Embedded Systems Design: A Unified Hardware/Software Introduction

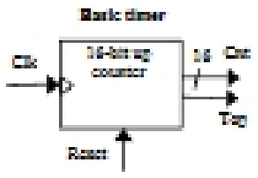
Chapter 4 Standard Single Purpose Processors: Peripherals

Introduction

- Single-purpose processors
  - Performs specific computation task
  - Custom single-purpose processors
    - Designed by us for a unique task
  - *Standard* single-purpose processors
    - "Off-the-shelf" -- pre-designed for a common task
    - a.k.a., peripherals
    - serial transmission
    - analog/digital conversions
  - Low NRE cost
  - Low unit cost

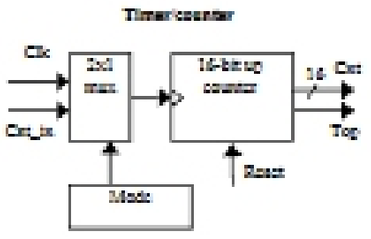
Timers, counters, watchdog timers

- Timer: measures time intervals - very common
  - To generate timed output events
    - e.g., hold traffic light green for 10 s
  - To measure input events
    - e.g., measure a car's speed
- Based on counting clock pulses
  - E.g., let Clk period be 10 ns
  - And we count 20,000 Clk pulses
  - Then 200 microseconds have passed
  - 16-bit counter would count up to 65,535*10 ns = 655.35 microsec., resolution = 10 ns
  - Top: indicates top count reached, wrap-around
    - Can be used to extend range with use of microprocessor



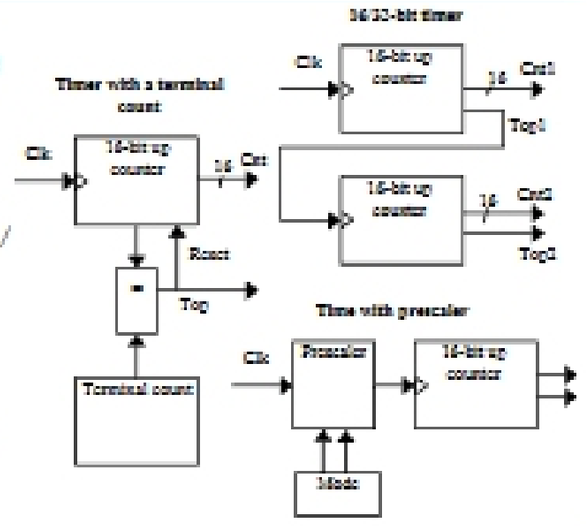
Counters

- Counter: like a timer, but counts pulses on a general input signal rather than clock
  - e.g., count cars passing over a sensor
  - Can often configure device as either a timer or counter

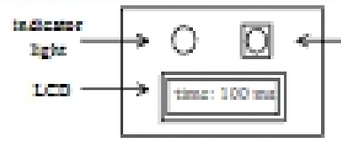


Other timer structures

- Interval timer
  - Indicates when desired time interval has passed
  - We set terminal count to desired interval
    - *Number of clock cycles = Desired time interval / Clock period*
- Cascaded counters
- Prescaler
  - Divides clock
  - Increases range, decreases resolution



Example: Reaction Timer

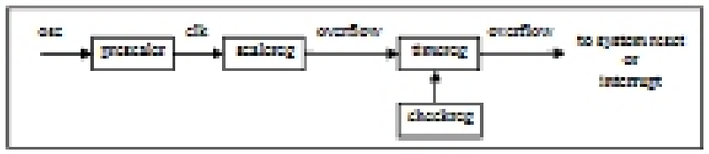


- Measure time between turning light on and user pushing button
  - 16-bit timer, clk period is 83.33 ns (12 MHz), counter increments every 6 cycles (once per instruction cycle - microcontroller specific)
  - Resolution = 6*83.33=0.5 microsec.
  - Range = 65535*0.5 microseconds = 32.77 milliseconds
  - Want program to count millisec., so initialize counter to 65535 – 1000/0.5 = 63535

```
/* reaction.c */

#define MAX_COUNT    65535
void main(void);
  int reaction_milliseconds = 0;

  configure timer mode
  set Cnt to MAX_COUNT

  wait a random amount of time
  turn on indicator light
  start timer

while (user has not pushed reaction button);
  stop timer
  read Cnt
  reaction_milliseconds = ...;
}
turn light off
printf("time: %d ms", reaction_milliseconds);
}
```


Watchdog timer

- Must reset timer every X time unit, else timer generates a signal
- Common use: detect failure, self-reset
- Another use: timeouts
  - e.g., ATM machine
  - 16-bit timer, 2 microsec. resolution
  - *timercnt* value = 2*(2^16-1)–X = 131070–X
  - For 2 min., X = 120,000 microsec.

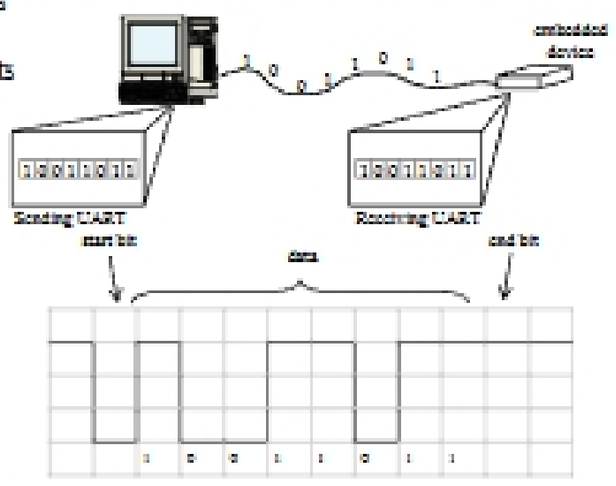



```
/* main.c */

main(){
  wait until card inserted
  call watchdog_reset_routine

  while(transaction in progress){
    if(button pressed){
      perform warning/ending action
      call watchdog_reset_routine
    }

/* if watchdog_reset_routine not called every
<2 minutes, interrupt_service_routine is
called */
}
```

```
watchdog_reset_routine(){
/* watchdog is set so next reset load value into
timercnt. Done is loaded into scaler=n and
131070 is loaded into timer=g */

  checking = 1
  scaler=n = 0
  timer=g = 131070
}

void interrupt_service_routine(){
  signal card
  reset screen
}
```

