Using the TMS370 Pact Module in Engine Control Application

TEXAS INSTRUMENTS

APPLICATION NOTE

USING THE TMS370 PACT MODULE IN ENGINE CONTROL APPLICATION

OCTOBER 1989

CONTENTS

Illustrations

USING THE TMS370 PACT MODULE IN ENGINE CONTROL APPLICATION

OCTOBER 1989

1. INTRODUCTION

In order to fit with more severe emission norms and to minimize fuel consumption without lowering the engine horse power, engine control systems need to be increasingly sophisticated. The increase of engine control functions such as static ignition, sequential fuel injection, exhaust gas recirculation, knock control..., implies the use of very powerful microcontrollers to process the timing functions. In engine control, most of the signals such as dwell time, spark time, fuel injection time must be synchronous with the engine cycle and must be generated by the microcontroller to a very precise time or a very precise engine angle. The TMS370 PACT module with its capabilities to combine external events with internal timers, to synchronize output to external events, to generate interrupt on event or event and time, allows the execution of software tasks (advance angle computation, fuel injection time computation) and output actions in perfect synchronization and at the optimum time of the engine cycle. This application note describes a high end 4 cylinder engine control system using the TMS370 PACT module.

Nota: A good understanding of this application note requires a minimum knowledge of the TMS370 family and PACT module.

2. ENGINE CONTROL SYSTEM DESCRIPTION

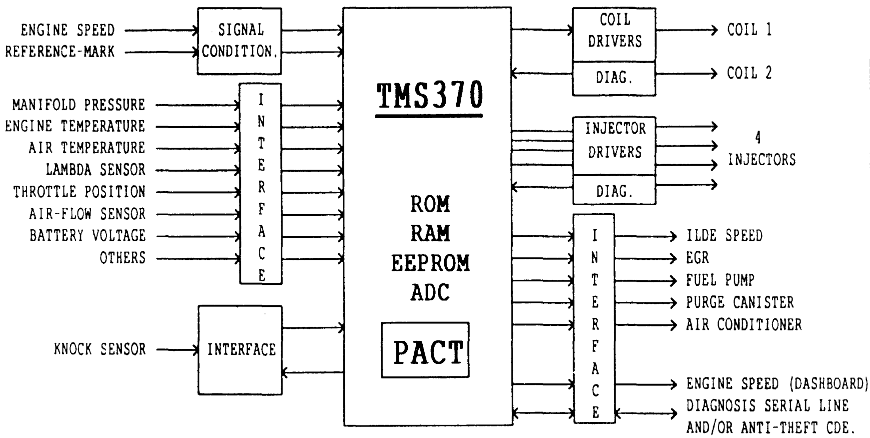
Basically, Engine Control Systems consist of two main functions, namely, ignition Control and injection Control.
At present, most ignition systems use a mechanical distributor to supply the high voltage current to the Spark plugs in a specific sequence. This ignition sytem is usually called dynamic ignition. Dynamic ignition will be increasingly replaced by semi-static or static ignition in which the distributor is replaced by two or four ignition coils driven by two or four power transistors. A semi- static ignition has two Spark plugs per coil. A static ignition has one Spark plug per coil.
In a fuel injection system, there are also two types of systems:

- The mono-point injection which requires one injector to supply all the cylinders. The gasoline is injected upstream of the throttle.
- The multi-point injection which requires one injector per cylinder. In this case, the gasoline is directly injected into the intake manifold near the intake valves by either all injectors at the same time (called full group injection), or by each injector sequentially (called sequential injection).

To meet emission control, fuel economy, and engine performance requirements, other subsystems in engine control are necessary, such as: exhaust gas recirculation (EGR) to reduce the peak combustion temperature in order to minimize the nitro-oxyde content of exhaust gas, idle speed control to keep a steady idle speed, knock control to eliminate knocking under all operating conditions, purge canister to suppress tank hydro-carbon vapors, adaptive strategies etc.

Figure 1 shows a typical architecture of a 4-cylinder semi-static ignition, sequential fuel injection.

FIGURE 1 : ENGINE CONTROL BLOCK DIAGRAM



The ignition timing and injection time values are mainly calculated from the two basic input signals which are the engine speed and the intake manifold pressure (engine load) described later. Other inputs, such as air temperature, coolant temperature, air flow, throttle position etc., with the help of system maps and computation, allow some correction of the calculated values. This application report describes in detail how the TMS370 PACT module can automatically drive all the above mentioned functions, such as ignition coil outputs, injector outputs, knock window generation, pwm signals etc. All the examples will help the reader to get familiar with the TMS370 PACT module and to better understand how to use and program the different PACT commands.

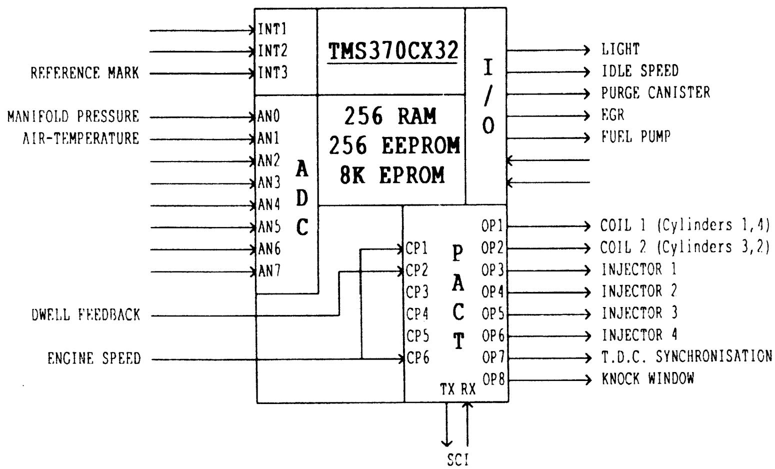
3- ENGINE CONTROL APPLICATION ARCHITECTURE

3.1 TMS370 CONFIGURATION

The engine control system described in this application report is a semi-static ignition, multi-points sequential fuel injection system for a four cylinder engine (Figure 2). The TMS370 micro-controller used is the TMS370C732 with the following configuration:

8K Bytes EPROM program	PACT
256 Bytes EEPROM Data	A/D Converter 8 channels
256 Bytes RAM	44-pins PLCC packaging

Figure 2 : TMS370 CONFIGURATION IN ENGINE CONTROL



As the continuation of the engine control system application report often refers to the PACT peripheral registers to configure the different PACT functions, the PACT peripheral control register table is shown below. Appendix A describes the function of each bit of each PACT control register.

TABLE 3-1: PACT PERIPHERAL FRAME DEFINITION

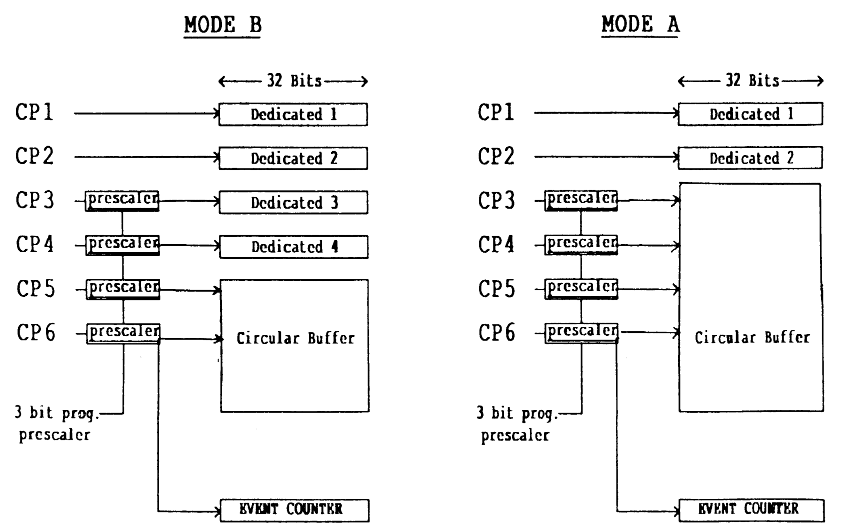
GLOBAL	.equ P04F	;Global function control register
WDKEY	.equ P04E	;Watchdog Key register
INPUTCTL	.equ P04D	;Event/Interrupt control register
CP65	.equ P04C	;Setup pins CP6 and CP5 register
CP43	.equ P04B	;Setup pins CP4 and CP3 register
CP21	.equ P04A	;Setup pins CP2 and CP1 register
CTLFLG	.equ P049	;Command/Definition entry flags register
OP1OP8	.equ P048	;Output pin 1-8 state register
SCITXDT	.equ P047	;SCI TX data register
SCIRXD	.equ P046	;SCI RX data register
SCICTL	.equ P045	;SCI control register
BUFPT	.equ P043	;Buffer pointer register
ENDCONT	.equ P042	;Command/Definition area end register
STARTCONT	.equ P041	;Command/Definition area start register
RESOL	.equ P040	;Setup control register

3.2 REGISTER FILE ORGANIZATION

The size of the register file used in this application is 256 bytes and the PACT access is limited to the 128 upper bytes. This RAM is primarily divided into three parts:
- General purpose register file accessed only by the CPU.
- Stack register file.
- Register file accessed by the PACT and the CPU.

In this engine control application, the PACT has been configured in mode A by clearing bit2 "GLOBAL peripheral register -P04F- (two dedicated input capture registers CP1, CP2) and the circular buffer length has been limited to one 32-bit word. The remaining 128 upper bytes are reserved for the COMMAND/DEFINITION area. Figure 3 shows the PACT mode A and mode B.

Figure 3 : INPUT FUNCTIONS



The start address which also fixes the circular buffer length is defined by programming the start address control register in the PACT peripheral frame ("STARTCNT": PO41). The stop address which defines the end of the COMMANDS/DEFINITIONS area is programmed in the stop address control register ("ENDCNT": PO42). In this application, "STARTCNT" is programmed to "EFH" and "ENDCNT" to "98H". In this case, the size of the COMMAND/DEFINITION area allows 22 commands or definitions to be programmed.

3.3 COMMAND/DEFINITION FILE FORMAT

As discussed below, this application uses two different contexts for the COMMAND/DEFINITION area. The first one uses only two commands and works only during the initialization phase of the program (missing teeth research) to get the engine synchronization. The second can accept up to 28 available commands and works during the normal running of the engine control programs. The following Figure 4 shows these two COMMAND/DEFINITION areas:

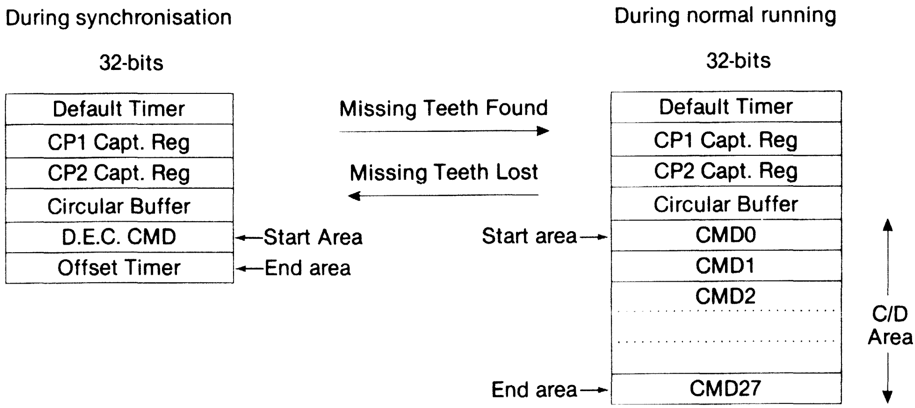


Figure 4: COMMAND/DEFINITION AREAS

Figure 5 shows the complete COMMAND/DEFINITION area programmed in this engine control system application.

TIMELSW TIMEMSW	>FF >FD	Event counter	20-bit Default Timer	>FF	
CP1RLSW CP1RLSW	>FB >F9	Storage Event Counter	CP1 Capture Dedicated Register 20-bit Default Timer	>FB	
CP2RLSW CP2RMSW	>F7 >F5	Storage Event Counter	CP2 Capture Dedicated Register 20-bit Default Timer	>F7	
BUFLSW BUFMSW	>F3 >F1	Circular Buffer		>F3	
COMØLSW COMØMSW	>EF >ED	Dummy Command		>EF	←Start Address Entry Interrupt Flag
COM1LSW COM1MSW	>EB >E9	End Injection 1 Command (S.C.) Reset OP3		>EB	2
COM2LSW COM2MSW	>E7 >E5	End Injection 2 Command (S.C.) Reset OP4		>E7	1
COM3LSW COM3MSW	>E3 >E1	End Injection 3 Command (S.C.) Reset OP5		>E3	0
COM4LSW COM4MSW	>DF >DD	End Injection 4 Command (S.C.) Reset OP6		>DF	7
COM5LSW COM5MSW	>DB >D9	Dwell Time 1 & 4 Command (S.C.) Set OP1		>DB	6
COM6LSW COM6MSW	>D7 >D5	Dwell Time 3 & 2 Command (S.C.) Set OP2		>D7	5
COM7LSW COM7MSW	>D3 >D1	Offset Timer Definition Event Max. = 115		>D3	4
COM8LSW COM8MSW	>CF >CD	Missing Tooth 57, 115 Command (D.E.C.) Interrupt		>CF	3
COM9LSW COM9MSW	>CB >C9	Start Injector 1 Command (D.E.C.) Set OP3. Interrupt		>CB	2
COM10LSW COM10MSW	>C7 >C5	Start Injector 2 Command (D.E.C.) Set OP4. Interrupt		>C7	1
COM11LSW COM11MSW	>C3 >C1	Start Injector 3 Command (D.E.C.) Set OP5. Interrupt		>C3	0
COM12LSW COM12MSW	>BF >BD	Start Injector 4 Command (D.E.C.) Set OP6. Interrupt		>BF	7
COM13LSW COM13MSW	>BB >B9	RPM IT 3, 33 (D.E.C.) Interrupt		>BB	6
COM14LSW COM14MSW	>B7 >B5	Pre-Spark Coil 1, 3, 4, 2 Command (D.E.C.) Interrupt		>B7	5
COM15LSW COM15MSW	>B3 >B1	1, 2, 3 & 4 Spark Time (C.C.) Reset OP, OP2		>B3	4
COM16LSW COM16MSW	>AF >AD	TDC 18, 48, 76, 106 (D.E.C.) Set/Reset OP7		>AF	3

→

COM17LSW	>AB	Knock Window 1, 4, 3 & 2 (C.C)	>AB	
COM17MSW	>A9	Set OP8		2
COM18LSW	>A7	Knock Window 1, 4, 3 & 2 (C.C.)	>A7	
COM18MSW	>A5	Reset OP8		1
COM19LSW	>A3	Baud Rate Definition	>A3	
COM19MSW	>A1	Not Programmed		0
COM20LSW	>9F	Dummy Command	>9F	
COM20MSW	>9D			7
COM21LSW	>9B	RPM IT 61, 91 (D.E.C.)	>9B	
COM21MSW	>99	Interrupt		7
COM22LSW	>97	Not Used	>97	←End Address
COM22MSW	>95			
COM23LSW	>93	Not Used	>93	
COM23MSW	>91			
COM24LSW	>8F	Not Used	>8F	
COM24MSW	>8D			
COM25LSW	>8B	Not Used	>8B	
COM25MSW	>89			
COM26LSW	>87	Not Used	>87	
COM26MSW	>85			
COM27LSW	>83	Not Used	>83	
COM27MSW	>81			

Figure 5: Engine control COMMAND/DEFINITION area

Noting the position of the "Offset Timer/Event Counter" definition in this engine control COMMAND/DEFINITION area, the different commands refer to Timers in the following way:

- referenced to the Default Timer: Command 1 to Command 6
- referenced to the Offset Timer: Commands 15, 17, 18

3.4 TIME SLOT AVAILABILITY AND PACT RESOLUTION

As it is explained in the PACT specifications, the number of definitions/ commands allowed, is governed by the crystal frequency and the resolution required by the application. These two parameters allow the definition of the number of time slots available and hence the number of definitions/ commands. For this engine control application, a 20 MHZ crystal frequency is used and a resolution of 2 μS has been chosen.

