A Comparison of the Event Processor Array (EPA) and High Speed Input/Output (HSIO) Unit

BRIAN HINTZMAN
AUTOMOTIVE APPLICATIONS ENGINEER

March 1991
Information in this document is provided in connection with Intel products. Intel assumes no liability whatsoever, including infringement of any patent or copyright, for sale and use of Intel products except as provided in Intel’s Terms and Conditions of Sale for such products.

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

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

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

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

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

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

or call 1-800-879-4683

COPYRIGHT © INTEL CORPORATION, 1995
A Comparison of the Event Processor Array (EPA) and High Speed Input/Output (HSIO) Unit

CONTENTS

1.0 INTRODUCTION .................................... 1
   1.1 General Overview ................................ 1
      1.1.1 HSI/O ..................................... 1
      1.1.2 EPA ....................................... 3
      1.1.3 PTS ....................................... 3
      1.1.4 How to Use This Application Note .......... 3

2.0 TIMERS ............................................ 4
   2.1a HSI/O Clocking .................................. 4
   2.1b EPA Clocking ................................... 4
   2.2a HSI/O Reset .................................... 5
   2.2b EPA Reset ..................................... 5
   2.3a HSI/O Count Direction ......................... 6
   2.3b EPA Count Direction ............................ 6

3.0 INPUTS ............................................ 7
   3.1a HSI/O Input Channels ............................ 7
   3.1b EPA Input Channels ............................. 7
   3.2a HSI/O Input Capture ............................ 7
   3.2b EPA Input Capture ............................. 7
   3.3a HSI/O Interrupts ............................... 8
   3.3b EPA Interrupts ................................. 8
   3.4a HSI/O Reading HSI Current State ............... 9
   3.4b EPA Reading EPA Current State ................. 9

4.0 OUTPUTS .......................................... 10
   4.1a HSI/O Output Channels .......................... 10
   4.1b EPA Output Channels ............................ 10
   4.2a HSI/O Output Events ............................ 11
   4.2b EPA Output Events ............................. 11
   4.3a HSI/O Output Event Buffering (CAM) ............ 12
   4.3b EPA Output Event Buffering ..................... 12
   4.4a HSI/O Event Locking ............................ 12
   4.4b EPA Event Locking ............................. 12

5.0 PULSE WIDTH MODULATION OUTPUT (PWM) .............. 13
   5.1a HSI/O Output Channels .......................... 13
   5.1b EPA Output Channels ............................ 13
   5.2a HSI/O Frequency ............................... 13
   5.2b EPA Frequency ................................. 13
   5.3a HSI/O Duty Cycle .............................. 13
   5.3b EPA Duty Cycle ................................. 13

6.0 EPA PWM EXAMPLES ................................ 14
   6.1 Example #1 ..................................... 14
   6.2 Example #2 ..................................... 15
   6.3 Example #3 ..................................... 16
   6.4 Example #4 ..................................... 17
   6.5 Example #5 ..................................... 18

7.0 CODED EXAMPLES .................................. 20
   7.1 Using the HSI/O to Capture Every Eighth Rising Edge 20
   7.2 Using the EPA to Capture Every Nth Edge ........... 21
   7.3 Using the HSI/O for an Eight Entry FIFO Unit .......... 23
   7.4 Using the EPA for an Eight Entry FIFO Unit .......... 24
   7.5 Using the EPA to Perform Multiple Output Events in Succession 26
   7.6 Using EPA2 to Clock TIMER2 Internally ............. 28

APPENDIX A .......................................... A-1
1.0 INTRODUCTION

In modern control applications, high speed communications with the outside world is an essential feature of today’s microcontroller. To handle such signals, Intel’s new generation of 16-bit microcontrollers, starting with the 8XC196KR, offers the Event Processor Array (EPA). The EPA’s many abilities make it versatile and ideal for such high speed signals.

Intel’s 16-bit microcontrollers prior to the 8XC196KR device provide the High Speed Input/Output unit (HSI/O) for high speed event control. The HSI/O serves as a basis for all high speed communications modules on Intel microcontrollers. The EPA is the result of an evolutionary process evaluating the needs of high speed input and output control. Because the EPA is a new peripheral module of the 16-bit microcontroller, the immediate question is how this module is comparable to its counterpart on previous MCS-96 devices. Though on the surface the HSI/O and EPA modules may appear to be vastly different, the reality is that the two are functionally very similar.

In this application note a general familiarity with the MCS-96 architecture is assumed. Its purpose is to compare the functions of the HSI/O and EPA and assist the programmer who wishes to use the EPA module and is already familiar with the HSI/O.
The HSI/O is capable of handling timed input and output events on a number of exterior pins. There are two dedicated input pins, four dedicated outputs, and two pins that are multiplexed between input and output. The module has two internal timers for the timing and scheduling of events. Up to a total of eight incoming events, rising or falling edges, on all input pins can be buffered at once in a FIFO storage unit. Up to eight output events for all output pins can be buffered at once in the Content Addressable Memory (CAM) unit. The HSI/O also features either one or three Pulse Width Modulation channels. The HSI/O appeared on the original 8X9X device and has changed only slightly as new parts have been released. Further information on the specific differences between these devices is given below. Also see the User's Guides for the 8096BH, 8XC196KB, and 8XC196KC in the 16-Bit Embedded Controllers Handbook.

1.1.2 EPA

The EPA had many attributes similar to the HSI/O. In the case of the 8XC196KR, there are ten EPA channels, called capture/compare modules, each of which can be selected as input or output. In addition there are two compare channels dedicated to the timing of internal events only. Again the module has two internal timers for the timing and scheduling of events. Two input events are buffered separately for each pin configured as an input and one output event for each pin configured as an output. More detailed information on all these functions is provided below as well as in the 8XC196KR User's Guide in the Automotive Products Handbook.

1.1.3 PTS

An additional module, first introduced on the 8XC196KC and 8XC196KR, called the Peripheral Transaction Server (PTS) greatly enhances the capabilities of the EPA. The PTS is an interrupt handler that performs very fast Direct Memory Access interrupts. These interrupts can be initiated by internal or external events, hence the close connection between the EPA and PTS. Though the PTS will not specifically be discussed here, more information can be obtained from the 8XC196KC and 8XC196KR User's Guide. Examples of using the PTS in conjunction with the EPA are given through this text.

1.1.4 HOW TO USE THIS APPLICATION NOTE

The purpose of this paper is to provide a direct, side by side comparison of the HSI/O module in the left-hand column and the EPA module in the right-hand column. Both accomplish nearly the same functions but their implementations, and therefore for the necessary software, can be quite different. Specific functions of HSI/O and EPA are discussed and, where applicable, actual code showing actions performed in both modules is provided. In some cases, examples are also given of functions that have no equivalent in the other module.

Most of the code examples for the EPA make use of the new windowing capabilities as well as the new port configuration techniques that, like the EPA itself, are new on the 8XC196KR device. The window and port functions are not described here but are discussed in the 8XC196KR User's Guide. Note that the new windowing allows many addresses outside of the 00 to FF address space to be referenced as registers. The equate statements defining the register names used in this text are defined in the appendix. Also, whenever actual times are referred to instead of state times, a 16 MHz clock frequency is assumed. Finally, where the letters BH, KB, KC, and KR appear, these refer to the 8X9XBH, 8XC196KB, 8XC196KC, and 8XC196KR respectively.
2.0 TIMERS

2.1a Clocking

Both TIMER1 and TIMER2 are 16-bit timers used for time stamping of incoming events and for scheduling of output events. TIMER1 is a free running timer whose only clocking mode is internal clocking once every eight state times. TIMER2 is clocked by transitions, both rising and falling edges, on either T2CLK or HSI.1 but the maximum clock rate is still once every eight state times. On the KB and KC, TIMER2 can be clocked once every state time in Fast Increment Mode and on the KC, TIMER2 can also be clocked internally.

Ex. 1a. Configure TIMER2 to Count Externally

```
LDB IOC0, #00H
LDB IOC1, #00H
LDB IOC2, #00H

• T2CLK pin is clock source
• Count up clocked by rising and falling edges on T2CLK pin
• External reset disabled
• Do not reset timer each write
• Disable overflow interrupt
• Disable fast increment mode - KB and KC only
```

2.1b Clocking

Both TIMER1 and TIMER2 are 16-bit timers. TIMER1 and TIMER2 can be clocked externally through T1CLK and T2CLK or clocked internally. The maximum clock rate, internal or external, is the chip clock rate divided by 4. Internally, prescaling allows clocking at several selectable fractions of the chip clock frequency. Externally, prescaling allows clocking at several selectable fractions of the T1CLK or T2CLK frequency. TIMER1 and TIMER2 can be chained together to produce a single 32-bit counter by clocking TIMER2 with the overflow of TIMER1. A quadrature counting option allows use of the EPA with an encoder wheel.

The principle difference between the EPA and HSI/O timers is that TIMER1 is not free running, it MUST be configured in order to count whereas it formerly would automatically begin counting on start up. Also note that all functions of the timers are held in the two TIMERn_CONTROL registers rather than mixed in with the IOCn registers allowing complete configuration of a timer in one memory write.

Ex. 1b. Configure TIMER2 to Count Externally

```
LDB WSR, #7EH
LDB P1IO_2, #0FFH
LDB P1SEL_2, #01H
LDB WSR, #7CH
LDB TIMER2_CONTROL_2, #0C8H

• P1.0 selected for input as T2CLK pin
• Count up clocked by rising and falling edges on T2CLK pin
```

Ex. 1c. Chaining Two Timers Together

```
LDB WSR, #7CH
LDB TIMER1_CONTROL_2, #0C2H
LDB TIMER2_CONTROL_2, #0F0H

• TIMER1 counts up with 1 µs period
• TIMER2 counts on TIMER1 overflow and in the same direction as TIMER1
```
HSI/O

2.2a Reset

TIMER1 can only be reset by a chip reset. TIMER2 can be reset by a chip reset, setting bit 1 in IOC0, setting either T2RST or HSI.0 depending on the value of IOC0.5, or by any of the HSO's. On the KB and KC, both timers may be written with any value, which may also be considered “resetting” the timer.

- Resets TIMER1 every 500 clocks
- Event locked in CAM (KB and KC only)
- No interrupts generated by CAM or TIMER1

Ex. 2a. Reset TIMER2 with HSO

Ex. 2a. Reset TIMER2 with HSO

LDB HSO__COMMAND, #8EH
LD HSO__TIME, #500

EPA

2.2b Reset

Both timers can be reset by any of the EPA channels. Choosing the appropriate EPA control mode allows either timer to reset itself or the opposite timer. Both timers can be loaded with any time value. Though there is no auto-reload capability this function can be produced using the PTS (see Appendix 7.3).

The main difference here is that the new system has no dedicated T2RST pin for external resetting of TIMER2. However, any of the 10 EPA channels can be configured to reset either timer or an input event as shown below.

