PWM Tutorial using 8051

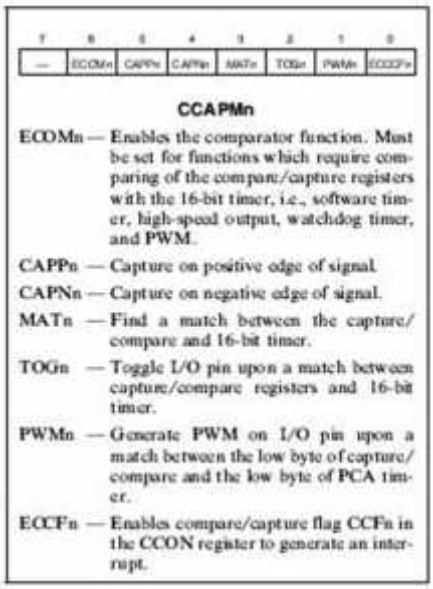
DC Motor Speed Control

The T89C51AC2:

The T89C51AC2 is an 8-bit microcontroller based on the 8051 architecture. It is an enhanced version of the 89C51and incorporates many new features including the Programmable Counter Array (PCA). Included in the Programmable Counter Array are a 16- bit free running timer and 5 separate modules.

The PCA timer has two 8-bit registers called CL (low byte) and CH (high byte), and is shared by all modules. It can be programmed to take input from four different sources. The inputs provide flexibility in choosing the count rate of the timer. The maximum count rate is 4 MHz ((/4 of the oscillator frequency). Some of the port 1 pins are used to interface each module and the timer to the outside world. When the PCA modules do not use the port pins, they may be used as regular I/O pins. The modules of the PCA can be programmed to perform in one of the following modes: capture mode, compare mode, high-speed output mode, pulse width modulator (PWM) mode, or watchdog timer mode (only module 4).

Every module has an 8-bit mode register called CCAPMn, and a 16-bit compare/capture register called CCAPnL & CCAPnH, where n can be any value from 0 to 4 inclusive. By setting the appropriate bits in the mode register you can program each module to operate in one of the aforementioned modes.



CCAPMn

ECOMn — Enables the comparator function. Must be set for functions which require comparing of the compare/capture registers with the 16-bit timer, i.e., software timer, high-speed output, watchdog timer, and PWM.

CAPPn — Capture on positive edge of signal.

CAPNn — Capture on negative edge of signal.

MATn — Find a match between the capture/compare and 16-bit timer.

TOGn — Toggle I/O pin upon a match between capture/compare registers and 16-bit timer.

PWMn — Generate PWM on I/O pin upon a match between the low byte of capture/compare and the low byte of PCA timer.

ECCFn — Enables compare/capture flag CCFn in the CCON register to generate an interrupt.

Fig. 16 - CCAPMn Register

When a module is programmed in capture mode, an external signal on the corresponding port pin will cause a capture of the current value of the 16-bit timer. By setting bits CAPPn or CAPNn or both, the module can be programmed to capture on the rising edge, falling, edge, or either edge of the signal. If enabled, an interrupt is generated at the time of capture. When module is to perform in one of the compare modes (software timer, high speed output, watch dog timer, PWM), the user loads the capture/compare registers with a calculated value, which is compared to the contents of the 16-bit timer, and causes an event as soon as the values match. It can also generate an interrupt.

PWM is one of the compare modes and is the only one, which uses only 8 bits of the capture/compare register. The user writes a value (0 to FFH) into the high byte (CCAPnH) of the selected module. This value is transferred into the lower byte of the same module and is compared to the low byte of the PCA timer. While CL < CCAPnL the output on the corresponding pin is a logic 0. When CL > CCAPnL, the output is a logic 1. In this application note we will see how a module can be programmed to perform as a PWM to control the speed and direction of a DC motor.

