Order this document
as AN476/D

Freescale Semiconductor

AN476

CPU16 and the configurable timer module (CTM) in engine control

By Ross Mitchell,
MCU Applications,
Motorola Ltd.,
East Kilbride, Scotland

Introduction

Engine control is very demanding not only for the hardware that must survive the harsh environment, but also for the MCU which may be used to manage the ignition, injectors and a number of other precisely controlled signals. A CPU and some port pins are not enough to do the job and so special timer modules have been developed to make the task of controlling ever more complex engines within the constraints of ever tightening emissions regulations.

The configurable timer module (CTM) was developed for automotive applications which require flexibility and performance from the MCU together with the ability to be designed very quickly for a specific customer's requirements. The CTM is a modular approach to solving this problem.

The CTM is in fact a variety of possible timers that may be constructed from a number of smaller building blocks. A version of the CTM, called CTM-2, is to be found on the M68HC16W1 device. This version contains 10 channels of flexible 16-bit double action capture/compare units plus 3 16-bit free running counters with various capabilities. The CTM-2 module does not have an other existing sub-module which is a single event capture/compare unit. Other modules are under development and as these become available, they may be included in new designs just as easily.

The code described in this document was generated specifically to demonstrate the method and performance of an application using the CTM timer module in a demanding environment. An engine management application requires a great variety of different functions from a timer and so is ideal in demonstrating how the CTM module may be used. The CPU16 is a very good match to this timer module with fast interrupt handling and good 16-bit performance. It should be noted that the CTM is also designed for use with CPU32 which provides significant improvements in CPU performance over CPU16, and supports both 16- and 32-bit reads and writes to make it an excellent match to the CTM.

The example does not cover all aspects of engine management. Aside from the event table generation code there are a number of demands on the software not explored here, such as prolonging or shortening a pulse that has already begun, since these events are mainly dependent upon the way the software is written rather than the specific features of the CTM. Some aspects of the engine control have been simplified and modified to more clearly demonstrate the techniques employed by the software.

It is assumed that the reader has access to detailed information regarding the CTM module, however a short description of the main elements of the CTM is provided to identify the functions associated with the CTM sub-modules.

1 CTM-2 – a brief description

1.1 Architecture and modularity

The CTM is a module for the 68HC16 and 68300 family of micro-controllers. These are collectively known as Inter Module Bus (IMB) devices due to their architecture of a silicon backplane off which hang all the modules that make up the device. The CTM has a very similar approach with a backplane along the centre line of the module and time base busses on the periphery of the CTM, thereby providing control and timing signals to the CTM submodules as efficiently as possible, while retaining maximum flexibility. All the sub-modules that make up the timer have a pre-defined height and are rectangular just as with the IMB modules. Figure 1.1 illustrates the physical layout of the CTM.

Since the sub-modules have been designed to fit together very easily (some designs can be completed in a matter of days), defining the CTM module is basically a case of deciding on the numbers of each sub-module and the order of the different sub-modules on the backplane. There are of course other things to consider, but designing a new CTM module with existing sub-modules is a remarkably easy task.

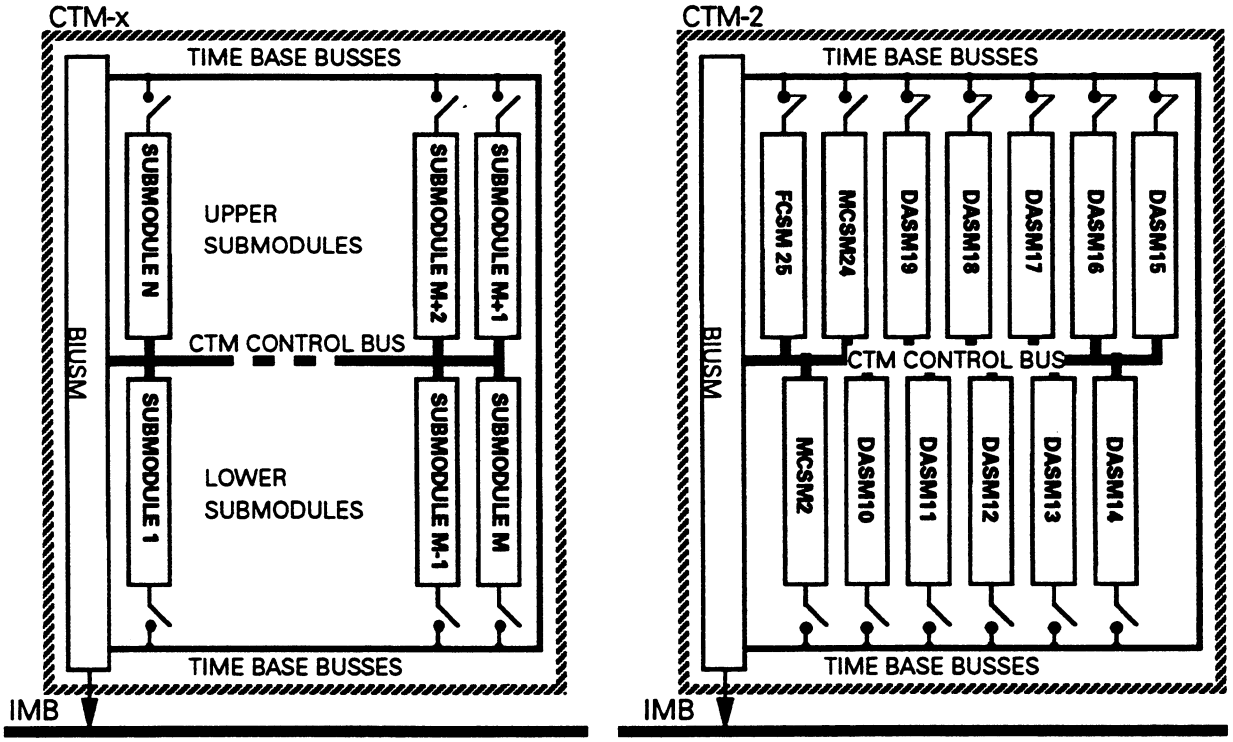


Figure 1.1 CTM-2 architecture

1.2 Bus Interface Unit Sub-module (BIUSM)

In the same way that the SIM module interfaces to the external memory, the Bus Interface Unit Sub-module (BIUSM) provides the interface between the timer sub-modules and the rest of the device. This unit contains the clock sources derived from the system clock and allows the user to set up some of the interrupt priority and arbitration control bits while the remainder are configured individually for each submodule. The clock sources can be from system clock divided by 2 (nominal 8.38MHz) down to a divide by 768 option (nominal 21.8kHz). Together with the control over the system clock frequency, this gives endless possibilities for the selection of the timebases.

1.3 Free-running Counter Sub-module (FCSM)

A 16-bit free-running up counter is basically the standard counter found on many other Motorola devices (eg HC11 and HC05) and very similar to the GPT free-running counter (found on the M68HC16Z1 and M68331 devices).

1.4 Modulus Counter Sub-module (MCSM)

In addition to the free-running counter, this has the additional capability of pre-loading the 16-bit counter. It remains an up counter but can be programmed to count from a specified value up to $FFFF before re-loading the counter as it overflows. Remember that this counter cannot count down. Also note that the counter load register is updated whenever the 16-bit up counter register is written and so saves updating both the load and counter registers separately.

1.5 Double action Sub-module (DASM)

A function that generates both edges of a pulse from a single CPU intervention is very useful in reducing the interrupt overhead where the pulse duration and start time are well defined before the pulse start edge. Equally, the capability of deriving the period or width of a pulse from a single CPU intervention can determine the input pulse characteristics with a minimum of interrupt overhead. These requirements are central to engine management where many interrupt sources are present and the worst case combination of events is of great importance.

The DASM sub-module was designed to perform these tasks and a few others with a minimum of CPU activity. The DASM also provides PWM capability and standard timer capture and compare functions similar to that of the HC11 timer.

2 Demands of engine management

Engine control involves a combination of fast responses to the rapid changes in engine demands required to maintain a high level of efficiency, while at the same time being able to control events ranging from seconds down to microseconds. Consequently, key factors for MCU performance are the maximum interrupt latency for engine control functions and the overall performance at high engine speeds. There are a number of boundary conditions, such as checking for overflows of timers for long or altered pulse periods and coping with sudden changes in demand from the driver, which must be considered for a full engine management system. The example described here is a basic engine control without checks for several boundary conditions.

In the example used to demonstrate the CTM capability, the engine is considered to be a 6-cylinder petrol engine with injection, ignition and knock gate signals provided by the CTM module. The engine parameters are considered to be a 60-tooth crank wheel to make calculations simple (6 degrees between teeth) and it is also the highest number of teeth normally found on petrol engines. Two missing teeth were again chosen because this is the most severe case in normal use and a good demonstration of one of the functions of the DASM sub-module.

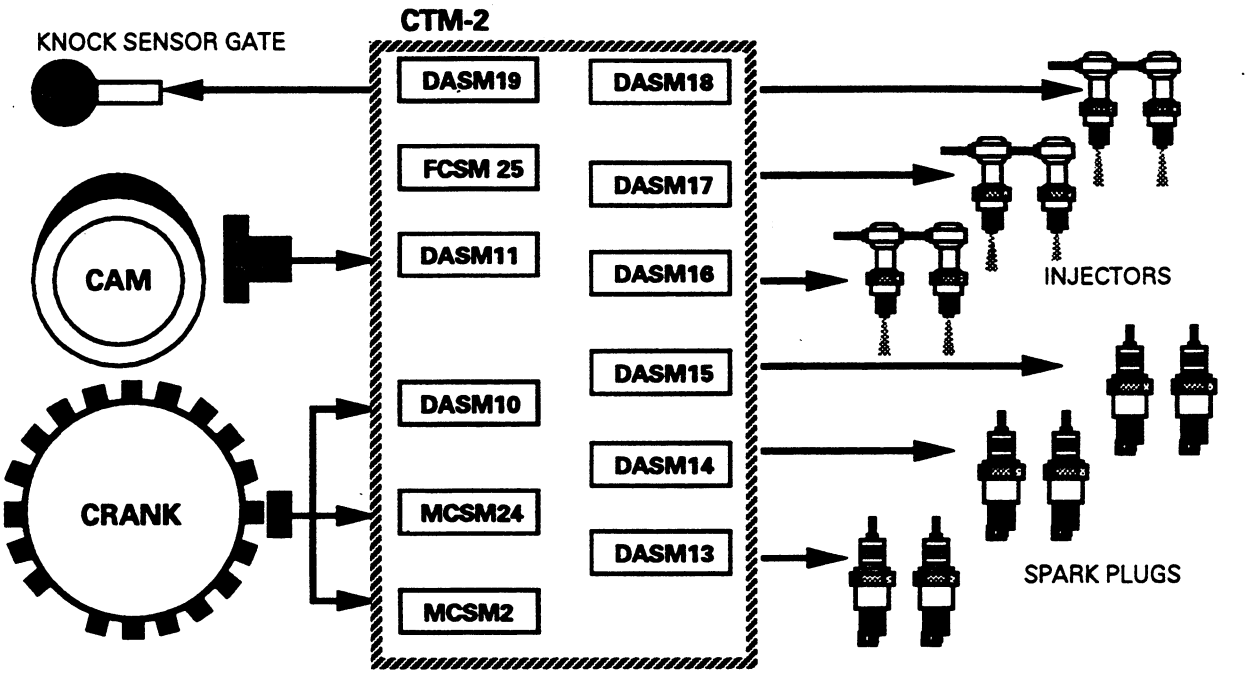


Figure 2.1 Engine control with CTM-2

3 System description

The hardware configuration of the CTM is simply to assign the 12 external pins of the CTM-2 module to engine input and output functions. Figure 3.1 shows the arrangement of external connections which are used for the demonstration software.

An engine has four phases to the engine cycle: induction, compression, ignition and exhaust. A 6-cylinder engine usually has three pairs of pistons, with each pair on opposing halves of the engine cycle. Therefore while one of the pair of pistons is compressing unburnt fuel for the ignition part of the engine cycle, the other piston has reached the exhaust phase of the engine cycle. As a consequence, ignition and injector control may be shared between these pairs of cylinders or be driven from individual outputs on the timer module.

This application demonstrates the principle of independent outputs for all timing pulses for injection and ignition. However, the CTM-2 module has a limited number of channels, and therefore some functions are shared on an output. Figure 3.1 shows the channels that have shared functions on the right hand side. The software is written as if these are separate outputs to acurately reflect the MCU activity and so could very easily be modified to use independent outputs for all the timing functions on a CTM with sufficient channels.

3.1 Hardware configuration

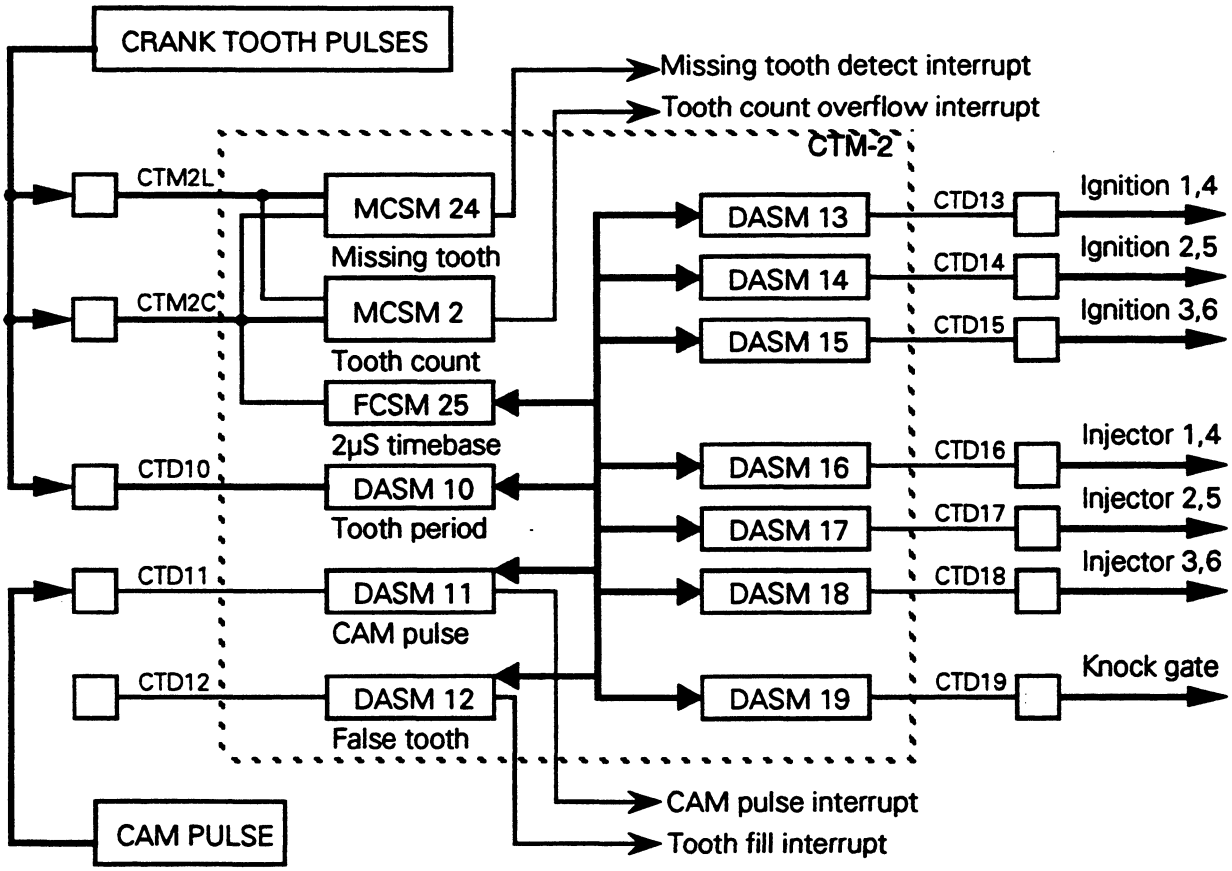


Figure 3.1 Configuration of CTM-2 for engine control

CTM initialisation

The timebase is chosen as 2µS (system clock divided by 32) to give 0.1 degrees resolution at 8,000 rpm and also allows an event of up to 131mS to occur before the 16-bit counter overflows. This equates to an engine speed of 460rpm for any signal that lasts half an engine cycle.