To get this 2 μS resolution, the fast mode must be programmed, thus the prescaler rate:

- Fast mode → bit 4 = 1 "RESOL" peripheral register (P040).
- Prescaler = 9 → bit 0-3 = 1001 "RESOL" peripheral register (P040).

This prescaler programming provides a system clock divided by 10, which means:

$$\frac{\text{Sysclk}}{10} = \frac{200\text{ ns}}{10} = 2\mu\text{S of resolution}$$

A table in the PACT specification ("number of Time slots available for each prescale setting table") gives for a divide rate of ten, a number of time slots (T.S) of 29. So, a 20 MHZ crystal frequency and a 2 μS resolution gives 29 T.S.

The commands/definitions need from 1 to 3 T.S., so the number of COMMANDS/DEFINITIONS available is included between 15 and 29 commands. This application uses only a part of these available commands/definitions.

3.5 Inputs Descriptions (Figure 2)
The Engine speed signal, which is described below, drives the input captures CP1 and CP6 of the PACT, on the one hand (CP1) for the average engine speed computation, and on the other hand (CP6) for the 16-bits offset timer capture into the circular buffer and the incrementation of the event counter (E.C.).
- Dwell feedback (coil current) drives the input captures CP2 (not programmed).
- The reference mark, which gives a pulse every engine cycle (every two engine rounds) drives the external interrupts pin INT3.
- Intake manifold pressure (engine load) drives the analog input AN0.
- Air temperature drives the analog input AN1.

3.6 Output Description (Figure 2)
The outputs OP1 and OP2 from the PACT module drive the two ignition coils, coil1 and coil2, which reciprocally drive the Spark plugs from cylinder 1 and 4 and the Spark plugs from cylinder 3 and 2.

The PACT outputs OP3 to OP6 drive the four injectors.

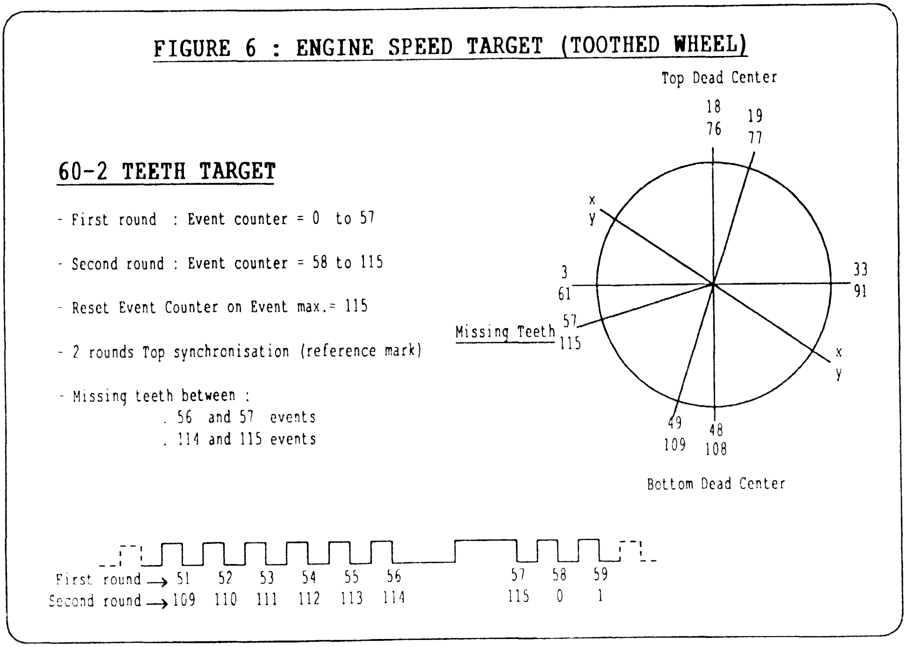
The PACT output OP7 supplies the Top Dead Center (TDC) synchronization signal (explained later).

The PACT output OP8 supplies the knock window signal.

Other TMS370 Standard I/Os drive the engine control ON/OFF or PWM outputs.

4. ENGINE CONTROL FUNCTIONS DESCRIPTION

4.1 Engine speed signal
The Engine speed signal is delivered by a variable reluctance sensor. This sensor supplies a signal whose range varies from ten millivolts to more than one hundred volts according to the signal frequency. This means that signal conditioning is necessary between the sensor and the micro-controller. This sensor is set in front of a toothed wheel having 60 teeth minus two (Figure 6). This wheel is fixed on the crankshaft. A turn of the wheel corresponds to an engine turn and an engine cycle corresponds to two engine turns (four stroke engine). The two missing teeth allow for locating the top dead center (TDC) of the cylinders. Generally, these two missing teeth are around 108 degrees before the TDC of cylinders 1 and 4 (corresponding to the bottom dead center (BDC) of cylinders 3 and 2). The 60-2 teeth target wheel chosen in this application report is a toothed wheel used by some car manufacturers. Other models are also used.



As described in the inputs description, the engine speed signal is input on the PACT CP1 and CP6 inputs (figure 2).
So, the CP1 and CP6 inputs are configured as follows:
- CP1 falling edge detect selected =→ set bit0 "CP21" peripheral register(P04A).
This enables the Default Timer capture in the CP1 dedicated capture register, with each falling edge of the engine speed signal.
- CP6 falling edge detect selected =→ set bit4 "CP65" peripheral register(P04C).
- Event only is selected (no Default Timer capture in the circular buffer) → set bit 2 "INPUTCTL" peripheral register (P04D).

Therefore, each falling edge of the engine speed signal increments the event counter, which numbers each tooth of the toothed wheel. As this toothed wheel has 60 teeth minus 2 teeth, the first engine round of the engine cycle has teeth numbered from 0 to 57, and the second round has teeth numbered from 58 to 115 (Figure 6).

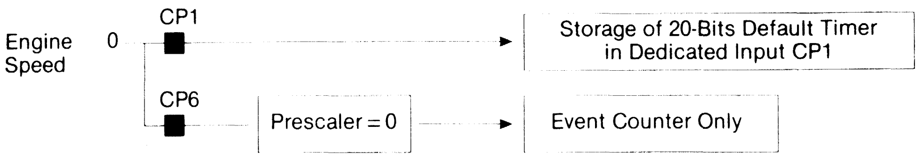


Figure 7: CP1, CP6 inputs configuration

As the CP6 input increments the 8-bit event counter on each falling edge of the engine speed signal, the input prescaler has to be programmed to zero (no prescale): bit 3,4,5 = 0 "INPUTCTL" peripheral register (P04D).

4.2 Engine synchronization (Initialization phase)

4.2.1 Offset Timer/Event Counter definition

In order to process tooth interpolation (to output events on very precise engine angles), the tooth signal period is captured in the circular buffer with an "Offset Timer/Event Counter" definition (Figure 8). This "Offset Timer/Event Counter" definition is programmed in the COMMAND/ DEFINITION area with the following features:

- Reset to zero the offset timer when an event (CP6) occurs. (This allows automatic capture of the tooth signal period in the circular buffer).

- Capture the 16-bit virtual offset timer (defined by the definition) in the circular buffer on every event on CP6 before it is cleared.

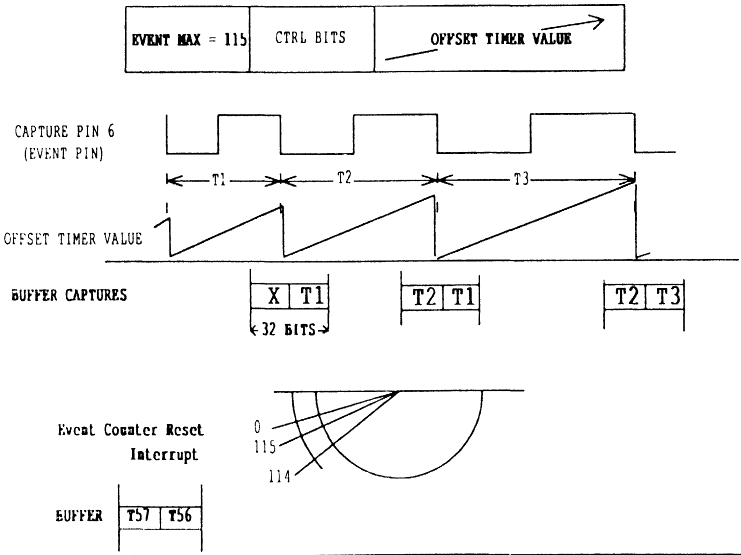
- Reset the event counter to zero on reaching the maximum event counter value of 115 (each engine cycle).

EVENT/OFFSET TIMER definition:

73H	34H	16-bits Offset Timer	1	Command 1
D31			D0	

D0 = 1 → required for the offset timer definition.
D16 = 0 → no step mode.
D17 = 0 → no interrupt when Event Counter reaches the maximum value.
D18 = 1 → start timer.
D19 = 0 → offset timer automatically reset to zero by every event on CP6.
D20 = 1 → reset Default Timer when the event reaches the maximum value. (Only during the initialization phase)
D21 = 1 → Store the 16-bit offset timer (defined by this definition) in the circular buffer on every event on CP6 before it is cleared.
D22 = 0 → no 32-bit data capture into the circular buffer.
D23 = 0 → no interrupt when an event on pin CP6 occurs. (During the initialization phase, The CP6 interrupt is programmed by "CP65" peripheral register).
D24 = 73h → event max = 115 (engine cycle).

FIGURE 8 : OFFSET TIMER DEFINITION



In this application, the "Offset Timer/Event counter" definition is the first command (COM1) during the initialization phase, and becomes the seventh command (COM7) once the engine synchronization has been done.

4.2.2 Command/Definition area definition during synchronization

Numbering the teeth of the toothed wheel permits the synchronization of software tasks with the engine cycle to a precise engine angle (missing teeth checking, ignition advance angle computation, dwell time computation, fuel injection time computation, intake manifold pressure measure etc.). This can be done by enabling an interrupt on a precise event by using the "Double Event Compare" command.

In engine control management, at the start of the engine, the first thing that the software has to do, is to find the missing teeth before taking any action like ignition command, injection commands etc., in order to get the synchronization with the engine cycle.

This missing teeth research is done at the initialization phase of the engine start or after a loss of the missing teeth synchro. During this missing teeth research, the event counter is always cleared (bit 1 = 1 "INPUTCTL" peripheral register (P04D)). Once the missing teeth are found, the event counter starts and is incremented every tooth, then cleared every engine cycle (event max = 115 programmed in the Offset Timer definition). During this initialization phase, the PACT works with a reduced COMMAND/DEFINITION area (Figure 4).

The PACT module configuration during the initialization phase is as below:

- Event Counter continuously cleared: bit1 = 1 "INPUTCTL" peripheral register (P04D).
- Interrupt on event (CP6) enable: bit7 = 1 "CP65" peripheral register(P04C)
- 16-bit virtual offset timer is captured into the circular buffer.

*The first "Double Event Compare" command in the reduced COMMAND/DEFINITION area, is used only to set the bit NEXDEF to indicate to the PACT that the next entry is a definition.

As this command is not used (it is only necessary to set the bit NEXDEF), the lower 16-bits of the 32-bits word are free and are used as a standard register file word to store the "Ti-2" tooth period as is described later.

*The second entry is an "Offset Timer/Event Counter" definition programmed as defined above.

So, at each falling edge of the Input CP6, the virtual offset timer is captured in the circular buffer and an interrupt is generated. The value captured in the circular buffer corresponds to the period of the tooth signal acting for the engine speed. The role of the CP6 interrupt subroutine is to find the missing teeth, in order to catch the engine synchronization.

The TMS370 source code to initialize the control registers and the COMMAND/DEFINITION area (during missing teeth research) is as follows:

```
;Control registers initialization

MOV     #0FFH,OP1OP8        ;OP1-OP8 = FF
MOV     #19H,RESOL          ;Resolution = 2µS. Disable cmdzone
MOV     #03H,GLOBAL         ;Disable Watchdog. ITs level 1
MOV     #0H,BUFPT           ;Circular Buffer Pointer Init
MOV     #0EFH,STARTCONT     ;Start Control Address
MOV     #0E8H,ENDCONT       ;End Control Address
MOV     #06H,INPUTCTL       ;CP6 event only,evt count=0,int cmdzone enable
MOV     #01H,CP21           ;CP1 falling edge & interruption disable
MOV     #90H,CP65           ;CP6 falling edge & interruption enable
MOV     #002H,INT3CTL       ;INT3 falling edge enable Level 2.
MOV     #000H,ADSTAT        ;A to D Interrupt disable.
MOVW    #0H,BUFLSW          ;Clear Circular Buffer
MOVW    #0H,BUFMSW          ;1*32-bit Word
```

Command/Definition Area Initialisation

DEC → "Double Event Compare" command
CC → "Conditional Compare" command
SC → "Standard Compare" command

```
MOVW    #00000H,COM0LSW     ;Next def cmd.(DEC)
MOVW    #01101H,COM0MSW     ;Reset DT on event=0.
MOVW    #00001H,COM1LSW     ;Offset Timer. 16-b capture.
MOVW    #07334H,COM1MSW     ;Reset DT on 115. Evt max=115
```

4.2.3 Missing Teeth Detection

As described just below, during the missing teeth research, the tooth period is captured every falling edge of the engine speed signal in the circular buffer. To detect the missing teeth, it is necessary to compare the last three captured values and to verify the following equation:

$$Ti - 1 > Ti + Ti - 2$$

where Ti – 2, Ti – 1, Ti are the last three captured values of the tooth signal period (see Figure 9 below).

