,vex+40	<pre># Interrupt Øx19. (Unimplemented)</pre>
4	5 TØØØØØØ6e	0028 67ddc000 01e00000	addr	badint,vex+44	<pre># Interrupt Øx1A. (Unimplemented)</pre>
4.	5 TØØØØØØ78	002c 67ddc000 01d60000	addr	badint,vex+48	<pre># Interrupt Øx1B. (Unimplemented)</pre>
4	7 100000082	0030 67ddc000 01cc0000	addr	badint,vex+52	<pre># Interrupt Øx1C. (Unimplemented)</pre>
4	B TØØØØØØ8c	0034 67ddc000 01c20000	addr	badint,vex+56	<pre># Interrupt Øx1D. Diagnostic: stop.</pre>
4	9 TØØØØØ96	0038 67ddc000 01b80000	addr	badint,vex+60	<pre># Interrupt Øx1E. (Unimplemented)</pre>
5	Ø TØØØØØØaØ	993c 67ddc999 91ae9999	addr	badint,vex+64	<pre># Interrupt Øx1F. (Unimplemented)</pre>
5	1 TØØØØØØaa	0040	addr	badint,vex+68	<pre># Interrupt Øx2Ø. (Unimplemented)</pre>
5	2 100000064	0044 67ddc000 019a0000	addr	badint,vex+72	<pre># Interrupt @x21. (Unimplemented)</pre>
5 5	3 4 TØØØØØØbe		movb	\$INIT,hpcctrl	# INITIALIZE command.
5		fffcØØ 54a5Ø5cØ fffeØØ	movb	\$5,hpcdata	<pre># RTC value: interval of 59 milliseconds.</pre>
5 5	7 TØØØØØØcc	5cd8c000 013e	movqb	\$Ø,flags	# Clear interrupt flags.
5 5 6	9	d4a614cØ	.set .set movb	rtcflg,Ø lcdflg,1 \$20,rtcctr	<pre># Bit Ø means RTC interrupt detected. # Bit 1 means LCD interrupt detected. # Clear RTC modulus counter (div by 2Ø).</pre>
6	1 TØØØØØØd9	000139 5fd8c000 0133	movqd	\$Ø,timent	# Clear seconds counter.
			un: bispsrw	\$0x800	# Enable interrupts from HPC.
6	6 7			# Neit	her communication port is selected yet.
					TL/DD/

69 70 Тророровс 4ec8a 70 Тророровс 54350 71 Тророровс 54350 72 Тророровс 54350 73 Тророровс 54350 74 Тророровс 54350 74 Тророровс 54350 75 Тророровс 54350 76 Тророровс 54350 76 Тророровс 9070 77 Тророровс 9470 78 Тророровс 54350 79 Тророровс 54350 79 Тророровс 9470 77 Тророровс 9470 78 Тророровс 54350 80 Тророровс 54353 91 Тророровс 54353 92 Тророровс 54353 93 Тророровс 54353 94 Тророровс 54353 95 Тророровс 54353 96 Тророровс 54	<pre>\$lcdflg,flags # Place cursor at first character of panel. \$SEND_LCD,hpcctrl \$Ø,hpcdata \$1,hpcdata \$\$vx80,hpcdata \$lcdflg,flags U1 \$lcdflg,flags # Write initial value of zeroes. \$SEND_LCD,hpcctrl \$0xFF,hpcdata \$0x30,hpcdata \$0x30,hpcdata \$0x30,hpcdata \$0x30,hpcdata \$0x30,hpcdata \$0x30,hpcdata \$0x30,hpcdata \$0x30,hpcdata \$0x30,hpcdata \$1cdflg,flags U2 \$rtcflg,flags mainlp \$rtcflg,flags</pre>