The crank pulses are input to a DASM and two MCSMs. The DASM gives a time for the rising edge of the pulse in addition to providing the time for the previous rising edge, with the result that the period of the pulse is easily determined. The first MCSM (MCSM2) will act as a pulse accumulator to accurately place the output pulses from the CTM. The counts from MCSM2 increment until the overflow interrupts at the desired tooth count. The second MCSM (MCSM24) will locate and verify the missing tooth position. The CAM pulse will go to a DASM with flag set on the negative (falling) edge. During initial starting sequence, the MCSM24 and DASM10 (input period mode) channels are used to locate the missing teeth and then DASM11 (input pulse width mode) is used to check that the missing tooth is at the correct half of the engine cycle before zeroing the tooth counter and synchronising the engine timing.

DASM12-19 are all configured in output pulse mode with either flag set on B (second edge) or flag set on A and B (both edges). The ignition pulses are generated using 3 DASM channels (13, 14, 15) in flag set on A and B mode and require an interrupt for both the rising and falling edges since these edges must be within a tenth of a degree of the optimum position and require the edge to be set as late as possible. The DASM is capable of generating a pulse from a single interrupt and so DASM 16,17 and 18 are used in flag set on B mode for injection pulses which do not have a very critical falling edge time. The knock gate signal is also generated using a single interrupt to set up DASM19 (flag set on B), since the first edge is most important and the pulse duration is short.

3.2 Software control

The software consists of three separate modules: the CTM initialisation, exception handler and background task.

To initialise the timing, the shortest period of tooth pulse is found and then it is assumed that the next longer period is the missing tooth location. This places the engine timing at either 0 degrees or 360 degrees of the 720 degrees full engine cycle so it is then necessary to confirm the presence of the CAM pulse.

Once the engine timing is synchronised, the crank pulses give a coarse angular position of the engine in the engine cycle. The missing teeth synchronise with the CAM pulse every second engine revolution. A modulus counter (MCSM2) counts the crank pulses, interrupting on overflow, while a double action submodule (DASM12) generates an interrupt for each of the missing tooth positions. Thus there is a possible interrupt every 6 degrees of engine revolution.

The missing teeth can be confirmed by setting the other modulus counter (MCSM24) to overflow between the predicted times for teeth 58 and 59 and again at the end of the engine cycle between teeth 118 and 119. This will enable the match flag when the tooth counter is at 58 or 118. Therefore, when DASM12 generates an interrupt on pulse position 58 or 118, the modulus counter overflow flag should not be set, but it will be set before pulse position 59 or 119. This provides a simple method of ensuring that the missing tooth positions are where they are expected to be. See figure 4.2.

The ignition pulse is generated by DASM13,14,15 (one output for a pair of cylinders). The position of the ignition pulse is normally determined by a number of sensors (engine load – vacuum, accelerator pedal position, engine speed, air temperature, etc) and a look-up table may be used to provide a schedule for the order and nature of the pulses to be generated. For example, the timing may be in terms of advance angle for the engine ignition while the injector timing could be an angular offset with a time for the pulse width.

The ignition pulses are known and angle-angle pulses since both edges are based directly on engine angle (ie the engine requires the falling edge at a given angle after top dead centre (TDC) and the rising edge at another, larger angle after TDC). The first edge starts energising the low tension coil, while the second edge cuts off the current supply abruptly to produce a high voltage in the high tension coil to produce a spark.

Since this timing is very critical to an engine, the second edge must be related to an engine angle and the low tension energising period is a time which is converted by the system software into an angle for the current engine speed.

The crank pulses yield a figure for the instantaneous speed of the engine by subtracting the first rising edge time from the 2nd rising edge time (DASM10 in period measurement mode).

The ignition output pulse timing is generated from this period and the table information to provide an interrupt between 1 and 2 crank tooth pulses before the required falling (start) edge and then a further interrupt prior to the rising (finish) edge to place this edge accurately for the ignition fire timing. This routine makes a fractional calculation to place the edge at a precise time based on the engine angle and engine speed. Similarly, the second edge of the pulse (rising edge) is a result of an angle to time calculation. In this example, the ignition pulses are negative polarity.

The reason for placing the edge at least 1 crank tooth period away is to allow for the worst case latency in the exception response time. In practice, this may be very much less than one tooth period, so the background software may take advantage of this, and could easily be made to place edges closer to the scheduling tooth.

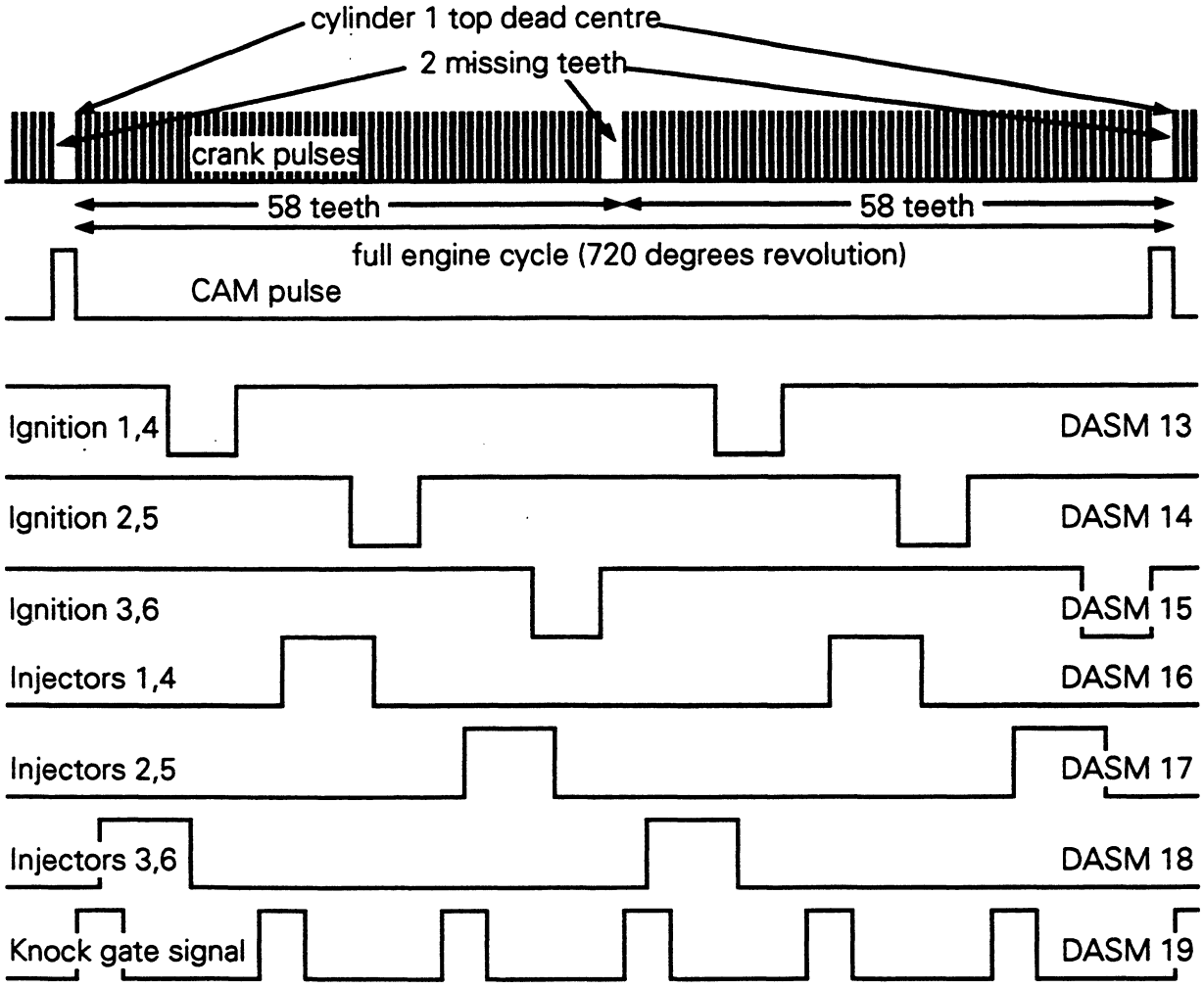


Figure 3.2 Engine control signals

The knock sensor gate has the same criteria as the ignition timing but in this case the first edge is most important and uses DASM19 in output pulse mode (flag set on B). This routine makes a fractional calculation to place the start edge at a precise time based on the engine angle and engine speed. The second edge of the pulse (falling edge) timing is also defined at this point and is again a conversion from angle to time. The knock gate pulses are known as angle-angle pulses with both edges based directly on engine angle. Because a single interrupt is used to set up the entire pulse, the second edge is calculated from an engine angle much earlier than the equivalent pulse for the ignition pulse. This will lead to a poorer estimate of the actual engine angle at the time of its second, falling edge but it reduces the interrupt overhead and still has sufficient accuracy for the engine to run efficiently. This pulse is normally relatively short and so there is no problem with this approach. One thing to note is the amount of calculation possible with CPU16, yet which still keeps the exception handler execution time much less than the shortest tooth period.

The injectors' pulse is not as critical as the ignition pulse and the more important parameter is the duration of the pulse, so DASM16,17,18 are used to generate the pulses for each of the 6 cylinders individually. This routine makes a fractional calculation to place the start edge at a precise time based on the engine angle and engine speed. The second edge of the pulse (falling edge) timing is also defined at this point and is a straight-forward duration of the pulse This is because the injector fire time is more important than the finish angle of the pulse and takes less CPU time to perform this task. The injector pulses are known and angle-time pulses with the first edge based on engine angle and the second on the duration of the pulse.

See section 5 for details of the performance figures relating to the different ways of generating a pulse. Here it can be seen that there is a trade-off between CPU time and edge placement accuracy.

The background program would normally do a lot more than just loop on the same instruction. Since very little of the CPU time is used to maintain the engine control signals, there is plenty of CPU performance left to perform the complex calculations required to place the control pulses. Since this does not help illustrate the CTM performance, such software was not written for this demonstration code.

4 The CTM modules in action

4.1 MCSM modes of operation

The tooth count is determined by using the MCSM as a pulse accumulator. Here the load pin is not used, but the clock input pin (CTM2C) increments the counter on each rising edge of the tooth pulse signal. By preloading the counter with a small negative number, an overflow indicates the occurrence of a particular tooth position.

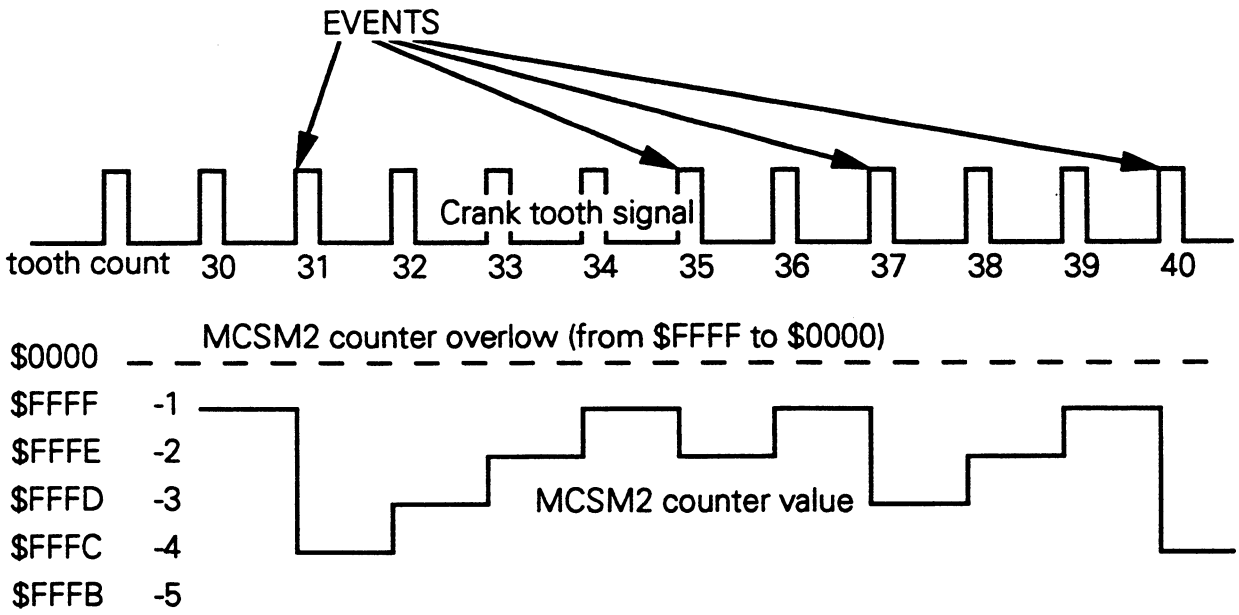


Fig 4.1 Angular position

4.1.1 Determining angular position

An interrupt is generated whenever the tooth counter (MCSM2) flag is set. This is controlled by the value in the modulus latch register which is set to 2's complement of the number of pulses remaining for the next scheduled event and so the modulus latch register is constantly updated to provide the interrupts at the correct point in the engine cycle. See figure 4.1 for an illustration of the MCSM2 counter being pre-loaded and then counting up to the overflow state. The exception handler then checks a table of events which have been written in chronological order to determine the next event and set up the appropriate pulse timing. The IY index register is used to point to the current data in this table. This table is normally built by the background program during the previous engine cycle. In this example the table has been fixed as constant data. The interrupt routine must execute in less than the minimum time between crank pulses and also include the worst case entry latency. This gives a maximum time of 100µS (10,000 rpm and 60 teeth per engine revolution).

4.1.2 Missing tooth pulses

By pre-loading a modulus counter (MCSM24) with a 2's complement value greater than the current tooth period and using the load pin (CTM2L) to reload the counter on each tooth pulse rising edge, it is possible to detect a missing tooth simply by the presence of the overflow flag being set. A safe value for the MCSM24 load register would be 1.5 times the current period as this would account for increasing and decreasing engine speed. Figure 4.2 shows the MCSM24 counter being constantly re-loaded until there is a longer period where the missing tooth position exists.