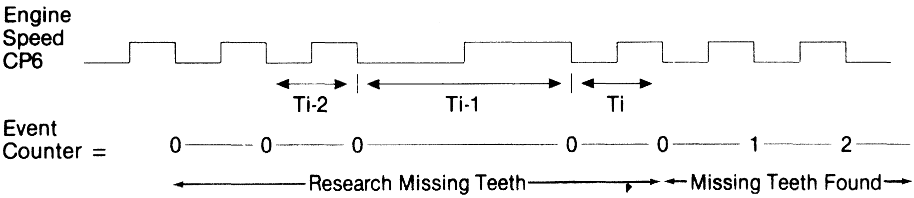


Figure 9: Missing teeth research

The verification of the relation Ti – 1 > Ti + Ti – 2 is done during each interrupt generated by the detection of the falling edge of the Input CP6 (enabled during this phase of the program). So on each falling edge on CP6 the following actions are taken:

- capture the 16-bit offset timer on the top of the circular buffer
- generate an interrupt (The interrupt subroutine algorithm is shown later in figure 10).

As the buffer pointer BUFPT (P043) is reset each interrupt, the 16-bit offset timer is always captured in the top of the circular buffer. That means at the beginning of the CP6 interrupt subroutine, the content of the circular buffer is the following:

		BUFPT	
Address FØ-F3	Ti – 1	Ti	One 32-bits Word Circular Buffer
Address EC-EF		Ti – 2	Start Command/Definition Area Command Ø (COMØ)

and at the end of the subroutine the content of the circular buffer is:

		BUFPT	
Address FØ-F3	Ti	Wait Next Capt.	One 32-bits Word Circular Buffer
Address EC-EF		Ti – 1	Start Command/Definition Area Command Ø (COMØ)

The 16-bit register which stores the value Ti-2 can be the 16-bit LSB of the first command word (COM0) in the COMMAND/DEFINITION area because this first command word during the initialization phase is only used to set on the NEXTDEF bit (bit16 = 1 "COM0").This is to indicate that the next entry in the COMMAND/DEFINITION area is the 16-bit "Offset Timer/Event Counter" definition.

The TMS370 source code for the missing teeth detection is as follows:

First Missing Teeth Research

Research Missing Tooth : $T(i-2) + T(i) < T(i-1)$
: Set Flag in Status if Synchro found

Ti – 1	Ti	Circular Buffer (BUFMSW – BUFLSW)
	Ti – 2	COMOMSW – COMOLSW

```
LOOKSYNC:
          MOV     #0H,BUFPT               ;Reset circular buffer pointer
          ADD     BUFLSW,COMOLSW          ;COMOLSW = T(i) + T(i-2)
          ADC     BUFLSW-1,COMOLSW-1      ;
          CMP     BUFMSW-1,COMOLSW-1      ;T(i-2)+T(i) < T(i-1)?
          JLO     SYNCOK
          JNE     NOSYNC
          CMP     BUFMSW,COMOLSW
          JLO     SYNCOK;
NOSYNC:
          MOVW    BUFMSW,COMOLSW          ;Ti-2 = Ti-1
          MOVW    BUFLSW,BUFMSW           ;Ti-1 = Ti
          RTS
SYNCOK:   OR      #01H,STATUS             ;Missing Teeth Recognized
          RTS
```

4.2.4 TOOTH INTERRUPT (CP6) (Enabled during the initialization phase)
The tooth interrupt subroutine is enabled only during the initialization phase until the missing teeth have been found. As this application is a multi-point sequential fuel injection, a reference mark signal is needed to indicate the number of the engine revolution in the engine cycle. (drive injector 1 or injector 4)

In the 4 cylinder engine cycle, the normal firing order, and therefore the normal fuel injection order, is the following:

1) Cylinder 1	first engine revolution	one engine cycle
2) Cylinder 3	(from 0 to 57)	
3) Cylinder 4	second engine revolution	
4) Cylinder 2	(from 58 to 115)	

That is why, before the missing teeth research, the tooth interrupt subroutine must check the reference mark signal (Camshaft mark) by polling the input INT3. (The INT3 input is programmed in disabled interrupt mode.) This INT3 input gives a "top synchro" each engine cycle.

The flowchart of the tooth interrupt subroutine is shown in figure 10 below: (this interrupt is enabled only during the initialization phase):

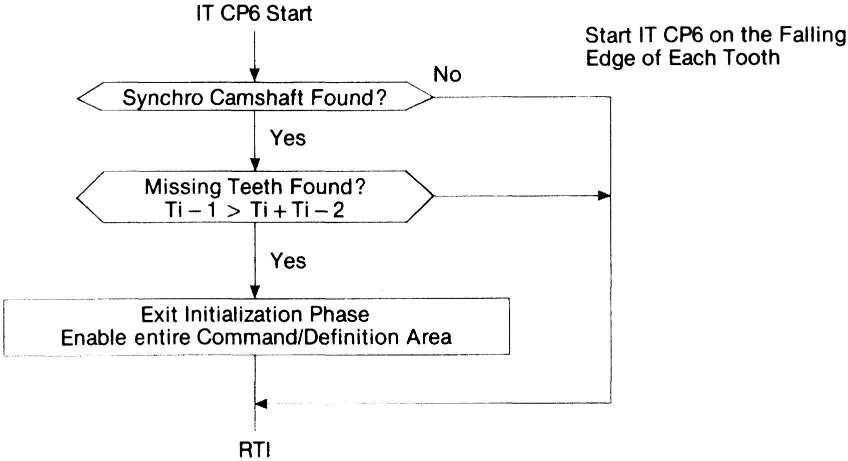


Figure 10: Flowchart of Missing teeth research (IT. CP6)

4.2.6 PACT context switch

When the missing teeth have been found, the engine control program can start (ignition, injection... enabled), so the COMMAND/DEFINITION area must be modified. To perform this, the following actions are taken during the last CP6 interrupt (PACT context switching):

1. Disable COMMAND/DEFINITION area.
   → Reset bit 5 "RESOL" peripheral register (PØ4Ø)
2. Disable interrupt on each tooth.
   → Reset bit 7 "CP65" peripheral register (PØ4C)
3. Enable the event counter to operate.
   → Reset bit 1 "INPUTCTL" peripheral register (PØ4D).
4. Change end address COMMANDS/DEFINITIONS area.
   → Store 98H in "ENDCONT" peripheral register (PØ42).
   → x Commands and Definitions words available.
5. Replace the two first commands.
6. Clear interrupt control flags.
   → Reset "CTLFFG" peripheral register (PØ49).
7. Enable COMMANDS/DEFINITIONS area
   → Set bit 5 "RESOL" peripheral register (PØ40).

The TMS370 source code for the PACT context switching is as follows:

```
AND     #0DFH,RESOL          ;Stop cmdzone.
MOV     #010H,CP65           ;CP6 Interrupt Disable.
MOV     #005H,INPUTCTL       ;Event Counter Start.
MOV     #098H,ENDCONT        ;End Control Address(+20 commands).
MOV     #00H,BUFPT           ;Reset Circular Buffer Pointer

MOVW    #00000H,COM0LSW      ;Reset 1st command word
MOVW    #00000H,COM0MSW      ;Not used

MOVW    #00000H,COM1LSW      ;End Injection 1.(SC)(OP3).
MOVW    #00808H,COM1MSW      ;Reset OP3.Init 0.

MOV     #00H,CTLFLG          ;Clear Interrupt Control Flags.
OR      #20H,RESOL           ;Enable Control Zone Area
```

To stop the COMMAND/DEFINITION area during this PACT, context switching is not mandatory, but the programmer has to take care during this exchange to avoid some bad command interpretations during the switching.

4.3 Engine control program (Synchronized phase)

4.3.1 Interrupt tasks

Once the missing teeth have been found, the tooth signal (engine speed) increments the event counter from:

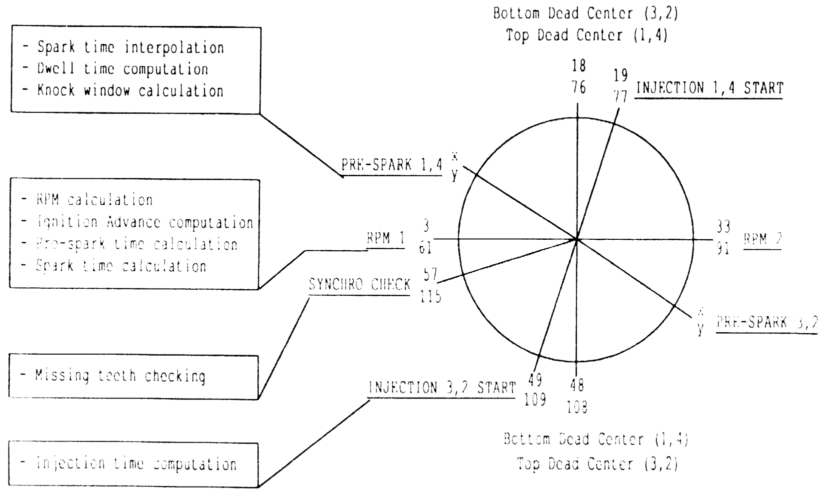
- 0 to 57 for the first engine revolution
- 58 to 115 for the second engine revolution.

When the event counter reaches the maximum value (115) programmed in the "Offset Timer/Event Counter" definition (COM7, cf above), it is cleared. So each tooth has a number and corresponds to an engine angle (6 degrees per tooth), the software tasks can be executed in synchronization with the engine cycle to a very precise angle. The "Conditional Compare" command and the "Double Event Compare" command make this possible by generating interrupts and/or actions (set/reset an output...) on an event and time (interpolation between two teeth) or an event.

Figure 11 shows the different interrupt tasks executed in this application. For example, the fuel injection of cylinder 1 starts at event 19 which corresponds to 6 degrees after the Top Dead Center (TDC) of this cylinder. The fuel injection of cylinder 3 starts at event 49, for cylinder 4 it starts at event 77 and for cylinder 2 it starts at event 109. Then, the event counter is cleared when it reaches the maximum value 115 and the fuel injection cycle restarts. The rest of this application note describes the different interrupt tasks as:

- synchronization check
- RPM calculation
- Pre-Spark interrupt (ignition)
- Injection interrupt
- etc.

FIGURE 11 : INTERRUPT TASKS DIAGRAM



As shown in figure 11, all the interrupt subroutines are perfectly synchronized to the engine angle which allows them to execute a task at the most appropriate time.

4.3.2 Missing teeth checking interrupt

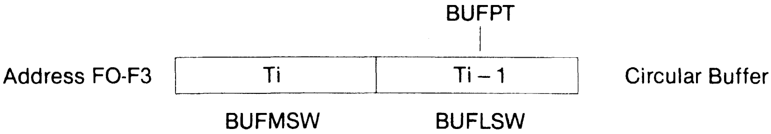
In order to stay perfectly synchronized to the engine cycle (TDC), the missing teeth must be checked every engine round. This is mandatory to avoid a possible desynchronization which could be destructive for the engine (for example an ignition at a bad engine angle). As the falling edge following the missing teeth corresponds to event "57" for the first round and "115" for the second round, a "Double Event Compare" command is programmed in the COMMAND/DEFINITION area and generates an interrupt when the event counter matches the compared values 57 or 115 (figure 12).

0 5	0 2	115	57	Command 8 (COM8)
D31		Event2	Event1	D0

Where:

D0-7	=	57	First engine revolution missing teeth check value
D8-15	=	115	Second engine revolution missing teeth check value
D17	=	1	Interrupt enable when E.C. matches "57" (Event1)
D23	=	0	Must be zero for this command to be valid
D24	=	1	Must be one for this command to be valid
D26	=	1	Interrupt enable when E.C. matches "115" (Event2)

After each engine revolution, the circular buffer pointer "BUFPT" is cleared (at the end of this interrupt), so the last two 16-bit virtual timer values captured in the circular buffer are stored in the following way:



So, the missing teeth checking interrupt subroutine only has to verify the equation Ti – 1 > 2*Ti. If the relation is true, the synchronization to the engine cycle is good, the program continues. If the relation is false, the program is immediately reset and restarts in the initialization phase to find the missing teeth again.

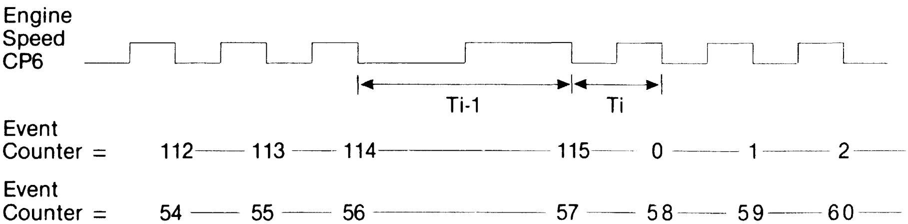


Figure 12: Missing teeth checking

The missing teeth checking interrupt flowchart is shown in figure 13:

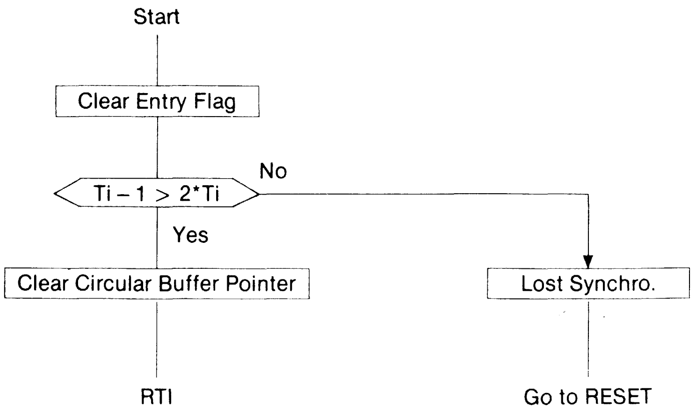


Figure 13: Missing teeth checking interrupt Flowchart

The TMS370 source code for the missing teeth checking interrupt subroutine is as follows:

Missing Teeth Checking Interrupt

```
;Synchro check : T57 > 2* T56
;               T115 > 2* T114                      if fail reset

SYNCHRO:
          MOV     #0F7H,CTLFLG              ;Clear Entry 3 Flag.
          CLRC                              ;2* BUFLSW → BUFLSW
          RLC     BUFLSW                    ;
          RLC     BUFLSW-1                  ;
          CMP     BUFMSW-1,BUFLSW-1         ;Compare T(i) with 2* T(i-1)
          JLO     ENDSYNC                   ;
          JNE     ERROR                     ;
LSBTEST:  CMP     BUFMSW,BUFLSW             ;
          JLO     ENDSYNC                   ;
ERROR:
          BR      START1                    ;Synchronization lost: RESET
ENDSYNC:
          RTI
```

4.3.3 Ignition control

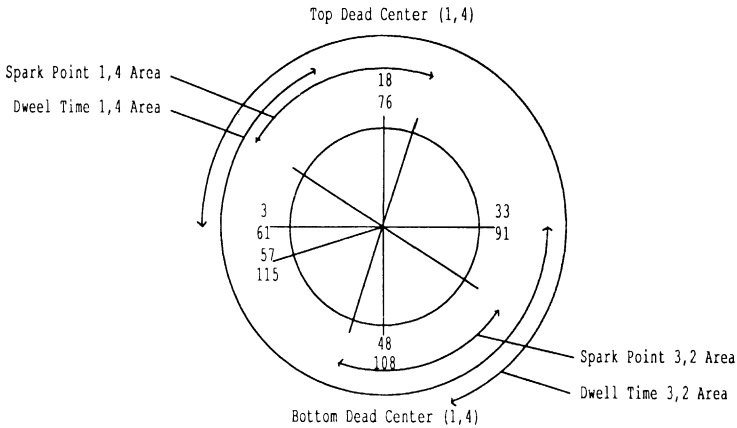
There are several ways to control the ignition in the engine control system. The one discussed in this application report describes a semi-static ignition, where one PACT output drives the spark plugs for cylinders 1 and 4 and another drives the spark plugs for cylinders 3 and 2.