- TIMER1 counts every 1 μs
- Resets TIMER1 every 500 clocks (500 μs)
- COMP0 time entry locked
- No interrupts generated by COMP0 or TIMER1

Ex. 2b. Reset TIMER2 with Compare Channel 0

Ex. 2b. Reset TIMER2 with Compare Channel 0

LDB WSR, #7CH
LDB COMP__CONTROL0, #0FE49H
LD COMP__TIME0, #500
LDB TIMER1__CONTROL, #0C2H

- Port 1.0 is EPA input
- Captures on rising edge
- TIMER2 counts every 1 μs
- Captures TIMER1, resets TIMER2, (opposite timer)
- NOTE: MUST capture one timer and reset the OPPOSITE timer to reset on input event
- NOTE: Cannot be done without interrupts on HSI/O

Ex. 2c. Reset TIMER2 on Input Rising Edge
**HSI/O**

**2.3a Count Direction**

In the BH part, both timers can only count up. On the KB and KC, the TIMER2 direction can be chosen by the value of Port 2.6. The timer direction cannot be controlled internally.

<table>
<thead>
<tr>
<th>LDB IOC2, #02H</th>
</tr>
</thead>
<tbody>
<tr>
<td>• TIMER2 counts down if Port 2.6 is high, up if low</td>
</tr>
</tbody>
</table>

**Ex. 3a. Control TIMER2 Direction Externally**

---

**EPA**

**2.3b Count Direction**

Both timers can determine their count direction from either their internal timer control register or on the T1DIR and T2DIR pins. Also, the count direction of TIMER2 can be set to match that of TIMER1 so that TIMER1 and TIMER2 together perform 32-bit up/down counting.

Additionally, because the T2DIR pin is the same as the EPA2 pin, T2DIR can be controlled directly using EPA2. Setting the T2DIR as the direction control for TIMER2 and configuring EPA2 as an output allows the value of EPA2 to control the count direction of TIMER2 (see Ex. 3c.)

<table>
<thead>
<tr>
<th>LDB WSR, #7CH</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDB P1IO_2, #0FFH</td>
</tr>
<tr>
<td>LDB P1SEL_2, #04H</td>
</tr>
<tr>
<td>LDB TIMER2__CONTROL_2, #90H</td>
</tr>
<tr>
<td>• P1.2 selected as T2DIR</td>
</tr>
<tr>
<td>• TIMER2 counts up if T2DIR is high, counts down if low</td>
</tr>
</tbody>
</table>

**Ex. 3b. Control TIMER2 Direction Externally**

<table>
<thead>
<tr>
<th>LDB WSR, #7CH</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDB P1IO_2, #0FFH</td>
</tr>
<tr>
<td>LDB P1SEL_2, #04H</td>
</tr>
<tr>
<td>LDB WSR, #7BH</td>
</tr>
<tr>
<td>LD EPA__CONTROL2_2, #0FE70H</td>
</tr>
<tr>
<td>LDB WSR, #7CH</td>
</tr>
<tr>
<td>LDB TIMER1__CONTROL_2, #0C2H</td>
</tr>
<tr>
<td>LDB TIMER2__CONTROL_2, #0C6H</td>
</tr>
<tr>
<td>• P1.2 selected as EPA2 input and T2DIR</td>
</tr>
<tr>
<td>• EPA2 toggles pin so it changes TIMER1 direction on EPA__TIME2 match</td>
</tr>
<tr>
<td>• NOTE: Time entry not locked here so a new EPA__TIME2 value must be written for each direction change</td>
</tr>
</tbody>
</table>

**Ex. 3c. Control TIMER2 Direction Internally with EPA2**
3.0 INPUTS

HSI/O

3.1a Input Channels
The HSI/O has a total of four input channels. Two are dedicated lines and two are selectable as input or output. All events recorded on these lines go directly to the FIFO unit.

3.2a Input Capture
The HSI records the time (value of TIMER1 only) of logic transitions along with the pin on which they occur in the FIFO structure. Up to eight events can be stored in the FIFO at one time. The HSI is capable of capturing events on rising edges only, falling edges only, rising and falling edges, or every 8th rising edge. Events on separate pins that occur within the same clock period are stored in the same FIFO entry. Additionally, on the KB and KC, rising edges on the P2.7 pin capture the value of TIMER2 in the T2CAP register for input event processing that bypasses the FIFO and creates its own interrupt.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR B</td>
<td>INT_PEND</td>
</tr>
<tr>
<td>LDB</td>
<td>IOC0, #10H</td>
</tr>
<tr>
<td>LDB</td>
<td>IOC1, #00H</td>
</tr>
<tr>
<td>LDB</td>
<td>HSI_MODE, #10H</td>
</tr>
<tr>
<td>LDB</td>
<td>INT_MODE, #04H</td>
</tr>
<tr>
<td>EI</td>
<td></td>
</tr>
</tbody>
</table>

- HSI.2 input enabled
- Captures each positive edge on HSI.2
- Tags event with TIMER1 time (no choice)
- HSI interrupts on Holding Register loaded

Ex. 4a. Choosing HSI.2 as Input

EPA

3.1b Input Channels
The EPA has 10 external pins, each of which can individually be selected for input capture. Unlike the HSI/O, the EPA pins also serve as port pins. This means the pins MUST first be configured to function for the EPA and then the EPA__CONTROLn register must also be configured for capturing the appropriate input event. See the 8XC196KR User's Guide for more information on configuring the ports.

3.2b Input Capture
The EPA can time stamp input events with the time value of TIMER1 and TIMER2. In the EPA, each channel stores its input events separately in its EPA__TIME register rather than mixed together as in the FIFO. Each time register is buffered allowing the storage of two input events at once. Input capture events include rising edges, falling edges, or rising and falling edges. Capture on every 8 rising edges as the HSI/O does can be handled by the EPA using the PTS with more flexibility (see Section 7.1). The T2CAP register of the KB and KC parts does not exist in the EPA because every channel is capable of performing this function and generating its own interrupt.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR B</td>
<td>INT_PEND</td>
</tr>
<tr>
<td>LDB</td>
<td>WSR, #7EH</td>
</tr>
<tr>
<td>LDB</td>
<td>P1REG__2, #0FFH</td>
</tr>
<tr>
<td>LDB</td>
<td>P1IO__2, #0FFH</td>
</tr>
<tr>
<td>LDB</td>
<td>P1SESEL__2, #04H</td>
</tr>
<tr>
<td>LDB</td>
<td>WSR, #7BH</td>
</tr>
<tr>
<td>LD</td>
<td>EPA__CONTROL2__2, #0FE20H</td>
</tr>
<tr>
<td>LDB</td>
<td>INT_MASK, #04H</td>
</tr>
<tr>
<td>EI</td>
<td></td>
</tr>
</tbody>
</table>

- P1.2 selected as EPA input
- Captures each rising edge
- Tags event with TIMER1 time
- EPA2 interrupts on capturing rising edge

Ex. 4b. Choosing EPA.2 as Input
HSI/O

3.3a Interrupts

All incoming events produce interrupts solely through the FIFO unit and vector through the same address in the vector table (except for TIMER2 capture on the KB and KC). Since the time of an input event from ANY pin is provided in the HSI.TIME register, this register MUST be read in the interrupt service routine to allow further interrupts from any HSI pin to occur. Also, if multiple HSI channels are used, because of the common FIFO storage unit, the interrupt service routine must decode the HSI.STATUS register to determine which channel created the interrupt.

Because of the storage space of the FIFO, several input events can be stored at once before an interrupt is required. On the BH part an interrupt can be created with one or six entries in the FIFO. With the KB and KC parts an interrupt can also be created on the fourth entry, or when the FIFO is half full.

EPA

3.3b Interrupts

Unlike the HSI/O where all input interrupts are vectored to the same interrupt service routine, several of the EPA channels generate their own interrupts directly with no decoding. EPA channels 0 through 3 each have their own bits in the INTPEND register and their own vectors allowing easy interrupt handling for these channels. EPA channels 4 through 9 and Compare channels 0 and 1 all generate interrupts to the EPAINTX bit in the INTPEND register. However, for these interrupts, the TJMP command has been added to the 8XC196KR instruction set to be used with the EPAIPV register allowing the use of individual interrupt routines for EVERY interrupt with very little decode overhead in determining the source of the interrupt (see Section 7.5). Similar to the HSI/O, however, the EPA_TIME register must be read during each interrupt service routine to allow further interrupts on that channel.

The addition of the PTS module makes the interrupt capability of the EPA even more flexible. For every interrupt produced by the EPA, it is possible to produce a PTS interrupt instead. This is done by setting the corresponding bit in the PTS_SELECT register. As long as this bit is set, all interrupts associated with this bit will be PTS interrupts. Again, see the 8XC196KR User’s Guide for information on using the PTS.

<table>
<thead>
<tr>
<th>CLRB</th>
<th>INTPEND</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDB</td>
<td>WSR, #7EH</td>
</tr>
<tr>
<td>LDB</td>
<td>P1REG_2, #0FFH</td>
</tr>
<tr>
<td>LDB</td>
<td>P1IO_2, #0FFH</td>
</tr>
<tr>
<td>LDB</td>
<td>P1SEL_2, #04H</td>
</tr>
<tr>
<td>LDB</td>
<td>WSR, #7BH</td>
</tr>
<tr>
<td>LD</td>
<td>EPA_CONTROL2_2, #0FE20H</td>
</tr>
<tr>
<td>LD</td>
<td>PTS_SELECT, #0004H</td>
</tr>
<tr>
<td>LDB</td>
<td>INT_MASK, #04H</td>
</tr>
<tr>
<td>EPTS</td>
<td></td>
</tr>
<tr>
<td>EI</td>
<td></td>
</tr>
</tbody>
</table>

• Same as previous example except uses PTS interrupt

Ex. 4c. Choosing EPA2 as Input, Using PTS to Service Interrupt
3.4a  Reading HSI Current State

The current state of any of the HSI pins can be read directly from the HSI_STATUS register. However, there are two difficulties in handling HSI interrupts.

The first is that reading the HSI_TIME register causes the next entry to be read out of the FIFO and written to the HSI_STATUS register. If two or more HSI events occurred in the same clock cycle then handling the first event will cause the HSI_STATUS register to be written over and the second event is lost. This is because all of the individual HSI interrupt bits are stored in the HSI_STATUS register. This means an extra copy of HSI_STATUS must be held in software to be sure no events are lost.

The second difficulty is that the IOS1 register is cleared every time it is read. The IOS1 register is used to indicate when the HSI Holding Register is full, when the FIFO is full, and when any of the software timers have expired. This means that an extra copy of this register also must be kept in software if one wishes to check the status of all conditions. Great care must be taken when writing software for both situations.

3.4b  Reading EPA Current State

Determining the current state of any of the EPA pins can be accomplished easily by reading either the P1PIN register (EPA 0 through 7) or the P6PIN register (EPA 8 and 9). Reading bits to decode which input channel caused an interrupt is not necessary as in the HSI/O because the decoding is performed by hardware. As stated in the previous section, EPA0 through 3 have their own interrupt bits and vectors. All other EPA channels are vectored through the EPAINTX vector. Nowhere is it necessary to keep extra copies of registers to avoid losing interrupt data.
# 4.0 OUTPUTS

## HSI/O

### 4.1a Output Channels

Four dedicated output lines are available to the HSI/O as well as the two multiplexed lines that are selectable as input or output. Output events to all six output lines are controlled solely by the CAM unit. While input events can only be time stamped by TIMER1, output events can be based on TIMER1 or TIMER2.