In the example it is started 8 tooth pulses prior to a missing tooth so that any synchronisation problems are detected and at the same time there is no chance for the engine to slow down sufficiently to cause an erroneous missing tooth interrupt.

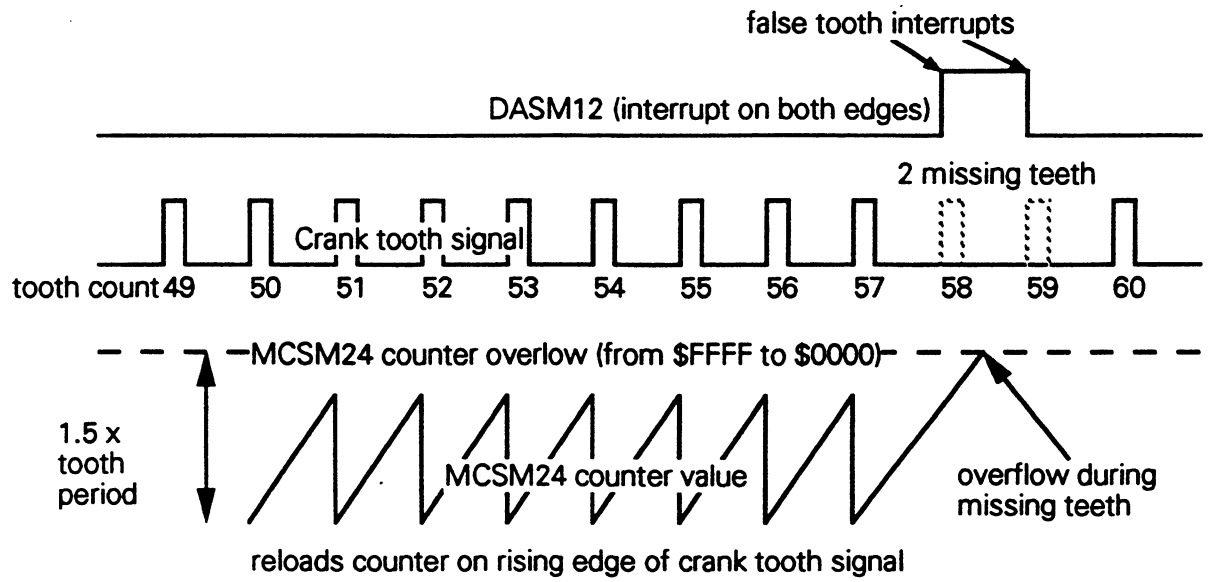


Figure 4.2 Missing tooth detection

4.2 DASM modes of operation

Input pulse length

This mode is used with the CAM signal exception handler. This routine checks to see that the CAM signal occurred at the correct time and saves the rising and falling edge times for the CAM pulse. In some systems the CAM pulse may be used to derive other information about the engine and perhaps the width of the pulse as well as the falling edge time is required for some calculations.

As can be seen this information is retrieved and stored in 1.2 microseconds with just two 32-bit HC16 instructions (LDED and STED).

Input period

A key parameter for the engine is the speed of rotation. By placing the DASM channel in input period measurement mode, a single 32-bit read followed by a subtraction reveals this value. Thus capturing the same edge on each pulse gives very fast determination of the engine speed with minimal interrupt overhead.

Output pulse, flag set on B

Again, to reduce the interrupt overhead, the DASM channel can place a pulse from a single CPU intervention. In this mode the injection pulses are determined before the start of the pulse and the flag is only set after the second edge. In this example there is no interrupt following the pulse, but the flag could be used to determine if the pulse is complete when there is a need to extend the pulse.

Output pulse, flag set on A and B

This mode is useful when the missing teeth occur. By placing an edge at both missing tooth positions, it is possible to generate an interrupt where the tooth pulses should have occurred from a single CPU intervention. See figure 4.3 for an illustration of the tooth filler interrupt generation.

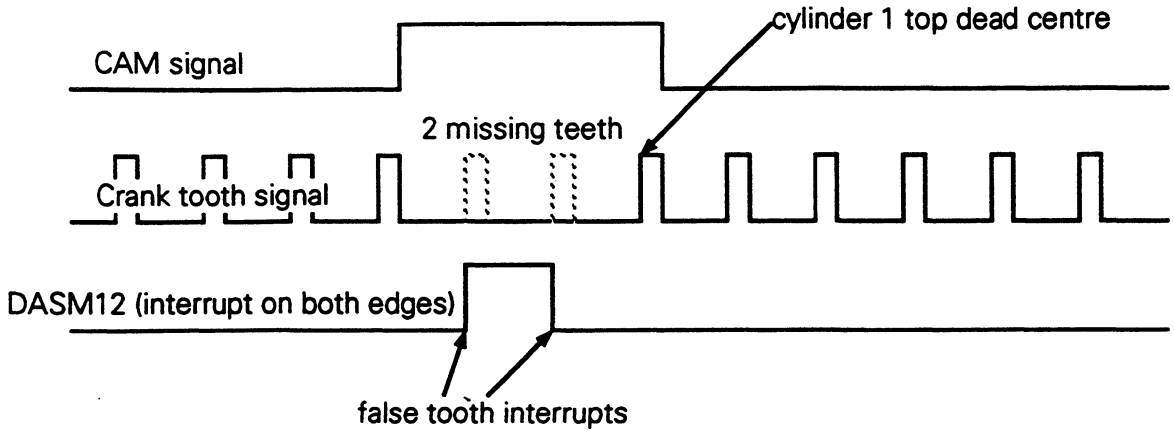


Figure 4.3 False tooth interrupt

This mode was also used for the ignition pulses since each edge is set up independently. Thus event A can occur before event B is determined. In the example, there are no interrupts generated by the pulse, but the flag is set each time there is a compare.

Prolonging or shortening a pulse that has already begun may be achieved very simply. Much depends upon the way the events are controlled, but it would be necessary to first check that the falling edge of the pulse was not imminent before scheduling another compare at the newly requested time. It is then simply a matter of writing a new value into the B register with the new time for the end of the pulse. Since the compare has not yet occurred, the revised match time can be rewritten and becomes effective immediately.

Pulse width modulation

While this example does not use PWM mode, it is frequently used in automotive applications. The DASM module has the capability of a wide range of periods and resolution, with a maximum of 62.5kHz PWM with 7 bits resolution.

4.2.1 Placing edges and pulses

In order to minimise interrupt handling 2 timebases are used; the first is effectively the tooth count rate which increments the modulus counter (MCSM2) and then switches to the second timebase which is derived from the much more stable and accurate system clock (15.991MHz in this example).

After finding the approximate position by counting the number of tooth pulses from cylinder 1 top dead centre (TDC) it is possible to obtain a coarse angular position for an output pulse. For finer resolution it is necessary to switch to a more accurate timebase. In this case a 2 µS timebase is provided by the free running counter (FCSM25).

The DASM10 channel provides the time for the past two crank tooth pulses and so gives much of the information required to place an output pulse. By adding the difference between the two values to the last tooth time, it is possible to immediately place the expected position of the next tooth pulse. By making a fractional calculation of how far through the next tooth period the output pulse should change state, it is possible to place an edge as accurately as the timebase will allow assuming the tooth period is approximately constant.

Crank tooth schedules an event to occur between one and two teeth from the current tooth. Tooth x interrupts and the previous period (z) of the crank signal is added to the fraction of the tooth period (y) to give an event z+y after tooth x. See fig 4.4 for an illustration on edge placement.

Thus one submodule (MCSM2) generates an interrupt a short time before the required time and the internal timebase is then used to actually place the pulse.

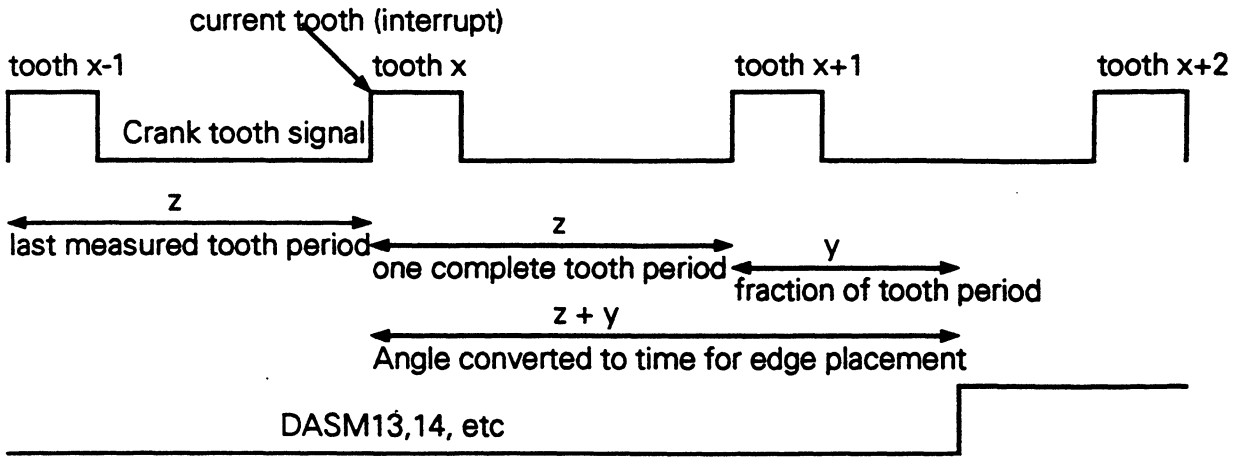


Figure 4.4 Edge placement

4.3 Exception handling

There are four sources of interrupt in the example. The highest interrupt mask was set at level 5 for the crank tooth interrupt and the false tooth interrupt since these events relate directly to the event scheduling and are regarded as critical timing within the system. Level 5 is also used for the missing tooth interrupt which should occur between the normal tooth pulse positions and so should normally occur when the false tooth pulse interrupt routine is complete. This interrupt could have been made level 6 to ensure it always was seen immediately, but keeping it at level 5 means that the normal tooth interrupt routine will complete before this routine starts and makes the worst case timing easier to define. Level 6 is reserved for other unspecified and more important tasks, while level 7 is used when the system needs to resynchronise the missing tooth position.

The CAM pulse interrupt is relatively unimportant and so has its interrupt mask at level 3.

5 The partnership of CPU16 and CTM

The CPU16 and CTM make a good partnership. Firstly they are both 16-bit designs and secondly the fast interrupt response of CPU16 makes best use of the CTM design. For software control, the GPT and CTM are similar. The CTM requires that the CPU controls all but the PWM function continuously to maintain the output signals, and so the improvements to reduce the interrupt handling overhead take advantage of CPU16. The fast multiply capability of CPU16 also makes it possible to construct an event table with fractions of an angle which can be converted to time immediately prior to the pulse to give the most accurate value possible for the output pulses. Just 5 µS is required to make this calculation and write to the DASM register.

5.1 Overall performance

The worst case latency is normally just the worst case entry delay into the exception handler of 38 cycles (2.28µS) for the EDIVS command. Assuming a total latency of 2.28µS the crank pulse exception handler must execute in less than 98µS.

The main exception handler has an interrupt priority of 5, but we must ensure that only one other interrupt source can delay its entry into the main routine. All other exception handlers should be kept as short an execution time as possible, with particular attention paid to exception handlers at level 5 and above. The routine's worst case timing is when an ignition pulse coincides with the same tooth count as that for other injector or ignition edges and where the crank tooth is present. It is assumed that the knock sensor gate is always separate from the ignition pulse's edges and that the injectors may overlap with 2 rising edges and 2 falling edges at the same time plus the next ignition pulse falling edge at the same time as the rising edge of a previous ignition pulse. A missing tooth interrupt takes this to 5 simultaneous edges from a single tooth position.

All of the following timings are given twice.

The first number (1) is for fast termination (ie internal ROM)
The second number (2) is for zero wait states (ie external 85ns EPROM)

5.1.1 Worst case latency

Interrupt entry latency (including EDIVS)		38 cycles (1) + 20 cycles (1) = 58 cycles (1) 39 cycles (2) + 26 cycles (2) = 65 cycles (2)
Main routine takes the following plus subroutine time to execute ...		
tooth 0	183 cycles (1),	230 cycles (2)
tooth 60	187 cycles (1),	236 cycles (2)
any other tooth	199 cycles (1),	252 cycles (2)
If entered from tooth filler interrupt then this routine takes the same number of cycles for teeth 59 and 119 and 14 fewer cycles for teeth 58,118		
Tooth missed detector interrupt		
If in synchronisation		54 cycles + 20 cycle entry latency = 64 cycles (1) 67 cycles + 26 cycle entry latency = 79 cycles (2)
Good cam position takes		20 cycles entry + 55 cycles = 74 cycles (1) 26 cycles entry + 66 cycles = 91 cycles (2)
tooth fill interrupt generator takes		55 cycles (1) and 70 cycles (2)
Ignition pulse - rising/falling takes		87 cycles (1) and 103 cycles (2)
Knock sensor gate pulse takes		203 cycles (1) and 244 cycles (2)
Injector - both edges min 916 rpm takes		99 cycles (1) and 118 cycles (2)
Injector - single edge min 8 rpm takes		87 cycles (1) and 103 cycles (2)
Missing tooth detect takes		38 cycles (1) and 53 cycles (2)

Worst case is interrupt entry + main routine + 2 ignition + 2 injector + missing tooth detect subroutines
With internal ROM: 58 + 199 + 87 + 87 + 99 + 99 + 38 =667 cycles (40.0µS) (1)
With zero wait state memory: 65 + 252 + 103 + 103 + 118 + 118 + 53 =812 cycles (48.7µS) (2)

5.1.2 Overall interrupt overhead
Each engine cycle requires the following, assuming each edge is scheduled from a separate tooth count (worst case interrupt overhead)

Missing tooth interrupt times 2
CAM interrupt times 1
Tooth filler interrupt times 4
tooth fill interrupt generator times 2
Ignition pulse - rising/falling times 12
Knock sensor gate pulse times 6
Injector - both edges min 916 rpm times 6
Missing tooth detect times 2

Total of 35 interrupts per engine cycle

Total time in exception handlers is calculated as follows:

Time with internal ROM (1)
35 interrupts + 2 missing tooth exceptions + CAM detection + 4 tooth fill + 28 normal main routines
(35x20) + (2x54) + 55 + (183 + 187 + 2x175) + (28x199) = 7155 cycles (1)
2 tooth fill + 12 ignition + 6 injection + 6 knock gate + 2 missing tooth detect subroutines
(2x55) + (12x87) + (6x203) + (6x99) + (2x38) = 3042 cycles (1)
TOTAL = 10197 cycles (612µS) (1)
With engine cycle (10,000 rpm)=12mS, exception handlers represent 5.1% of CPU16 performance (1)

Time with external zero wait state memory (2)
(35x26) + (2x67) + 65 + (229 + 235 + 2x237) + (28x252) = 9103 cycles (2)
(2x70) + (12x103) + (6x244) + (6x118) + (2x53) = 3654 cycles (2)
TOTAL = 12757 cycles (765µS) (2)
With engine cycle (10,000 rpm)=12mS, exception handlers represent 6.4% of CPU16 performance (2)