In ignition control, two timings must be very precise: the spark point and the dwell time. The spark time must be very exact to generate the spark at the right engine advance angle, and the dwell time to avoid either an under-charging of the ignition coil, or an over-heating of the ignition coil.

As shown in figure 13, the spark point area may vary from +10 degrees to −60 degrees from the Top Dead Center (TDC) of the referenced cylinder. That means that the spark time takes place between:

Tooth # 8 to 19 for cylinder 1
Tooth # 38 to 49 for cylinder 3
Tooth # 66 to 77 for cylinder 4
Tooth # 98 to 109 for cylinder 2

Figure 14 : SEMI-STATIC IGNITION TIMING



The sparks for the cylinders arrive sequentially. In order to optimize the size of the COMMAND/DEFINITION area, only one command is used to generate the spark point for each cylinder. The content of this command is updated each half engine turn at the moment of the engine advance angle computation. This computation takes place every half engine turn during the RPM interrupt subroutine described below.

The angle covered by the dwell time depends on the engine speed. The dwell time is a constant time depending on the ignition coil features. However, this constant time may be adjusted (according to the engine control system) to take into account variations of battery voltage and to have better control of the saturation current in the ignition coil. In this application the dwell time is a constant of 5 ms. As an overlapping of the dwells of cylinders 1, 4 and cylinders 3, 2 (during high engine speed) is possible, it is easier, for the Dwell control, to use two different commands to drive the two ignition coil outputs (OP1,OP2).

As shown in figures 14 and 16, the start of the dwell is done by a "Standard Compare" command which sets the ignition coil output (OP1 PACT output for cylinders 1 and 4, OP2 PACT output for cylinders 3 and 2) and the spark is generated by a "Conditional Compare" command which resets the same PACT outputs.

To determine the spark time and hence the dwell start which is 5 ms before the spark time, the engine advance angle must be computed previously. This advance computation must be done as near as possible to the referred cylinder spark. In this application, this advance computation takes place during the RPM 1 or RPM 2 interrupt subroutine generated at the beginning of the quarter engine revolution in which the spark is located (figure 11).

4.3.4 RPM interrupt subroutine
The RPM subroutine has three main tasks to execute (figure 11):

- RPM computation needed for the advance every half revolution.
- Ignition advance computation.
- Tooth pre-spark time and spark time computation and load commands in the COMMAND/DEFINITION area.

The interrupt of the RPM subroutine is programmed by two "Double Event Compare" commands in the COMMAND/DEFINITION area. These two commands allow the event counter to be compared to four different values namely 3, 33, 61, 91. The content of these commands is described below:

5	2	33	3	Command 13 (COM13)
D31		Event 2	Event 1 D0	

5	2	91	61	Command 21 (COM21)
D31		Event 2	Event 1 DO	

Where:

- The values 3, 33, 61, 91 permit generation of an interrupt when the event counter (E.C.) matches these values.
- D17 = 1 Interrupt enable when E.C. matches event 1
- D23 = 0 Must be zero for this command to be valid
- D24 = 0 Must be one for this command to be valid
- D26 = 1 Interrupt enable when E.C matches event 2

The addresses of these two commands in the COMMAND/DEFINITION area are equivalent modulo 8. This causes both commands to generate the same interrupt vector and branch at the same interrupt subroutine as explained in the PACT specification.

The different tasks executed by the RPM interrupt subroutine are listed below:

A. Average engine speed computation
The average is processed every half engine revolution. This is done by capturing the 20-bit Default Timer (using CP1) every half revolution and then computing the difference since the last capture.

B. Ignition advance computation
The ignition advance computation is made using a table lookup. The table correlates advance angle according to the engine speed and the intake manifold pressure. The subroutine described in the listing (ADVCAL) performs a triple interpolation. The result supplied by this routine is the advance angle given by a byte which has an accuracy of 1/16 of tooth (0,375 degrees).

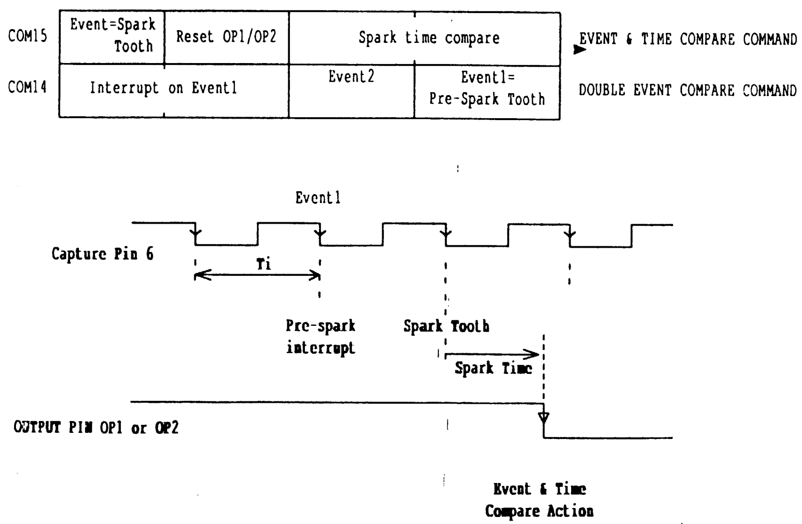
The format of the advance angle is:

# Tooth	# 1/16 tooth	ADVANCE
← ms nibble →	← ls nibble →	

The intake manifold pressure is obtained from the Analog/digital converter, and the conversion is done during the RPM interrupt subroutine. So, this convertion is synchronized to the engine cycle (90 degrees before TDC). This allows for possible intake manifold pressure disturbances.

C. Pre-spark tooth and spark time computation
The advance value is represented in sixteenths of a tooth. To drive an ignition output with this accuracy, it is necessary to convert the engine angle into time by making an interpolation using the number of fractional teeth and the period of a single tooth signal. Since the advance is calculated at the RPM point (figure 11), which is 30 to 100 degrees before the spark time, the tooth period may vary and thus alter the result of the interpolation. This is why a pre-spark interrupt is programmed in order to compute this interpolation just before the spark time (figure 15).

Figure 15 : SPARK TIME GENERATION



A "Double Event Compare" command (COM14) is used to program the pre-spark interrupt where the value of the event 1 of the command corresponds to the integer value of the ignition advance minus one (spark tooth minus one). This command is updated during each RPM interrupt subroutine.

Figure 16 shows the flowchart of the RPM subroutine:

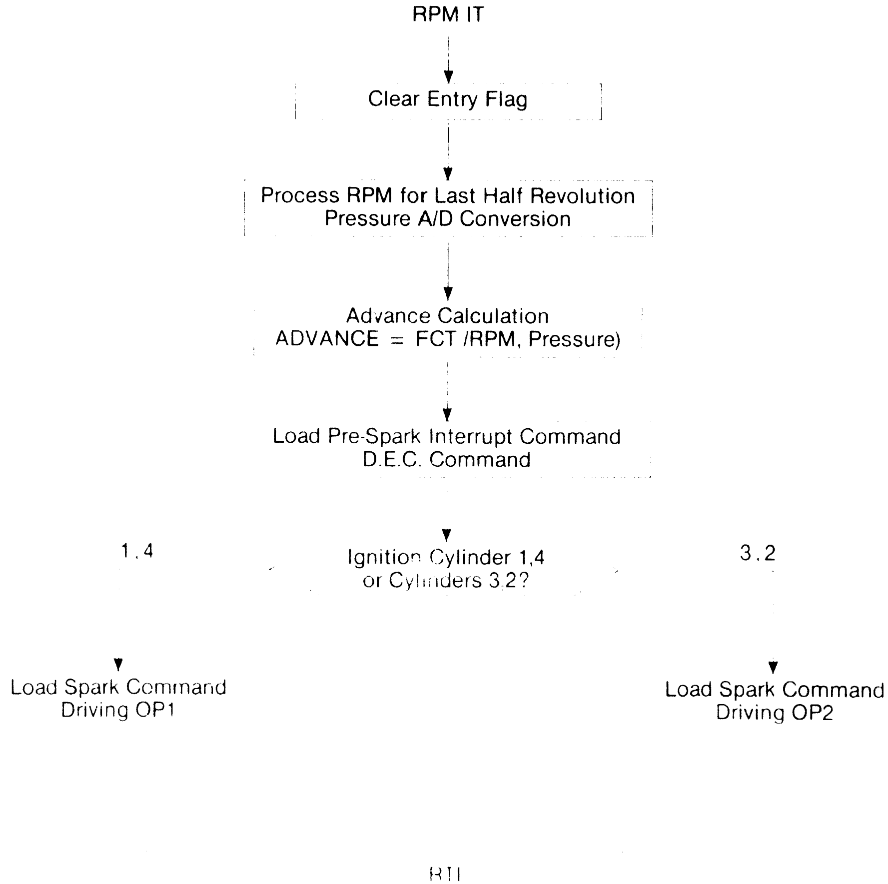


Figure 16: RPM interrupt subroutine flowchart.

4.3.5 Spark time command

As is shown in figure 15, a "Conditional Compare" command (COM15), combining Event + Time drives the ignition outputs OP1 or OP2. The number of the spark tooth event is loaded during the RPM interrupt and the spark time compare value during the pre-spark interrupt subroutine (result of the conversion angle/time). This value is compared to the Offset Timer (COM7) defined above in the COMMAND/DEFINITION area. The same command drives both outputs OP1 and OP2. The output selected is updated during each RPM interrupt.

4.3.6 Dwell command

The dwell time, in this application, is a 5 ms constant time. Two "Standard Compare" commands (COM5, COM6) which refer to the free running Default Timer, set the ignition outputs OP1 and OP2 (figure 17). Since the dwell time occurs before the spark time, the dwell time command has to be previously programmed. This is done during the pre-spark interrupt subroutine.

Depending on the engine speed, in order to take care of the fast engine speed variation, the Dwell command can be programmed either a half engine revolution or a whole revolution before the spark time of the referred cylinder.

- For an engine speed greater than 2000 rpm, the command is loaded one engine round before. For example, the dwell command of cylinder 4 is loaded during the pre-spark interrupt subroutine of cylinder 1 and reciprocally. In this case, the equation and computation to get the compare value to load in the "Standard Compare" command is the following:

```
Dwell time compare value = Default timer
                           + one engine round period
                           + one tooth period
                           + time interpolation spark time 4
                           - 5 msec.
```

- For an engine speed lower than 2000 rpm, the command is loaded a half engine revolution before. In this case, the Dwell command for cylinder 4 is loaded during the pre-spark subroutine of cylinder 3. The equation to compute and get the compare value from is the following:

```
Dwell time compare value = Default timer
                           + half engine round period
                           + one tooth period
                           + time interpolation spark time 4
                           - 5 msec.
```

The TMS370 source code for this last case is as follows:

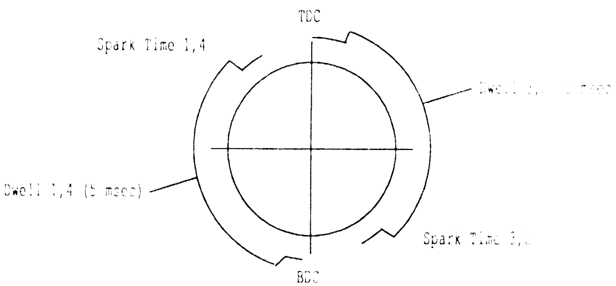
Dwell time computation

```
- if engine speed > 2000 rpm
    Dwell time = DT + 2 RPM + tooth + spark time -5 msec
    Load Dwell time one turn before
- if engine speed < 2000 rpm
    Dwell time = DT + RPM + tooth + spark time -5 msec
    Load Dwell time 1/2 turn before

DWEL
        MOVW    CP1RLSW,TEMP            ;Read Last capture D.T.
        ADD     RPM,TEMP                ;Time = D.T. + RPM
        ADC     RPM-1,TEMP-1
        SUB     #0C4H,TEMP
        SBB     #09H,TEMP-1             ;Time = RPM + D.T. -9C4H (5 msec.)
        ADD     COM15LSW,TEMP           ;Time = time + tooth inter.
        ADC     COM15LSW-1,TEMP-1
        ADD     CAPT16,TEMP             ;Time = time + tooth period
        ADC     CAPT16-1,TEMP-1
```

Figure 17 : DWELL TIME GENERATION

COM5	Set OP1	Dwell Time 1,4	STANDARD COMPARE COMMAND
COM6	Set OP2	Dwell Time 3,2	STANDARD COMPARE COMMAND



4.3.7 Injection control

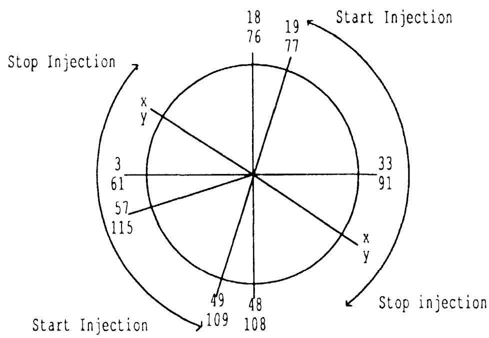
A four cylinder sequential fuel injection system requires four distinct injectors and four independent PACT outputs as opposed to a full group fuel injection system where only one output drives the 4 injectors). A lot of different solutions can be used to drive the injectors and they depend on the chosen engine control strategy.