<table>
<thead>
<tr>
<th>LDB</th>
<th>HSO_COMMAND, #20H</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>HSO_TIME, TIMER1, #200</td>
</tr>
</tbody>
</table>

- HSO.0 sets pin in 200 clocks of TIMER1
- TIMER1 counts every 8 state times (no choice)

**Ex. 5a. Set HSO.0 Pin Based on TIMER1**

## EPA

### 4.1b Output Channels

All ten of the external EPA pins can individually be configured for output events. Each channel has its own EPA_CONTROLn and EPA_TIMEn register for the scheduling of events. EPA channels 0 and 1 and channels 2 and 3 can be “mapped” together such that all of their external events appear on EPA pins 1 and 3 respectively. Compare channels 0 and 1, though they do not seem to have external pins, can be configured to set or reset pins. Output events of Compare channel 0 appear on EPA8 and those of Compare channel 1 appear on EPA9 (EPA8/Compare0 and EPA9/Compare1 can be thought of as permanently mapped together). The remapping of channels allows two events to be scheduled to occur on one pin in quick succession, faster than could be done with interrupts. Note: the mapping of two channels together on one pin is for output events only. This does not work for input.

<table>
<thead>
<tr>
<th>LDB</th>
<th>WSR, #7EH</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDB</td>
<td>P1REG , #0FEH</td>
</tr>
<tr>
<td>LDB</td>
<td>P1IO , #0FEH</td>
</tr>
<tr>
<td>LDB</td>
<td>P1SSEL , #01H</td>
</tr>
<tr>
<td>LDB</td>
<td>WSR, #7BH</td>
</tr>
<tr>
<td>LD</td>
<td>EPA_CONTROL0 , #0FE60H</td>
</tr>
<tr>
<td>LD</td>
<td>EPA_TIME0 , #200</td>
</tr>
<tr>
<td>LDB</td>
<td>WSR, #7CH</td>
</tr>
<tr>
<td>LDB</td>
<td>TIMER1_CONTROL , #0C2H</td>
</tr>
</tbody>
</table>

- P1.0 selected as EPA, output, initially low
- EPA0 sets pin in 200 clocks of TIMER1
- TIMER1 counts every 1 µs

**Ex. 5b. Set EPA0 Pin Based on TIMER1**
**HSI/O**

**4.2a Output Events**

Output events are initiated immediately when the time tag of an output event matches the value on the selected timer. The HSI/O can produce a number of different "output" events, not all affecting the external pins. The unit can raise or lower the logic levels of one or a group of the pins, reset TIMER2, or initiate an A/D conversion, while each action can also optionally cause an interrupt. Using the 4 software timers the unit can cause an interrupt without taking any other action. When several software timers are used, however, the source of interrupt must be decoded because all timers use the same interrupt vector.

*NOTE: Decoding of interrupt source by reading IOS1*

---

**EPA**

**4.2b Output Events**

EPA output events can be initiated from a time tag match with either timer. The various events controlled are similar to those of the HSI/O including setting or resetting a pin or toggling the pin, resetting either timer, and starting an A/D conversion, where each event can be accompanied by an interrupt. The four software timers of the HSI/O are replaced in the EPA by simply allowing an interrupt to occur without performing another action.

---

**Ex. 6a. Use Software TIMER0 to Create Interrupt**

<table>
<thead>
<tr>
<th>CLRB</th>
<th>INT__PEND</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDB</td>
<td>HSO_COMMAND, #58H</td>
</tr>
<tr>
<td>ADD</td>
<td>HSO_TIME, TIMER2, #1000</td>
</tr>
<tr>
<td>LDB</td>
<td>INT__MASK, #04H</td>
</tr>
</tbody>
</table>

*Software timer 0 set to interrupt in 1000 clocks of TIMER2*

*NOTE: Decoding of interrupt source by reading IOS1*

---

**Ex. 6b. Use COMP0 to Create Interrupt**

<table>
<thead>
<tr>
<th>LDB</th>
<th>WSR, #7CH</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD</td>
<td>COMP__CONTROL0__2, #0FE80H</td>
</tr>
<tr>
<td>ADD</td>
<td>COMP__TIME0__2, TIMER2, #1000</td>
</tr>
<tr>
<td>LDB</td>
<td>INT__MASK, #01H</td>
</tr>
<tr>
<td>EI</td>
<td></td>
</tr>
</tbody>
</table>

*COMP0 set to interrupt in 1000 clocks of TIMER2*

*NOTE: TJMP along with EPAIPV must be used to get to interrupt service routine*

---

**Ex. 6c. Using Remap Function to Produce Two Events on One Pin**

<table>
<thead>
<tr>
<th>LDB</th>
<th>WSR, #7EH</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDB</td>
<td>P1REG__2, #0FDH</td>
</tr>
<tr>
<td>LDB</td>
<td>P1IO__2, #0FDH</td>
</tr>
<tr>
<td>LDB</td>
<td>P1SEL__2, #02H</td>
</tr>
<tr>
<td>LDB</td>
<td>WSR, #7BH</td>
</tr>
<tr>
<td>LD</td>
<td>EPA__CONTROL0__2, #0FE60H</td>
</tr>
<tr>
<td>LD</td>
<td>EPA__TIME0__2, #500</td>
</tr>
<tr>
<td>LD</td>
<td>EPA__CONTROL1__2, #0FF50H</td>
</tr>
<tr>
<td>LD</td>
<td>EPA__TIME1__2, #501</td>
</tr>
<tr>
<td>LDB</td>
<td>WSR, #7CH</td>
</tr>
<tr>
<td>LDB</td>
<td>TIMER1__CONTROL__2, #0C0</td>
</tr>
</tbody>
</table>

*EPA0 and 1 remapped together*

*P1.1 selected as EPA output*

*EPA0 sets pin in 500 clocks of TIMER1, EPA1 reset pin 1 clock later*

*TIMER1 counts every 250 ns*
HSI/O

4.3a Output Event Buffering (CAM)

The CAM unit allows up to eight output events to be written, or buffered, at once. Each event occurs when its time tag matches the corresponding timer value, regardless of the order events are written to the CAM.

The CAM is loaded by writing the event to the HSO__COMMAND register followed by writing the time to the HSO__TIME register. When the HSO__TIME register is written it may take up to eight state times to actually load the CAM. When writing multiple events to the CAM one must be careful to observe the eight state time limit in writing the HSO__TIME register.

Once an event is written to the CAM it cannot be removed if, for instance, it needs to be rescheduled. For commands that affect the external pins, writing the opposite action for the same time and pin will prevent any action from being taken while taking up two of the eight CAM entries. For other actions, such as resetting TIMER2, there is no appropriate “opposite” action. However, the entire CAM can be cleared to remove the event by setting IOC2.7 (not always a viable option).

4.4a Event Locking

On the KB and KC parts, events written to the CAM can be “locked”. These events remain in the CAM instead of disappearing after executing once as normally occurs. They occur EVERY time the clock counts back around to match the time tag of the event. Care must be taken when using this option as these events can only be cleared from the CAM by clearing the entire CAM (setting bit IOC2.7) or resetting the chip.

Ex. 7a. Lock Event in CAM

<table>
<thead>
<tr>
<th>LDB HSO__COMMAND, #0A0H</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD HSO__TIME, TIMER1, #200</td>
</tr>
</tbody>
</table>

EPA

4.3b Output Event Buffering

One of the main differences between the HSI/O and EPA structures is that the EPA has no CAM unit. Instead, each pin holds one timed output event in the EPA__TIMEEn register.

An EPA output event is set up by first writing to the EPA__CONTROLn register for the desired channel and then writing the EPA__TIMEEn register. As soon as the event occurs, normal software or and interrupt routine can set up the next timed event.

Since all of the EPA channels have their own separate EPA__CONTROLn registers, events on different channels can be written as fast as desired. The eight state time wait period does not have to be observed as in the HSI/O. If a pending event must be changed or rescheduled, the appropriate register, either control or time, simply needs to be written over before the event takes place. Clearing all pending events is never necessary as is sometimes the case in the HSI/O.

The one advantage lost in the EPA is the scheduling of up to eight events at once as in the CAM. However, the fast interrupts of the PTS can be used to schedule many new events as soon as the previous event occurs. Also many more than eight events can be scheduled using this technique. Please see Appendix for an example.

4.4b Event Locking

The EPA also has locked events that occur every time the time tag matches the appropriate timer. Unlike the HSI/O they are easily removed when they are no longer desired. Simply writing the EPA__CONTROLX register with a new value will enable a new event that may or may not be locked, depending on the value of the RE bit.

Ex. 7b. Lock Event in EPA__TIMEEn

<table>
<thead>
<tr>
<th>LD EPA__CONTROL0__2, #0FE68H</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD EPA__TIME0__2, #200</td>
</tr>
</tbody>
</table>

• Same as Ex. 5b but event is now locked in EPA__TIME0 register
5.0 PULSE WIDTH MODULATION OUTPUT (PWM)

5.1a Output Channels

Another feature of Intel 16-bit microcontrollers prior to the KR device, yet functionally separate from the HSI/O structure, is the Pulse Width Modulation Output or PWM. Intended for slow response analog devices such as meters and motors, the PWM acts as a D/A converter producing a constant frequency, variable duty cycle square pulse train. The unit has its own eight-bit counter. An eight-bit PWM__CONTROL register determines on which of the 256 counts the output goes low. The BH part has one such dedicated channel while the KB and KC parts have 3 PWM pins.

5.2a Frequency

On the BH part, the PWM frequency is set where one period is equal to 256 state times. On the KB and KC parts the period can equal 256 or 512 state times. The actual frequency is then completely determined by the chip clock frequency.

5.3a Duty Cycle

The PWM duty cycle is only selectable with eight bits of precision, it must be one of 256 discrete values. Even when the period is equal to 512 state times there is no gain in duty cycle resolution, only a change in PWM frequency. The duty cycle can be of either polarity as long as care is taken to note that time in the PWM__CONTROL register is the time the signal goes LOW. Also note that a 0% duty cycle (always low) is achievable while 99.6% duty cycle (high) is the maximum.

LDB IOC1, #01H
LDB PWM__CONTROL, #80H

• PWM with 50% duty cycle

Ex. 8. Typical Use of PWM Unit

5.1b Output Channels

The KR device has no PWM output unit. The EPA by itself or using the PTS performs the functions of a dedicated PWM unit. Using the PTS PWM Mode or PTS PWM Toggle Mode the user is able to produce any signal the PWM unit produces with much more control over the actual shape of the wave. A variety of PWM outputs can be used on any of the 10 EPA channels. A number of those possible are shown in complete program examples in Section 6.

5.2b Frequency

Where the HSI/O’s PWM unit gives the user at most two frequencies to choose from, the EPA and PTS combination provides a very wide range of possible output frequencies. Depending on the process used a period range of as little as two state times up to 65536 state times (or longer) is possible. (Again, see Section 6 for examples.)

