- **The 9S12 Pulse Accumultor**
- ECT_16B8C Block User Guide
- Huang, Sections 8.8

Pulse Accumulator on the HCS12

• A pulse accumulator counts the number of active edges at the input of its channel.
• The HCS12 has four 8-bit pulse accumulators, configurable as two 16- bit pulse accumulators. In the following we will discuss the 16-bit pulse accumulator PACA, made up of the two 8-bit pulse accumulators PACN3 and PACN2.

• To use the pulse accumulator connect an input to Port T7 (PT7).

• The pulse accumulator operates in two modes:
1. Event-Count Mode
2. Gated Time Accumulation Mode

• In Event-Count Mode, the pulse accumulator counts the number of rising or falling edges on Port T7
– You can set up the pulse accumulator to select which edge to count
– The counts are held in the 16-bit PACA register
– On each selected edge the PAIF flag of the PAFLG register is set
– When PACA overflows from 0xFFFF to 0x0000, the PAOVF flag of the PAFLG register is set

• In Gated Time Accumulation Mode the pulse accumulator counts clock cycles while in the input to Port T7 is high or low
– In Gated Time Accumulation Mode the pulse accumulator uses the Timer Clock. To use the pulse accumulator in Gated Time Accumulation Mode you must enable the Timer Clock by writing a 1 to the TEN bit of TSCR
– You can set up the pulse accumulator to count while PT7 is high or to count while PT7 is low
– The clock for the pulse accumulator is the bus clock divided by 64
– With an 24 MHz bus clock, the clock frequency of the pulse accumulator is 375 kHz, for a period of 2.67 μs
– For example, if the pulse accumulator is set up to count while Port T7 is high, and it counts 729 clock pulses, then the input to Port T7 was high for 729 x 2.67 μs = 1.94 ms

The Pulse Accumulator

• The pulse accumulator uses PT7 as an input
– To use the pulse accumulator make sure bit 7 of TIOS is 0 (otherwise PT7 used as output compare pin)
– To use the pulse accumulator make sure bits 6 and 7 of TCTL3 are 0 (otherwise timer function connected to PT7)

• The pulse accumulator uses three registers: PACTL, PAFLG, PACA

• To use the pulse accumulator you have to program the PACTL register

• The PAFLG register has flags to indicate the status of the pulse accumulator
– You clear a flag bit by writing a 1 to that bit

• The count value is stored in the 16-bit PACA register
– You may write a value to PACA
– Suppose you want an interrupt after 100 events on PT7
– Write -100 to PACA, and enable the PAOVI interrupt
– After 100 events on PT7, PACA will overflow, and a PAOVI interrupt will be generated

0	PAEN	PAMOD	PEDGE	CLK1	CLK0	PAOVI	PAI	PACTL 0x0060

0	0	0	0	0	0	PAOFV	PAIF	PAFLG 0x0061

PAEN: 1= Enable PA

PAMOD: 0=Event count mode
1=Gated time accumulator mode

PEDGE: 0=Falling edge (event) High enable (gated)
1=Rising edge (event) Low enable (gated)

PAI: 0=Interrupt disabled
1=Interrupt enabled

PAOVI: 1=Enable interrupt, when edge on PT7
If PEDGE=0, interrupt on falling edge
If PEDGE=1, interrupt on rising edge

The 16-bit PACA register is at address 0x0062

The Pulse Accumulator PACA

• Here is a C program which counts the number of rising edges on PT7:

```
#include "hcs12.h"
#include "DBug12.h"
#define PACA *(unsigned int *) 0x62; /* pulse accumulator*/

int start_count,end_count,total_count;

main()
{
int i;
TIOS = TIOS & ~0x80; /* PT7 input */
TCTL3 = TCTL3 & ~0xC0 /* Disconnect IC/OC logic from PT7 */
PACTL = 0x50;  /* 0 1 0 1 0 0 0 0 */
               /*   | | |     | | */
               /*   | | |     | \_ No interrupt on edge */
               /*   | | |     \___ No interrupt on OV */
               /*   | | \___________ Rising Edge */
               /*   | \_____________ Event Count Mode */
               /*   \_______________ Enable PACA (16 bit)*/
start_count = PACA;
for (i=0;i<10000;i++) ; /* Software Delay */
end_count = PACA;
total_count = end_count - start_count;
DB12FNP->printf("Total counts = %d\r\n",total_count);
}
```

The Pulse Accumulator PACA

• Here is a C program which determines how long the input on PA7 is high:

```
#include "hcs12.h"
#include "DBug12.h"
#define PACA *(unsigned int *) 0x62; /* pulse accumulator */

int start_count,end_count,total_count;

main()
{
int i;
TSCR = 0x80; /* Turn on timer clock */
TIOS = TIOS & ~0x80; /* PT7 input */
TCTL3 = TCTL3 & ~0xC0 /* Disconnect IC/OC logic from PT7 */
PACTL = 0x60;  /* 0 1 1 0 0 0 0 0 */
               /*   | | |     | | */
               /*   | | |     | \_ No interrupt on edge */
               /*   | | |     \___ No interrupt on OV */
               /*   | | \_____ Count while input high */
               /*   | \_______ Gated Counter Mode */
               /*   \_________ Enable PACA (16 bit mode) */
start_count = PACA;
while ((PTT & 0x80) == 0) ;    /* Wait input goes high */
while ((PTT & 0x80) == 0x80) ; /* Wait input goes low */
end_count = PACA;
total_count = end_count - start_count;
DB12FNP->printf("Total clock cycles = %d\r\n",total_count);
}
```