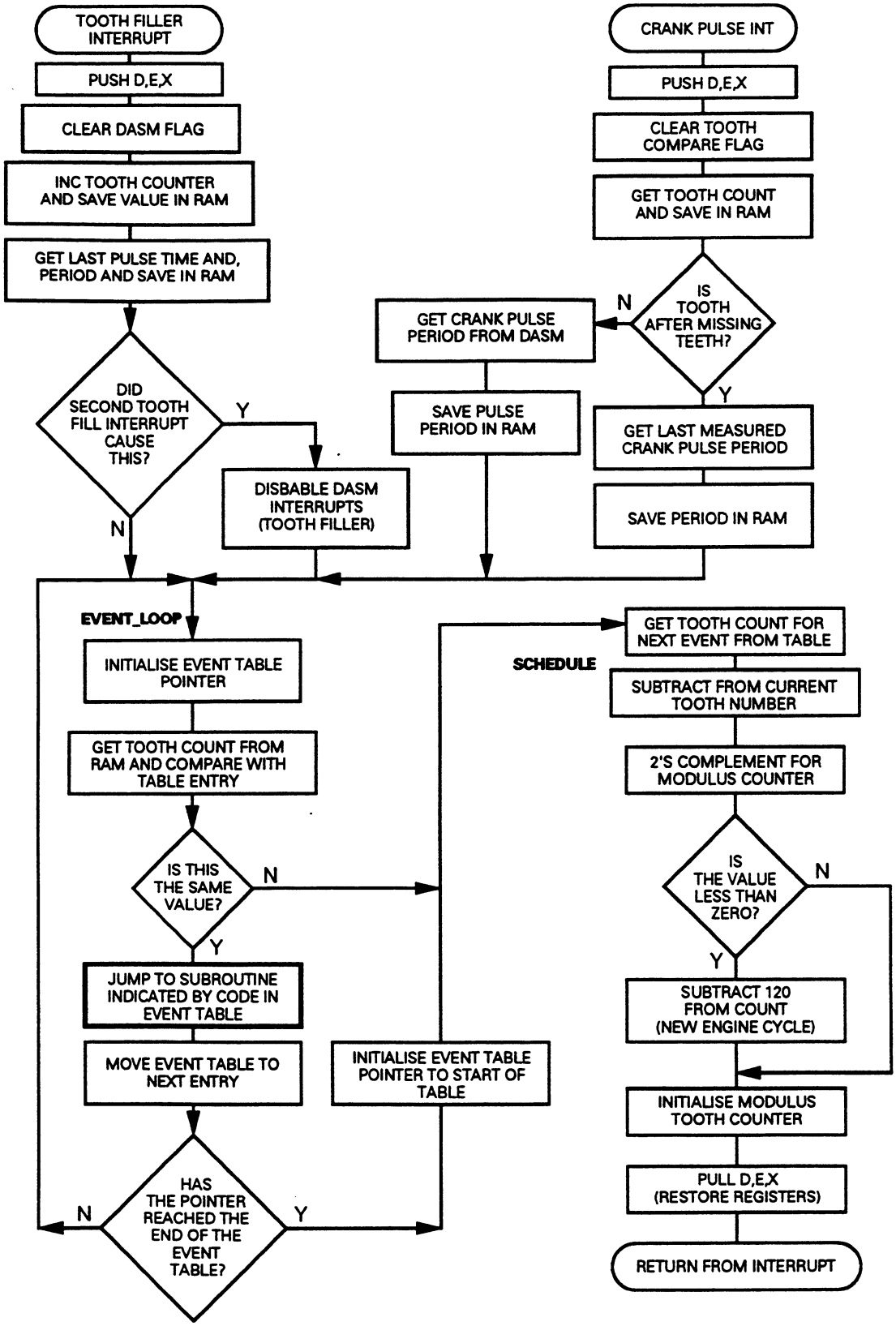
5.1.3 Results of the above calculations
Note that although the example uses a system clock of 15.991MHz, the performance timings are related to a system clock of 16.78MHz (the maximum system clock frequency for this device).

For fast termination (ie internal ROM) timings (1)
Total time for one ignition pulse (angle-angle with 2 interrupts) is 36µS (612 cycles)
Total time for one injector pulse (angle-time with a single interrupt) is19µS (318 cycles)
Total time for one knock gate pulse (angle-angle with a single interrupt) is 25µS (422 cycles)
Worst case interrupt latency of 40µS (much less than 100µS) plus the time of the longest exception handler that could be invoked and interrupt handling taking 612µS at 10,000 rpm (5.1% of CPU16 performance)

For zero wait states (ie external 85ns EPROM) timings (2)
Total time for one ignition pulse (angle-angle with 2 interrupts) is 43µS (714 cycles)
Total time for one injector pulse (angle-time with a single interrupt) is 24µS (396 cycles)
Total time for one knock gate pulse (angle-angle with a single interrupt) is 31µS (522 cycles)
Worst case interrupt latency of 49µS (much less than 100µS) plus the time of the longest exception handler that could be invoked and interrupt handling taking 765µS at 10,000 rpm (6.4% of CPU16 performance)

Clearly the performance of the CTM and the CPU16 together allows for more comprehensive engine control than has been demonstrated in this example. It shows that a high performance timer can be used very efficiently with a 16-bit CPU and frees the CPU to perform the many other tasks required for modern engine control.

