SIEMENS

Microcontrollers

ApNote

AP1635

☐ additional file
APXXXX01.EXE available

Fuel Injection Control for a 12 Cylinder Formula 1 Engine

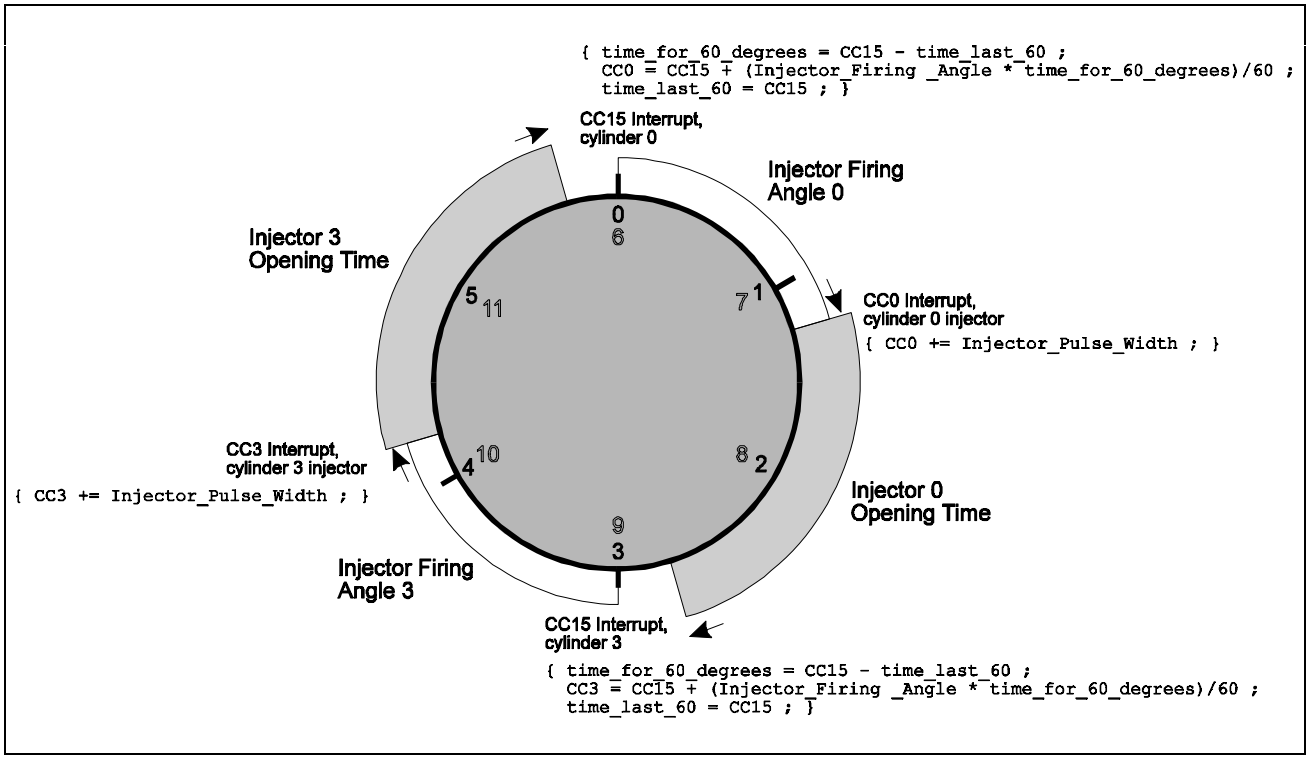
Consequently, the C166 finds many applications in areas such as motor control and automotive engine management where the production of precisely timed pulse trains are required.