5.3b Duty Cycle

Any duty cycle of either polarity is achievable using the appropriate example found in Section 6. The resolution in selecting the duty cycle is always 1 state time. The number of different duty cycles is only limited by the number of state times in the period of the PWM signal produced.
6.0 EPA PWM EXAMPLES

6.1 Example #1

The first example stores and locks a time in an EPA_TIMEn register while the EPA_CONTROLn register
instructs the EPA channel to toggle the pin. Every time the clock counts around to the time value in the EPA_-
TIMEn register the pin toggles. This makes a low frequency, 50% duty cycle square wave. The frequency is variable
by changing the TIMER1 prescaling. No CPU overhead required.

```
EXAMPLE MODULE MAIN, STACKSIZE(20)

$NOLIST
$INCLUDE (KR.INC)
$LIST

STK EQU 200H

CSEG AT 2000H

LD SP,#STK ; Initialize Stack Pointer
CLRB INT_PEND ; Clear out interrupts
CLRB INT_PEND1 ;

LDB WSR,#7EH ; 32 byte window on 1FC0H
LDB P1REG_2,#0FFH ; Turn off pull down
LDB P1IO_2,#0FFH ; P1.0 is output
LDB P1SEL_2,#01H ; P1.0 is EPA0

LDB WSR,#78H ; 32 byte window on 1F60H
LD EPA_CONTROL0_2,#0FE78H ; TIMER1, toggle output, lock time entry
LD EPA_TIME0_2,#100 ; Time tag to toggle output

LDB WSR,#7CH ; 32 byte window on 1F80H
LD TIMER1_CONTROL_2,#0C2H ; Enable timer, count up, 1us period

SELF:
    SJMP SELF ; Let EPA take over

END
```

270968-4
6.2 Example #2

Here the PTS is used to produce a square wave of 50% duty cycle where the selection of frequency is much greater than in the previous example. Using the PTS PWM Mode, the interrupt adds the value of CONST1 to the value in the EPA_TIMEO register. On each interrupt the external pin is toggled. The PWM frequency is then selectable with 16 bits of resolution. This example produces a 50% duty cycle 1 KHz square wave. Very high frequency waves cannot be made, however, because two interrupt latencies will occur during each period.

```
EXAMPLE MODULE MAIN, STACKSIZE(20)
$Nolist
$include(KR. INC)
$List
STK EQU 200H
CSEG AT 2048H
   DCW PTS_UNUSED0
CSEG AT 2008H
   DCW EPA0_ISR
RSEG AT 70H
   PTS_UNUSED0: DSB 1
   PTS_CONTROL0: DSB 1
   PTS_SRC0: DSW 1
   CONST1: DSW 1
CSEG AT 2080H
DI
DPTS
   LD SF,#STK ; Initialize Stack Pointer
   CLR INT_PEND ; Clear out interrupts
   CLR INTPEND1
   LDB PTS_CONTROL0,#40H ; PTS PWM Mode
   LD PTS_SRC0,#EPA_TIME0 ; Source is EPA_TIME0
   LD CONST1,#500 ; Constant for half period
   LDB WSR,#7EH ; 32 byte window on 1FC0H
   LDB P1REG2,#0FH ; Turn off pull down
   LDB P1IO 2,#0FH ; P1.0 is output
   LDB P1SEL 2,#01H ; P1.0 is EPA0
   LDB WSR,#7BH ; 32 byte window on 1F60H
   LD EPA_CONTROL0 2,#0FE70H ; TIMER1, toggle output
   LD EPA_TIME0 2,#100 ; Initial toggle time
   LDB WSR,#7CH ; 32 byte window on 1F80H
   LD TIMER1_CONTROL 2,#0C2H ; Enable timer, 1us period
   LD PTS_SELECT,#0010H ; Enable PTS interrupt for EPA0
   LD INT_MASK,#10H ; Enable EPA0 interrupts
   EPTS
   EI

SELF:
   SJMP SELF ; Let EPA take over
EPA0_ISR:
   ORB INT_PEND,#10H ; Select PTS interrupt
END
270968-5
```
6.3 Example #3

The PTS Toggle Mode is used to produce a square wave with other than 50% duty cycle using only one EPA channel. On each interrupt either CONST1 or CONST2 is added to the EPA_TIME0 register. On the following interrupt the other constant is added. Each time the external pin is toggled. This allows specifying the high time of the signal in one constant and the low time in the other. The period of the signal is the sum of CONST1 and CONST2, the high time and the low time. This sum must be greater than the maximum interrupt latency.

```
EXAMPLE MODULE MAIN, STACKSIZE(20)

$NOLIST
$INCLUDE(KR.INC)
$LIST

STK EQU 200H

CSEG AT 2048H
DCW PTS_UNUSED0

CSEG AT 2008H
DCW EPA0_ISR

RSEG AT 70H

PTS_UNUSED0 D8B1
PTS_CTRL0 D8B1
PT0_SRC0 D8W1
PTS_CONST0 D8W1
PTS_CONST1 D8W1

CSEG AT 2009H
DI
DI

PTS

LD SP,#STK ; Initialize stack
CLRB INT_PEND ; Clear out interrupts
CLRB INT_PEND1 ;

LDB PTS_CTRL0,#42H ; PTS Toggle Mode
LD PTS_SRC0,#EPA_TIME0 ; Source is EPA_TIME0
LD PTS_CONST0,#500 ; Constant for positive half cycle
LD PTS_CONST1,#800 ; Constant for negative half cycle

LDB WSR,#7EH ; 32 byte window on 1FC0H
LDB P1REG_2,#0FPH ; Turn off pull down
LDB P1IO_2,#0FH ; P1.0 is output
LDB P1SEL_2,#01H ; P1.0 is EPA0

LDB WSR,#7BH ; 32 byte window on 1F60H
LD EPA_CTRL0_2,#0FE70H ; TIMER1, toggle output
LD EPA_TIME0_2,#100 ; Initial toggle time

LDB WSR,#7CH ; 32 byte window on 1F80H
LD TIMER1_CTRL0_2,#0C2H ; Enable timer, 1us period
LD PTS_SELECT,#0010H ; Enable PTS interrupt for EPA0
LD INT_MASK,#10H ; Enable EPA0 interrupts

EPTS
EI

SELF:
SJMP SELF ; Let EPA take over

EPA0_ISR:
CLR INT_PEND,#10H ; Select DTO interrupt

END

270968-6

16
6.4 Example #4

This example does not use the PTS, it simply uses the capabilities of the EPA. EPA0 and 1 are remapped together on EPA1. EPA0 sets the pin and resets TIMER1 while EPA1 takes care of resetting the pin. No interrupts are required, however TIMER1 is dedicated to this process alone as it resets in phase with the frequency of the signal produced. This is the fastest PWM frequency possible.

```
EXAMPLE     MODULE MAIN, STACKSIZE(20)
$NOLIST
$INCLUDE(KR.INC)
$LIST

STK        EQU 200H

CSEG AT 2080H
DI
DPTS

LD   SP,#STK ; Initialize stack
CLRB  INT_PEND ; Clear out interrupts
CLRB  INT_PEND1 ;

LDB  WSR,#7EH ; 32 byte window on 1FC0H
LDB  P1REG_2,#0FFH ; Turn off pull down
LDB  P1IO_2,#0FDH ; P1.1 is output
LDB  P1SEL_2,#02H ; P1.1 is EPA1

LDB  WSR,#78H ; 32 byte window on 1F60H
LD  EPA_CONTROL0_2,#0FE69H ; TIMER1, set pin, reset timer, lock time
LD  EPA_TIME0_2,#500 ; Time tag for setting pin
LD  EPA_CONTROL1_2,#0FF58H ; TIMER1, reset pin, lock time entry
LD  EPA_TIME1_2,#150 ; Time to reset pin

LDB  WSR,#7CH ; 32 byte window on 1F80H
LD  TIMER1_CONTROL_2,#0C2H ; Enable timer, count up, 1us period

SELF:
   SJMP SELF ; Let EPA take over

END
```
6.5 Example #5

Finally, the PTS PWM Toggle Mode is used with two channels mapped together to produce a square wave of arbitrary frequency and arbitrary duty cycle. EPA0 is set up to set the pin while EPA1 is set up to reset the pin. Each time each channel performs its function it also performs a PTS interrupt that adds the constant value from its PTS Control Block to the value in its EPA_TIMEn register. The advantage of the example over the previous one is that the timer can be free running and used for other functions as well rather than resetting for every period of the square wave. Note that the constant values of the two PTS channels must be the same to preserve the duty cycle. The duty cycle is chosen by the initial value of the two EPA_TIMEn registers.

```
EXAMPLE MODULE MAIN, STACKSIZE(20)
$NOLIST
$INCLUDE(KR.INC)
$LIST

STK EQU 200H
CSEG AT 2048H
   DCW PTS_UNUSED0
CSEG AT 2046H
   DCW PTS_UNUSED1
CSEG AT 2008H
   DCW EPA0_ISR
CSEG AT 2006H
   DCW EPA1_ISR
RSEG AT 70H
   PTS_UNUSED0: DSB 1
   PTS_CONTROL0: DSB 1
   PTS_SRC0: DSW 1
   CONST0: DSW 1
RSEG AT 80H
   PTS_UNUSED1: DSB 1
   PTS_CONTROL1: DSB 1
   PTS_SRC1: DSW 1
   CONST1: DSW 1
CSEG AT 2080H
   DI
   DPTS
   LD SP,#STK ; Initialize stack
   CLR B INTPEND ; Clear out interrupts
   CLR B INTPEND1 ;
   LDB PTS_CONTROL0,#40H
   LD PTS_SRC0,#EPA_TIME0
   LD CONST0,#500
   LDB PTS_CONTROL1,#40H
   LD PTS_SRC1,#EPA_TIME1
   LD CONST1,#500
```
6.5 Example #5 (Continued)

```
LDB    WSR,#7EH          ; 32 byte window on 1FC0H
LDB    P1REG_2,#0FFH     ; Turn off pull down
LDB    P1IO_2,#0FDH      ; P1.1 is output
LDB    P1SEL_2,#02H      ; P1.1 is EPA1

LDB    WSR,#7BH          ; 32 byte window on 1F60H
LD     EPA_CONTROL0_2,#0FE50H ; TIMER1, set pin
LD     EPA_TIME0_2,#500   ; 500us high time
LD     EPA_CONTROL1_2,#0FF50H ; TIMER1, reset pin
LD     EPA_TIME1_2,#800   ; 800us low time

LDB    WSR,#7CH          ; 32 byte window on 1F80H
LD     TIMER1_CONTROL_2,#0C2H ; Enable timer, count up, 1us period
LD     PTS_SELECT,#0018H
LDB    INT_MASK,#10H

EPTS
EI

SELF:
      SJMP SELF          ; Let EPA take over

EPA0_ISR:
      ORB INT_PEND,#10H
      RET

EPA1_ISR:
      ORB INT_PEND,#08H
      RET

END
```
7.0 CODED EXAMPLES

7.1 Using the HSIO to Capture Every Eighth Rising Edge

This program sets up HSI2 for capturing on every eighth edge. On interrupt, HSI TIME is read to allow further interrupts to occur.