In this application, the start of the fuel injection pulse is synchronized to an engine angle (6 degrees after the TDC). Figure 18 shows which engine angle (the tooth number) has been chosen for the different cylinders:

- Tooth # 19 → Start injection cylinder 1 (6 degrees after cylinder 4 TDC)
- Tooth # 109 → Start injection cylinder 2 (6 degrees after cylinder 3 TDC)
- Tooth # 49 → Start injection cylinder 3 (6 degrees after cylinder 2 TDC)
- Tooth # 77 → Start injection cylinder 4 (6 degrees after cylinder 1 TDC)

Figure 18 : INJECTION COMMANDS

Set OPx Interrupt on Event	Event2= X	Event1= Start injection	DOUBLE EVENT COMPARE COMMAND
Reset OPx	Injection Time Compare		STANDARD TIME COMPARE COMMAND



Four "Double Event Compare" commands are used to start the fuel injection (one per cylinder). These commands execute the following functions (figure 18):

- Set the dedicated injector output.
- Generate an interrupt to execute the injection subroutine.

So four different injection interrupts are generated. The tasks of these interrupt subroutines are to compute the injection time and to load the result in the "Standard Compare" commands (COM1, COM2, COM3, COM4)(figure 18). The injection time compare value is equal to the computed injection time added to the Default timer value (obtained by the CP1 input capture dedicated register at the start of the injection pulse). So, when the Default Timer matches this injection Time Compare value, the "Standard Compare" command resets the dedicated injector output.

As said before, in this application, four "Double Event Compare" commands are used to start the fuel injection of the four cylinders. In the case where the COMMAND/DEFINITION area must be optimized, only one "Double Event Compare" command can be programmed. Then, during the injection interrupt subroutine the start injection event for the next cylinder has to be updated (this each half engine revolution). On the other hand, for high engine speed and strong engine load, an overlap of the different fuel injection times for the four cylinders is possible. So four "Standard Compare" commands are necessary to close the injectors.

Figure 19 gives the flowchart of an injection interrupt subroutine:

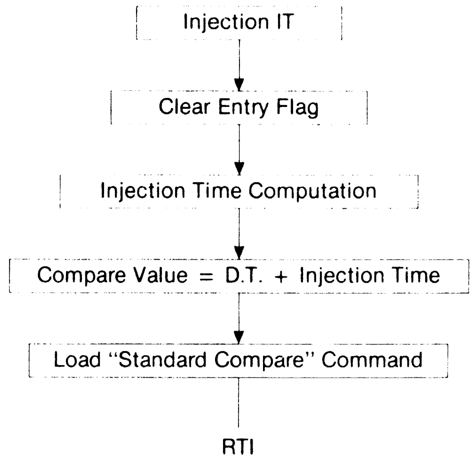


Figure 19: Injection interrupt subroutine flowchart

The TMS370 source code for the injector 1 interrupt subroutine is as follows:

```
INJ1
    MOV     #0,COM1MSW - 1          ;Disable end injection cmd
    MOV     #0FBH,CTLFLG            ;Clear Entry 2 Flag.
    CALL    TINJEC                  ;Time Injection computation
    ADD     TINJ,CP1RLSW
    ADC     TINJ-1,CP1RLSW-1        ;COM1LSW = CP1LSW + TINJ
    MOVW    CP1RLSW,COM1LSW         ;Load command
    MOV     #8,COM1MSW-1            ;Re-enable command
    RTI
```

The disabling of the end injection command (COM1) avoids this command resetting the injector output before being updated with the new compare value computed in this interrupt subroutine.

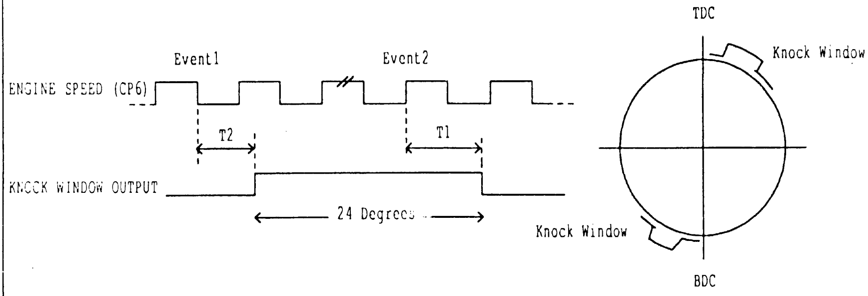
4.3.8 "Knock window" generation

In order to extract the knocking information from the signal provided by the knock sensor, a measuring angle window (knock window) synchronized to the crankshaft is necessary. This window is generally angularly constant (9 to 33 degrees after the Top Dead Center in this application) but can be modified according to the Engine Control strategy.

As is shown in figure 20, the knock window appears after each Top Dead Center of each cylinder, so each half engine round.

Figure 20 : KNOCK WINDOW COMMANDS

COM17	Event1=(Set Knock Tooth)	Set OP7	Angle Adjustment Time (T1)	EVENT & TIME COMMAND TIME
COM18	Event2=(Reset Knock Tooth)	Reset OP7	Angle Adjustment Time (T2)	EVENT & TIME COMPARE COMMAND



Two "Conditional Compare" commands (COM17, COM18) are used to generate the knock window. The first one sets the PACT output OP7, the second one resets this same output. These commands refer to the offset timer defined previously in the COMMAND/DEFINITION area. In this application the values of the "Conditional Compare" commands are updated every half engine round during the RPM and the pre-spark interrupt subroutines. In this application the constant angular knock window goes from 9 to 33 degrees after the Top Dead Center of the referred cylinder.

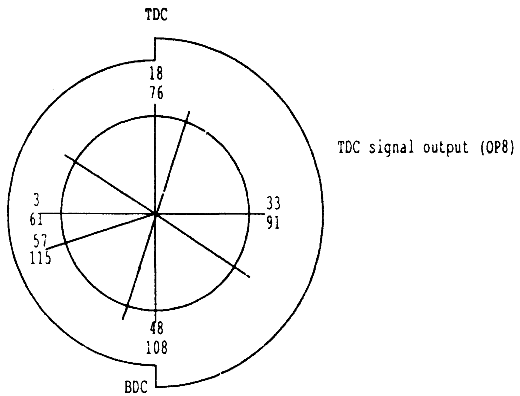
4.3.9 Top Dead Center (TDC) signal generation

For diagnosis and adjustment, or to drive the engine speedometer of a dashboard, it is very useful to output a signal synchronized with the Top Dead Center (TDC) and the Bottom Dead Center of cylinders. This makes available the engine speed signal synchronized to the TDC. A "Double Event Compare" command (COM16) is used to set the PACT output OP8 at the cylinder 1,4 TDC (event 1 = 18 or 76) and to reset the same output at cylinder 1,4 BDC (event 2 = 48, 108). The event compare values are updated each engine revolution during the RPM interrupt subroutine.

Figure 21 shows the Top Dead Center signal generation.

Figure 21 : TDC SIGNAL OUTPUT GENERATION

COM16	Set on Event1/Reset on Event2	Event2=48	Event1=18	DOUBLE EVENT COMPARE COMMAND
COM16	Set on Event1/Reset on Event2	Event2=106	Event1=76	DOUBLE EVENT COMPARE COMMAND



The programming of the "Double Event Compare" command is the following:

Control bits	event 2	event 1	Command 16 COM16
D31		D0	

D0 – 7 = Event1 = TDC1 = 18 during first engine revolution
Event1 = TDC4 = 76 during second engine revolution

D8-15 = Event2 = BDC1 = 48 during first engine revolution
Event2 = BDC4 = 108 during second engine revolution

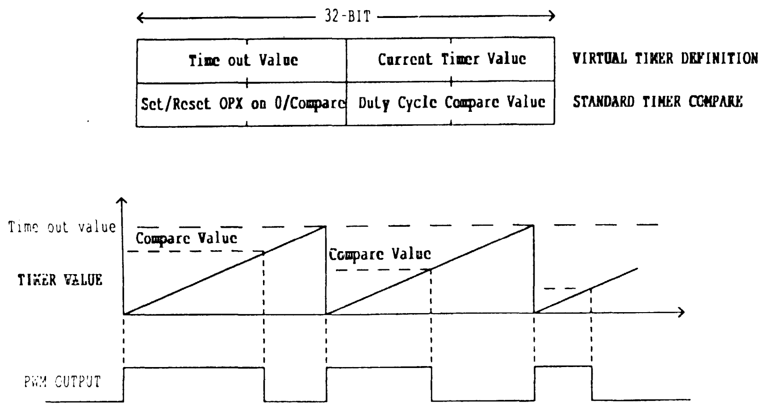
D18-20 = 111 → OP8 output selected
D21 = 1 → Set output when Event Counter matches EVENT 1
D23 = 0 Must be zero for this command to be valid
D24 = 1 Must be one for this command to be valid
D25 = 1 Invert action on selected output pin when event counter matches EVENT 2

4.3.10 PWM signal generation

In engine control systems, some actuators for functions like EGR, Idle speed, fuel pump, throttle... need to be driven by PWM signals. As the features of these actuators differ most of the time, the PWM signal frequencies need to be specific to each actuator. The number of the 16-bit virtual timer being programmable, the PACT module can provide as many PWM signals with independent frequencies as the engine control system requires.

The PWM signal generations are not used in this application, but the description of their programming in the COMMAND/DEFINITION area is described just below and shows in figure 22.

Figure 22 : PWM GENERATION



To generate a PWM signal, two commands are necessary: a 16-bit "virtual timer" definition and a "Standard Compare" command (figure 22).

- The "Virtual Timer" incremented every resolution, defines the PWM signal frequency. This frequency is fixed by the value loaded in the maximum virtual timer value.
- The "Standard Compare" command, which refers to the previous virtual timer defined, controls the duty cycle of the PWM signal and drives the selected output. This duty cycle is programmed by loading its value in the duty cycle compare value.

The control bits in this command allow:

- To select an output pin whose state will be modified when the compare value is matched by the referred virtual timer.
- To set (or reset) the pin defined when the compare value is matched by the referred timer.
- To reset (or set) the same pin defined when the referred timer is reset to zero (when it reaches its maximum virtual timer value).

The programming of the "Virtual Timer" to obtain 50 Hz is as follows:

- The previous command must have D16 = NexDef = 1
- D0 must be zero for this command to be valid
- PWM frequency 50 HZ → period = max value* resolution
  → 20000μS = 2710H* 2μS

→ max value = 0010 0111 0001 0000b = 2710H

→ D19 = Prec	= 1	
D22-21-20	= 001	
D31-23	= 001110001	
D18 = ENable	= 1	→ D31-16 = 0011 1000 1001 1100
D17 = I0	= 0	
D16	= 0	= 389Ch

The programming of the "Standard Compare" command associated with the above "Virtual Timer" is as follows:

- To have a 75% PWM, the compare value must be 15000μ = 1D4Ch* 2μS

→ compare value = 0001 1101 0100 1100b

- The progamming of the other bits are:

	D16 = NexDef	= 0	
	D17 = Int Comp	= 0	
select OP4	D18-20	= 011	
	D21 = Comp act	= 0	
	D22 = step mode	= 0	D31-16 = 0000 1010 0000 1100
	D23-24	= 00	
	D25 = Reset act	= 1	= 0A0Ch
	D26 = Int 0	= 0	
	D27 = Ena Pin	= 1	
	D28-31	= 0000	

The TMS370 source code to program a 100HZ PWM signal and the previous example (50HZ PWM signal) in the COMMAND/DEFINITION area is as follow:

```
MOVW    #00000H,COM22LSW     ;Virtual Timer Definition (100HZ).
MOVW    #09C34H,COM22MSW     ;PACT resolution is 2µS)

MOVW    #09C4H,COM23LSW      ;Standard Compare
MOVW    #00A09H,COM23MSW     ;OP3 Duty Cycle = 50% Next is Definition

MOVW    #00000H,COM24LSW     ;Virtual Timer Definition (50HZ)
MOVW    #0389CH,COM24MSW     ;

MOVW    #01D4CH,COM25LSW     ;Standard Compare
MOVW    #00A0CH,COM25MSW     ;OP4 Duty Cycle = 75%
```

This source code provides two PMW signals with two different frequencies 100HZ and 50HZ on the OP3, OP4 outputs.

5. Engine control program listing

5.1 Engine control program listing contents

A. Definition
- variable definition
- constant definition
- COMMAND/DEFINITION area definition
- PACT peripheral Frame definition
- TMS370 control register definition

B. Program
- Initialization
  - Control register
  - COMMAND/DEFINITION area
  - Enable control zone area

- Background program
  - A/D conversion of AN1 (temperature), AN2, AN3

- Interrupt subroutines
  - Missing teeth research
  - Tooth interrupt for synchro research (only after reset)
  - RPM interrupt
    - RPM computation
    - Tooth spark time computation and load command
    - Tooth pre-spark time computation and load command.
  - Advance calculation (triple interpolation)
  - Pre-spark interrupt
    - interpolation computation for spark time, load command
    - Dwell time computation, load command
    - Knock window computation, load command
  - Spark interrupt
  - Missing teeth checking interrupt
  - Injection interrupt
  - Ignition advance law table
  - Interrupt vector table