Michael Beach / Hitex

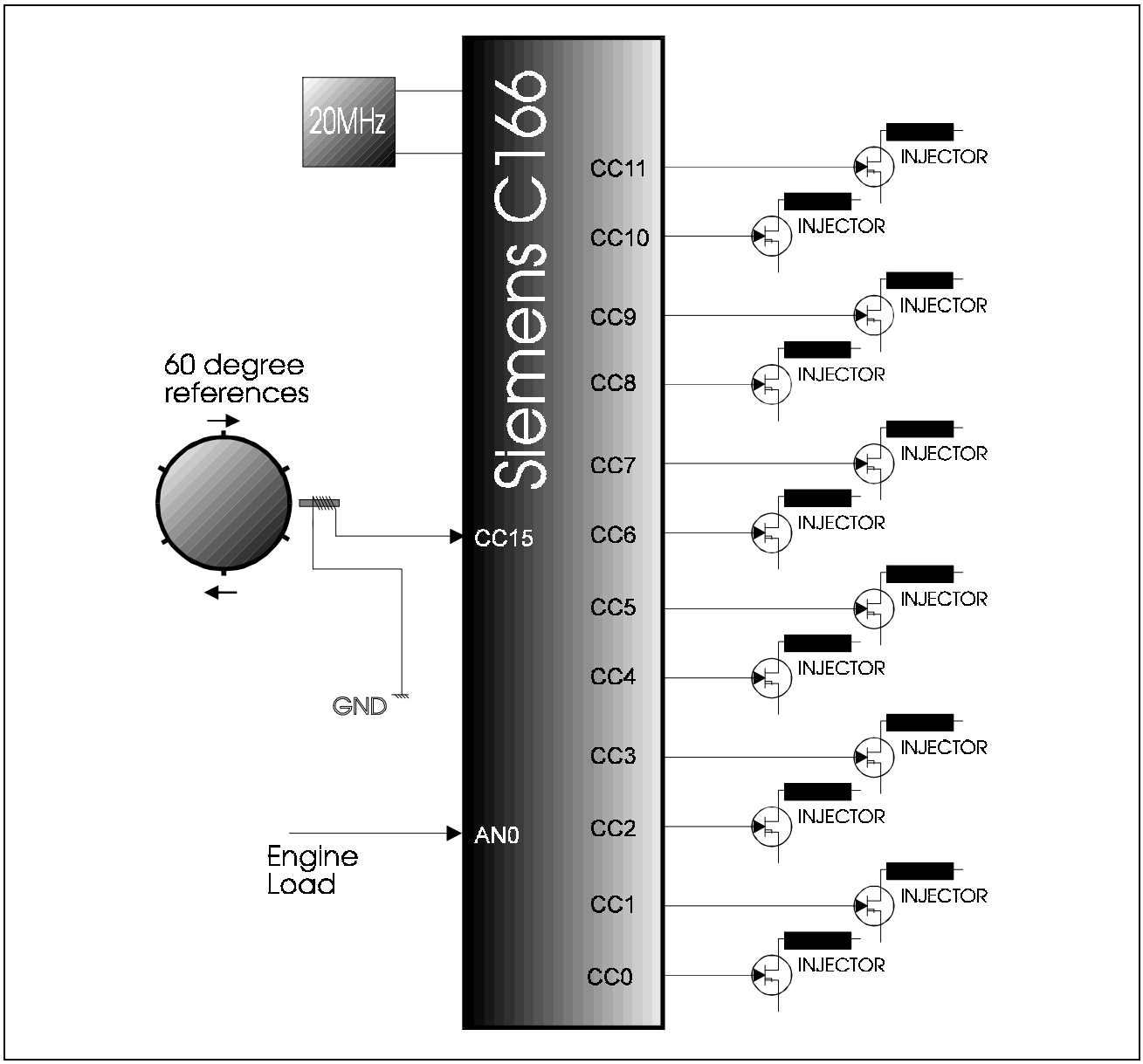
Introduction

The C166 family is perhaps unique in having 16 high speed I/O pins coupled to a capture and compare (CAPCOM) unit that can be used either as pulse generators or event detectors. The combination of a very fast RISC-like CPU and a 16 channel capture and compare unit (CAPCOM) allows the easy generation of multiple pulse trains, all locked to a single timebase. Consequently, the C166 finds many applications in areas such as motor control and automotive engine management where the production of precisely timed pulse trains are required. As the CAPCOM unit is an integral part of the CPU, it can be fully configured and controlled directly from assembler or C and does not require the use of microcode.

This application note outlines how 13 such C166 channels can be used to provide fuel injector waveform drives, as used on a 12 cylinder formula 1 engine. The overall CPU loading to achieve this is just 3.6% at 18000rpm, even when programmed in C!



The input pulse train is fed into input capture channel 15 from a sensor giving a negative edge at each cylinder's TDC (top dead center). In a 12 cylinder engine, successive TDCs are 60 degrees apart and at the maximum operating speed of 18000rpm, this corresponds to a time of 555µs between TDC "references".