71 ТØØØØØØeb 54350 72 ТØØØØØØF2 54350 73 ТØØØØØF2 54350 73 ТØØØØØF2 54350 74 ТØØØØØF2 54350 74 ТØØØØØF7 54350 75 ТØØØØØF7 7460 76 ТØØØØØF7 780000 77 78 ТØØØØ0100 79 ТØØØØ0110 4ce8a 80 TØØØØ0120 54353 77 TØØØØ0120 54353 77 TØØØØ0118 54353 79 TØØØØ0120 54353 81 TØØØØØ135 54353 76 TØØØØØ135 54353 83 TØØØØØ142 54353 76 TØØØØØ150 54353 71 TØØØØØ150 54353 71 TØØØØØ157 54353 76 TØØØØØ150 54353 70 TØØØØØ155 54353 70 TØØØØØ155 54353 <t< td=""><td><pre>\$\$ hpcdata \$\$ hpcdata \$\$ hpcdata \$\$ hpcdata \$\$ lcdflg,flags ll \$lcdflg,flags # Write initial value of zeroes. \$\$ END_LCD,hpcctrl \$\$ kpcdata \$\$ hpcdata \$\$ hpcda</pre></td></t<>	<pre>\$\$ hpcdata \$\$ hpcdata \$\$ hpcdata \$\$ hpcdata \$\$ lcdflg,flags ll \$lcdflg,flags # Write initial value of zeroes. \$\$ END_LCD,hpcctrl \$\$ kpcdata \$\$ hpcdata \$\$ hpcda</pre>
72 TØØØØØØF2 54359 73 TØØØØØF9 54358 74 TØØØØØF9 54358 75 TØØØØØF9 54358 75 TØØØØØF9 54358 76 TØØØØØF9 9a79 77 TØØØØØF19 4ec8a 79 TØØØØØF19 4ec8a 79 TØØØØØF15 54358 77 TØØØØØF15 54359 79 TØØØØØF15 54358 81 TØØØØØF15 54353 81 TØØØØØF15 54353 76 TØØØØØF15 54353	<pre>\$1,hpcdata \$0x80,hpcdata \$lcdflg,flags 11 \$lcdflg,flags # Write initial value of zeroes. \$SEND_LCD,hpcctrl \$0xFF,hpcdata \$0x30,hpcdata \$0</pre>
73 ТРРРРРД 5455 74 ТРРРРД 5458 74 ТРРРРД 1445 75 ТРРРРД 1445 76 ТРРРРД 1445 76 ТРРРРД 1445 77 ТРРРРД 1445 78 ТРРРД 1445 79 ТРРРД 1445 79 ТРРРД 1445 80 ТРРРД 145 81 ТРРРД 145 82 ТРРРД 145 83 ТРРРД 145 84 ТРРД 145 84 ТРРД 145 85 ТРРД 145 86 ТРРД 145 87 ТРРД 145 88 ТРРД 145 87 ТРРД 145 88 ТРРД 145 87 ТРРД 145 88 ТРРД 145 89 ТРРД 145 90 ТРРД 145 91 ТРРД 145 92 ТРРД 145 93 ТРД 145 94 ТРД 145 94 ТРД 145	<pre>\$\$\phix8\$\phi,hpcdata \$\$lcdflg,flags 11 \$lcdflg,flags # Write initial value of zeroes. \$\$END_LCD,hpcctrl \$\$0xFF,hpcdata \$\$0x3\$\phi,hpcdata \$\$1cdflg,flags 12 \$rtcflg,flags mainlp</pre>
74 Т00000100 54558 75 Т00000107 fffep 75 Т00000107 fffep 76 Т00000107 f4ac0 77 Т00000107 f4ac0 79 Т00000110 4ec8a 6 Т00000111 54a50 79 Т00000114 54a51 80 Т00000126 54a53 76 Т00000135 54a53 76 Т00000142 54a53 76 Т00000150 54a53 76 700000150 54a53 <td><pre>\$lcdflg,flags [1 \$lcdflg,flags # Write initial value of zeroes. \$SEND_LCD,hpcctrl \$ØxFF,hpcdata \$Øx30,hpcdata \$Icdflg,flags [2 \$rtcflg,flags mainlp</pre></td>	<pre>\$lcdflg,flags [1 \$lcdflg,flags # Write initial value of zeroes. \$SEND_LCD,hpcctrl \$ØxFF,hpcdata \$Øx30,hpcdata \$Icdflg,flags [2 \$rtcflg,flags mainlp</pre>