```
EXAMPLE MODULE MAIN, STACKSIZE(20)

$NOLIST
$INCLUDE (RC.INC)
$LIST

STK EQU 200H

CSEG AT 2004H
DCW HSI2_ISR

CSEG AT 2080H

DI
LD SP,#STK
CLRB INT_FEND
CLRB INT_FEND1

LDB IOC0,#10H
LDB IOC1,#00H
LDB IOC2,#80H
LDB HSI_MODE,#00H
LDB INT_MASK,#04H
EI

SELF:
SJMP SELF

HSI2_ISR:
PUSHP
LD R0,#HSI_TIME
POFF
RET

END
```
7.2 Using the EPA to Capture Every Nth Edge

This program demonstrates the use of the EPA and PTS to perform the eighth rising edge capture capability of the HSIO. In fact any arbitrary number of rising edges, falling edges, or both can be chosen by adjusting the EPA capture mode and the number of PTS cycles executed before action is taken. This program works by capturing all rising edges on EPA2. 7 rising edges are captured, each creating a PTS interrupt that simply copies EPA_TIME2 to the Zero Register to allow further interrupts. The eighth edge creates a normal interrupt that also reads EPA_TIME2 and resets further interrupts to be PTS interrupts. This algorithm is very flexible because the EPA capture mode can be changed to falling edges or rising and falling edges. Also by adjusting the number of PTS_COUNT2 up to every 255th edge can be captured for interrupt processing.

```
EXAMPLE MODULE MAIN, STACKSIZE(20)

$NOLIST
$INCLUDE (KR.INC)
$LIST

STK EQU 200H
PTS_CYCLES EQU 8

CSEG AT 2044H
DCW PTS_COUNT2

CSEG AT 2004H
DCW EPA2_ISR

RSEG AT 70H
PTS_COUNT2: DSB 1
PTS_CONTROL2: DSB 1
PTS_SRC2: DSW 1
PTS_DEST2: DSW 1

RSEG AT 76H
INCR: DSW 1

CSEG AT 2080H
DI
DPTS

LD SP, #STK
CLR #INT_PEND
CLR #INT_PEND1

LDB PTS_COUNT2, #PTS_CYCLES ; 8 PTS cycles then normal interrupt
LDB PTS_CONTROL2, #90H ; PTS Single Transfer Mode
LD PTS_SRC2, #EPA_TIME2 ; Source is EPA_TIME2
LD PTS_DEST2, #R0 ; Destination is R0

LDB WSR, #7EH ; 32 byte window on 1FC0H
LDB P1REG_2, #0FFH ; Turn off pull down
LDS P1IO_2, #0FFH ; EPA2 is input
LDB P1SSEL_2, #04H ; EPA2 is special function

LDB WSR, #7BH ; 32 byte window on 1F60H
LD EPA_CONTROL2_2, #0FE20H ; TIMER1, capture positive edges
```

270968-11
7.2 Using the EPA to Capture Every Nth Edge (Continued)

LDB NSR,#7CH ; 32 byte window on 1F80H
LD TIMER1_CONTROL_2,#0C2H ; Enable TIMER1, 1us period
LD PTS_SELECT,#0004H ; Select PTS interrupt for EPA2
LDB INT_MASK,#04H ; Enable EPA2 interrupts
EPTS
EI

SELF:
SJMP SELF ; Let EPA take over

EPA2_ISR:
PUSH A
JRS PTS_SELECT,1,ERROR ; Check for bug
LD R0,#EPA_TIME2 ; Read EPA TIME2 to allow new interrupts
LDB PTS_COUNT2,#PTS_CYCLES ; Reset PTS for next 8 rising edges
OR PTS_SELECT,#0004H ; Return to PTS cycles
POPA R0
RET
ERROR:
ORB INT_PEND,#04H
POPA R0
RET

END
7.3 Using the HSIO for an Eight Entry FIFO Unit

The HSIO is set up to perform the functionality of a FIFO in hardware as part of its normal operation. One use of this structure is to capture up to eight events on one pin at a time for interrupt processing, as in this example.

```
EXAMPLE MODULE MAIN, STACKSIZE(20)

$Nolist
$Include (KC.INC)
$List

Stk EQU 200H

Cseg AT 2004H
Dcw HS12_ISR

Cseg AT 2080H

Di
Ld SP,#STK ; Initialize Stack Pointer
Clrb INT_PEND ; Clear all interrupts
Clrb INT_PEND1 ;
Ldb IOC0,#40H ; Enable HSI.3
Ldb IOC1,#80H ; HSI interrupt on FIFO full
Ldb IOC2,#80H ; Clear entire CAM
Ldb HSI_MODE,#40H ; HSI.3 captures rising edges
Ldb INT_MASK,#04H ; Enable HSI Data available interrupt
Ei

Self:
    Jmp  Self

HS12_ISR:
    Pushf
    Orb IOS1_SAVE,IOS1 ; Clear FIFO by reading out all
    Jbc IOS1_SAVE,7,FIFO_EMPTY ; entries
    Andb IOS1_SAVE,#7FH
    Ld  R0,#HSI_TIME
    Jmp  HS12_ISR

FIFO_EMPTY:
    Andb IOS1_SAVE,#7FH
    Popp
    Ret

End
```
7.4 Using the EPA for an Eight Entry FIFO Unit

This program demonstrates the use of the EPA and PTS to perform the eight entry FIFO function of the HSIO. In fact any arbitrary number of buffered events can be captured up to 256, the maximum number of PTS cycles without interrupt. The events occurring on one channel are stored separately from the events on another channel thus removing the need for decode to discover where the event originated. The software FIFO is created by the PTS cycles where the EPA_TIME3 value is copied to memory and then the destination address is incremented. After the desired number of events have been captured, eight here, a normal interrupt occurs allowing processing of the information just obtained. This example only utilizes one channel, EPA3, and therefore only creates one software FIFO.

```assembly
EXAMPLE MODULE MAIN, STACKSIZE(20)

$NLIST
$INCLUDE(KR.INC)
$LIST

STK EQU 200H
PTS_CYCLES EQU 8

CSEG AT 2042H
DCW PTS_COUNT3

CSEG AT 2002H
DCW EPA3_ISR

RSEG AT 70H
PTS_COUNT3: DSB 1
PTS_CONTROL3: DSB 1
PTS_SRC3: DSW 1
PTS_DEST3: DSW 1

RSEG AT 76H
CLRREG: DSW 1

RSEG AT 80H
FIFO: DSW 8

CSEG AT 2080H
DI
DPTS

LD SP,#STK ; Initialize Stack Pointer
CLR8 INT_PEND ; Clear all pending interrupts
CLR8 INT_PENDI

LD CLRREG,#FIFO

CLEAR:
ST R0,[CLRREG]+ ; Routine to clear register area for FIFO

CMP CLRREG,#00A0H
BNE CLEAR
```
7.4 Using the EPA for an Eight Entry FIFO Unit (Continued)

```
LDB PTS_COUNT3,#PTS_CYCLES ; 8 PTS cycles then normal interrupt
LDB PTS_CONTROL3,#85H     ; Single xsfer, incr and update dest
LD PTS_SRC3,#EPA_TIME3     ; Source is EPA_TIME3
LD PTS_DEST3,#FIFO        ; Destination is FIFO area

LDB WSR,#7EH               ; 32 byte window on 1FC0H
LDB P1REG_2,#0FFH          ; Turn off pull down
LDB P1IO_2,#0FFH           ; P1.3 is input
LDB P1SEL_2,#08H           ; P1.3 is EPA3

LDB WSR,#7BH               ; 32 byte window on 1F60H
LD EPA_CONTROL3_2,#0FE20H  ; TIMER1, capture positive edges

LDB WSR,#7CH               ; 32 byte window on 1F80H
LD TIMER1_CONTROL_2,#0C6H  ; Enable TIMER1, 16us period

LD PTS_SELECT,#0002H       ; Select PTS interrupt for EPA3
ORB INT_MASK,#02H          ; Enable EPA3 interrupts
EPTS EI
SELF:
  SJMP SELF                  ; Let EPA take over

EPA3_ISR:
  PUSHA
  JBS PTS_SELECT,1,ERROR     ; Check for bug
  LDB PTS_COUNT3,#PTS_CYCLES ; Reset PTS for next 8 rising edges
  LDB PTS_DEST3,#FIFO       ; Put dest back at beginning of FIFO
  ORB PTS_SELECT,#0002H     ; Return to PTS cycles
  POPA
  RET
ERROR:
  ORB INT_END,#02H
  POPA
  RET
END
```
7.5 Using the EPA to Perform Multiple Output Events in Succession

This program demonstrates the use of the EPA and PTS to perform the eight entry CAM of the HSIO for the scheduling of multiple events on one output channel. In fact any arbitrary number of buffered events can be set up to occur limited only by the speed of the PTS to set up the next pending event and the maximum number of PTS cycles without software intervention (256).

```
EXAMPLE MODULE MODE, STACKSIZE(20)
$NOLIST
$INCLUDE(KR.INC)
$LIST

PTS_CYCLES EQU 8
CSEG AT 2042H
   DCW PTS_COUNT3
CSEG AT 2002H
   DCW EPA3_ISR
RSEG AT 070H
   PTS_COUNT3: DSB 1
   PTS_CONTROL3: DSB 1
   PTS_SRC3: DSW 1
   PTS_DEST3: DSW 1
RSEG AT 076H
   TEMPREG: DSW 1
RSEG AT 080H
   CAM0: DSW 1
   CAM1: DSW 1
   CAM2: DSW 1
   CAM3: DSW 1
   CAM4: DSW 1
   CAM5: DSW 1
   CAM6: DSW 1
CSEG AT 2080H
   DI
   DPS
   LD TEMPREG,#CAM0
CLEAR:
   ST R0,[TEMPREG]+
   CMP TEMPREG,#00A0H
   BNE CLEAR
   LD TEMPREG,#CAM0
```

; Clearing register area for CAM
7.5 Using the EPA to Perform Multiple Output Events in Succession (Continued)

LOAD:  ; Loading "CAM" for many timed events
LD   CAM0,#0070H
LD   CAM1,#00C6H
LD   CAM2,#00D1H
LD   CAM3,#0113H
LD   CAM4,#0145H
LD   CAM5,#0168H
LD   CAM6,#0172H

LDB   PTS_COUNT3,#PTS_CYCLES  ; 8 PTS cycles then normal interrupt
LDB   PTS_CONTROL3,#0AH       ; Single transfer mode, incr and update src
LD   PTS_SRC3,#CAM0           ; Source is CAM
LD   PTS_DEST3,#EPA_TIME3     ; Destination is FIFO area

LDB   WSR,#7EH                 ; 32 byte window on 1FC0H
LDB   PIRO_2,#0F7H             ; P1.3 begins low
LDB   PISO_2,#0F7H             ; P1.3 is output
LDB   PISSEL_2,#08H           ; P1.3 is EPA3

LDB   WSR,#7BH                 ; 32 byte window on 1F60H
LD   EPA_CONTROL3_2,#0FE70H    ; TIMER1, toggle output
LD   EPA_TIME3_2,#0032H        ; Load EPA_TIME3

LDB   WSR,#7CH                 ; 32 byte window on 1F80H
LD   COMP_CONTROL0_2,#0FE49H   ; TIMER1, lock time entry, reset timer
LD   COMP_TIME0_2,#0172H       ; Reset timer after 370 counts

LD   TIMER1_2,#0FFC0H          ; Set TIMER1 to count up to 0000H
LD   TIMER1_CONTROL_2,#0C6H    ; Enable TIMER1, 16us period

LD   PTS_SELECT,#0002H         ; Select PTS interrupt for EPA3
ORB   INT_MASK,#02H            ; Enable EPA3 interrupts

EPTS
EI

SELF:  ; Let EPA take over
SJMP   SELF

EPA3_ISR:
PUSH   A
JSR   PTS_SELECT,1,ERROR       ; Check for bug
LDB   PTS_COUNT3,#PTS_CYCLES   ; Reset PTS for next 8 rising edges
LDB   PTS_SRC3,#CAM0           ; Put dest back at beginning of FIFO
LDB   WSR,#7BH                 ; 32 byte window on 1F60H
LD   EPA_TIME3_2,#0032H        ; Reset first event time
OR   PTS_SELECT,#0002H         ; Return to PTS cycles
POPA
RET

ERROR:
ORB   INT_PEND,#02H            ; Send to PTS cycle
POPA
RET

END
7.6 Using EPA2 to Clock TIMER2 Internally

The KR device has some interesting characteristics due to the fact that the T2CLK and T2DIR pins are shared with EPA pins. Because of the way the output drivers are set up it is possible to configure an EPA channel as an output and drive one of these two clock pins with no additional hardware. This example uses EPA0 to drive T2CLK and hence clock TIMER2 "externally". Note that one can then base an EPA output event on TIMER2. Interestingly, if TIMER1 counts at its slowest rate, 16 μs period, the TIMER2 is clocked at its slowest rate, once every 64 edges, it is possible to schedule an event to occur approximately 50 days in the future.