5.2 Engine control program listing

```
;*************************************************************************
;*                                                                       *
;*   4-cylinders semi-static ignition, sequential fuel injection program *
;*                                                                       *
;*************************************************************************


;*************REGISTER FILE VARIABLE DEFINITION***************************


TEMPER      .equ    R026      ; Air temperature value
STATUS      .equ    R027      ; Status : bit0=1 ==> Missing tooth find
AN2DATA     .equ    R028      ; AN2 ADC result
AN3DATA     .equ    R029      ; AN3 ADC result
EVENTC      .equ    R02B      ; Event Counter value
ADVANC      .equ    R02C      ; Advance value
PRESS       .equ    R02E      ; Pressure value

CAPT16      .equ    R030      ; 16-bit tooth period
RPM         .equ    R032      ; 16-bit 1/2 engine round period
OLDCP1      .equ    R034      ; 16-bit last CP1 capture during RPM IT.
TEMP        .equ    R036      ; 16-bit Dwell time value
TINJ        .equ    R038      ; 16-bit injection time value

;************* REGISTER FILE VARIABLE FOR TRIPLE INTERPOLATION ****************

DP          .equ    R039      ; Delta pressure Pi+1-Pi
DPX         .equ    R03A      ; Delta pressure Px-Pi
P           .equ    R03B      ; Advance table pressure mark
AV1         .equ    R03C      ; Advance Px+1,Rx    + temporary Reg. DR msb
DR          .equ    R03D      ; Delta Engine speed Ri+1-Ri
AV3         .equ    R03E      ; Advance Px+1,Rx+1 + temporary Reg. DRX msb
DRX         .equ    R03F      ; Delta Engine speed Rx-Ri
N           .equ    R040      ; Advance table engine speed mark
AV2         .equ    R041      ; Advance Px,Rx+1    +temporary Reg. Pi msb
AV0         .equ    R042      ; Advance Px,Rx      +temporary Reg. Pi lsb


;********  Constants definition  ********************************************

C13         .equ   0CH        ; Number of engine speed rows (advance law table)
C9          .equ   08H        ; Number of pressure lines (advance Law table)
```

```
;********* Command/Definition area Definition **********************************

COM27MSW      .equ    R081      ;} Command 27
COM27LSW      .equ    R083
COM26MSW      .equ    R085      ;} Command 26
COM26LSW      .equ    R087
COM25MSW      .equ    R089      ;} Command 25
COM25LSW      .equ    R08B
COM24MSW      .equ    R08D      ;} Command 24
COM24LSW      .equ    R08F
COM23MSW      .equ    R091      ;} Command 23
COM23LSW      .equ    R093
COM22MSW      .equ    R095      ;} Command 22
COM22LSW      .equ    R097
COM21MSW      .equ    R099      ;} Command 21
COM21LSW      .equ    R09B
COM20MSW      .equ    R09D      ;} Command 20
COM20LSW      .equ    R09F
COM19MSW      .equ    R0A1      ;} Command 19
COM19LSW      .equ    R0A3
COM18MSW      .equ    R0A5      ;} Command 18
COM18LSW      .equ    R0A7
COM17MSW      .equ    R0A9      ;} Command 17
COM17LSW      .equ    R0AB
COM16MSW      .equ    R0AD      ;} Command 16
COM16LSW      .equ    R0AF
COM15MSW      .equ    R0B1      ;} Command 15
COM15LSW      .equ    R0B3
COM14MSW      .equ    R0B5      ;} Command 14
COM14LSW      .equ    R0B7
COM13MSW      .equ    R0B9      ;} Command 13
COM13LSW      .equ    R0BB
COM12MSW      .equ    R0BD      ;} Command 12
COM12LSW      .equ    R0BF
COM11MSW      .equ    R0C1      ;} Command 11
COM11LSW      .equ    R0C3
COM10MSW      .equ    R0C5      ;} Command 10
COM10LSW      .equ    R0C7
COM9MSW       .equ    R0C9      ;} Command 9
COM9LSW       .equ    R0CB
COM8MSW       .equ    R0CD      ;} Command 8
COM8LSW       .equ    R0CF
COM7MSW       .equ    R0D1      ;} Command 7
COM7LSW       .equ    R0D3
COM6MSW       .equ    R0D5      ;} Command 6
COM6LSW       .equ    R0D7
COM5MSW       .equ    R0D9      ;} Command 5
COM5LSW       .equ    R0DB
```

```
COM4MSW     .equ   R0DD      ;} Command 4
COM4LSW     .equ   R0DF
COM3MSW     .equ   R0E1      ;} Command 3
COM3LSW     .equ   R0E3
COM2MSW     .equ   R0E5      ;} Command 2
COM2LSW     .equ   R0E7
COM1MSW     .equ   R0E9      ;} Command 1
COM1LSW     .equ   R0EB
COM0MSW     .equ   R0ED      ;} Command 0
COM0LSW     .equ   R0EF
BUFMSW      .equ   R0F1      ;] Circular buffer
BUFLSW      .equ   R0F3      ;] one*32-bits word
CP2RMSW     .equ   R0F5      ;[
CP2RLSW     .equ   R0F7      ;[ CP2 capture register
CP1RMSW     .equ   R0F9      ;]
CP1RLSW     .equ   R0FB      ;] CP1 capture register
TIMEMSW     .equ   R0FD      ;[
TIMELSW     .equ   R0FF      ;[ Event Counter (E.C.) & Default Timer (D.T.) Reg.

;********** HSCA PERIPHERAL FRAME DEFINITION **********************************

GLOBAL      .equ   P04F      ; Global function control register
WDKEY       .equ   P04E      ; Watchdog Key register
INPUTCTL    .equ   P04D      ; Event/Interrupt control register
CP65        .equ   P04C      ; Setup pins CP6 and CP5 register
CP43        .equ   P04B      ; Setup pins CP4 and CP3 register
CP21        .equ   P04A      ; Setup pins CP2 and CP1 register
CTLFLG      .equ   P049      ; Command/Definition entry flags register
OP1OP8      .equ   P048      ; Output pin 1-8 state register
SCITXDT     .equ   P047      ; SCI TX data register
SCIRXD      .equ   P046      ; SCI RX data register
SCICTL      .equ   P045      ; SCI control register
BUFPT       .equ   P043      ; Buffer pointer register
ENDCONT     .equ   P042      ; Command/Definition area end register
STARTCONT   .equ   P041      ; Command/Definition area start register
RESOL       .equ   P040      ; Setup control register

;********** GENERAL PERIPHERAL FRAME DEFINITION *****************************

PORTADATA   .equ   P022      ; Port A data register
PORTADIR    .equ   P023      ; Port A direction register
INT2CTL     .equ   P018      ; INT2 control register
INT3CTL     .equ   P019      ; INT3 control register
ADCTL       .equ   P070      ; ADC control register
ADSTAT      .equ   P071      ; ADC status register
ADDATA      .equ   P072      ; ADC data register


         .list
         .text   7800H
```

```
;********** INITIALIZATION ***************************************************

START:
        MOV     #00FH,PORTADIR          ;Port A bit 0 to 3 output
        MOV     #000H,PORTADATA         ;Port A = 0
START1  DINT
        MOV     #00H,STATUS             ;STATUS Initialization
        MOV     #060H,B                 ;Stack Pointer Initialization
        LDSP

;Control registers initialization

        MOV     #0FFH,OP1OP8            ;OP1-OP8 = FF
        MOV     #19H,RESOL              ;Resolution = 2us. Disable cmdzone
        MOV     #03H,GLOBAL             ;Disable Watchdog. ITs level 1
        MOV     #0H,BUFPT               ;Circular Buffer Pointer Init
        MOV     #0EFH,STARTCONT         ;Start Control Address
        MOV     #0E8H,ENDCONT           ;End Control Address
        MOV     #06H,INPUTCTL           ;CP6 event only,evt count=0,int cmdzone enable
        MOV     #01H,CP21               ;CP1 falling edge & interruption disable
        MOV     #90H,CP65               ;CP6 falling edge & interruption enable
        MOV     #002H,INT2CTL           ;INT2 falling edge disable Level 2.
        MOV     #002H,INT3CTL           ;INT3 falling edge enable Level 2.
        MOV     #000H,ADSTAT            ;A to D Interrupt disable.

        MOVW    #0H,BUFLSW              ;Clear Circular Buffer
        MOVW    #0H,BUFMSW              ; 1 * 32-bit Word

;************* COMMAND/DEFINITION AREA INITIALISATION *************************
;
; DEC ==> "Double Event Compare" command
; CC  ==> "Conditional Compare" command
; SC  ==> "Standart Compare" command
;
;******************************************************************************

        MOVW    #00000H,COM0LSW          ; Next def cmd.(DEC)
        MOVW    #01101H,COM0MSW          ; Reset DT on event=0.

        MOVW    #00001H,COM1LSW          ; Offset Timer. 16-b capture.
        MOVW    #07334H,COM1MSW          ; Reset DT on 115.evt max=115

        MOVW    #00000H,COM2LSW          ; End Injection 2.(SC)(OP4).
        MOVW    #0080CH,COM2MSW          ; Reset OP4.Init 0.

        MOVW    #00000H,COM3LSW          ; End Injection 3.(SC)(OP5).
        MOVW    #00810H,COM3MSW          ; Reset OP5.Init 0.

        MOVW    #00000H,COM4LSW          ; End Injection 4.(SC)(OP6).
        MOVW    #00814H,COM4MSW          ; Reset OP6.Init 0.Nextdef.
```

```
MOVW    #00000H,COM5LSW     ; Dwell time 1,4 (SC) OP1
MOVW    #00820H,COM5MSW     ; Set OP1

MOVW    #00000H,COM6LSW     ; Dwell time 2,3 (SC) OP2
MOVW    #00825H,COM6MSW     ; Set OP2.Next def.

MOVW    #00001H,COM7LSW     ; Offset Timer. 16-b capture.
MOVW    #07324H,COM7MSW     ; Reset evc on 115.

MOVW    #07339H,COM8LSW     ; Missing Teeth 57, 115 (DEC)
MOVW    #00502H,COM8MSW     ;

MOVW    #00013H,COM9LSW     ; Start Injection 1 (DEC)(OP3)
MOVW    #0092AH,COM9MSW     ; Set OP3. IT. Init 19.

MOVW    #0006BH,COM10LSW    ; Start Injection 2 (DEC)(OP4)
MOVW    #0092EH,COM10MSW    ; Set OP4. IT. Init 49.

MOVW    #00031H,COM11LSW    ; Start Injection 3 (DEC)(OP5)
MOVW    #00932H,COM11MSW    ; Set OP5. IT. Init 77.

MOVW    #0004DH,COM12LSW    ; Start Injection 4 (DEC)(OP6)
MOVW    #00936H,COM12MSW    ; Set OP6. IT. Init 107.

MOVW    #02103H,COM13LSW    ; RPM IT 3, 33 (DEC)
MOVW    #00502H,COM13MSW    ; Init 2103

MOVW    #000FFH,COM14LSW    ; Pre spark 1,3,4,2 (DEC) IT
MOVW    #00102H,COM14MSW    ; IT compare 1

MOVW    #00002H,COM15LSW    ; Spark time (CC)
MOVW    #010C0H,COM15MSW    ; Reset OP1 or OP2.Reset on ev+1.

MOVW    #00000H,COM16LSW    ; TDC 18,48,76,106 (DEC)(OP8)
MOVW    #00B3CH,COM16MSW    ; Set on 18,76 and reset on 48,106

MOVW    #00000H,COM17LSW    ; Knock window (CC) Set OP7
MOVW    #000B8H,COM17MSW    ; Init 19

MOVW    #00000H,COM18LSW    ; Knock window (CC) Reset OP7
MOVW    #00098H,COM18MSW    ; Init 21

MOVW    #00000H,COM19LSW    ; Baud rate definition
MOVW    #00000H,COM19MSW    ; (not programmed)

MOVW    #00000H,COM20LSW    ; Dummy Command
MOVW    #00000H,COM20MSW    ;

MOVW    #05B3DH,COM21LSW    ; RPM IT 61, 91 (DEC)
MOVW    #00502H,COM21MSW    ; Same IT as COM13
```

```
        MOVW    #00000H,COM22LSW    ; Dummy Command
        MOVW    #00000H,COM22MSW    ;

        MOVW    #00000H,COM23LSW    ; Dummy Command
        MOVW    #00000H,COM23MSW    ;

        MOVW    #00000H,COM24LSW    ; Dummy Command
        MOVW    #00000H,COM24MSW    ;

        MOVW    #00000H,COM25LSW    ; Dummy Command
        MOVW    #00000H,COM25MSW    ;

        MOVW    #00000H,COM26LSW    ; Dummy Command
        MOVW    #00000H,COM26MSW    ;

        MOVW    #00000H,COM27LSW    ; Dummy Command
        MOVW    #00000H,COM27MSW    ;

        OR      #20H,RESOL          ; Enable Control Zone Area
        MOV     #00H,CTLFLG         ; Clear Interrupt Control Flags.
        EINT                        ; Enable Interruptions

;***************************    MAIN PROGRAM    ********************************
;                               BACKGROUND

MLOOP

        MOV     #041H,ADCTL         ;Start sampling  AN1
        MOV     #0C1H,ADCTL         ;Start Convert AN1
        BTJZ    #04H,ADSTAT,$       ;Wait end convertion
        MOV     ADDATA,TEMPER       ;Read converter

        MOV     #042H,ADCTL         ;Start sampling  AN2
        MOV     #0C2H,ADCTL         ;Start Convert AN2
        BTJZ    #04H,ADSTAT,$       ;Wait end convertion
        MOV     ADDATA,AN2DATA      ;Read converter

        MOV     #043H,ADCTL         ;Start sampling  AN3
        MOV     #0C3H,ADCTL         ;Start Convert AN3
        BTJZ    #04H,ADSTAT,$       ;Wait end convertion
        MOV     ADDATA,AN3DATA      ;Read converter
        BR      MLOOP
```

```
;****************** FIRST MISSING TEETH RESEARCH ****************************
;
; Research missing tooth   :   T(i-2)+T(i) < T(i-1)
;                          :   Set flag in status if synchro found
;
;
;        ---------------------------------
;        !   Ti-1     !     Ti       !  Circular Buffer (BUFMSW-BUFLSW)
;        ---------------------------------
;        !            !    Ti-2      !  COMOMSW-COMOLSW
;        ---------------------------------
;
;
;****************************************************************************


LOOKSYNC:
         MOV       #0H,BUFPT             ; Reset circular buffer pointer
         ADD       BUFLSW,COMOLSW        ; COMOLSW = T(i) + T(i-2)
         ADC       BUFLSW-1,COMOLSW-1    ;
         CMP       BUFMSW-1,COMOLSW-1    ; T(i-2)+T(i) < T(i-1) ?
         JLO       SYNCOK
         JNE       NOSYNC
         CMP       BUFMSW,COMOLSW
         JLO       SYNCOK                ;
NOSYNC:
         MOVW      BUFMSW,COMOLSW        ; Ti-2 = Ti-1
         MOVW      BUFLSW,BUFMSW         ; Ti-1 = Ti
         RTS
SYNCOK:  OR        #01H,STATUS           ; Missing Teeth Recognized
         RTS

;****************** TOOTH INTERRUPT (Before Synchronisation) ****************

TOOTH:
         PUSH      A
         PUSH      B
         AND       #0BFH,CP65             ; Clear CP6 Interrupt Flag.
         BTJZ      #80H,INT3CTL,ENDTOOTH ;Check synchro 2 engine cycle.
         CALL      LOOKSYNC               ; Missing teeth Research
         BTJZ      #01H,STATUS,ENDTOOTH ; Missing teeth found ?

         AND       #0DFH,RESOL           ; Stop cmdzone.
         MOV       #010H,CP65            ; CP6 Interrupt Disable.
         MOV       #005H,INPUTCTL        ; Event Counter Start.
         MOV       #098H,ENDCONT         ; End Control Address(+20 commands).
         MOV       #00H,BUFPT            ; Reset Circular Buffer Pointer

         MOVW      #00000H,COMOLSW       ; Reset 1st command word
         MOVW      #00000H,COMOMSW       ; Not used
```

```
        MOVW    #00000H,COM1LSW     ; End Injection 1.(SC)(OP3).
        MOVW    #00808H,COM1MSW     ; Reset OP3.Init 0.

        MOV     #00H,CTLFLG         ; Clear Interrupt Control Flags.
        OR      #20H,RESOL          ; Enable Control Zone Area
ENDTOOTH:
        POP     B
        POP     A
        RTI

;**************************** RPM  INTERRUPT *********************************
;
; 1- 1/2 RPM computation :    RPM = CP1RLSW - OLDCP1 = 1/2 Engine round period
;                             OLDCP1 = CP1RLSW
; 2- Tooth spark time computation and load command
; 3- Tooth pre-spark time computation and load command
;
;*****************************************************************************

ITRPM
        PUSH    A
        PUSH    B
        MOV     #0BFH,CTLFLG        ;Clear entry 6 flag

        MOV     #040H,ADCTL         ;Start sampling  AN0
        MOV     #0C0H,ADCTL         ;Start Convert AN0
        BTJZ    #04H,ADSTAT,$       ;Wait end convertion
        MOV     ADDATA,PRESS        ;Read converter

        MOV     #0,BUFPT            ;Init buffer pointer
        MOVW    BUFLSW,CAPT16       ;Store tooth period
        MOV     TIMEMSW-1,EVENTC    ;Store event counter
        MOVW    CP1RLSW,RPM         ;Compute RPM period
        SUB     OLDCP1,RPM
        SBB     OLDCP1-1,RPM-1
        MOVW    CP1RLSW,OLDCP1      ;RPM=CP1RLSW-OLDCP1

;******** Pre-spark & Spark tooth number  computation ************************

        CALL    ADVCAL              ;Call advance calculation
        MOV     ADVANC,A            ;Read ignition advance
        SWAP    A
        AND     #0FH,A              ;Read number of teeth
        ADD     A,EVENTC            ;Compute spark tooth
        MOV     EVENTC,COM15MSW-1   ;Load spark tooth
        SUB     #01H,EVENTC         ;pre-spark tooth = spark tooth -1
        MOV     EVENTC,COM14LSW     ;Load pre-spark tooth
        CMP     #57,EVENTC          ;test engine revolution (1st or 2nd)