Setting Up the PCA:

The T89C51AC2 has several Special Function Registers (SFRs) that are unknown to ASM51 versions before 2.4. The names of these SFRs must be defined by DATA directive or be defined in a separate file and be included at the time of compilation.
Two special function registers are dedicated to the PCA timer to allow mode selection and control of the timer. These registers are CCON and CMOD and are shown in figure contains the PCA timer ON/OFF bit (CR), timer rollover flag (CF) and module flags (CCFn). Module flags are used to determine which module causes the PCA interrupt.

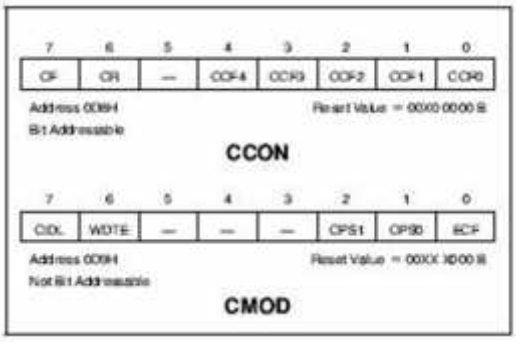


Fig. 17 - CCON and CMOD registers

First the clock source for the PCA timer must be defined. The 16-bit timer may have one of four sources for its input. These sources are: osc freq/4, osc freq/12, timer 0 overflow, and external clock. Two bits in the CMOD register are dedicated to selecting one of the sources for the PCA timer input. They are bits 1 and 2 of CMOD, which are called CPS0 and CPS1. CMOD is not bit addressable, thus the value must be loaded as a byte. As shown in figure all the sources and the corresponding values of CPS0 and CPS1.

CPS1	CPS0	TIMER INPUT SOURCE
0	0	Internal clock, Fosc/12
0	1	Internal clock, Fosc/4
1	0	Timer 0 overflow
1	1	External clock (input on P1.2)

Fig. 18 - Timer Input Source

Next the appropriate module must be programmed as a PWM. As it was noted earlier, the 8-bit mode register for each module is called CCAPMn. Bit 1 of each register is called PWMn. This bit along with ECOMn (bit 6 of the same register) must be set to program the module in the PWM mode. PWM is one of the compare functions of the PCA, and ECOMn enables the compare function. Thus, the hex value that must be loaded into the appropriate CCAPMn register is 42H.

Now that the module is programmed as a PWM, a value must be loaded in the high byte of the compare register to select the duty cycle. The value can be any number from 0 to 255. In the T89C51AC2 loading 0 in the CCAPnH will yield 100% duty cycle, and 255 (0FFh) will generate a 0.4% duty cycle as seen in figure below.

The next step is to start the PCA timer. The bit that turns the timer on and off is called CR and is bit 6 of CCON register as shown in the figure of the CCON register figures. Since this register is bit addressable, you can use bit instructions to turn the timer on and off. In the following example module 2 has been selected to provide a PWM signal to a motor driver. An external clock will be provided for the timer input, so the value that needs to be loaded into CMOD is 06H.

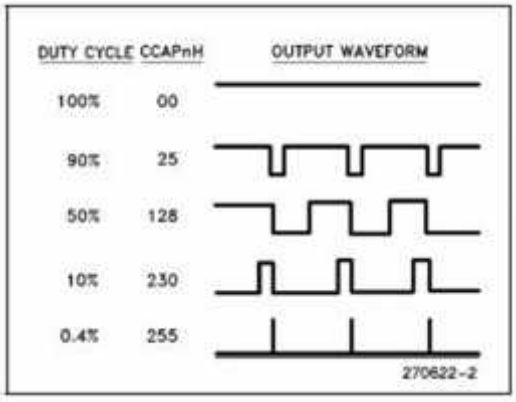


Fig. 19 - Selected Duty Cycles and Waveforms

The following table will give the percentages of the ON time for the two linear DC motors that will drive the robot. As explained above the 8-bit decimal value (0 – 255) that will be inputted in the CCAPxH and CCAPxL will be the OFF time so here is an example to show how to achieve the ON time percentage. Code example:

```
MOV CCAP0H, #90D
MOV CCAP0L, CCAP0H
```

Calculation for the ON time percentage:

- (255D – 90D) = 165 D (ON time)
- (165 / 255) * 100% = 65%