75 TØØØØØ107 f4a60 ØØ10 76 TØØØØ0109 9010 76 TØØØØ010 4ec8a cØØØ 78 TØØØØØ115 54a50 78 TØØØØ0115 54a50 80 TØØØØ0126 54a50 81 TØØØØ0126 54a53 82 TØØØØ0126 54a53 83 TØØØØ0135 54a53 84 TØØØØ0142 54a53 85 TØØØØ0142 54a53 86 TØØØØ0159 54a53 87 TØØØØ0159 54a53 88 TØØØØ0155 54a53 90 TØØØØ0156 54a53 91 TØØØØØ156 54a53 92 TØØØØØ165 64a59 93 TØØØØØ166 9a79 93 TØØØØØ166 54a59 94 TØØØØØ166 9a79	<pre>l1 slcdflg,flags # Write initial value of zeroes. ssEND_LCD,hpcctrl s0xFF,hpcdata s8,hpcdata s0x30,hpcdata s0x30,hpcdata s0x30,hpcdata s0x30,hpcdata s0x30,hpcdata s0x30,hpcdata s1cdflg,flags l2 srtcflg,flags mainlp</pre>
76 TØØØØØ10e 9a79 77 TØØØØ010e 4ec8a 78 TØØØØ0110 4ec8a 79 TØØØØ0110 4ec8a 79 TØØØØ0110 54a50 80 TØØØØ0116 54a50 81 TØØØØ0126 54a50 82 TØØØØ0126 54a50 83 TØØØØ0130 54a53 84 TØØØØ0130 54a53 85 TØØØØ0142 54a53 86 TØØØØ0142 54a53 87 TØØØØ0159 54a53 88 TØØØØ0159 54a53 90 TØØØØ0157 54a53 90 TØØØØ0155 54a53 91 TØØØØ0156 54a53 91 TØØØØØ156 54a53 91 TØØØØØ156 54a53 92 TØØØØØ156 54a53 91 TØØØØØ156 54a53 92 TØØØØØ156 54a53 94 TØØØØØ156 54a53 <td><pre>\$lcdflg,flags # Write initial value of zeroes. \$SEND_LCD,hpcctrl \$\$xFF,hpcdata \$\$,hpcdata \$\$x30,hpcdata \$\$x40,hpcdata \$\$x40</pre></td>	<pre>\$lcdflg,flags # Write initial value of zeroes. \$SEND_LCD,hpcctrl \$\$xFF,hpcdata \$\$,hpcdata \$\$x30,hpcdata \$\$x40,hpcdata \$\$x40</pre>
78 ТРФФФФ110 4-ec8. 79 ТРФФФФ118 54a50 79 ТРФФФФ118 54a50 80 ТРФФФФ118 54a50 81 ТРФФФФ118 54a50 82 ТРФФФФ126 54a50 83 ТРФФФФ126 54a53 84 ТРФФФФ126 54a53 84 ТРФФФФ126 54a53 84 ТРФФФФ126 54a53 85 ТРФФФФ129 54a53 86 ТРФФФФ157 54a53 87 ТРФФФФ157 54a53 86 ТРФФФФ157 54a53 87 ТРФФФФ157 54a53 88 ТРФФФФ157 54a53 90 ТРФФФФ155 54a53 91 ТРФФФФ155 54a53 92 ТРФФФФ155 54a53 94 ТРФФФФ155 54a53 94 ТРФФФФ155 54a53 97 ТРФФФФ155 54a53 96 ТРФФФФ155 54a53 </td <td><pre>\$SEND_LCD,hpcctrl \$0xFF,hpcdata \$0,hpcdata \$0,x30,hpcdata \$0x30,hpcdata \$0x30,hpc</pre></td>	<pre>\$SEND_LCD,hpcctrl \$0xFF,hpcdata \$0,hpcdata \$0,x30,hpcdata \$0x30,hpcdata \$0x30,hpc</pre>