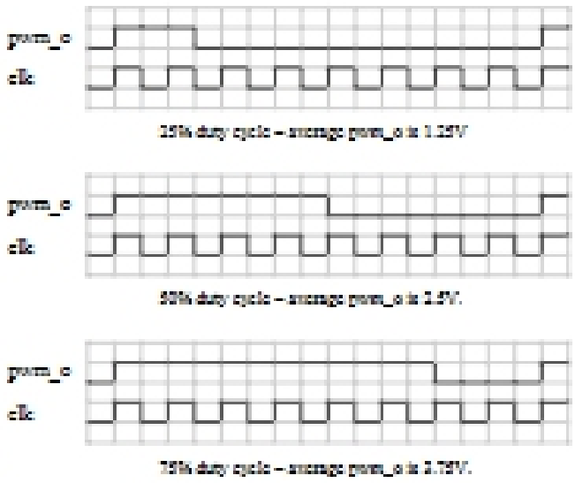

Serial Transmission Using UARTs

- UART: Universal Asynchronous Receiver Transmitter
  - Takes parallel data and transmits serially
  - Receives serial data and converts to parallel
- Parity: extra bit for simple error checking
- Start bit (receiver continually monitors for it), stop bit
- Baud rate
  - signal changes per second
    - Bits shifted out of buffer
  - Speed of communication
  - Configured via configuration register

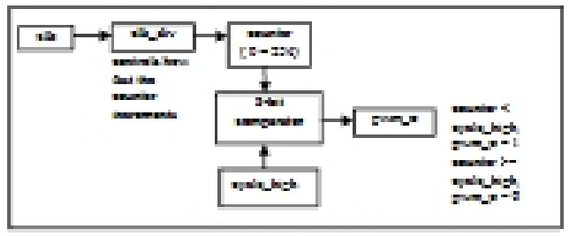


Pulse width modulator

- Generates pulses with specific high/low times
- Duty cycle: % time high
  - Square wave: 50% duty cycle
- Common use: control average voltage to electric device
  - Simpler than DC-DC converter or digital-analog converter
  - DC motor speed, dimmer lights
- Another use: encode commands, receiver uses timer to decode



Controlling a DC motor with a PWM



Internal Structure of PWM

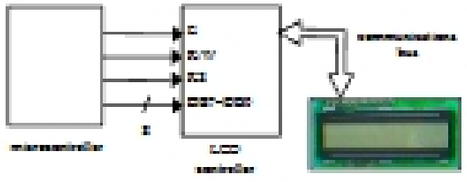
Input Voltage	% of Maximum Voltage Applied	Motor RPM
0	0	0
0.5	20	1500
2.75	75	6000
3.0	100	8000

Relationship between applied voltage and speed of the DC Motor

```
void main (void) {

    /* enable peripheral */
    PERIPH = 0x01;
    /* enable duty cycle */
    PDCYC = 0x75;

    while(1) {}
}
```

The PWM alone cannot drive the DC motor, a possible way to implement a driver is shown below using an NPN PWM transistor.

LCD controller

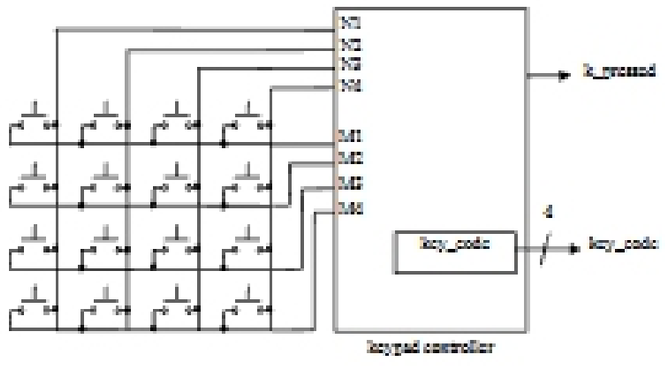



```
void WriteCharByte n(){

  RS = 1;                 /* indicate data being sent */
  DATA_BUS = a;           /* send data to LCD */
  EnableLCD(E2);          /* toggle the LCD with appropriate delay */
}
```

CODES	
I/D = 1 cursor moves left	DL = 1 8-bits
I/D = 0 cursor moves right	DL = 0 4-bits
S = 1 with display shift	N = 1 2 rows
S/C = 1 display shift	N = 0 1 row
S/C = 0 cursor movement	F = 1 5x10 dots
R/L = 1 shift to right	F = 0 5x7 dots
R/L = 0 shift to left	

RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0	Description
0	0	0	0	0	0	0	0	0	1	Clear all display, return cursor home
0	0	0	0	0	0	0	0	1	*	Return cursor home
0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and/or specifies entire display shift
0	0	0	0	0	0	1	D	C	B	Sets entire display (D) on/off, cursor on/off (C), and blink position (B)
0	0	0	0	0	1	S/C	R/L	*	*	Move cursor and shifts display
0	0	0	0	1	DL	N	F	*	*	Sets interface data length, number of display lines, and character font
1	0	WRITE DATA								Write Data

Keypad controller



N=4, M=4