```
EXAMPLE    MODULE MAIN, STACKSIZE(20)
$NOLIST
$INCLUDE (KR.INC)
$LIST
STK       EQU   200H
CSEG AT 2080H
    LD    SP,#STK
    CLR8# INT_PEND
    CLR8# INT_PEND1
    LDB# WSR,#1FH       ; 128 byte window on 1F80H
    LDB# P1REG_0,#0FEH ; Turn off pull down on P1.0
    LDB# P1IO_0,#0FEH ; P1.0 is output
    LDB# P1SEL_0,#01H  ; P1.0 is T2CLK
    LDB# WSR,#1EH       ; 128 byte window on 1F00H
    LD    EPA_CONTROL3_0,#0FE78H ; TIMER1, toggle output, lock time
    LD    EPA_TIME3_0,#500  ; Time tag for toggle
    LDB# WSR,#3EH
    LDB# TIMER1_CONTROL_1,#0C6H ; Enable timer, count up, 16μs period
    LDB# TIMER2_CONTROL_1,#0CEH ; Enable timer, external clock
SELF:
    SJMP SELF          ; Let EPA take over
END
```

270968-18
APPENDIX A

;******************************************************************************
; SFRS.KR - DEFINITION OF SYMBOLIC NAMES FOR THE I/O REGISTERS
; OF THE 8XC156KR
; (C) INTEL CORPORATION 1989
;******************************************************************************

RO EQU 00H:WORD ; R ZERO REGISTER
ZERO EQU 00H:WORD ; R ZERO REGISTER
PTS_SELECT EQU 04H:WORD ; R/W
PTS_SRW EQU 06H:WORD ; R/W
INT_MASK EQU 08H:BYTE ; R/W
INT_PEND EQU 09H:BYTE ; R/W
WATCHDOG EQU 0AH:BYTE ; W WATCHDOG TIMER
INTPEND1 EQU 12H:BYTE ; R/W
INT_MASK1 EQU 13H:BYTE ; R/W
MSR EQU 14H:BYTE ; R/W
SP EQU 16H:WORD ; R/W

******************************************************************************

;******************************************************************************
; SFR DEFINITIONS FOR REGISTERS OUTSIDE 00 TO FFH ALONG WITH ALL
; WINDOW DEFINITIONS FOR THE SAME REGISTER.
;******************************************************************************

;LABEL: FOR USE WITH WSR VALUE:
\\---------------------

POPIN EQU 01FD6H:BYTE ; R/W ---
POPIN_0 EQU 0D6H:BYTE ; R/W 1FH
POPIN_1 EQU 0D7H:BYTE ; R/W 3FH
POPIN_2 EQU 0FAH:BYTE ; R/W 7EH

PIPIN EQU 01FD6H:BYTE ; R ---
PIPIN_0 EQU 0D6H:BYTE ; R 1FH
PIPIN_1 EQU 0D7H:BYTE ; R 3FH
PIPIN_2 EQU 0FAH:BYTE ; R 7EH

P1REG EQU 01FD4H:BYTE ; R/W ---
P1REG_0 EQU 0D4H:BYTE ; R/W 1FH
P1REG_1 EQU 0D5H:BYTE ; R/W 3FH
P1REG_2 EQU 0D6H:BYTE ; R/W 7EH

P2IO EQU 01FD2H:BYTE ; R/W ---
P2IO_0 EQU 0D2H:BYTE ; R/W 1FH
P2IO_1 EQU 0D3H:BYTE ; R/W 3FH
P2IO_2 EQU 0D4H:BYTE ; R/W 7EH

P1SEL EQU 01FD0H:BYTE ; R/W ---
P1SEL_0 EQU 0D0H:BYTE ; R/W 1FH
P1SEL_1 EQU 0D1H:BYTE ; R/W 3FH
P1SEL_2 EQU 0D2H:BYTE ; R/W 7EH

P2PIN EQU 01FCFH:BYTE ; R ---
P2PIN_0 EQU 0CFH:BYTE ; R 1FH
P2PIN_1 EQU 0CHF:BYTE ; R 3FH
P2PIN_2 EQU 0FH:BYTE ; R 7EH

P2REG EQU 01FCDH:BYTE ; R/W ---
P2REG_0 EQU 0CDH:BYTE ; R/W 1FH
P2REG_1 EQU 0CEH:BYTE ; R/W 3FH
P2REG_2 EQU 0DFH:BYTE ; R/W 7EH
P2IO EQU 01FCFH:BYTE ; R ---

270968-19
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Equivalent</th>
<th>Mode</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2IO_0</td>
<td>EQU 0CBH:BYTE</td>
<td>R/W</td>
<td>1FH</td>
</tr>
<tr>
<td>P2IO_1</td>
<td>EQU 0CBH:BYTE</td>
<td>R/W</td>
<td>3FH</td>
</tr>
<tr>
<td>P2IO_2</td>
<td>EQU 0EBH:BYTE</td>
<td>R/W</td>
<td>7EH</td>
</tr>
<tr>
<td>P2SEL_0</td>
<td>EQU 0C9H:BYTE</td>
<td>R/W</td>
<td>1FH</td>
</tr>
<tr>
<td>P2SEL_1</td>
<td>EQU 0C9H:BYTE</td>
<td>R/W</td>
<td>3FH</td>
</tr>
<tr>
<td>P2SEL_2</td>
<td>EQU 0E9H:BYTE</td>
<td>R/W</td>
<td>7EH</td>
</tr>
<tr>
<td>P3PIN</td>
<td>EQU 01FFEH:BYTE</td>
<td>R</td>
<td>---</td>
</tr>
<tr>
<td>P3PIN_0</td>
<td>EQU 0FFEH:BYTE</td>
<td>R</td>
<td>1FH</td>
</tr>
<tr>
<td>P3PIN_1</td>
<td>EQU 0FFEH:BYTE</td>
<td>R</td>
<td>3FH</td>
</tr>
<tr>
<td>P3PIN_2</td>
<td>EQU 0FFEH:BYTE</td>
<td>R</td>
<td>7FH</td>
</tr>
<tr>
<td>P3REG</td>
<td>EQU 01FFEH:BYTE</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>P3REG_0</td>
<td>EQU 0FFCH:BYTE</td>
<td>R/W</td>
<td>1FH</td>
</tr>
<tr>
<td>P3REG_1</td>
<td>EQU 0FFCH:BYTE</td>
<td>R/W</td>
<td>3FH</td>
</tr>
<tr>
<td>P3REG_2</td>
<td>EQU 0FFCH:BYTE</td>
<td>R/W</td>
<td>7FH</td>
</tr>
<tr>
<td>P4PIN</td>
<td>EQU 01FFFH:BYTE</td>
<td>R</td>
<td>---</td>
</tr>
<tr>
<td>P4PIN_0</td>
<td>EQU 0FFEH:BYTE</td>
<td>R</td>
<td>1FH</td>
</tr>
<tr>
<td>P4PIN_1</td>
<td>EQU 0FFEH:BYTE</td>
<td>R</td>
<td>3FH</td>
</tr>
<tr>
<td>P4PIN_2</td>
<td>EQU 0FFEH:BYTE</td>
<td>R</td>
<td>7FH</td>
</tr>
<tr>
<td>P4REG</td>
<td>EQU 01FFDH:BYTE</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>P4REG_0</td>
<td>EQU 0FFDH:BYTE</td>
<td>R/W</td>
<td>1FH</td>
</tr>
<tr>
<td>P4REG_1</td>
<td>EQU 0FFDH:BYTE</td>
<td>R/W</td>
<td>3FH</td>
</tr>
<tr>
<td>P4REG_2</td>
<td>EQU 0FFDH:BYTE</td>
<td>R/W</td>
<td>7FH</td>
</tr>
<tr>
<td>P5PIN</td>
<td>EQU 01FF7H:BYTE</td>
<td>R</td>
<td>---</td>
</tr>
<tr>
<td>P5PIN_0</td>
<td>EQU 0FF7H:BYTE</td>
<td>R</td>
<td>1FH</td>
</tr>
<tr>
<td>P5PIN_1</td>
<td>EQU 0FF7H:BYTE</td>
<td>R</td>
<td>3FH</td>
</tr>
<tr>
<td>P5PIN_2</td>
<td>EQU 0FF7H:BYTE</td>
<td>R</td>
<td>7FH</td>
</tr>
<tr>
<td>P5REG</td>
<td>EQU 01FF8H:BYTE</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>P5REG_0</td>
<td>EQU 0FF8H:BYTE</td>
<td>R/W</td>
<td>1FH</td>
</tr>
<tr>
<td>P5REG_1</td>
<td>EQU 0FF8H:BYTE</td>
<td>R/W</td>
<td>3FH</td>
</tr>
<tr>
<td>P5REG_2</td>
<td>EQU 0FF8H:BYTE</td>
<td>R/W</td>
<td>7FH</td>
</tr>
<tr>
<td>P5IO</td>
<td>EQU 01FF9H:BYTE</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>P5IO_0</td>
<td>EQU 0FF9H:BYTE</td>
<td>R/W</td>
<td>1FH</td>
</tr>
<tr>
<td>P5IO_1</td>
<td>EQU 0FF9H:BYTE</td>
<td>R/W</td>
<td>3FH</td>
</tr>
<tr>
<td>P5IO_2</td>
<td>EQU 0FF9H:BYTE</td>
<td>R/W</td>
<td>7FH</td>
</tr>
<tr>
<td>P5SEL</td>
<td>EQU 01FFAH:BYTE</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>P5SEL_0</td>
<td>EQU 0FFAH:BYTE</td>
<td>R/W</td>
<td>1FH</td>
</tr>
<tr>
<td>P5SEL_1</td>
<td>EQU 0FFAH:BYTE</td>
<td>R/W</td>
<td>3FH</td>
</tr>
<tr>
<td>P5SEL_2</td>
<td>EQU 0FFAH:BYTE</td>
<td>R/W</td>
<td>7FH</td>
</tr>
<tr>
<td>P6PIN</td>
<td>EQU 01FD7H:BYTE</td>
<td>R</td>
<td>---</td>
</tr>
<tr>
<td>P6PIN_0</td>
<td>EQU 0D7H:BYTE</td>
<td>R</td>
<td>1FH</td>
</tr>
<tr>
<td>P6PIN_1</td>
<td>EQU 0D7H:BYTE</td>
<td>R</td>
<td>3FH</td>
</tr>
<tr>
<td>P6PIN_2</td>
<td>EQU 0D7H:BYTE</td>
<td>R</td>
<td>7FH</td>
</tr>
<tr>
<td>P6REG</td>
<td>EQU 01FD8H:BYTE</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>P6REG_0</td>
<td>EQU 0D8H:BYTE</td>
<td>R/W</td>
<td>1FH</td>
</tr>
<tr>
<td>P6REG_1</td>
<td>EQU 0D8H:BYTE</td>
<td>R/W</td>
<td>3FH</td>
</tr>
<tr>
<td>P6REG_2</td>
<td>EQU 0D8H:BYTE</td>
<td>R/W</td>
<td>7EH</td>
</tr>
<tr>
<td>P6IO</td>
<td>EQU 01FD9H:BYTE</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>P6IO_0</td>
<td>EQU 0D9H:BYTE</td>
<td>R/W</td>
<td>1FH</td>
</tr>
<tr>
<td>P6IO_1</td>
<td>EQU 0D9H:BYTE</td>
<td>R/W</td>
<td>3FH</td>
</tr>
<tr>
<td>P6IO_2</td>
<td>EQU 0D9H:BYTE</td>
<td>R/W</td>
<td>7EH</td>
</tr>
<tr>
<td>P6SEL</td>
<td>EQU 01FD1H:BYTE</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>P6SEL_0</td>
<td>EQU 0D1H:BYTE</td>
<td>R/W</td>
<td>1FH</td>
</tr>
<tr>
<td>P6SEL_1</td>
<td>EQU 0D1H:BYTE</td>
<td>R/W</td>
<td>3FH</td>
</tr>
</tbody>
</table>
P6SSEL_2 EQU 0F1H:BYTE ; R/W 7EH

TIMER1 EQU 01F9AH:WORD ; R/W ---
TIMER1_0 EQU 09AH:WORD ; R/W 1FH
TIMER1_1 EQU 0DAH:WORD ; R/W 3EH
TIMER1_2 EQU 0FAH:WORD ; R/W 7CH
TIMER1_CONTROL EQU 01F9BH:BYTE ; R/W ---
TIMER1_CONTROL_0 EQU 09BH:BYTE ; R/W 1FH