79 TØØØØØ118 54a50 80 TØØØØØ114 54a50 81 TØØØØØ124 54a50 82 TØØØØØ124 54a53 83 TØØØØØ124 54a53 84 TØØØØØ124 54a53 85 TØØØØØ135 54a53 86 TØØØØØ142 54a53 87 TØØØØØ149 54a53 86 TØØØØØ157 54a53 87 TØØØØØ155 54a53 90 TØØØØØ155 54a53 91 TØØØØØ155 54a53 92 TØØØØØ156 54a53 91 TØØØØØ156 54a53 92 TØØØØØ156 54a53 93 TØØØØØ156 54a53 94 TØØØØØ156 54a53 972 TØØØØØ0156 54a53	<pre>\$PxFF,hpcdata \$PxFF,hpcdata \$Px3P,hpcdata \$PxSP,hpcdata \$PxSP,hpcda</pre>
80 T0000011f 5455 81 T00000126 54350 82 T00000126 54350 83 T00000135 54353 84 T00000135 54353 84 T00000135 54353 84 T00000135 54353 85 T00000150 54353 86 T00000150 54353 87 T00000150 54353 88 T00000155 54353 90 T000000155 54353 91 T000000155 54353 92 T000000155 54353 93 T000000155 54353 94 T000000155 54353 95 T000000155 54353 96 T000000155 54353 97 T000000155 54353 97 T000000155 54353 98 T000000155 54353 99 T000000155 94353 99 T000000155 943535<	<pre>\$8, hpcdata \$9x39, hpcdata \$1cdflg, flags l2 \$rtcflg, flags mainlp</pre>
81 TØØØØØ126 54558 82 TØØØØØ124 54a53 82 TØØØØØ124 54a53 83 TØØØØØ134 54a53 84 TØØØØØ134 54a53 85 TØØØØØ142 54a53 85 TØØØØØ142 54a53 86 TØØØØØ149 54a53 87 TØØØØØ159 54a53 88 TØØØØØ159 54a53 90 TØØØØØ159 54a53 91 TØØØØØ155 54a53 91 TØØØØØ155 54a53 91 TØØØØØ155 54a53 92 TØØØØØ165 64a59 93 TØØØØØ166 9a79 93 TØØØØØ166 64a69 90 4<	\$0x30, hpcdata \$0x30, hpcdata \$0x30, hpcdata \$0x30, hpcdata \$0x30, hpcdata \$0x30, hpcdata \$0x30, hpcdata \$0x30, hpcdata \$1cdflg, flags 12 \$rtcflg, flags mainlp
82 TØØØØØ12d 54533 83 TØØØØØ134 54533 84 TØØØØØ135 5453 84 TØØØØØ134 5453 85 TØØØØØ135 5453 86 TØØØØØ142 5453 87 TØØØØ0150 5453 87 TØØØØ0150 5453 88 TØØØØ0157 5453 90 TØØØØ0156 5453 91 TØØØØØ157 5453 91 TØØØØØ165 64669 92 TØØØØØ166 64669 93 TØØØØØ166 64669 94 TØØØØØ166 64669 92 TØØØØØ166 74669	\$\$x30, hpcdata \$\$x30, hpcdata \$\$x30, hpcdata \$\$x30, hpcdata \$\$x30, hpcdata \$\$x30, hpcdata \$\$x30, hpcdata \$\$x30, hpcdata \$\$zcdflg, flags l2 \$rtcflg, flags mainlp
83 Т909090134 54533 84 Т00909013b 54533 85 Т009090142 54533 86 Т009090142 54533 86 Т009090142 54533 87 Т009090149 54533 87 Т009090159 54533 88 Т009090157 54533 98 Т000909157 54533 99 Т000909155 54653 90 Т000909156 54653 91 Т000909156 54653 91 Т00000156 94379 92 73 Т00000166 94379 93 Т00000175 9479 94 Т00000175 9479	\$\$x30, hpcdata \$\$x30, hpcdata \$\$x30, hpcdata \$\$x30, hpcdata \$\$x30, hpcdata \$\$x30, hpcdata \$\$x30, hpcdata \$\$ucdflg, flags l2 \$rtcflg, flags mainlp