Appendix A – Exception handler



Appendix B – Engine management software

```
                          4  ************************************************************** ************
                          5  **** M O T O R O L A   C O N F I D E N T I A L   P R O P R I E T A R Y ****
                          6  ************************************************************** ************
                          7  * ENGINE.ASM      Rev 1.0     last edit : 8th September 1992
                          8  * Demonstration code written for CTM-2 module to show how the Configurable Timer
                          9  * Module may be used in an engine management application.
                         10  * This program configures the CTM-2 module to run a petrol engine and
                         11  * contains the major elements of the interrupt handling to perform this task.
                         12  *
                         13  * All commented timings are for zero wait states access, assuming
                         14  * external EPROM or RAM
                         15  *
                         16  * Revision 1.0 Written for the  68HC16W1 which contains the CTM-2 module.
                         17  *
                         18  * This code is written for demonstration purposes only and is not guaranteed
                         19  * to function in a given application.
                         20  * Copyright Motorola Inc 1991, 1992
                         23  *
                         24  * GENERAL DESCRIPTION
                         25  * The following code was generated specifically to demonstrate the method and
                         26  * performance of an application using the CTM timer module in a demanding
                         27  * environment. An engine management application requires a great variety
                         28  * of different functions from a timer and so is ideal in demonstrating
                         29  * how the CTM module may be used. The CPU16 is a very good match to this
                         30  * timer module with fast interrupt handling and good 16-bit performance.
                         31  *
                        317  ************************************************************** **************
                        318  *
00000                   319  rom      equ     $1000
00000                   320  ram      equ     $0
00000                   321  vectors  equ     $0
00000                   322  top_stack equ    $0dfe
                        323  *
00000                   324  sim      equ     $A00
00000                   325  sram     equ     $B00
00000                   326  qsm      equ     $C00
00000                   327  ctm      equ     $400
00000                   328  ctm_module equ   $80             ; ctm module vectors start at vector 64
                        329  *
                        477  *****  CTM Module Registers  *****
00000                   478  CTMMCR     equ   $F400
00000                   479  TBR        equ   $F404
00000                   480  CPSMCR     equ   $F408
00000                   481  MCSM2SIC   equ   $F410
00000                   482  MCSM2CNT   equ   $F412
00000                   483  CMSM2MOD   equ   $F414
00000                   496  dasm10sic equ   $F450
00000                   497  dasm10a    equ   $F452
00000                   498  dasm10b    equ   $F454
00000                   499  dasm11sic equ   $F458
00000                   500  dasm11a    equ   $F45A
00000                   501  dasm11b    equ   $F45C
00000                   502  dasm12sic equ   $F460
00000                   503  dasm12a    equ   $F462
00000                   504  dasm12b    equ   $F464
00000                   505  dasm13sic equ   $F468
00000                   506  dasm13a    equ   $F46A
00000                   507  dasm13b    equ   $F46C
00000                   508  dasm14sic equ   $F470
00000                   509  dasm14a    equ   $F472
00000                   510  dasm14b    equ   $F474
00000                   511  dasm15sic equ   $F478
00000                   512  dasm15a    equ   $F47A
00000                   513  dasm15b    equ   $F47C
00000                   514  dasm16sic equ   $F480
00000                   515  dasm16a    equ   $F482
00000                   516  dasm16b    equ   $F484
00000                   517  dasm17sic equ   $F488
00000                   518  dasm17a    equ   $F48A
00000                   519  dasm17b    equ   $F48C
00000                   520  dasm18sic equ   $F490
00000                   521  dasm18a    equ   $F492
00000                   522  dasm18b    equ   $F494
00000                   523  dasm19sic equ   $F498
00000                   524  dasm19a    equ   $F49A
00000                   525  dasm19b    equ   $F49C
00000                   534  mcsm24sic equ   $F4C0
```

```
00000                   535  mcsm24cnt equ   $F4C2
00000                   536  mcsm24mod equ   $F4C4
00000                   537  fcsm25sic equ   $F4C8
00000                   538  fcsm25cnt equ   $F4CA
00000                   932          org     ram
00000                   933  period      rmb 2       ; period of previous pulse
00002                   934  next_pulse  rmb 2       ; time predicted for next pulse from crank
00004                   935  tenths      rmb 2       ; edge position as a number of tenths of a degree
00006                   936  tooth       rmb 1       ; tooth count for current time
00007                   937  next_tooth  rmb 1       ; next tooth count for event
00008                   938  cam_tooth   rmb 2       ; cam tooth position
0000A                   939  cam_pulse   rmb 2       ; CAM pulse edges
0000C                   940  knock_tooth rmb 2       ; knock tooth fraction for start edge
                        941  *
                        942  * CONSTANTS
                        943  *
0000E                   944  event   equ     $400
0000E                   945  table   equ     $500
0000E                   946  sixty   equ     60          ; constant value of value 60 (10 * 6 degrees)
0000E                   948  dasm_fall equ   $8010       ; change edge, use BCLRW
0000E                   949  dasm_rise equ   $00         ; change edge, use BSET (and clear flag with BCLR)
0000E                   950  flag    equ     $80         ; flag for DASM and SASM
0000E                   951  int_en  equ     $84         ; SASM interrupt enable bit
                        952  *
0000E                   953  K       equ     $0110      ;  not used, ZK=$0, SK=$1, PK=$0
0000E                   954  SP      equ     $0DFE      ;  stack pointer starts at address $0DFE
0000E                   955  IZ      equ     $0000      ;  index pointer set for registers
                        956
00000                   957          org     vectors
00000      0110         958          dw    K           ;  initial ZK, SK, PK
00002      1000         959          dw    reset       ;  initial program counter value
00004      0DFE         960          dw    SP          ;  initial stack pointer value
00006      0000         961          dw    IZ          ;  initial direct page select (IZ)
                        962  *
                        963  * CTM module vectors
                        964  *
00008                   965  reserved equ    $0000
                        966  *
00080                   967          org     ctm_module
00080      156A         968          fdb     ctm_dummy       ; biusm (reserved)
00082      156A         969          fdb     ctm_dummy       ; cpsm  (reserved)
00084      11B0         970          fdb     int             ; mcsm 2
00086      156A         971          fdb     ctm_dummy       ; mcsm 2 (reserved)
00088      0000         972          fdb     reserved        ; sasm 4a (not available on CTM-2)
0008A      0000         973          fdb     reserved        ; sasm 4b (not available on CTM-2)
0008C      0000         974          fdb     reserved        ; sasm 6a (not available on CTM-2)
0008E      0000         975          fdb     reserved        ; sasm 6b (not available on CTM-2)
00090      0000         976          fdb     reserved        ; sasm 8a (not available on CTM-2)
00092      0000         977          fdb     reserved        ; sasm 8b (not available on CTM-2)
00094      156A         978          fdb     ctm_dummy       ; dasm 10
00096      116C         979          fdb     cam             ; dasm 11
00098      118A         980          fdb     tooth_filler    ; dasm 12
0009A      156A         981          fdb     ctm_dummy       ; dasm 13
0009C      156A         982          fdb     ctm_dummy       ; dasm 14
0009E      156A         983          fdb     ctm_dummy       ; dasm 15
000A0      156A         984          fdb     ctm_dummy       ; dasm 16
000A2      156A         985          fdb     ctm_dummy       ; dasm 17
000A4      156A         986          fdb     ctm_dummy       ; dasm 18
000A6      156A         987          fdb     ctm_dummy       ; dasm 19
000A8      0000         988          fdb     reserved        ; sasm 20a (not available on CTM-2)
000AA      0000         989          fdb     reserved        ; sasm 20b (not available on CTM-2)
000AC      0000         990          fdb     reserved        ; sasm 22a (not available on CTM-2)
000AE      0000         991          fdb     reserved        ; sasm 22b (not available on CTM-2)
000B0      1142         992          fdb     tooth_missed    ; mcsm 24
000B2      156A         993          fdb     ctm_dummy       ; fcsm 25
                        994  *
```

```
                        995  * EVENT correlation
                        996  * Table maps event code to a subroutine address
                        997  * event code is always an even number since the table is made up of word values
                        998  *
00400                   999          org     event
                       1000  event_list
00400    1568          1001          fdb     dummy           ; 00              dummy event
00402    11B0          1002          fdb     int             ; 02    mcsm2     main interrupt routine
00404    1540          1003          fdb     checkmis        ; 04    mcsm24    check for missing tooth
00406    122E          1004          fdb     tooth_fill      ; 06    dasm12    missing tooth position
00408    124A          1005          fdb     ignit1f         ; 08    dasm13    ignition 1 falling edge
0040A    126A          1006          fdb     ignit2f         ; 10    dasm14    ignition 2 falling edge
0040C    128A          1007          fdb     ignit3f         ; 12    dasm15    ignition 3 falling edge
0040E    12AA          1008          fdb     ignit4f         ; 14    dasm13    ignition 4 falling edge
00410    12CA          1009          fdb     ignit5f         ; 16    dasm14    ignition 5 falling edge
00412    12EA          1010          fdb     ignit6f         ; 18    dasm15    ignition 6 falling edge
00414    130A          1011          fdb     ignit1r         ; 20    dasm13    ignition 1 rising edge
00416    132A          1012          fdb     ignit2r         ; 22    dasm14    ignition 2 rising edge
00418    134A          1013          fdb     ignit3r         ; 24    dasm15    ignition 3 rising edge
0041A    136A          1014          fdb     ignit4r         ; 26    dasm13    ignition 4 rising edge
0041C    138A          1015          fdb     ignit5r         ; 28    dasm14    ignition 5 rising edge
0041E    13AA          1016          fdb     ignit6r         ; 30    dasm15    ignition 6 rising edge
00420    13CA          1017          fdb     knock           ; 32    dasm19    knock sensor gate
00422    1568          1018          fdb     dummy           ; 34
00424    155C          1019          fdb     toggle1         ; 36              toggle port E-7 high
00426    1562          1020          fdb     toggle0         ; 38              toggle port E-7 low
00428    141C          1021          fdb     inject1         ; 40    dasm16    injector 1 both edges
0042A    1442          1022          fdb     inject2         ; 42    dasm17    injector 2 both edges
0042C    1468          1023          fdb     inject3         ; 44    dasm18    injector 3 both edges
0042E    148E          1024          fdb     inject4         ; 46    dasm16    injector 4 both edges
00430    14B4          1025          fdb     inject5         ; 48    dasm17    injector 5 both edges
00432    14DA          1026          fdb     inject6         ; 50    dasm18    injector 6 both edges
00434    1568          1027          fdb     dummy           ; 52
00436    1568          1028          fdb     dummy           ; 54
00438    1500          1029          fdb     inject1r        ; 56    dasm16    injector 1 rising edge
0043A    1520          1030          fdb     inject1f        ; 58    dasm16    injector 1 falling edge
                       1031  *       fdb     inject2r        ; 60    dasm17    injector 2 rising edge
                       1032  *       fdb     inject2f        ; 62    dasm17    injector 2 falling edge
                       1033  *       fdb     inject3r        ; 64    dasm18    injector 3 rising edge
                       1034  *       fdb     inject3f        ; 66    dasm18    injector 3 falling edge
                       1035  *       fdb     inject4r        ; 68    dasm16    injector 4 rising edge
                       1036  *       fdb     inject4f        ; 70    dasm16    injector 4 falling edge
                       1037  *       fdb     inject5r        ; 72    dasm17    injector 5 rising edge
                       1038  *       fdb     inject5f        ; 74    dasm17    injector 5 falling edge
                       1039  *       fdb     inject6r        ; 76    dasm18    injector 6 rising edge
                       1040  *       fdb     inject6f        ; 78    dasm18    injector 6 falling edge
                       1041  *
                       1042  *
                       1043  * Timing event schedule table
                       1044  *
                       1045  * This table would normally be calculated in background and built in RAM
                       1046  * to provide a continuously varying set of timings for the engine pulses.
                       1047  *
                       1048  * ignitx     event, tooth, fraction
                       1049  * knock      event, tooth, fraction, tooth count duration, fraction duration
                       1050  * injectx    event, tooth, fraction, time in units of 2uS, number of overflows
                       1051  * Missing tooth check flags missing teeth when unexpected and can
                       1052  * be used to re-sync the system in the event of a missing tooth pulse.
                       1053  * Toggle is a marker for the start of the engine cycle
                       1054  * Table entry starts with first byte 0 to allow 16-bit load of tooth event type
                       1055  *
00500                  1056          org     table
```

```
                        1057  event_table
00500       00080500    1058          db      0,08,05,00,0,0          ; ignition 1 falling edge
            0000
00506       002E070C    1059          db      0,46,07,12,2,00         ; injector 4
            0200
0050C       00140C00    1060          db      0,20,12,00,0,0          ; ignition 1 rising edge
            0000
00512       00200F2C    1061          db      0,32,15,44,05,31        ; knock sensor gate
            051F
00518       000A1932    1062          db      0,10,25,50,0,0          ; ignition 2 falling edge
            0000
0051E       00301B00    1063          db      0,48,27,00,2,10         ; injector 5
            020A
00524       00162028    1064          db      0,22,32,40,0,0          ; ignition 2 rising edge
            0000
0052A       00202322    1065          db      0,32,35,34,05,51        ; knock sensor gate
            0533
00530       000C2D28    1066          db      0,12,45,40,0,0          ; ignition 3 falling edge
            0000
00536       00322F17    1067          db      0,50,47,23,2,00         ; injector 6
            0200
0053C       00043200    1068          db      0,04,50,0,0,0           ; missing tooth check
            0000
00542       00183414    1069          db      0,24,52,20,0,0          ; ignition 3 rising edge
            0000
00548       00203718    1070          db      0,32,55,24,06,0         ; knock sensor gate
            0600
0054E       00043900    1071          db      0,04,57,0,0,0           ; missing tooth check
            0000
00554       00063900    1072          db      0,06,57,0,0,0           ; tooth fill interrupt
            0000
0055A       00243B00    1073          db      0,36,59,0,0,0           ; toggle port E-7 high
            0000
00560       000E4132    1074          db      0,14,65,50,0,0          ; ignition 4 falling edge
            0000
                        1075  *       db      0,56,65,00,0,0          ; injector 1 rising edge
00566       00284300    1076          db      0,40,67,00,2,20         ; injector 1
            0214
0056C       001A4828    1077          db      0,26,72,40,0,0          ; ignition 4 rising edge
            0000
                        1078  *       db      0,58,73,40,0,0          ; injector 1 falling edge
00572       00204B2A    1079          db      0,32,75,42,05,39        ; knock sensor gate
            0527
00578       00105528    1080          db      0,16,85,40,0,0          ; ignition 5 falling edge
            0000
0057E       002A571E    1081          db      0,42,87,30,2,10         ; injector 2
            020A
00584       001C5C14    1082          db      0,28,92,20,0,0          ; ignition 5 rising edge
            0000
0058A       00205F24    1083          db      0,32,95,36,05,34        ; knock sensor gate
            0522
00590       00126932    1084          db      0,18,105,50,0,0         ; ignition 6 falling edge
            0000
00596       002C6B14    1085          db      0,44,107,20,2,10        ; injector 3
            020A
0059C       00046E00    1086          db      0,04,110,0,0,0          ; missing tooth check
            0000
005A2       001E701E    1087          db      0,30,112,30,0,0         ; ignition 6 rising edge
            0000
005A8       00207300    1088          db      0,32,115,00,06,00       ; knock sensor gate
            0600
005AE       00067500    1089          db      0,06,117,0,0,0          ; missing tooth interrupt
            0000
005B4       00067500    1090          db      0,06,117,0,0,0          ; tooth fill interrupt
            0000
005BA       00267700    1091          db      0,38,119,0,0,0          ; toggle port E-7 low
            0000
                        1092  table_end
                        1093
01000                   1094          org     rom
01000 [02]  274C        1095  reset   nop
                        1096                                  ;give initial values for extension registers
                        1097                                  ;and initialize system clock and COP
01002 [02]  F50F        1098          LDAB    #$0F
01004 [02]  27FA        1099          TBEK                    ; point EK to bank F for register access
01006 [02]  F500        1100          LDAB    #$00
01008 [02]  379C        1101          TBXK                    ; point XK to bank 0
0100A [02]  379D        1102          TBYK                    ; point YK to bank 0
0100C [02]  379E        1103          TBZK                    ; point ZK to bank 0
                        1104
0100E [04]  37B50003    1105          LDD     #$0003          ; at reset, the CSBOOT block size is 512k.
```

```
01012 [06] 37FAFA48  1106          STD     CSBARBT         ; this line sets the block size to 64k since
                     1107                                  ;  that is what physically comes with the EVB16
                     1108
01016 [02] 757C      1109          LDAA    #$7C            ; w=0, x=1, y=111100
01018 [06] 177AFA04  1110          STAA    SYNCR           ; set system clock to 15.991 Mhz
                     1111
0101C [08] 1735FA21  1112          CLR     SYPCR           ; turn COP (software watchdog) off,
                     1113                                  ; since COP is on after reset
                     1114
                     1115                                  ;initialize the SCI
01020 [04] 37B50037  1116          LDD     #$0037
01024 [06] 37FAFC08  1117          STD     SCCR0           ;set the SCI baud rate to 9600 baud
                     1118