        JHS     TDC1                ;
        MOVW    #3012H,COM16LSW     ;Load set & reset TDC command
        JMP     TDC2
```

```
TDC1     MOVW     #6A4CH,COM16LSW      ;Load set & reset TDC command

TDC2     SUB      #18,EVENTC           ;Test which 1/2 round
         JLO      IGN14                ;jump to ignition cylinder 1-4
         SUB      #30,EVENTC
         JLO      IGN32                ;jump to ignition cylinder 3-2
         SUB      #30,EVENTC
         JLO      IGN14                ;jump to ignition cylinder 1-4
IGN32
         MOV      #0C6H,COM15MSW       ;Load spark cde.
         MOV      #008H,COM6MSW-1      ;Enable Dwell cde
         JMP      ENDRPM
IGN14    MOV      #0C2H,COM15MSW       ;Load spark cde.
         MOV      #008H,COM5MSW-1      ;Enable Dwell cde

ENDRPM:
         POP      B
         POP      A
         RTI


;******************** ADVANCE CALCULATION SUBROUTINE ***************************
;*********************************************
;*       Triple interpolation               *
;*********************************************
;
;               Ri     Ri+1
;             ---------------
;
;         Pi ! Av0   !  Av2 !
;            !-------!------!                 ADVANC = FCT(Engine Speed, pressure)
;       Pi+1 ! Av1   !  Av3 !
;            ---------------
;
;
;
;    1)    Av0 = Av0 + (Px-Pi)/(Pi+1 - Pi)(Av1-Av0)
;
;    2)    Av2 = Av2 + (Px-Pi)/(Pi+1 - Pi)(Av3-Av2)
;
;    3)    Av0 = Av0 + (Rx-Ri)/(Ri+1 - Ri)(Av2-Av0)
;
;
;*****************************************************************************

;******** DP, DPX, P computation *****************

ADVCAL   MOV    #C9,B
         MOV    PRESS,A
         CMP    PRES(B),A
         JHS    PRESNI        ;Jump if P end of table
         DEC    B
```

```
PRESD    CMP    PRES(B),A
         JHS    PRES2              ;Jump if Px>Pi
         DJNZ   B,PRESD
PRES2    MOV    PRESS,DPX
         MOV    PRES+1(B),A        ;Read Pi+1
         MOV    A,DP
         MOV    PRES(B),A          ;Read Pi
         SUB    RO,DP              ;DP=Pi+1-Pi
         SUB    RO,DPX             ;DPX=PX-Pi
         JMP    PRESF
PRESNI   CLR    DPX
         MOV    #1,DP
PRESF    MOV    B,P

;******** DR, DRX, N Computation *****************

         CMP    #4,CAPT16-1              ;Rpm < 480 rpm ?
         JLO    REG1
         JMP    REGNI

REG1     MOV    #C13,B
         MOV    RPM-1,A
         CMP    REG(B),A
         JLO    REGNI                    ;Jump if R end of table
         JEQ    REGNI
         DEC    B
REGD     CMP    REG(B),A
         JLO    REG2                     ;Jump if Rx<Ri
         JEQ    REGLSB                   ;Need lsb analysis
         DJNZ   B,REGD
         JMP    REG2
REGLSB   MOV    RPM,A
         CMP    REG+1+C13(B),A           ;RPM lsb analysis
         JLO    REG2
         JEQ    REG2
         DEC    B
REG2     MOV    REG+1(B),A               ;Read Ri+1 msb
         MOV    A,AVO-1
         MOV    REG+1+C13+1(B),A         ;Read Ri+1 lsb
         MOV    A,AVO
         MOV    REG(B),A                 ;Read Ri    msb
         MOV    A,DR-1
         MOV    A,DRX-1
         MOV    REG+1+C13(B),A           ;Read Ri    lsb
         MOV    A,DR
         MOV    A,DRX
         SUB    RPM,DRX                  ;DRX=Ri-RX      lsb
         SBB    RPM-1,DRX-1              ;               msb
         SUB    AVO,DR                   ;DR=Ri-Ri+1     lsb
         SBB    AVO-1,DR-1               ;               msb
         RRC    DR-1
```

```
          RRC     DR
          RRC     DR-1
          RRC     DR
          RRC     DR-1
          RRC     DR
          RRC     DR-1
          RRC     DR                      ; 4 right shifts
          RRC     DRX-1
          RRC     DRX
          RRC     DRX-1
          RRC     DRX
          RRC     DRX-1
          RRC     DRX
          RRC     DRX-1
          RRC     DRX
          JMP     REGF

REGNI     CLR     DRX
          MOV     #1,DR
REGF      MOV     B,N

;********** 4 advance values read *************

AVANCE    CLRC
          MPY     #C13+1,P        ;@AV0=TAVANC+C13*P+N
          ADD     N,B
          MOV     TAVANC(B),A
          MOV     A,AV0
          INC     B
          MOV     TAVANC(B),A   ;@AV2=TAVANC+C13*P+N+1
          MOV     A,AV2
          ADD     #C13,B
          MOV     TAVANC(B),A   ;@AV1=TAVANC+C13*(P+1)+N
          MOV     A,AV1
          INC     B
          MOV     TAVANC(B),A   ;@AV3=TAVANC+C13*(P+1)+N+1
          MOV     A,AV3

;********** Advance computation ****************

          SUB     AV0,AV1         ; A0=A0+DPX/DP(A1-A0)
          CLRC
          MPY     AV1,DPX
          DIV     DP,A
          ADD     R0,AV0
          SUB     AV2,AV3         ; A2=A2+DPX/DP(A3-A2)
          CLRC
          MPY     AV3,DPX
          DIV     DP,A
          ADD     R0,AV2
```

```
        SUB   AV0,AV2          ; A0=A0+DRX/DR(A2-A0)
        CLRC
        MPY   AV2,DRX
        CLRC
        DIV   DR,A
        ADD   R0,AV0
        INV   AV0              ; Complement advance
        MOV   AV0,ADVANC
        RTS

;******************** PRE-SPARK INTERRUPT ************************************
;
; 1- Interpollation computation for spark time and load command
; 2- Dwell time computation and load command
; 3- Knock window computation and load command
;
;*****************************************************************************

PRESPARK:
        PUSH       A
        PUSH       B
        MOV        #0DFH,CTLFLG        ;Clear Entry 5 Flag.
        MOV        TIMEMSW-1,EVENTC    ;Store event counter

;************* Tooth interpollation (205 cycles) ***************************
;
;        COM15 = (tooth period) * (number of 1/16 of tooth)
;

        MOVW       BUFLSW,CAPT16       ;Store tooth period
        AND        #0FH,ADVANC
        MPY        ADVANC,BUFLSW
        MOVW       R1,COM15LSW         ;COM15 = capt16*((Advance AND 0Fh)/16)
        MPY        ADVANC,BUFLSW-1
        ADD        R1,COM15LSW-1
        RRC        COM15LSW-1
        RRC        COM15LSW
        RRC        COM15LSW-1
        RRC        COM15LSW
        RRC        COM15LSW-1
        RRC        COM15LSW
        RRC        COM15LSW-1
        RRC        COM15LSW
        AND        #0FH,COM15LSW-1
        CMP        #0H,COM15LSW-1      ;Compare value in cde must be > 0
        JNZ        DWEL
        CMP        #0H,COM15LSW
        JNZ        DWEL
        MOV        #1H,COM15LSW
```

```
;************** Dwell time computation ***********************************
;
; - if engine speed > 2000 rpm
;      Dwell time = DT + 2 RPM + tooth + spark time - 5 msec
;      Load Dwell time one turn before
; - if engine speed < 2000 rpm
;      Dwell time = DT + RPM + tooth + spark time - 5 msec
;      Load Dwell time 1/2 turn before


DWEL
         MOVW     CP1RLSW,TEMP         ; Read Last capture D.T.
         ADD      RPM,TEMP             ; Time = D.T. + RPM
         ADC      RPM-1,TEMP-1
         SUB      #0C4H,TEMP
         SBB      #09H,TEMP-1          ; Time = RPM + D.T. - 9C4H (5 msec.)
         ADD      COM15LSW,TEMP        ; Time = time + tooth inter.
         ADC      COM15LSW-1,TEMP-1
         ADD      CAPT16,TEMP          ; Time = time + tooth periode
         ADC      CAPT16-1,TEMP-1

;**************** Knock window computation (9-33degrees) ************************

         MOVW     CAPT16,COM17LSW      ; Load 1/2 period tooth cmd
         CLRC
         RRC      COM17LSW-1           ; 9 = 6 + 3
         RRC      COM17LSW             ; 33 = 30 + 3
         MOVW     CAPT16,COM18LSW      ; Load 1/2 period tooth cmd
         CLRC
         RRC      COM18LSW-1           ; 9 = 6 + 3
         RRC      COM18LSW             ; 33 = 30 + 3

;**************** Load Dwell time and knock window cmd ************************

         CMP      #1,CAPT16-1          ;Rpm < 2000 r/mn
         JLO      LDWELL

;******** if RPM > 2000

         SUB      #19,EVENTC           ;Check cylinder number
         JLO      DWELL1L              ;jump to dwell cylinder 1
         SUB      #30,EVENTC
         JLO      DWELL3L              ;jump to dwell cylinder 3
         SUB      #30,EVENTC
         JLO      DWELL4L              ;jump to dwell cylinder 4
DWELL2L  MOVW     TEMP,COM5LSW         ;Load dwell time 1,4
         MOV      #107,COM17MSW-1      ;Load start Knock window
         MOV      #111,COM18MSW-1      ;Load end Knock window
         JMP      ENDPSP
```

```
DWELL4L  MOVW   TEMP,COM6LSW         ;Load dwell time 3,2
         MOV    #77,COM17MSW-1       ;Load start Knock window
         MOV    #81,COM18MSW-1       ;Load end Knock window
         JMP    ENDPSP
DWELL3L  MOVW   TEMP,COM5LSW         ;Load dwell time 1,4
         MOV    #49,COM17MSW-1       ;Load start Knock window
         MOV    #53,COM18MSW-1       ;Load end Knock window
         JMP    ENDPSP
DWELL1L  MOVW   TEMP,COM6LSW         ;Load dwell time 3,2
         MOV    #19,COM17MSW-1       ;Load start Knock window
         MOV    #23,COM18MSW-1       ;Load end Knock window
         JMP    ENDPSP

;******** if RPM < 2000

LDWELL   ADD    RPM,TEMP             ;Check cylinder number
         ADC    RPM-1,TEMP-1
         SUB    #19,EVENTC
         JLO    DWELL1               ;jump to dwell cylinder 1
         SUB    #30,EVENTC
         JLO    DWELL3               ;jump to dwell cylinder 3
         SUB    #30,EVENTC
         JLO    DWELL4               ;jump to dwell cylinder 4
DWELL2   MOVW   TEMP,COM6LSW         ;Load dwell time 3,2
         MOV    #107,COM17MSW-1      ;Load start Knock window
         MOV    #111,COM18MSW-1      ;Load end Knock window
         JMP    ENDPSP
DWELL4   MOVW   TEMP,COM5LSW         ;Load dwell time 1,4
         MOV    #77,COM17MSW-1       ;Load start Knock window
         MOV    #81,COM18MSW-1       ;Load end Knock window
         JMP    ENDPSP
DWELL3   MOVW   TEMP,COM6LSW         ;Load dwell time 3,2
         MOV    #49,COM17MSW-1       ;Load start Knock window
         MOV    #53,COM18MSW-1       ;Load end Knock window
         JMP    ENDPSP
DWELL1   MOVW   TEMP,COM5LSW         ;Load dwell time 1,4
         MOV    #19,COM17MSW-1       ;Load start Knock window
         MOV    #23,COM18MSW-1       ;Load end Knock window
ENDPSP
         POP    B
         POP    A
         RTI

;******************* SPARK INTERRUPT ****************************************

SPARK
         MOV    #0EFH,CTLFLG         ;Clear Entry 4 Flag.
         RTI
```

```
;****************** MISSING TEETH CHECKING INTERRUPT ***************************
;
;  Synchro check : T57  > 2 * T56
;                  T115 > 2 * T114          if fail reset
;
;

SYNCHRO:
         MOV       #0F7H,CTLFLG          ;Clear Entry 3 Flag.
         CLRC                            ; 2 * BUFLSW --> BUFLSW
         RLC       BUFLSW                ;
         RLC       BUFLSW-1              ;
         CMP       BUFMSW-1,BUFLSW-1     ;Compare T(i) with 2 * T(i-1)
         JLO       ENDSYNC               ;
         JNE       ERROR                 ;
LSBTEST: CMP       BUFMSW,BUFLSW         ;
         JLO       ENDSYNC               ;
ERROR:
         BR        START1                ;Synchronisation lost : RESET
ENDSYNC:
         RTI

;******************** INJECTION INTERRUPT  *************************************

INJ1     MOV       #0,COM1MSW-1          ;Disable end injection cmd
         MOV       #0FBH,CTLFLG          ;Clear Entry 2 Flag.
         CALL      TINJEC                ;Time Injection computation
         ADD       TINJ,CP1RLSW
         ADC       TINJ-1,CP1RLSW-1      ;COM1LSW = CP1LSW + TINJ
         MOVW      CP1RLSW,COM1LSW       ;Load command
         MOV       #8,COM1MSW-1          ;Re-enable command
         RTI

INJ2     MOV       #0,COM2MSW-1          ;Disable end injection cmd
         MOV       #0FDH,CTLFLG          ;Clear Entry 1 Flag.
         CALL      TINJEC                ;Time Injection computation
         ADD       TINJ,CP1RLSW
         ADC       TINJ-1,CP1RLSW-1      ;COM2LSW = CP1LSW + TINJ
         MOVW      CP1RLSW,COM2LSW       ;Load command
         MOV       #8,COM2MSW-1          ;Re-enable command
         RTI

INJ3     MOV       #0,COM3MSW-1          ;Disable end injection cmd
         MOV       #0FEH,CTLFLG          ;Clear Entry 0 Flag.
         CALL      TINJEC                ;Time Injection computation
         ADD       TINJ,CP1RLSW
         ADC       TINJ-1,CP1RLSW-1      ;COM3LSW = CP1LSW + TINJ
         MOVW      CP1RLSW,COM3LSW       ;Load command
         MOV       #8,COM3MSW-1          ;Re-enable command
         RTI
```

```
INJ4    MOV     #0,COM4MSW-1            ;Disable end injection cmd
        MOV     #07FH,CTLFLG            ;Clear Entry 7 Flag.
        CALL    TINJEC                  ;Time Injection computation
        ADD     TINJ,CP1RLSW
        ADC     TINJ-1,CP1RLSW-1        ;COM4LSW = CP1LSW + TINJ
        MOVW    CP1RLSW,COM4LSW         ;Load command
        MOV     #8,COM4MSW-1            ;Re-enable command
        RTI

;************* Injection time computation ****************************
;
;   The injection time formula is defined just for the software
;   demonstration and doesn't correspond to a physical phenomenon.
;


TINJEC
        PUSH    A
        PUSH    B

        MPY     TEMPER,PRESS            ;TINJ = TEMPER*PRESS
        MOVW    R1,TINJ
        CMP     #2,TINJ-1               ;minimun TINJ = 200h = 1 ms
        JHS     TI1
        MOVW    #20H,TINJ               ;if TINJ<1ms => TINJ=20h = cut off
TI1     POP     B
        POP     A
        RTS

;********************* IGNITION ADVANCE LAW TABLE ******************************
;
;       4 msb of advance value = number of teeth = x * 6 degrees
;       4 lsb of advance value = number of 1/16 of tooth = x * 0.37 degree
;
;       ==> Advance value = number of 1/16 of tooth
;

PRES    .BYTE 00H,1EH,3CH,5AH,78H,96H,0B4H,0D2H,0F0H           ;Pression
REG     .BYTE 75H,27H,1DH,17H,13H,010H,00EH,00BH,009H,08H,07H,06H,005H ;RPM msb
        .BYTE 30H,10H,4CH,70H,88H,0BDH,0A6H,0B8H,0C4H,5EH,53H,82H,0DCH ;lsb
                ;   ;   ;   ;   ;    ;    ;    ;    ;   ;   ;   ;     ;
                ;   ;   ;   ;   ;    ;    ;    ;    ;   ;   ;   ;     ;
TAVANC  .BYTE 00H,07H,0EH,15H,1CH,023H,02AH,031H,038H,3FH,46H,4BH,50H  ;00H
        .BYTE 0AH,11H,18H,1FH,26H,02DH,034H,03BH,042H,49H,50H,55H,5AH  ;1EH
        .BYTE 14H,1BH,22H,29H,30H,037H,03EH,045H,04CH,53H,5AH,5FH,64H  ;3CH
        .BYTE 1EH,25H,2CH,33H,3AH,041H,048H,04FH,056H,5DH,64H,69H,6EH  ;5AH
        .BYTE 28H,2FH,36H,3DH,44H,04BH,052H,059H,060H,67H,6EH,73H,78H  ;78H
        .BYTE 32H,39H,40H,47H,4EH,055H,05CH,063H,06AH,71H,78H,7DH,82H  ;96H
        .BYTE 3CH,43H,4AH,51H,58H,05FH,066H,06DH,074H,7BH,82H,87H,8CH  ;B4H
        .BYTE 47H,4DH,54H,56H,62H,069H,070H,077H,07EH,85H,8CH,91H,96H  ;D2H
        .BYTE 50H,57H,5EH,65H,6CH,073H,07AH,081H,088H,8FH,96H,96H,0A0H ;F0H
            ;Avance allant de 0 a 60 degres: 0 a 160 "1/16 de dent"
```

```
;********** Interrupt vector table *****************************************

        .sect  inphsca,7FB2H
        .word   TOOTH             ; IT CP6 (during initialization phase)

        .sect  inthsca,7FA0H
        .word   INJ3              ; IT injector 3
        .word   INJ2              ; IT injector 2
        .word   INJ1              ; IT injector 1
        .word   SYNCHRO           ; IT Missing teeth checking
        .word   SPARK             ; IT spark time
        .word   PRESPARK          ; IT pre-spark time
        .word   ITRPM             ; IT RPM
        .word   INJ4              ; IT injector 4

        .sect RESET,7FF8H
        .WORD  #0FFFFH
        .WORD  #0FFFFH
        .WORD  #0FFFFH
        .WORD  START              ; IT start program
        .end
```