TIMER1_CONTROL_1 EQU 0DBH:BYTE ; R/W 3EH
TIMER1_CONTROL_2 EQU 0FBH:BYTE ; R/W 7CH

TIMER2 EQU 01F9CH:WORD ; R/W ---
TIMER2_0 EQU 0E9H:WORD ; R/W 1FH
TIMER2_1 EQU 0DEH:WORD ; R/W 3EH
TIMER2_2 EQU 0FEH:WORD ; R/W 7CH
TIMER2_CONTROL EQU 01F9CH:BYTE ; R/W ---
TIMER2_CONTROL_0 EQU 09CH:BYTE ; R/W 1FH
TIMER2_CONTROL_1 EQU 0DCCH:BYTE ; R/W 3EH
TIMER2_CONTROL_2 EQU 0FCCH:BYTE ; R/W 7CH

SP_BAUD EQU 01FBCH:WORD ; W ---
SP_BAUD_0 EQU 0BCH:WORD ; W 1FH
SP_BAUD_1 EQU 0FCCH:WORD ; W 3EH
SP_BAUD_2 EQU 0FCH:WORD ; W 7DH
SP_CONTROL EQU 01FBBH:BYTE ; R/W ---
SP_CONTROL_0 EQU 0BBH:BYTE ; R/W 1FH
SP_CONTROL_1 EQU 0FBH:BYTE ; R/W 3EH
SP_CONTROL_2 EQU 0FCH:BYTE ; R/W 7DH
SP_STATUS EQU 01FBB9H:BYTE ; R/W ---
SP_STATUS_0 EQU 0B9H:BYTE ; R/W 1FH
SP_STATUS_1 EQU 0F9H:BYTE ; R/W 3EH
SP_STATUS_2 EQU 0F9BH:BYTE ; R/W 7DH
SUBF_TX EQU 01FBAAH:BYTE ; R/W ---
SUBF_TX_0 EQU 0BAH:BYTE ; R/W 1FH
SUBF_TX_1 EQU 0FAH:BYTE ; R/W 3EH
SUBF_TX_2 EQU 0FACH:BYTE ; R/W 7DH
SUBF_RX EQU 01FBA9H:BYTE ; R/W ---
SUBF_RX_0 EQU 0BA9H:BYTE ; R/W 1FH
SUBF_RX_1 EQU 0FACH:BYTE ; R/W 3EH
SUBF_RX_2 EQU 0F8H:BYTE ; R/W 7DH

EPAIPv EQU 01FA8H:BYTE ; R ---
EPAIPv_0 EQU 0A8H:BYTE ; R 1FH
EPAIPv_1 EQU 0E8H:BYTE ; R 3EH
EPAIPv_2 EQU 0E9H:BYTE ; R 7DH
EPA_PEND EQU 01FA2H:WORD ; R/W ---
EPA_PEND_0 EQU 0A2H:WORD ; R/W 1FH
EPA_PEND_1 EQU 0E2H:WORD ; R/W 3EH
EPA_PEND_2 EQU 0E2H:WORD ; R/W 7DH
EPA_PEND1 EQU 01FA6H:BYTE ; R/W ---
EPA_PEND1_0 EQU 0A6H:BYTE ; R/W 1FH
EPA_PEND1_1 EQU 0E6H:BYTE ; R/W 3EH
EPA_PEND1_2 EQU 0E6H:BYTE ; R/W 7DH

EPA_MASK EQU 01FA0H:WORD ; R ---
EPA_MASK_0 EQU 0A0H:WORD ; R/W 1FH
EPA_MASK_1 EQU 0E0H:WORD ; R/W 3EH
EPA_MASK_2 EQU 0E0H:WORD ; R/W 7DH
EPA_MASK1 EQU 01FA4H:WORD ; R/W --- BUG...must write as word
EPA_MASK1_0 EQU 0A4H:WORD ; R/W 1FH

270968-21
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPA_MASK1_1</td>
<td>EQU 0E4H:WORD; R/W 3EH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EPA_MASK1_2</td>
<td>EQU 0E4H:WORD; R/W 7DH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USFR</td>
<td>EQU 0FF6H:BYTE; W ---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USFR_0</td>
<td>EQU 0F6H:BYTE; W 1FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USFR_1</td>
<td>EQU 0F6H:BYTE; W 3FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USFR_2</td>
<td>EQU 0F6H:BYTE; W 7FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLP_CMD</td>
<td>EQU 01FFAH:BYTE; R/W ---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLP_CMD_0</td>
<td>EQU 0FAH:BYTE; R/W 1FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLP_CMD_1</td>
<td>EQU 0FAH:BYTE; R/W 3FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLP_CMD_2</td>
<td>EQU 0FAH:BYTE; R/W 7FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLP_STAT</td>
<td>EQU 01FF6H:BYTE; R/W ---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLP_STAT_0</td>
<td>EQU 0F6H:BYTE; R/W 1FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLP_STAT_1</td>
<td>EQU 0F6H:BYTE; R/W 3FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLP_STAT_2</td>
<td>EQU 0F6H:BYTE; R/W 7FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLP_FUNREG</td>
<td>EQU 01FFBH:BYTE; R/W ---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLP_FUNREG_0</td>
<td>EQU 0FBH:BYTE; R/W 1FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLP_FUNREG_1</td>
<td>EQU 0FBH:BYTE; R/W 3FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLP_FUNREG_2</td>
<td>EQU 0FBH:BYTE; R/W 7FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD_TIME</td>
<td>EQU 01FACH:BYTE; R/W ---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD_TIME_0</td>
<td>EQU 0AACH:BYTE; R/W 1FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD_TIME_1</td>
<td>EQU 0EACH:BYTE; R/W 3EH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD_TIME_2</td>
<td>EQU 0EACH:BYTE; R/W 7DH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD_TEST</td>
<td>EQU 01FACH:BYTE; R/W ---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD_TEST_0</td>
<td>EQU 0AACH:BYTE; R/W 1FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD_TEST_1</td>
<td>EQU 0EACH:BYTE; R/W 3EH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD_COMMAND</td>
<td>EQU 01FACH:BYTE; R/W ---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD_COMMAND_0</td>
<td>EQU 0AACH:BYTE; R/W 1FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD_COMMAND_1</td>
<td>EQU 0EACH:BYTE; R/W 3EH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD_COMMAND_2</td>
<td>EQU 0EACH:BYTE; R/W 7DH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD_RESULT</td>
<td>EQU 01FAAH:WORD; R/W ---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD_RESULT_0</td>
<td>EQU 0AAGH:WORD; R/W 1FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD_RESULT_1</td>
<td>EQU 0EAGH:WORD; R/W 3EH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD_RESULT_2</td>
<td>EQU 0EAGH:WORD; R/W 7DH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSIO_BAUD</td>
<td>EQU 01FB4H:BYTE; W ---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSIO_BAUD_0</td>
<td>EQU 0B4H:BYTE; W 1FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSIO_BAUD_1</td>
<td>EQU 0B4H:BYTE; W 3EH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSIO_BAUD_2</td>
<td>EQU 0B4H:BYTE; W 7DH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSIO_STCR1</td>
<td>EQU 01FB3H:BYTE; R/W ---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSIO_STCR1_0</td>
<td>EQU 0B3H:BYTE; R/W 1FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSIO_STCR1_1</td>
<td>EQU 0F3H:BYTE; R/W 3EH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSIO_STCR1_2</td>
<td>EQU 0F3H:BYTE; R/W 7DH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSIO_STB1</td>
<td>EQU 01FB2H:BYTE; R/W ---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSIO_STB1_0</td>
<td>EQU 0B2H:BYTE; R/W 1FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSIO_STB1_1</td>
<td>EQU 0F2H:BYTE; R/W 3EH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSIO_STB1_2</td>
<td>EQU 0F2H:BYTE; R/W 7DH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSIO_STCRO</td>
<td>EQU 01FB1H:BYTE; R/W ---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSIO_STCRO_0</td>
<td>EQU 0B1H:BYTE; R/W 1FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSIO_STCRO_1</td>
<td>EQU 0F1H:BYTE; R/W 3EH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSIO_STCRO_2</td>
<td>EQU 0F1H:BYTE; R/W 7DH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSIO_STB0</td>
<td>EQU 01FB0H:BYTE; R/W ---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSIO_STB0_0</td>
<td>EQU 0B0H:BYTE; R/W 1FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSIO_STB0_1</td>
<td>EQU 0F0H:BYTE; R/W 3EH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSIO_STB0_2</td>
<td>EQU 0F0H:BYTE; R/W 7DH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td>Address</td>
<td>Read</td>
<td>Write</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------</td>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>COMP_TIME1</td>
<td>01F8EH</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>COMP_TIME1_0</td>
<td>08EH</td>
<td>R/W</td>
<td>1FH</td>
</tr>
<tr>
<td>COMP_TIME1_1</td>
<td>0CEH</td>
<td>R/W</td>
<td>3EH</td>
</tr>
<tr>
<td>COMP_TIME1_2</td>