01028 [04] 37B5000C  1119          LDD     #$000C
0102C [06] 37FAFC0A  1120          STD     SCCR1           ;enable the SCI receiver and transmitter
                     1121
01030 [04] 37B5FF00  1122          LDD     #$FF00          ; enable TPU RAM at $FF0000
01034 [06] 37FAFB04  1123          STD     RAMBAR          ; store high ram array, bank 15 and enable RAM
01038 [08] 1735FB00  1124          CLR     RAMMCR          ; Array operates normally
                     1125
0103C [02] F50F      1126          LDAB    #$0F
0103E [02] 379F      1127          TBSK                ; set SK to bank 15 for system stack
01040 [04] 37BF0DFE  1128          LDS     #top_stack  ; put SP at top of 3.5k internal SRAM
                     1129  *
                     1130  * Initialisation of the SIM and CPU registers
                     1131  *
01044 [04] 37BD0000  1132          ldy     #0
01048 [04] 37B57830  1133          ldd     #$7830
0104C [06] 37FAFA4A  1134          std     CSORBT           ;set up CSBOOT with 0 wait states
01050 [04] 37BE0000  1135          ldz     #0000
                     1136  *
                     1137  * Set up RAM variables
                     1138  *
01054 [04] 37BD0500  1139          ldy     #event_table
01058 [04] 37BE0000  1140          ldz     #0000
                     1141  *
                     1142  * set up real time debug pins on port E
                     1143  * Test purposes only (for real-time debug)
                     1144  *
0105C [08] 1735FA17  1145          clr     pepar           ; set port E to data (DB8=0 at reset does this)
01060 [08] 39F0FA15  1146          bset    ddre,#$F0       ; SIZ1/PE7: 1 = tooth 59, 0 = tooth 119
                     1147                                  ; SIZ0/PE6: 1 = in CAM pulse exception handler
                     1148                                  ; AS/PE5: 1 = in Tooth filler exception handler
                     1149                                  ; DS/PE4: 1 = in MCSM exception handler
01064 [08] 1735FA13  1150          clr     porte
                     1151  *
                     1152  * Set up the CTM configuration
                     1153  *
01068 [04] 37B50F20  1154          ldd     #$0F20          ; vectors set to $4x, IARB0-2 = 7
                     1155                                  ; time base 3 selected
0106C [06] 37FAF400  1156          std     ctmmcr
                     1157  *
                     1158  * Set up the counter prescaler module and free running counter
                     1159  * 2uS timebase on TBB2
                     1160  *
01070 [04] 37B50008  1161          ldd     #$0008          ; prescaler running, divide by 64 on VSPCLK6
01074 [06] 37FAF408  1162          std     cpsmcr          ; divide by 2
01078 [04] 37B50904  1163          ldd     #$0904          ; no interrupts, arb3=1, timebase B driven
                     1164                                  ; divide by 32 selected (2uS timebase).
0107C [06] 37FAF4C8  1165          std     fcsm25sic
                     1166  *
                     1167  * Tooth counter (crank pulse counter)
                     1168  * MCSM 2 counting external pulses
                     1169  * Tooth count on TBB4
                     1170  *
01080 [04] 37B55A07  1171          ldd     #$5a07          ; interrupt level 5, arb3=1, drive timebase A
                     1172                                  ; positive edge input (IN2 pin)
                     1173                                  ; IN1 input disabled (no LOAD externally)
01084 [06] 37FAF410  1174          std     mcsm2sic
                     1175  *
                     1176  * Missing tooth detector
                     1177  * MCSM 24 generates interrupt on a missing tooth
                     1178  * Set interrupt level 6 to enable interrupts just prior to a missing tooth
                     1179  * to generate an interrupt on a missing tooth position.
                     1180  * Used to test missing tooth 58 ( CAM interrupt synchronises tooth 117)
                     1181  *
01088 [04] 37B50814  1182          ldd     #$0814          ; iarb3=1, interrupt disabled, driving bus off
                     1183                                  ; positive edge load (IN1 pin)
                     1184                                  ; IN2 input disabled
                     1185                                  ; divide by 32 clock selected
0108C [06] 37FAF4C0  1186          std     mcsm24sic
```

```
                        1187  *
                        1188  *
                        1189  * CAM pulse capture
                        1190  * DASM11 routine set up
                        1191  * DASM11 captures the 2uS timebase on the falling edge of the cam pulse
                        1192  * This routine intialises the DASM11 channel in input pulse length mode and an
                        1193  * interrupt is generated on a capture of a negative edge
                        1194  *
01090 [04] 37B53911     1195          ldd     #$3911          ; interrupt level 3 arb3=1, interrupt enabled
                        1196                                  ; timebase B (2uS)
                        1197                                  ; input pulse length mode, negative edge
01094 [06] 37FAF458     1198          std     dasm11sic
                        1199  *
                        1200  * False tooth generation
                        1201  * DASM12 routine set up
                        1202  * DASM12 compares for a time when a pulse should have been seen but where
                        1203  * the tooth is missing.
                        1204  * This routine intialises the DASM12 channel in output mode and an interrupt
                        1205  * is generated on a compare
                        1206  *
01098 [04] 37B50905     1207          ldd     #$0905          ; interrupt level 5 (when active), IARB3=1
                        1208                                  ; timebase B (2uS timebase), polarity=0
                        1209                                  ; output compare mode (flag set on A or B)
0109C [06] 37FAF460     1210          std     dasm12sic
                        1211  *
                        1212  * Crank pulse period capture
                        1213  * DASM10 routine set up
                        1214  * DASM10 gives the period of the crank timing signal
                        1215  * This routine intialises the DASM10 channel in period measurement mode
                        1216  *
010A0 [04] 37B50902     1217          ldd     #$0902          ; interrupt level 0, arb3=1, positive edge i/p
                        1218                                  ; timebase B (2uS), reset output flip-flop
                        1219                                  ; mode IPM, (mode = $2)
010A4 [06] 37FAF450     1220          std     dasm10sic
                        1221  *
                        1222  * Ignition (CTD13,14,15)
                        1223  * DASM13 routine set up
                        1224  * DASM13 compares for a time the pulse goes either high or low.
                        1225  * channel A is the falling edge, channel B the rising edge
                        1226  *
010A8 [04] 37B50915     1227          ldd     #$0915          ; interrupts disabled, IARB3=1
                        1228                                  ; timebase B (2uS timebase), EDPOL=1
                        1229                                  ; output compare mode (flag set on A or B)
010AC [06] 37FAF468     1230          std     dasm13sic
                        1231  *
                        1232  * Ignition (CTD13,14,15)
                        1233  * DASM14 routine set up
                        1234  * DASM14 compares for a time the pulse goes either high or low.
                        1235  * channel A is the falling edge, channel B the rising edge
                        1236  *
010B0 [04] 37B50915     1237          ldd     #$0915          ; interrupts disabled, IARB3=1
                        1238                                  ; timebase B (2uS timebase), EDPOL=1
                        1239                                  ; output compare mode (flag set on A or B)
010B4 [06] 37FAF470     1240          std     dasm14sic
                        1241  *
                        1242  * Ignition (CTD13,14,15)
                        1243  * DASM15 routine set up
                        1244  * DASM15 compares for a time the pulse goes either high or low.
                        1245  * channel A is the falling edge, channel B the rising edge
                        1246  *
010B8 [04] 37B50915     1247          ldd     #$0915          ; interrupts disabled, IARB3=1
                        1248                                  ; timebase B (2uS timebase), EDPOL=1
                        1249                                  ; output compare mode (flag set on A or B)
010BC [06] 37FAF478     1250          std     dasm15sic
                        1251  *
                        1252  * Injection (CTD16,17,18)
                        1253  * DASM16,17,18 routine set up
                        1254  * DASM16 compares for a time the pulse goes either high or low.
                        1255  * channel A is the rising edge, channel B the falling edge
                        1256  *
010C0 [04] 37B50904     1257          ldd     #$0904          ; interrupts disabled, IARB3=1
                        1258                                  ; timebase B (2uS timebase), EDPOL=0
                        1259                                  ; output compare mode (flag set on B)
010C4 [06] 37FAF480     1260          std     dasm16sic
                        1261  *
                        1262  * Injection (CTD16,17,18)
                        1263  * DASM17 routine set up
                        1264  * DASM17 compares for a time the pulse goes either high or low.
                        1265  * channel A is the rising edge, channel B the falling edge
                        1266  *
010C8 [04] 37B50904     1267          ldd     #$0904          ; interrupts disabled, IARB3=1
```

```
                        1268                                    ; timebase B (2uS timebase), EDPOL=0
                        1269                                    ; output compare mode (flag set on B)
010CC [06] 37FAF488     1270            std     dasm17sic
                        1271    *
                        1272    * Injection (CTD16,17,18)
                        1273    * DASM18 routine set up
                        1274    * DASM18 compares for a time the pulse goes either high or low.
                        1275    * channel A is the rising edge, channel B the falling edge
                        1276    *
010D0 [04] 37B50904     1277            ldd     #$0904          ; interrupts disabled, IARB3=1
                        1278                                    ; timebase B (2uS timebase), EDPOL=0
                        1279                                    ; output compare mode (flag set on B)
010D4 [06] 37FAF490     1280            std     dasm18sic
                        1281    *
                        1282    * knock gate (CTD19)
                        1283    * DASM19 routine set up
                        1284    * DASM19 compares for a time the pulse goes either high or low.
                        1285    * channel A is the rising edge, channel B the falling edge
                        1286    *
010D8 [04] 37B50904     1287            ldd     #$0904          ; interrupts disabled, IARB3=1
                        1288                                    ; timebase B (2uS timebase), EDPOL=0
                        1289                                    ; output compare mode (flag set on B)
010DC [06] 37FAF498     1290            std     dasm19sic
                        1291    *
                        1292    * INITIALISATION CODE TO START UP SYSTEM
                        1293    *
                        1294    sync
010E0 [10] 3616         1295            bsr     search_miss
                        1296    *
                        1297    * ENABLE INTERRUPT ON MCSM2
010E2 [08] 3880F410     1298            bclr    mcsm2sic,#flag
010E6 [04] 37B5FFFF     1299            ldd     #$FFFF
010EA [06] 37FAF412     1300            std     mcsm2cnt        ; interrupt on next crank pulse (overflow)
                        1301    *
010EE [02] 372C         1302            tpd
010F0 [04] 37B6FF1F     1303            andd    #$ff1f          ; clear the interrupt mask
010F4 [02] 372D         1304            tdp
                        1305    *
                        1306    * Starts with pending interrupts - highest interrupt mask is tooth counter
                        1307    * (MCSM2 = ILV 6)
                        1308    *
                        1309    main_loop               ; this would normally calculate the event table
010F6 [08] 3870FA13     1310            bclr    porte,#bit4+bit5+bit6   ; CPU not in interrupt routine
010FA [06] B0F6         1311            bra     main_loop
                        1312    *
                        1313    *
                        1314    * Missing tooth search
                        1315    * MCSM 24 runs off 2uS timebase A
                        1316    * Finds the shortest period and then exits from the routine when the gap
                        1317    * between pulses is greater that 1.5 times the previous pulse period
                        1318    *
                        1319    search_miss
010FC [08] 2771F452     1320            lded    dasm10a         ; load dasm10a and dasm10b pulse times
01100 [02] 2779         1321            sde                     ; calculate the pulse period from the crank
01102 [02] 27FB         1322            ted                     ; save accE
01104 [02] 27FF         1323            lsrd                    ; divide period by 2
01106 [02] 2778         1324            ade                     ; add to period of last pulse
01108 [02] 2772         1325            nege                    ; obtain 2's complement for modulus counter
0110A [06] 377AF4C2     1326            ste     mcsm24cnt       ; place number in MCSM 24 modulus latch
                        1327                                    ; and initialise counter at same time
0110E [08] 3880F4C0     1328            bclr    mcsm24sic,#flag ; clear the overflow flag
01112 [08] 3880F450     1329            bclr    dasm10sic,#flag
01116 [10] 3B80F450     1330    no_gap  brset   dasm10sic,#flag,search_miss
           FFE0
0111C [10] 3A80F4C0     1331            brclr   mcsm24sic,#flag,no_gap
           FFF4
                        1332    *
                        1333    * Find CAM signal as start condition
                        1334    * Could do the following, but could wait for 2 revolutions of the engine
                        1335    *
01122 [10] 3A80F459     1336            brclr   dasm11sic+1,#$80,no_gap  ; check CTD11 pin state
           FFEE
                        1337                                            ; if =1 then at cam position
                        1338    *
                        1339    *
01128 [08] 2771F452     1340            lded    dasm10a         ;10 load dasm10a and dasm10b pulse times
0112C [02] 2779         1341            sde                     ; 3 calculate the pulse period from the crank
0112E [06] 377A0000     1342            ste     period          ; 8 and save this in RAM
01132 [06] 3771F452     1343            adde    dasm10a         ; 8 add this pulse time to the period
                        1344                                    ;   could use ALSE, ADE (4 cycles) but would
                        1345                                    ;  overflow on ALSE when period >=$8000 (16rpm)
```

```
01136 [06] 377A0002  1346          ste     next_pulse      ; 8 and save next pulse time in RAM
0113A [02] 7500      1347          ldaa    #0
0113C [06] 177A0007  1348          staa    next_tooth
01140 [12] 27F7      1349          rts
                     1350  *
01142                1351    PAGE
                     1352  ********************************************************************* *******
                     1353  *       TOOTH MISSED INTERRUPT ROUTINE
                     1354  *
                     1355  *       MCSM24 overflows 2uS timebase is missing tooth detected
                     1356  *       and then generates an interrupt.
                     1357  *       Normally run from tooth position 57.
                     1358  *       Once engine is running normally this routine just checks that
                     1359  *       the tooth count is 58 when the interrupt occurs (ie tooth filler
                     1360  *       edge at appropriate time)
                     1361  *       If tooth filler in wrong position, tooth count will be 57
                     1362  *
                     1363  *       If correct - 48 cycles + 20 cycle entry latency = 64 cycles (1)
                     1364  *                    59 cycles + 20 cycle entry latency = 79 cycles (2)
                     1365  ********************************************************************* *******
                     1366  *                                  ; cycles
                     1367  tooth_missed
01142 [04] 3401      1368          pshm    d                 ; 7
01144 [08] 38F0F4C0  1369          bclr    mcsm24sic,#$F0    ; 8 clear the flag and disable interrupts
01148 [06] 17F50006  1370          ldab    tooth             ; 7 get the current tooth number
0114C [02] F83A      1371          cmpb    #58               ; 3 is it tooth number 58?
0114E [02] B702      1372          beq     miss_good         ;7,3 If not then it must be a fault
01150 [02] F876      1373          cmpb    #118              ; 3 is it tooth number 118?
01152 [02] B7FE      1374          beq     miss_good         ;7,3 If not then must be a fault
01154 [06] B000      1375          bra     go_miss           ; 7
                     1376  miss_good
01156 [04] 3540      1377          pulm    d                 ; 7
01158 [12] 2777      1378          rti                       ;14
                     1379  go_miss
0115A [08] 39E00DFD  1380          bset    top_stack-1,#$E0 ; interrupt mask will retrurn at level 7
0115E [04] 37B510E6  1381          ldd     #sync+6           ; RTI subtracts 4 from the PC on return
01162 [06] 37FA0DFE  1382          std     top_stack         ; return address is start of sync routine
01166 [04] 37BF0DFA  1383          lds     #top_stack-4
0116A [12] 2777      1384          rti                       ;14
                     1385  *
                     1386  ********************************************************************* *******
                     1387  *       CAM PULSE INTERRUPT ROUTINE
                     1388  *
                     1389  *       DASM11 used to capture timebase B (2uS)
                     1390  *       Once engine is running normally this routine just checks that
                     1391  *       the tooth count is 117 when the CAM pulse reaches the rising edge
                     1392  *
                     1393  *       Good cam position takes 20 cycles entry + 55 cycles = 74 cycles (1)
                     1394  *                                               + 66 cycles = 85 cycles (2)
                     1395  *       Bad cam position is 78 cycles (this should not happen)
                     1396  *
                     1397  ********************************************************************* *******
                     1398  *                                  ; cycles
0116C [08] 3940FA13  1399  cam     bset    porte,#bit6       ; set SIZ0/PE6 =1 as a marker (test only)
01170 [04] 3403      1400          pshm    d,e               ; 9
01172 [08] 3880F458  1401          bclr    dasm11sic,#flag ;10  clear the flag
01176 [06] 17F50006  1402          ldab    tooth             ; 7
0117A [02] F800      1403          cmpb    #0                ; 3  check which tooth number the CAM pulse
0117C [02] B604      1404          bne     badcam            ;7,3 occurs at
0117E [08] 2771F45A  1405          lded    dasm11a           ;10
01182 [08] 2773000A  1406          sted    cam_pulse         ;10 save the times for the CAM pulse edges
                     1407  badcam
01186 [04] 3560      1408          pulm    d,e               ; 9
01188 [12] 2777      1409          rti                       ;14
0118A                1410    PAGE
```

```