APPENDIX A

PACT Peripheral Frame

ADDR	PF	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	
1040h	P040	DEFTIM OVRFL INT ENA	DEFTIM OVRFL INT FLAG	CMD/DEF AREA ENA	FAST MODE SELECT	PACT PRECALE SELECT 3	PACT PRESCALE SELECT 2	PACT PRESCALE SELECT 1	PACT PRESCALE SELECT 0	PACTSCR
1041h	P041	CMD/DEF AREA INT ENA	---	CMD/DEF AREA START BIT 5	CMD/DEF AREA START BIT 4	CMD/DEF AREA START BIT 3	CMD/DEF AREA START BIT 2	---	---	CDSTART
1042h	P042	---	CMD/DEF AREA END BIT 6	CMD/DEF AREA END BIT 5	CMD/DEF AREA END BIT 4	CMD/DEF AREA END BIT 3	CMD/DEF AREA END BIT 2	---	---	CDEND
1043h	P043	---	---	BUFFER POINTER BIT 5	BUFFER POINTER BIT 4	BUFFER POINTER BIT 3	BUFFER POINTER BIT 2	BUFFER POINTER BIT 1	---	BUFPTR
1044h	P044	RESERVED								
1045h	P045	PACT RXRDY	PACT TXRDY	PACT PARITY	PACT FE	PACT SCI RX INT ENA	PACT SCI TX INT ENA	---	PACT SCI SW RESET	SCICTLP
1046h	P046	PACT RXDT7	PACT RXDT6	PACT RXDT5	PACT RXDT4	PACT RXDT3	PACT RXDT2	PACT RXDT1	PACT RXDT0	RXBUFP
1047h	P047	PACT TXDT7	PACT TXDT6	PACT TXDT5	PACT TXDT4	PACT TXDT3	PACT TXDT2	PACT TXDT1	PACT TXDT0	TXBUFP
1048h	P048	PACT OP8 STATE	PACT OP7 STATE	PACT OP6 STATE	PACT OP5 STATE	PACT OP4 STATE	PACT OP3 STATE	PACT OP2 STATE	PACT OP1 STATE	OPSTATE
1049h	P049	CMD/DEF INT 7 FLAG	CMD/DEF INT 6 FLAG	CMD/DEF INT 5 FLAG	CMD/DEF INT 4 FLAG	CMD/DEF INT 3 FLAG	CMD/DEF INT 2 FLAG	CMD/DEF INT 1 FLAG	CMD/DEF INT 0 FLAG	CDFLAGS
104Ah	P04A	CP2 INT ENA	CP2 INT FLAG	CP2 CAPT RISING EDGE	CP2 CAPT FALLING EDGE	CP1 INT ENA	CP1 INT FLAG	CP1 CAPT RISING EDGE	CP1 CAPT FALLING EDGE	CPCTL1
104Bh	P04B	CP4 INT ENA	CP4 INT FLAG	CP4 CAPT RISING EDGE	CP4 CAPT FALLING EDGE	CP3 INT ENA	CP3 INT FLAG	CP3 CAPT RISING EDGE	CP3 CAPT FALLING EDGE	CPCTL2
104Ch	P04C	CP6 INT ENA	CP6 INT FLAG	CP6 CAPT RISING EDGE	CP6 CAPT FALLING EDGE	CP5 INT ENA	CP5 INT FLAG	CP5 CAPT RISING EDGE	CP5 CAPT FALLING EDGE	CPCTL3
104Dh	P04D	BUFFER HALF/ FULL INT ENA	BUFFER HALF/ FULL INT FLAG	INPUT CAPT PRESCALE SELECT 3	INPUT CAPT PRESCALE SELECT 2	INPUT CAPT PRESCALE SELECT 1	CP6 EVENT ONLY	EVENT COUNTER SW RESET	OP SET / CLR SELECT	CPPRE
104Eh	P04E	WATCHDOG RESET KEY								WDRST
104Fh	P04F	PACT STEST	---	PACT GROUP 1 PRIORITY	PACT GROUP 2 PRIORITY	PACT GROUP 3 PRIORITY	PACT MODE SELECT	PACT WD PRESCALE SELECT 1	PACT WD PRESCALE SELECT 0	PACTPRI

APPENDIX B

TMS370CX32 Pin Assignment

PIN NAME	PIN NO.	I/O	DESCRIPTION
A0	20	I/O	Port A is a general purpose bidirectional port.
A1	19	I/O	
A2	18	I/O	
A3	17	I/O	
A4	16	I/O	
A5	15	I/O	
A6	13	I/O	
A7	12	I/O	
			Port D is a general purpose bidirectional port.
D3	23	I/O	Also configurable as CLKOUT (see Note 1).
D4/CP3	22	I/O	PACT input capture 3 (see Note 2).
D6/CP4	24	I/O	PACT input capture 4 (see Note 2).
D7/CP5	21	I/O	PACT input capture 5 (see Note 2).
INT1	7	I	External interrupt (non-maskable or maskable)/general purpose input pin.
INT2	8	I/O	Externable maskable interrupt input/general purpose bidirectional pin.
INT3	9	I/O	Externable maskable interrupt input/general purpose bidirectional pin.
CP1	40	I	PACT input capture pin 1.
CP2	36	I	PACT input capture pin 2.
CP6	34	I	PACT input capture pin 6. External Event input pin (for event counter).
TXD	41	O	PACT SCI transmit output pin.
RXD	35	I	PACT SCI receive input pin.
OP1	42	O	PACT output pin 1.
OP2	43	O	PACT output pin 2.
OP3	44	O	PACT output pin 3.
OP4	1	O	PACT output pin 4.
OP5	2	O	PACT output pin 5.
OP6	3	O	PACT output pin 6.
OP7	4	O	PACT output pin 7.
OP8	5	O	PACT output pin 8.
AN0	25	I	A/D analog input (AN0—AN7) or positive reference pins (AN1—AN7).
AN1	26	I	
AN2	27	I	
AN3	28	I	The analog port may be individually programmed as general purpose input pins if not used as A/D converter analog input or positive reference input.
AN4	30	I	
AN5	31	I	
AN6	32	I	
AN7	33	I	
$\overline{RESET}$	6	I/O	System reset bidirectional pin. As input it initializes microcontroller, as open-drain output it indicates an internal failure was detected by the watchdog or oscillator fault circuit.
MC	39	I	Microcomputer mode control input pin; also enables EEPROM write protection override (WPO) mode.
XTAL2/ CLKIN	38	I	Internal oscillator crystal input/External clock source input.
XTAL1	37	O	Internal oscillator output for crystal.
V_{CC1}	10		Positive supply voltage for digital logic and digital I/O pins.
V_{SS1}	14		Ground reference for digital logic and digital I/O pins.
V_{CC3}	11		A/D converter positive supply voltage and optional positive reference input.
V_{SS3}	29		A/D converter ground supply and low reference input pin.

NOTES: 1. D3 may be configured as CLKOUT by appropriately programming the DPORT1 and DPORT2 registers.
2. These digital I/O buffers are internally connected to some of the PACT module's input capture pins. This allows the microcontroller to read the level on the input capture pin, or if the Port D pin is configured as an output, to generate a capture. Be careful to leave the Port D pin configured as an input if the corresponding input capture pin is being driven by external circuitry.

APPENDIX C

Glossary

BDC	: BOTTOM DEAD CENTER
C.C.	: "CONDITIONAL COMPARE" COMMAND
D.E.C.	: "DOUBLE EVENT COMPARE" COMMAND
D.T	: DEFAULT TIMER ON THE TOP OF THE REGISTER FILE
DWELL	: TIME DURING WHICH CURRENT IS SUPPLIED TO IGNITION COIL
DWELL TIME	: START TIME FOR THE DWELL
E.C	: EVENT COUNTER (INCREMENTED BY INPUT CP6)
NEXTDEF	: BIT16 IN COMMAND TO INDICATE THAT NEXT COMMAND IS A DEFINITION
RF	: REGISTER FILE: DATA RAM FROM O TO FF
RPM	: ROUND PER MINUTE
S.C.	: "STANDARD COMPARE" COMMAND
PACT	: TIME AND ACQUISITION SPECIFIC COPROCESSOR
TDC	: TOP DEAD CENTER
T.S	: TIME SLOTS: TIME NEEDED BY THE PACT TO READ OR WRITE IN THE COMMAND/DEFINITION AREA

APPENDIX D

REFERENCES

- "COMMANDE ELECTRONIQUE DES MOTEURS A ALLUMAGE" PAR M. LE DR HELMUT SCHWARZ (ROBERT BOSCH GMBH) SIA OCT.84
- "UNE NOUVELLE GENERATION D'INJECTION ELECTRONIQUE" PAR MM. LEMONNIER ET HOONHORST (SBAE) SIA OCT.84
- AUTOMOTIVE HANDBOOK SAE (BOSCH)
- MOTRONIC TECHNICAL BOOK (BOSCH)
- TMS370 FAMILY DATA MANUAL (1988)
- TMS370 PACT MODULE SPECIFICATIONS (1989)

NOTES

NOTES

NOTES

NOTES

NOTES