<td>0EEH</td>
<td>R/W</td>
<td>7CH</td>
</tr>
<tr>
<td>COMP_CONTROL1</td>
<td>01F8CH</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>COMP_CONTROL1_0</td>
<td>08CH</td>
<td>R/W</td>
<td>1FH</td>
</tr>
<tr>
<td>COMP_CONTROL1_1</td>
<td>0CECH</td>
<td>R/W</td>
<td>3EH</td>
</tr>
<tr>
<td>COMP_CONTROL1_2</td>
<td>0EECH</td>
<td>R/W</td>
<td>7CH</td>
</tr>
<tr>
<td>COMP_TIME0</td>
<td>01F8AH</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>COMP_TIME0_0</td>
<td>08AH</td>
<td>R/W</td>
<td>1FH</td>
</tr>
<tr>
<td>COMP_TIME0_1</td>
<td>0CEAH</td>
<td>R/W</td>
<td>3EH</td>
</tr>
<tr>
<td>COMP_TIME0_2</td>
<td>0EAEH</td>
<td>R/W</td>
<td>7CH</td>
</tr>
<tr>
<td>COMP_CONTROL0</td>
<td>01F8BH</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>COMP_CONTROL0_0</td>
<td>08BH</td>
<td>R/W</td>
<td>1FH</td>
</tr>
<tr>
<td>COMP_CONTROL0_1</td>
<td>0CEBH</td>
<td>R/W</td>
<td>3EH</td>
</tr>
<tr>
<td>COMP_CONTROL0_2</td>
<td>0EAEH</td>
<td>R/W</td>
<td>7CH</td>
</tr>
<tr>
<td>EPA_TIME9</td>
<td>01F86H</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>EPA_TIME9_0</td>
<td>086H</td>
<td>R/W</td>
<td>1FH</td>
</tr>
<tr>
<td>EPA_TIME9_1</td>
<td>0C6H</td>
<td>R/W</td>
<td>3EH</td>
</tr>
<tr>
<td>EPA_TIME9_2</td>
<td>0E6H</td>
<td>R/W</td>
<td>7CH</td>
</tr>
<tr>
<td>EPA_CONTROL9</td>
<td>01F84H</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>EPA_CONTROL9_0</td>
<td>084H</td>
<td>R/W</td>
<td>1FH</td>
</tr>
<tr>
<td>EPA_CONTROL9_1</td>
<td>0C4H</td>
<td>R/W</td>
<td>3EH</td>
</tr>
<tr>
<td>EPA_CONTROL9_2</td>
<td>0E4H</td>
<td>R/W</td>
<td>7CH</td>
</tr>
<tr>
<td>EPA_TIME8</td>
<td>01F82H</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>EPA_TIME8_0</td>
<td>082H</td>
<td>R/W</td>
<td>1FH</td>
</tr>
<tr>
<td>EPA_TIME8_1</td>
<td>0C2H</td>
<td>R/W</td>
<td>3EH</td>
</tr>
<tr>
<td>EPA_TIME8_2</td>
<td>0E2H</td>
<td>R/W</td>
<td>7CH</td>
</tr>
<tr>
<td>EPA_CONTROL8</td>
<td>01F80H</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>EPA_CONTROL8_0</td>
<td>080H</td>
<td>R/W</td>
<td>1FH</td>
</tr>
<tr>
<td>EPA_CONTROL8_1</td>
<td>0C0H</td>
<td>R/W</td>
<td>3EH</td>
</tr>
<tr>
<td>EPA_CONTROL8_2</td>
<td>0E0H</td>
<td>R/W</td>
<td>7CH</td>
</tr>
<tr>
<td>EPA_TIME7</td>
<td>01F7EH</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>EPA_TIME7_0</td>
<td>0F6H</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_TIME7_1</td>
<td>0FCH</td>
<td>R/W</td>
<td>3DH</td>
</tr>
<tr>
<td>EPA_TIME7_2</td>
<td>0FEH</td>
<td>R/W</td>
<td>7BH</td>
</tr>
<tr>
<td>EPA_CONTROL7</td>
<td>01F7CH</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>EPA_CONTROL7_0</td>
<td>0F0H</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_CONTROL7_1</td>
<td>0FCH</td>
<td>R/W</td>
<td>3DH</td>
</tr>
<tr>
<td>EPA_CONTROL7_2</td>
<td>0FEH</td>
<td>R/W</td>
<td>7BH</td>
</tr>
<tr>
<td>EPA_TIME6</td>
<td>01F7AH</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>EPA_TIME6_0</td>
<td>0FAH</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_TIME6_1</td>
<td>0FAH</td>
<td>R/W</td>
<td>3DH</td>
</tr>
<tr>
<td>EPA_TIME6_2</td>
<td>0FAH</td>
<td>R/W</td>
<td>7BH</td>
</tr>
<tr>
<td>EPA_CONTROL6</td>
<td>01F78H</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>EPA_CONTROL6_0</td>
<td>0F8H</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_CONTROL6_1</td>
<td>0F8H</td>
<td>R/W</td>
<td>3DH</td>
</tr>
<tr>
<td>EPA_CONTROL6_2</td>
<td>0F8H</td>
<td>R/W</td>
<td>7BH</td>
</tr>
<tr>
<td>EPA_TIME5</td>
<td>01F76H</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>EPA_TIME5_0</td>
<td>0F6H</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_TIME5_1</td>
<td>0F6H</td>
<td>R/W</td>
<td>3DH</td>
</tr>
<tr>
<td>EPA_TIME5_2</td>
<td>0F6H</td>
<td>R/W</td>
<td>7BH</td>
</tr>
<tr>
<td>EPA_CONTROL5</td>
<td>01F74H</td>
<td>R/W</td>
<td>---</td>
</tr>
<tr>
<td>EPA_CONTROL5_0</td>
<td>0F4H</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_CONTROL5_1</td>
<td>0F4H</td>
<td>R/W</td>
<td>3DH</td>
</tr>
<tr>
<td>EPA_CONTROL5_2</td>
<td>0F4H</td>
<td>R/W</td>
<td>7BH</td>
</tr>
<tr>
<td>Symbol</td>
<td>Value</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>EPA_TIME4</td>
<td>01F72H:WORD</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_TIME4_0</td>
<td>0F2H:WORD</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_TIME4_1</td>
<td>0F2H:WORD</td>
<td>R/W</td>
<td>3DH</td>
</tr>
<tr>
<td>EPA_TIME4_2</td>
<td>0F2H:WORD</td>
<td>R/W</td>
<td>7BH</td>
</tr>
<tr>
<td>EPA_CONTROL4</td>
<td>01F70H:WORD</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_CONTROL4_0</td>
<td>0F0H:WORD</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_CONTROL4_1</td>
<td>0F0H:WORD</td>
<td>R/W</td>
<td>3DH</td>
</tr>
<tr>
<td>EPA_CONTROL4_2</td>
<td>0F0H:WORD</td>
<td>R/W</td>
<td>7BH</td>
</tr>
<tr>
<td>EPA_TIME3</td>
<td>01F6EH:WORD</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_TIME3_0</td>
<td>0EEH:WORD</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_TIME3_1</td>
<td>0EEH:WORD</td>
<td>R/W</td>
<td>3DH</td>
</tr>
<tr>
<td>EPA_TIME3_2</td>
<td>0EEH:WORD</td>
<td>R/W</td>
<td>7BH</td>
</tr>
<tr>
<td>EPA_CONTROL3</td>
<td>01F6CH:WORD</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_CONTROL3_0</td>
<td>0ECH:WORD</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_CONTROL3_1</td>
<td>0ECH:WORD</td>
<td>R/W</td>
<td>3DH</td>
</tr>
<tr>
<td>EPA_CONTROL3_2</td>
<td>0ECH:WORD</td>
<td>R/W</td>
<td>7BH</td>
</tr>
<tr>
<td>EPA_TIME2</td>
<td>01F6AH:WORD</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_TIME2_0</td>
<td>0EAH:WORD</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_TIME2_1</td>
<td>0EAH:WORD</td>
<td>R/W</td>
<td>3DH</td>
</tr>
<tr>
<td>EPA_TIME2_2</td>
<td>0EAH:WORD</td>
<td>R/W</td>
<td>7BH</td>
</tr>
<tr>
<td>EPA_CONTROL2</td>
<td>01F68H:WORD</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_CONTROL2_0</td>
<td>0E8H:WORD</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_CONTROL2_1</td>
<td>0E8H:WORD</td>
<td>R/W</td>
<td>3DH</td>
</tr>
<tr>
<td>EPA_TIME1</td>
<td>01F66H:WORD</td>
<td>R/W</td>
<td>3DH</td>
</tr>
<tr>
<td>EPA_TIME1_0</td>
<td>0E6H:WORD</td>
<td>R/W</td>
<td>3DH</td>
</tr>
<tr>
<td>EPA_TIME1_1</td>
<td>0E6H:WORD</td>
<td>R/W</td>
<td>7BH</td>
</tr>
<tr>
<td>EPA_CONTROL1</td>
<td>01F64H:WORD</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_CONTROL1_0</td>
<td>0E4H:WORD</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_CONTROL1_1</td>
<td>0E4H:WORD</td>
<td>R/W</td>
<td>3DH</td>
</tr>
<tr>
<td>EPA_CONTROL1_2</td>
<td>0E4H:WORD</td>
<td>R/W</td>
<td>7BH</td>
</tr>
<tr>
<td>EPA_TIME0</td>
<td>01F62H:WORD</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_TIME0_0</td>
<td>0E2H:WORD</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_TIME0_1</td>
<td>0E2H:WORD</td>
<td>R/W</td>
<td>3DH</td>
</tr>
<tr>
<td>EPA_TIME0_2</td>
<td>0E2H:WORD</td>
<td>R/W</td>
<td>7BH</td>
</tr>
<tr>
<td>EPA_CONTROL0</td>
<td>01F60H:WORD</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_CONTROL0_0</td>
<td>0E0H:WORD</td>
<td>R/W</td>
<td>1EH</td>
</tr>
<tr>
<td>EPA_CONTROL0_1</td>
<td>0E0H:WORD</td>
<td>R/W</td>
<td>3DH</td>
</tr>
<tr>
<td>EPA_CONTROL0_2</td>
<td>0E0H:WORD</td>
<td>R/W</td>
<td>7BH</td>
</tr>
</tbody>
</table>