                        1411  ********************************************************************* *******
                        1412  *       TOOTH FILLER INTERRUPT ROUTINE
                        1413  *
                        1414  *       Interrupt generated by DASM12 (2uS timebase).
                        1415  *       Increments tooth count for teeth 58,59,118 and 119.
                        1416  *       Teeth 59 and 119 are special cases since the interrupt must be
                        1417  *       disabled after these to prevent a tooth filler interrupt at
                        1418  *       teeth 0 and 60
                        1419  *       This routine also calculates the next pulse position for the
                        1420  *       missing teeth based on the most recent, valid period measurement
                        1421  *       There is no need to use the MCSM counter interrupt in this case
                        1422  *       as there is always an event following the tooth filler event
                        1423  *
                        1424  *       EK = $F for this routine
                        1425  *       IY points to event table.
                        1426  *
                        1427  *       interrupt handler execution times
                        1428  *       tooth 58,118    56 cycles (1),     73 cycles (2)
                        1429  *       tooth 59,119    65 cycles (1),     87 cycles (2)
                        1430  *
                        1431  ********************************************************************* *******
                        1432  *                                       ; cycles
                        1433  tooth_filler
0118A [08] 3920FA13     1434          bset    porte,#bit5     ; set AS/PE5 = 1 as a marker (test only)
0118E [04] 3407         1435          pshm    d,e,x           ;11 push D, E and IX onto stack
01190 [08] 3880F460     1436          bclr    dasm12sic,#flag ;10 clear flag
01194 [08] 17330006     1437          inc     tooth           ; 8 increment tooth count
01198 [06] 37F50002     1438          ldd     next_pulse      ; 8 Get this pulse time
0119C [06] 37F10000     1439          addd    period          ; 8 and add last measured period
011A0 [06] 37FA0002     1440          std     next_pulse      ; 8 and save next pulse time in RAM
011A4 [10] 3B80F461     1441          brset   dasm12sic+1,#$80,event_a
           0004
                        1442                                  ;13,17 if ctd12=1 then first event
                        1443  event_b                         ;if ctd12=0 then must be 2nd tooth fill
011AA [08] 3870F460     1444          bclr    dasm12sic,#$70  ;10 disable interrupts from tooth filler
                        1445  event_a                         ; first tooth fill, so leave dasm12 running
011AE [06] B036         1446          bra     event_loop      ; 7
                        1447  *
                        1448  ********************************************************************* *******
                        1449  *       CRANK PULSE MATCH INTERRUPT ROUTINE
                        1450  *
                        1451  *       Interrupt generated by MCSM2  (overflow on a tooth count)
                        1452  *       Program first read the value of the tooth count (next_tooth) and then
                        1453  *       selects the appropriate event to be set up. After each event has
                        1454  *       been initialised the next table entry is checked until all events
                        1455  *       related to this tooth count are complete. The routine then returns
                        1456  *       control to the background task (RTI).
                        1457  *
                        1458  *       Timebase is assumed to be 2uS
                        1459  *       With 60 teeth and a 16 bit counter, minimum engine speed is
                        1460  *       8rpm for ignition routines
                        1461  *       approximately 120 rpm for knock routines since valid for up to 1/4th
                        1462  *       of an engine revolution (15 teeth duration max)
                        1463  *       960 rpm for injector pulse (this can extend over 2 engine revolutions)
                        1464  *       8 rpm for independent rising and falling edges on injectors
                        1465  *
                        1466  *       EK = $F for this routine
                        1467  *       IY points to event table.
                        1468  *
                        1469  *       interrupt entry latency
                        1470  *                         38 cycles (1), 39 cycles (2) (for EDIVS)
                        1471  *                       + 20 cycles (1), 26 cycles (2) interrupt routine entry
                        1472  *                       = 58 cycles (1),=65 cycles (2)
                        1473  *
                        1474  *       Main routine takes the following plus subroutine time to execute ...
                        1475  *       tooth 0                 183 cycles (1),     230 cycles (2)
                        1476  *       tooth 60                187 cycles (1),     236 cycles (2)
                        1477  *       any other tooth         199 cycles (1),     252 cycles (2)
                        1478  *
                        1479  *       If entered from tooth filler interrupt then this routine takes
                        1480  *       fewer cycles and so is not considered for worst case timing
                        1481  *
                        1482  ********************************************************************* *******
                        1483  *
```

```
                        1484  *                               ; cycles
                        1485  int
011B0 [08] 3910FA13     1486          bset    porte,#bit4     ;10 entering interrupt routine (test only)
011B4 [04] 3407         1487          pshm    d,e,x           ;11 push D, E and IX onto stack
011B6 [08] 3880F410     1488          bclr    mcsm2sic,#flag  ;10 clear status reg flag
011BA [06] 17F50007     1489          ldab    next_tooth      ; 7 get the tooth count
011BE [06] 17FA0006     1490          stab    tooth           ; 5 save the tooth count
011C2 [02] 3716         1491          tstb                    ; 3 check if tooth 0 or 60 since the missing
011C4 [02] B714         1492          beq     tooth_060       ;7,3 teeth will corrupt the period measurement
011C6 [02] F83C         1493          cmpb    #60             ; 3
011C8 [02] B710         1494          beq     tooth_060       ;7,3
011CA [08] 2771F452     1495          lded    dasm10a         ;10 load dasm10a and dasm10b pulse times
011CE [02] 2779         1496          sde                     ; 3 calculate the pulse period from the crank
011D0 [06] 377A0000     1497          ste     period          ; 8 and save this in RAM
011D4 [06] 3771F452     1498          adde    dasm10a         ; 8 add this pulse time to the period
                        1499                                  ;   could use ALSE, ADE (4 cycles) but would
                        1500                                  ;  overflow on ALSE when period >=$8000 (16rpm)
011D8 [06] 377A0002     1501          ste     next_pulse      ; 8 and save next pulse time in RAM
011DC [06] B008         1502          bra     event_loop      ; 7
                        1503  tooth_060
011DE [06] 37750002     1504          lde     next_pulse      ; 8 Get this pulse time
011E2 [06] 37710000     1505          adde    period          ; 8 and add last measured period
011E6 [06] 377A0002     1506          ste     next_pulse      ; 8 and save next pulse time in RAM
                        1507  *
                        1508  * now check which event(s) needs to be initialised
                        1509  *
                        1510  event_loop
011EA [04] 37BC0400     1511          ldx     #event_list     ; 6 IX points to top of event correlation table
011EE [06] 17750006     1512          ldaa    tooth           ; 8 get tooth count again
011F2 [06] 5802         1513          cmpa    2,y             ; 8 compare with next element in event table
011F4 [02] B614         1514          bne     schedule        ;7,3 quit int routine if no event scheduled
011F6 [06] 37550000     1515          lde     0,y             ; 8 get the event code from the event table
011FA [06] 2785         1516          ldd     e,x             ; 7 get address
011FC [02] 37CC         1517          xgdx                    ; 3 and place in X
011FE [12] 89000000     1518          jsr     0,x             ;14 jump to event interrupt handler
01202 [02] 3D06         1519          aiy     #6              ; 3 move event table pointer to next entry
01204 [04] 377D05C0     1520          cpy     #table_end      ; 6 check if past end of event table listing
01208 [02] B6DC         1521          bne     event_loop      ;7,3 if not then continue around loop
0120A [04] 37BD0500     1522          ldy     #event_table    ; 6 if yes then set pointer to start of table
                        1523                                  ;   and return to main routine again
                        1524  *
                        1525  * Now that the pulses have been set up, load up MCSM2 modulus register
                        1526  * with the tooth count for the next interrupt
                        1527  * Don't worry about missing tooth position as this is self governing.
                        1528  * This is because the tooth filler handles the tooth count and
                        1529  * there are always events on teeth numbers 57,58,59,117,118,119
                        1530  * and teeth 0 and 60 are special cases where the tooth period is taken
                        1531  * from 3 teeth previously
                        1532  *
                        1533  schedule
0120E [06] D502         1534          ldab    2,y             ; 7 subtract next tooth event
01210 [06] 17FA0007     1535          stab    next_tooth      ; 5 save this value in RAM
01214 [02] 3705         1536          clra                    ; 3
01216 [06] 17F00006     1537          subb    tooth           ; 7 get current tooth number and then
0121A [02] B4FE         1538          bcc     no_neg          ; 7,3
0121C [02] 3700         1539          coma                    ; 3
0121E [02] 27F2         1540  no_neg  negd                    ; 3 2's compliment for modulus counter
01220 [02] BB00         1541          bmi     sched_ok        ;7,3 check for going back to start of table
01222 [04] 37B00078     1542          subd    #120            ; 6 if yes, then subtract 120 teeth
                        1543  sched_ok
01226 [06] 37FAF412     1544          std     mcsm2cnt        ; 8 initialise modulus count
0122A [04] 3570         1545          pulm    d,e,x           ;11 restore accumulators D and E and index IX
0122C [12] 2777         1546          rti                     ;14 return from interrupt
                        1547  *
```

```
                        1548  ********************************************************************* *******
                        1549  *
                        1550  * SUBROUTINES FOR INITIALISING PULSES FOR IGNITION, INJECTION AND KNOCK GATE
                        1551  *
                        1552  * tooth fill interrupt generator    takes  55 cycles (1) and  70 cycles (2)
                        1553  * Ignition pulse - rising/falling   takes  87 cycles (1) and 103 cycles (2)
                        1554  * Knock sensor gate pulse           takes 203 cycles (1) and 244 cycles (2)
                        1555  * Injector - both edges min 916 rpm takes  99 cycles (1) and 118 cycles (2)
                        1556  * Injector - single edge min 8 rpm  takes  87 cycles (1) and 103 cycles (2)
                        1557  * Missing tooth detect              takes  38 cycles (1) and  53 cycles (2)
                        1558  *
                        1559  ********************************************************************* *******
                        1560  *
                        1561  * missing tooth interrupt generator
                        1562  * takes 55 cycles (1),  70 cycles (2)
                        1563  *
                        1564  *                                  ;cycles
                        1565  tooth_fill                        ;   06     dasm12    missing tooth position
0122E [06] 37F50002     1566          ldd     next_pulse        ; 7 set up an interrupt on the missing tooth
01232 [06] 37FAF462     1567          std     dasm12a           ; 8 set up compare on next tooth position
01236 [06] 37F10000     1568          addd    period            ; 8 set up interrupt on 2nd missing tooth
0123A [06] 37FAF464     1569          std     dasm12b           ; 8
0123E [08] 3880F460     1570          bclr    dasm12sic,#flag   ;10 clear flag of dasm12
01242 [02] 7549         1571          ldaa    #$49              ; 7
01244 [06] 177AF460     1572          staa    dasm12sic         ; 8 enable interrupts on DASM12
01248 [12] 27F7         1573          rts                       ;14
                        1574  *
                        1575  * Ignition pulse - falling edge
                        1576  * takes 87 cycles (1),   103 cycles (2)
                        1577  *
                        1578  *                                  ;cycles
                        1579  ignit1f                           ;   08     dasm13a    ignition 1 falling edge
0124A [08] 3880F468     1580          bclr    dasm13sic,#flag   ;10 set to logic 0 on compare also clears flag
0124E [06] 37750000     1581          lde     period            ; 8 get period from RAM
01252 [06] D503         1582          ldab    3,y               ; 7 get duty as tenths of a degree
01254 [02] 3705         1583          clra                      ; 3
01256 [10] 3725         1584          emul                      ;11
01258 [04] 37BC003C     1585          ldx     #sixty            ; 6 period * tenths
0125C [24] 3728         1586          ediv                      ;25 ---------------  =  edge offset
0125E [02] 37CC         1587          xgdx                      ; 3     sixty
01260 [06] 37F10002     1588          addd    next_pulse        ; 8 add time for next crank pulse
01264 [06] 37FAF46A     1589          std     dasm13a           ; 8
01268 [12] 27F7         1590          rts                       ;14
                        1591  *
                        1592  *                                  ;cycles
                        1593  ignit2f                           ;   10     dasm14a    ignition 1 falling edge
0126A [08] 3880F470     1594          bclr    dasm14sic,#flag   ;10 set to logic 0 on compare also clears flag
0126E [06] 37750000     1595          lde     period            ; 8 get period from RAM
01272 [06] D503         1596          ldab    3,y               ; 7 get duty as tenths of a degree
01274 [02] 3705         1597          clra                      ; 3
01276 [10] 3725         1598          emul                      ;11
01278 [04] 37BC003C     1599          ldx     #sixty            ; 6 period * tenths
0127C [24] 3728         1600          ediv                      ;25 ---------------  =  edge offset
0127E [02] 37CC         1601          xgdx                      ; 3     sixty
01280 [06] 37F10002     1602          addd    next_pulse        ; 8 add time for next crank pulse
01284 [06] 37FAF472     1603          std     dasm14a           ; 8
01288 [12] 27F7         1604          rts                       ;14
                        1605  *
                        1606  *                                  ;cycles
                        1607  ignit3f                           ;   12     dasm15a    ignition 1 falling edge
0128A [08] 3880F478     1608          bclr    dasm15sic,#flag   ;10 set to logic 0 on compare also clears flag
0128E [06] 37750000     1609          lde     period            ; 8 get period from RAM
01292 [06] D503         1610          ldab    3,y               ; 7 get duty as tenths of a degree
01294 [02] 3705         1611          clra                      ; 3
01296 [10] 3725         1612          emul                      ;11
01298 [04] 37BC003C     1613          ldx     #sixty            ; 6 period * tenths
0129C [24] 3728         1614          ediv                      ;25 ---------------  =  edge offset
0129E [02] 37CC         1615          xgdx                      ; 3     sixty
012A0 [06] 37F10002     1616          addd    next_pulse        ; 8 add time for next crank pulse
012A4 [06] 37FAF47A     1617          std     dasm15a           ; 8
012A8 [12] 27F7         1618          rts                       ;14
                        1619  *
                        1620  *                                  ;cycles
                        1621  ignit4f                           ;   14     dasm13a    ignition 1 falling edge
012AA [08] 3880F468     1622          bclr    dasm13sic,#flag   ;10 set to logic 0 on compare also clears flag
012AE [06] 37750000     1623          lde     period            ; 8 get period from RAM
012B2 [06] D503         1624          ldab    3,y               ; 7 get duty as tenths of a degree
012B4 [02] 3705         1625          clra                      ; 3
012B6 [10] 3725         1626          emul                      ;11
012B8 [04] 37BC003C     1627          ldx     #sixty            ; 6 period * tenths
012BC [24] 3728         1628          ediv                      ;25 ---------------  =  edge offset
```

```
012BE [02] 37CC      1629          xgdx                    ; 3     sixty
012C0 [06] 37F10002  1630          addd    next_pulse      ; 8 add time for next crank pulse
012C4 [06] 37FAF46A  1631          std     dasm13a         ; 8
012C8 [12] 27F7      1632          rts                     ;14
                     1633  *
                     1634  *                               ;cycles
                     1635  ignit5f                         ;   16     dasm14a    ignition 1 falling edge
012CA [08] 3880F470  1636          bclr    dasm14sic,#flag ;10 set to logic 0 on compare also clears flag
012CE [06] 37750000  1637          lde     period          ; 8 get period from RAM
012D2 [06] D503      1638          ldab    3,y             ; 7 get duty as tenths of a degree
012D4 [02] 3705      1639          clra                    ; 3
012D6 [10] 3725      1640          emul                    ;11
012D8 [04] 37BC003C  1641          ldx     #sixty          ; 6 period * tenths
012DC [24] 3728      1642          ediv                    ;25 ---------------  =  edge offset
012DE [02] 37CC      1643          xgdx                    ; 3     sixty
012E0 [06] 37F10002  1644          addd    next_pulse      ; 8 add time for next crank pulse
012E4 [06] 37FAF472  1645          std     dasm14a         ; 8
012E8 [12] 27F7      1646          rts                     ;14
                     1647  *
                     1648  *                               ;cycles
                     1649  ignit6f                         ;   18     dasm15a    ignition 1 falling edge
012EA [08] 3880F478  1650          bclr    dasm15sic,#flag ;10 set to logic 0 on compare also clears flag
012EE [06] 37750000  1651          lde     period          ; 8 get period from RAM
012F2 [06] D503      1652          ldab    3,y             ; 7 get duty as tenths of a degree
012F4 [02] 3705      1653          clra                    ; 3
012F6 [10] 3725      1654          emul                    ;11
012F8 [04] 37BC003C  1655          ldx     #sixty          ; 6 period * tenths
012FC [24] 3728      1656          ediv                    ;25 ---------------  =  edge offset
012FE [02] 37CC      1657          xgdx                    ; 3     sixty
01300 [06] 37F10002  1658          addd    next_pulse      ; 8 add time for next crank pulse
01304 [06] 37FAF47A  1659          std     dasm15a         ; 8
01308 [12] 27F7      1660          rts                     ;14
                     1661  *
                     1662  * Ignition pulse - rising edge
                     1663  * takes 87 cycles (1),   103 cycles (2)
                     1664  *
                     1665  *                               ;cycles
                     1666  ignit1r                         ;   20     dasm13b    ignition 1 rising edge
0130A [08] 3880F468  1667          bclr    dasm13sic,#flag ;10 set to logic 1 on compare also clears flag