At each TDC, the capture register CC15 latches the value of a free running 16 bit timer, running at a 1.6µs per count. An interrupt service routine is subsequently called, during which the C166 must make an analog to digital conversion of the throttle position (and hence mass airflow), read the injector firing angle, relative to the TDC from a table and then create a turn-on time-out using a compare output to turn that cylinder's injector on. It must also immediately switch off the injector, in case it is still on from the previous cycle, as might happen under high acceleration conditions. This foreshortening of the truncated pulse requires the fuel deficit to be made up by a transient fueling strategy which is outside the scope of this piece.

```
/*** Crankshaft TDC Interrupt ***/

void tdc_int(void) interrupt 0x1f using TDCREGS {

   unsigned int temp, time_for_60 ;  // Make this a register variable
   unsigned int time_last_60 ;  // These are really static but
dedicated
   unsigned int tooth_count ;   // regbank TDCREGS makes them like
static
                                // register variables

   /*** Read Pressure Channel For Synchronous Fuelling ***/

   ADCON = 0 ;       // Set AN0
   ADST = 1 ;        // Start conversion

   time_for_60 = CC15 - time_last_60 ;  // Find time between references
```

In a conventional processor, the CPU working registers would have to be pushed onto the stack on entry to the interrupt routine, resulting a delay of many microseconds before the CPU is in a position to start processing. The C166 allows a simple context switch to be performed to make available a fresh set of working registers. This uses the SCXT ("switch context") instruction and takes 100ns - the "USING" control in the function prototype above performs the switch from C. With this dedicated set of registers, all the data used can be held purely in registers so that most instructions take just 100ns - in keeping with the RISC-like CPU design.

The conversion from the angle domain into the time domain, necessary to create the injector firing angle time-out, is performed by a scaling calculation, based on the measured time since the last TDC interrupt. As the compare registers are attached to the same timer as the input capture, the calculated firing angle time-out is simply added to the captured value of the time at the start of the TDC interrupt.

The preceding calculations give sufficient time to allow the A/D conversion to be completed. The result is fed into a two-dimensional look-up table interpolator, along with the engine speed to obtain the required injector pulsewidth, based on the A/D reading and the engine speed. By structuring the C source for the interpolator function to take best advantage of the C166 core design, the run time can be got down to just 24µs.

Overall, the software effectively simulates the fuel distributor found in a mechanical system, such as Bosch K-Jetronic. The distribution of injector pulses is performed via a reference count, which allows each individual cylinder's TDC to be identified and hence the appropriate injector to be fired via one of the 12 CAPCOM outputs.

When the compare register matches the timer value after the calculated time-out period, the corresponding compare pin changes state to switch the injector on. A simple interrupt routine is now required to set the turn off time-out, to define the injector opening time. This injector-off interrupt takes a total of just 0.4µs!

```
/*** Injector On Interrupts To Determine On-Time ***/

void CC0_int(void) interrupt 0x10 {

   CC0 += inj_PW ;
   CC0IE = 0 ;      // No more interrupts until next ref
   }
```

The problem arises when the engine is accelerating hard as the injector firing angle to time-out conversion will be wrong as it was based on a time between references that is no longer current.

```
 temp = inj_delay_angle/Max_Timeout_Angle ;
   timeout_angle[tooth_count] = MDH ;  // Decide angle to timeout over.
This
                                       // direct access to MDH gives
the
                                       // remainder without doing a DIV
again.

   temp += tooth_count ;               // Decide which tooth to time
out from
   if(temp >= No_Of_Cylinders) {       // Limit to correct no of teeth
per cycle
      temp -= No_Of_Cylinders;
      }

   timeout_tooth[tooth_count] = temp ;
```

The solution is to time-out each firing angle from the reference just before the required angle. This simply amounts to subtracting multiples of 60 degrees from the angle and then just timing out over the remainder of angle/60. The overall effect is to count references until the one just before the required firing point and then to set the time-out.

```
  /*** Check If We Need To Timeout From This Tooth ***/

   if(timeout_tooth[0] == tooth_count) {

      temp = timeout_angle[0] ;
      temp = ((unsigned long)time_for_60 * temp)/Max_Timeout_Angle ; //
Inline DIV
      CC0 = temp + T0  ;
      CC0IR = 0 ;   // Could be done with a pointer CCxIC!
      CC0IE = 1 ;
      }

   if(timeout_tooth[1] == tooth_count) {
      }
... etc.
```

The overall run time of the whole TDC interrupt is 45.9µs, including 25µs for the 2-dimensional look up table interpolation, at a CPU clock of 20MHz with a 16 bit non-multiplexed bus.

The overall CPU load to perform the injector drive signal amounts to just 3.6% at 18000 rpm or 7% when the interpolation is included.