Review for Final Exam

• Numbers
– Decimal to Hex (signed and unsigned)
– Hex to Decimal (signed and unsigned)
– Binary to Hex
– Hex to Binary
– Addition and subtraction of fixed-length hex numbers
– Overflow, Carry, Zero, Negative bits of CCR

• Programming Model
– Internal registers – A, B, (D = AB), X, Y, SP, PC, CCR

• Addressing Modes and Effective Addresses
– INH, IMM, DIR, EXT, REL, IDX (Not Indexed Indirect)
– How to determine effective address

• Instructions
– What they do (HCS12 Core Users Guide)
– What machine code is generated
– How many cycles to execute
– Effect on CCR
– Branch instructions – which to use with signed and which with unsigned

• Machine Code
– Reverse Assembly

• Stack and Stack Pointer
– What happens to stack and SP for instructions (e.g., PSHX, JSR)
– What happens to stack and SP for interrupt
– What happens to stack and SP when program leaves an interrupt service routine

• Assembly Language
– Be able to read and write simple assembly language program
– Know basic psuedo-ops – e.g., equ, dc.b, ds.w
– Flow charts

• C Programming
– Setting and clearing bits in registers
  PORTA = PORTA | 0x02;
  PORTA = PORTA & ~0x0C;
– Using pointers to access specific memory location or port.
  * (unsigned char *) 0x0400 = 0xaa;
  #define PORTX (* (unsigned char *) 0x400)

• Interrupts
– Interrupt Vectors (and reset vector)
_ How to set interrupt vector in assembly
_ How to set interrupt vector in C
– How do you enable interrupts (specific mask and general mask)
– What happens to stack when you receive an enabled interrupt
– What happens when you leave ISR with RTI instruction?
– What setup do you need to do before enabling interrupts?
– What do you need to do in interrupt service routine (clear source of interrupt, exit with RTI instruction)?

• SPI (Synchronous Serial Communications)
– Pins used – SCLK, MOSI, MISO, SS
– Difference of use in Master and Slave mode
– SPI0CR1 Register
_ Enable SPI
_ Master or Slave
_ Enable interrupts
_ Clock polarity
_ Clock phase
_ Automatically operate SS for single-byte transfers
_ LSB or MSB first
– SPI0CR2 — Bidirectional vs. normal mode
– SPI0BR Register — Set speed (master only)
– SPI0SR Register — SPIF flag - clear by reading SPI0SR, the read SPI0DR
– SPI0SR Register — SPITF flag - clear by reading SPI0SR, the write SPI0DR
– SPI0DR Register — shift register – master starts transfer by writing data to SPI0DR

• A/D Converter
– How to power-up A/D converter (ATDCTL2)
– Write 0x05 to ATDCTL4 to set at fastest conversion speed and 8-bit conversions
– Write 0x85 to ATDCTL4 to set at fastest conversion speed and 10-bit conversions
– Left justified or right justified, signed or unsigned
– How to set modes of A/D converter (ATDCTL5)
_ Continuous Scan vs. Single Scan
_ Multichannel vs. Single Channel conversions
· Channel for single channel scan
· Starting channel for multichannel scan
– How to tell when conversion is complete - ATDSTAT register
– How to read results of A/D conversions
_ Be able to convert from digital number to voltage, and from voltage to digital number (need to know VRH and VRL).

• SCI (Asynchronous Serial Communications)
– Baud rate
– Start and stop bits, data synchronization
– LSB first
– Parity, number of data bits
– 9S12 SCI initialization and use
_ SCI0CR1 — enable parity, set parity type
_ SCI0CR2 — enable transmitter and reciever, enable interrupts
_ SCI0BDH and SCI0BDR — set baud rate
_ SCI0SR1 — check to see if data available, okay to transmit, check or errors
_ SCI0DRL — 8 bit data register (both receive and transmit)
_ SCI0DRH — 9th bit of data in 9-bit data mode

• The HC12 in Expanded Mode
– Getting into expanded mode — MODA, MODB, MODC pins or MODE Register
– PEAR Register — enable ECLK, LSTRB, R/W on external pins
– Ports A and B in expanded mode
_ Port A – AD 15-8 (Port A is for data for high byte, even addresses)
_ Port B – AD 7-0 (Port B is for data for low byte, odd addresses)
– E clock
_ Address on AD 15-0 when E low, Data on AD 15-0 when E high
_ Need to latch address on rising edge of E clock
_ On write (output), external device latches data on signal initiated by falling edge of E
_ On read (input), HC12 latches data on falling edge of E
– R/W Line
– LSTRB line
– Single-byte and two-byte accesses
_ 16-bit access of even address – A0 low, LSTRB low – accesses even and odd bytes
_ 8-bit access of even address – A0 low, LSTRB high – accesses even byte only
_ 8-bit access of odd address – A0 high, LSTRB low – accesses odd byte only