84 TØØØØØ13b 54533 85 TØØØØØ142 54533 85 TØØØØØ142 54533 86 TØØØØØ159 54533 87 TØØØØØ159 54533 88 TØØØØØ157 54533 89 TØØØØØ157 54533 90 TØØØØØ156 54533 91 TØØØØØ155 54533 92 TØØØØØ156 64630 93 TØØØØØ166 9379 93 TØØØØ0166 54653 94 TØØØØØ166 74640 99 94 TØØØØØ166	\$0x30, hpcdata \$0x30, hpcdata \$0x30, hpcdata \$0x30, hpcdata \$0x30, hpcdata \$0x30, hpcdata \$1cdflg, flags 12 \$rtcflg, flags mainlp
85 TØØØØØ142 5453 86 TØØØØØ149 54653 87 TØØØØØ159 54653 87 TØØØØØ157 54653 88 TØØØØØ157 54653 89 TØØØØØ157 54653 90 TØØØØØ155 54653 91 TØØØØØ156 54653 92 TØØØØØ166 9279 93 TØØØØØ166 64609 94 TØØØØØ166 74609 94 TØØØØØ166 9279	\$0x30, hpcdata \$0x30, hpcdata \$0x30, hpcdata \$0x30, hpcdata \$0x30, hpcdata \$1cdflg, flags 12 \$rtcflg, flags mainlp
86 ТØØØØØ149 5453 87 ТØØØØØ15Ø 5453 87 ТØØØØØ15Ø 5453 88 ТØØØØØ157 5453 89 ТØØØØØ155 5453 90 ТØØØØØ155 5453 90 ТØØØØØ155 5453 91 ТØØØØØ155 6463 92 22 93 ТØØØØØ16C 93 ТØØØØØ16E 64640 99 94 ТØØØØ0162 9379	\$0x30,hpcdata \$0x30,hpcdata \$0x30,hpcdata \$lcdflg,flags l2 \$rtcflg,flags mainlp
87 TØØØØØ15Ø 5453 88 TØØØØØ157 5453 89 TØØØØØ157 5453 90 TØØØØØ156 5453 90 TØØØØØ156 5453 90 TØØØØØ156 5453 91 TØØØØØ156 64650 92 93 TØØØØØ166 9479 93 TØØØØØ166 64660 ØØØØ9 94 TØØØØØ156 54653	\$0x30,hpcdata \$0x30,hpcdata \$lcdflg,flags l2 \$rtcflg,flags mainlp
88 TØØØØØ157 54533 89 TØØØØØ155 54a53 90 TØØØØ0155 54a53 90 TØØØØ0155 54a53 91 TØØØØ0155 54a53 92 93 TØØØØØ166 64a50 93 TØØØØ0166 64a60 94 TØØØØ0167 9a79	\$0x30,hpcdata \$lcdflg,flags l2 \$rtcflg,flags mainlp
89 ТØØØ0015e 54653 90 TØØ000165 fffe0 91 TØØ00016c 9a79 92 93 TØØ00016e f4a60 94 TØØ000167 9a79 94 TØØ000175 9a79	<pre>\$lcdflg,flags l2 \$rtcflg,flags mainlp</pre>
90 T00000165 f4a60 0000 91 T0000016c 9a79 92 93 T0000016e f4a60 00009 94 T00000175 9a79	l2 \$rtcflg,flags mainlp
91 TØØØØØ16C 9a79 92 93 TØØØØØ16e f4a6Ø ØØØØ9 94 TØØØØØ175 9a79	<pre>\$rtcflg,flags mainlp</pre>
93 TØØØØ016e f4a6Ø ØØØØ 94 TØØØØ0175 9a79	mainlp
94 TØØØØØ175 9a79	
95 TØØØØØ177 4ec8a cØØØØ	
96 97 10000017f 7ca10	b \$0x01
98 TØØØØØ182 4effa ØØØØØ ØØØØ8	\$ØxØ1,timcnt # Increment BCD elapsed time.
99	TL/DD/9976

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

- Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
- A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.



National does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied and National reserves the right at any time without notice to change said circuitry and specifications.