0130E [06] 37750000  1668          lde     period          ; 8 get period from RAM
01312 [06] D503      1669          ldab    3,y             ; 7 get duty as tenths of a degree
01314 [02] 3705      1670          clra                    ; 3
01316 [10] 3725      1671          emul                    ;11
01318 [04] 37BC003C  1672          ldx     #sixty          ; 6 period * tenths
0131C [24] 3728      1673          ediv                    ;25 ---------------  =  edge offset
0131E [02] 37CC      1674          xgdx                    ; 3     sixty
01320 [06] 37F10002  1675          addd    next_pulse      ; 8 add time for next crank pulse
01324 [06] 37FAF46C  1676          std     dasm13b         ; 8
01328 [12] 27F7      1677          rts                     ;14
                     1678  *
                     1679  *                               ;cycles
                     1680  ignit2r                         ;   22     dasm14b    ignition 1 rising edge
0132A [08] 3880F470  1681          bclr    dasm14sic,#flag ;10 set to logic 1 on compare also clears flag
0132E [06] 37750000  1682          lde     period          ; 8 get period from RAM
01332 [06] D503      1683          ldab    3,y             ; 7 get duty as tenths of a degree
01334 [02] 3705      1684          clra                    ; 3
01336 [10] 3725      1685          emul                    ;11
01338 [04] 37BC003C  1686          ldx     #sixty          ; 6 period * tenths
0133C [24] 3728      1687          ediv                    ;25 ---------------  =  edge offset
0133E [02] 37CC      1688          xgdx                    ; 3     sixty
01340 [06] 37F10002  1689          addd    next_pulse      ; 8 add time for next crank pulse
01344 [06] 37FAF474  1690          std     dasm14b         ; 8
01348 [12] 27F7      1691          rts                     ;14
                     1692  *
                     1693  *                               ;cycles
                     1694  ignit3r                         ;   24     dasm15b    ignition 1 rising edge
0134A [08] 3880F478  1695          bclr    dasm15sic,#flag ;10 set to logic 1 on compare also clears flag
0134E [06] 37750000  1696          lde     period          ; 8 get period from RAM
01352 [06] D503      1697          ldab    3,y             ; 7 get duty as tenths of a degree
01354 [02] 3705      1698          clra                    ; 3
01356 [10] 3725      1699          emul                    ;11
01358 [04] 37BC003C  1700          ldx     #sixty          ; 6 period * tenths
0135C [24] 3728      1701          ediv                    ;25 ---------------  =  edge offset
0135E [02] 37CC      1702          xgdx                    ; 3     sixty
01360 [06] 37F10002  1703          addd    next_pulse      ; 8 add time for next crank pulse
01364 [06] 37FAF47C  1704          std     dasm15b         ; 8
01368 [12] 27F7      1705          rts                     ;14
                     1706  *
                     1707  *                               ;cycles
                     1708  ignit4r                         ;   26     dasm13b    ignition 1 rising edge
0136A [08] 3880F468  1709          bclr    dasm13sic,#flag ;10 set to logic 1 on compare also clears flag
```

```
0136E [06] 37750000  1710          lde     period          ; 8 get period from RAM
01372 [06] D503      1711          ldab    3,y             ; 7 get duty as tenths of a degree
01374 [02] 3705      1712          clra                    ; 3
01376 [10] 3725      1713          emul                    ;11
01378 [04] 37BC003C  1714          ldx     #sixty          ; 6 period * tenths
0137C [24] 3728      1715          ediv                    ;25 ---------------  =  edge offset
0137E [02] 37CC      1716          xgdx                    ; 3     sixty
01380 [06] 37F10002  1717          addd    next_pulse      ; 8 add time for next crank pulse
01384 [06] 37FAF46C  1718          std     dasm13b         ; 8
01388 [12] 27F7      1719          rts                     ;14
                     1720  *
                     1721  *                               ;cycles
                     1722  ignit5r                         ;   28     dasm14b    ignition 1 rising edge
0138A [08] 3880F470  1723          bclr    dasm14sic,#flag ;10 set to logic 1 on compare also clears flag
0138E [06] 37750000  1724          lde     period          ; 8 get period from RAM
01392 [06] D503      1725          ldab    3,y             ; 7 get duty as tenths of a degree
01394 [02] 3705      1726          clra                    ; 3
01396 [10] 3725      1727          emul                    ;11
01398 [04] 37BC003C  1728          ldx     #sixty          ; 6 period * tenths
0139C [24] 3728      1729          ediv                    ;25 ---------------  =  edge offset
0139E [02] 37CC      1730          xgdx                    ; 3     sixty
013A0 [06] 37F10002  1731          addd    next_pulse      ; 8 add time for next crank pulse
013A4 [06] 37FAF474  1732          std     dasm14b         ; 8
013A8 [12] 27F7      1733          rts                     ;14
                     1734  *
                     1735  *                               ;cycles
                     1736  ignit6r                         ;   30     dasm15b    ignition 1 rising edge
013AA [08] 3880F478  1737          bclr    dasm15sic,#flag ;10 set to logic 1 on compare also clears flag
013AE [06] 37750000  1738          lde     period          ; 8 get period from RAM
013B2 [06] D503      1739          ldab    3,y             ; 7 get duty as tenths of a degree
013B4 [02] 3705      1740          clra                    ; 3
013B6 [10] 3725      1741          emul                    ;11
013B8 [04] 37BC003C  1742          ldx     #sixty          ; 6 period * tenths
013BC [24] 3728      1743          ediv                    ;25 ---------------  =  edge offset
013BE [02] 37CC      1744          xgdx                    ; 3     sixty
013C0 [06] 37F10002  1745          addd    next_pulse      ; 8 add time for next crank pulse
013C4 [06] 37FAF47C  1746          std     dasm15b         ; 8
013C8 [12] 27F7      1747          rts                     ;14
                     1748  *
                     1749  * Knock sensor gate pulse
                     1750  * takes 203 cycles (1),   244 cycles (2)
                     1751  *
                     1752  *                               ;cycles
                     1753  knock                           ;   32     dasm11     knock sensor gate
013CA [04] 3420      1754          pshm    #$20            ; 7 Push K register
013CC [08] 3880F498  1755          bclr    dasm19sic,#flag ;10
                     1756  * rising edge
013D0 [06] 37750000  1757          lde     period          ; 8 get period from RAM
013D4 [06] D503      1758          ldab    3,y             ; 7 get duty as tenths of a degree
013D6 [02] 3705      1759          clra                    ; 3
013D8 [10] 3725      1760          emul                    ;11
013DA [04] 37BC003C  1761          ldx     #sixty          ; 6 period * tenths
013DE [24] 3728      1762          ediv                    ;25 ---------------  =  edge offset
013E0 [02] 37CC      1763          xgdx                    ; 3     sixty
013E2 [06] 37FA000C  1764          std     knock_tooth     ; 8 save the start edge fraction
013E6 [06] 37F10002  1765          addd    next_pulse      ; 8 add time for rising edge of next crank pulse
013EA [06] 37FAF49A  1766          std     dasm19a         ; 8
                     1767  * falling edge
013EE [06] 37750000  1768          lde     period          ; 8 get period from RAM
013F2 [06] D505      1769          ldab    5,y             ; 7 get the fractional offset
013F4 [02] 3705      1770          clra                    ; 3
013F6 [10] 3725      1771          emul                    ;11                          tenths of a degree
013F8 [04] 37BC003C  1772          ldx     #sixty          ; 6 period * tenths
013FC [24] 3728      1773          ediv                    ;25 ---------------  =  edge offset
                     1774                                  ;       sixty
013FE [06] 37750000  1775          lde     period          ; 8 get period from RAM
01402 [06] D504      1776          ldab    4,y             ; 7 get number of teeth
01404 [02] 3705      1777          clra                    ; 3
01406 [10] 3725      1778          emul                    ;11 result (period * teeth) in accD (16 bits)
01408 [02] 37CD      1779          adx                     ; 3 add partial tooth time to integer teeth time
                     1780                                  ;   note that ADX corrupts XK if accD msb set
0140A [02] 37CC      1781          xgdx                    ; 3 put result in acc D
0140C [06] 37F1000C  1782          addd    knock_tooth     ; 8 add the start edge fraction
01410 [06] 37F10002  1783          addd    next_pulse      ; 8 add time for rising edge of next crank pulse
01414 [06] 37FAF49C  1784          std     dasm19b         ; 8
01418 [04] 3502      1785          pulm    #$02            ; 7 Pull K register
0141A [12] 27F7      1786          rts                     ;14
                     1787  *
                     1788  * Injector - both edges
                     1789  * minumum 916 rpm
                     1790  * takes 99 cycles (1),   118 cycles (2)
```

```
                        1791  *
                        1792  *                                ;cycles
                        1793  inject1                          ;   40     dasm16    injector 1 both edges
0141C [08] 3880F480     1794          bclr    dasm16sic,#flag ;10
                        1795  * rising edge
01420 [06] 37750000     1796          lde     period          ; 8 get period from RAM
01424 [06] D503         1797          ldab    3,y             ; 7 get duty as tenths of a degree
01426 [02] 3705         1798          clra                    ; 3
01428 [10] 3725         1799          emul                    ;11
0142A [04] 37BC003C     1800          ldx     #sixty          ; 6 period * tenths
0142E [24] 3728         1801          ediv                    ;25 ---------------  =  edge offset
01430 [02] 37CC         1802          xgdx                    ; 3     sixty
01432 [06] 37F10002     1803          addd    next_pulse      ; 8 add time for rising edge of next crank pulse
01436 [06] 37FAF482     1804          std     dasm16a         ; 8
                        1805  * falling edge
0143A [06] 9104         1806          addd    4,y             ; 7 add time for injector on
0143C [06] 37FAF484     1807          std     dasm16b         ; 8
01440 [12] 27F7         1808          rts                     ;14
                        1809  *
                        1810  *                                ;cycles
                        1811  inject2                          ;   42     dasm17    injector 1 both edges
01442 [08] 3880F488     1812          bclr    dasm17sic,#flag ;10
                        1813  * rising edge
01446 [06] 37750000     1814          lde     period          ; 8 get period from RAM
0144A [06] D503         1815          ldab    3,y             ; 7 get duty as tenths of a degree
0144C [02] 3705         1816          clra                    ; 3
0144E [10] 3725         1817          emul                    ;11
01450 [04] 37BC003C     1818          ldx     #sixty          ; 6 period * tenths
01454 [24] 3728         1819          ediv                    ;25 ---------------  =  edge offset
01456 [02] 37CC         1820          xgdx                    ; 3     sixty
01458 [06] 37F10002     1821          addd    next_pulse      ; 8 add time for rising edge of next crank pulse
0145C [06] 37FAF48A     1822          std     dasm17a         ; 8
                        1823  * falling edge
01460 [06] 9104         1824          addd    4,y             ; 7 add time for injector on
01462 [06] 37FAF48C     1825          std     dasm17b         ; 8
01466 [12] 27F7         1826          rts                     ;14
                        1827  *
                        1828  *                                ;cycles
                        1829  inject3                          ;   44     dasm18    injector 1 both edges
01468 [08] 3880F490     1830          bclr    dasm18sic,#flag ;10
                        1831  * rising edge
0146C [06] 37750000     1832          lde     period          ; 8 get period from RAM
01470 [06] D503         1833          ldab    3,y             ; 7 get duty as tenths of a degree
01472 [02] 3705         1834          clra                    ; 3
01474 [10] 3725         1835          emul                    ;11
01476 [04] 37BC003C     1836          ldx     #sixty          ; 6 period * tenths
0147A [24] 3728         1837          ediv                    ;25 ---------------  =  edge offset
0147C [02] 37CC         1838          xgdx                    ; 3     sixty
0147E [06] 37F10002     1839          addd    next_pulse      ; 8 add time for rising edge of next crank pulse
01482 [06] 37FAF492     1840          std     dasm18a         ; 8
                        1841  * falling edge
01486 [06] 9104         1842          addd    4,y             ; 7 add time for injector on
01488 [06] 37FAF494     1843          std     dasm18b         ; 8
0148C [12] 27F7         1844          rts                     ;14
                        1845  *
                        1846  *                                ;cycles
                        1847  inject4                          ;   46     dasm16    injector 1 both edges
0148E [08] 3880F480     1848          bclr    dasm16sic,#flag ;10
                        1849  * rising edge
01492 [06] 37750000     1850          lde     period          ; 8 get period from RAM
01496 [06] D503         1851          ldab    3,y             ; 7 get duty as tenths of a degree
01498 [02] 3705         1852          clra                    ; 3
0149A [10] 3725         1853          emul                    ;11
0149C [04] 37BC003C     1854          ldx     #sixty          ; 6 period * tenths
014A0 [24] 3728         1855          ediv                    ;25 ---------------  =  edge offset
014A2 [02] 37CC         1856          xgdx                    ; 3     sixty
014A4 [06] 37F10002     1857          addd    next_pulse      ; 7 add time for rising edge of next crank pulse
014A8 [06] 37FAF482     1858          std     dasm16a         ; 8
                        1859  * falling edge
014AC [06] 9104         1860          addd    4,y             ; 7 add time for injector on
014AE [06] 37FAF484     1861          std     dasm16b         ; 8
014B2 [12] 27F7         1862          rts                     ;14
                        1863  *
                        1864  *                                ;cycles
                        1865  inject5                          ;   48     dasm17    injector 1 both edges
014B4 [08] 3880F488     1866          bclr    dasm17sic,#flag ;10
                        1867  * rising edge
014B8 [06] 37750000     1868          lde     period          ; 8 get period from RAM
014BC [06] D503         1869          ldab    3,y             ; 7 get duty as tenths of a degree
014BE [02] 3705         1870          clra                    ; 3
014C0 [10] 3725         1871          emul                    ;11
```

```
014C2 [04] 37BC003C  1872          ldx     #sixty          ; 6 period * tenths
014C6 [24] 3728      1873          ediv                    ;25 ---------------  =  edge offset
014C8 [02] 37CC      1874          xgdx                    ; 3     sixty
014CA [06] 37F10002  1875          addd    next_pulse      ; 8 add time for rising edge of next crank pulse
014CE [06] 37FAF48A  1876          std     dasm17a         ; 8
                     1877  * falling edge
014D2 [06] 9104      1878          addd    4,y             ; 7 add time for injector on
014D4 [06] 37FAF48C  1879          std     dasm17b         ; 8
014D8 [12] 27F7      1880          rts                     ;14
                     1881  *
                     1882  *                               ;cycles
                     1883  inject6                         ;   50     dasm18    injector 1 both edges
014DA [08] 3880F490  1884          bclr    dasm18sic,#flag ;10
                     1885  * rising edge
014DE [06] 37750000  1886          lde     period          ; 8 get period from RAM
014E2 [06] D503      1887          ldab    3,y             ; 7 get duty as tenths of a degree
014E4 [02] 3705      1888          clra                    ; 3
014E6 [10] 3725      1889          emul                    ;11
014E8 [04] 37BC003C  1890          ldx     #sixty          ; 6 period * tenths
014EC [24] 3728      1891          ediv                    ;25 ---------------  =  edge offset
014EE [02] 37CC      1892          xgdx                    ; 3     sixty
014F0 [06] 37F10002  1893          addd    next_pulse      ; 8 add time for rising edge of next crank pulse
014F4 [06] 37FAF492  1894          std     dasm18a         ; 8
                     1895  * falling edge
014F8 [06] 9104      1896          addd    4,y             ; 7 add time for injector on
014FA [06] 37FAF494  1897          std     dasm18b         ; 8
014FE [12] 27F7      1898          rts                     ;14
                     1899  *
                     1900  * Injector - rising edge
                     1901  * minumum 8 rpm
                     1902  * DASM in SSOP mode where event A is separately from event B
                     1903  * takes 87 cycles (1),   103 cycles (2)
                     1904  *
                     1905  *                               ;cycles
                     1906  inject1r                        ;   56     dasm16a    injector 1 rising edge
01500 [08] 3880F480  1907          bclr    dasm16sic,#flag ;10
01504 [06] 37750000  1908          lde     period          ; 8 get period from RAM
01508 [06] D503      1909          ldab    3,y             ; 7 get duty as tenths of a degree
0150A [02] 3705      1910          clra                    ; 3
0150C [10] 3725      1911          emul                    ;11
0150E [04] 37BC003C  1912          ldx     #sixty          ; 6 period * tenths
01512 [24] 3728      1913          ediv                    ;25 ---------------  =  edge offset
01514 [02] 37CC      1914          xgdx                    ; 3     sixty
01516 [06] 37F10002  1915          addd    next_pulse      ; 8 add time for rising edge of next crank pulse
0151A [06] 37FAF482  1916          std     dasm16a         ; 8
0151E [12] 27F7      1917          rts                     ;14
                     1918  *
                     1919  * Injector - falling edge
                     1920  * minumum 8 rpm
                     1921  * DASM in SSOP mode where event B is separately from event A
                     1922  * takes 87 cycles (1),   103 cycles (2)
                     1923  *
                     1924  *                               ;cycles
                     1925  inject1f                        ;   58     dasm16b    injector 1 falling edge
01520 [08] 3880F480  1926          bclr    dasm16sic,#flag ;10
01524 [06] 37750000  1927          lde     period          ; 8 get period from RAM
01528 [06] D503      1928          ldab    3,y             ; 7 get duty as tenths of a degree
0152A [02] 3705      1929          clra                    ; 3
0152C [10] 3725      1930          emul                    ;11
0152E [04] 37BC003C  1931          ldx     #sixty          ; 6 period * tenths
01532 [24] 3728      1932          ediv                    ;25 ---------------  =  edge offset
01534 [02] 37CC      1933          xgdx                    ; 3     sixty
01536 [06] 37F10002  1934          addd    next_pulse      ; 8 add time for rising edge of next crank pulse
0153A [06] 37FAF484  1935          std     dasm16b         ; 8
0153E [12] 27F7      1936          rts                     ;14
                     1937  *
                     1938  * Missing tooth detect
                     1939  * MCSM 24 runs off 2uS timebase A
                     1940  * Places an interrupt at 1.5 times the current tooth period.
                     1941  * If set at tooth 58, should interrupt between tooth position 59 and 60.
                     1942  * 38 cycles (1),   53 cycles (2)
                     1943  *
                     1944  *                               ;cycles
                     1945  checkmis
01540 [06] 37F50000  1946          ldd     period          ; 7 get period
01544 [02] 277B      1947          tde                     ; 3 save in accE
01546 [02] 27FF      1948          lsrd                    ; 3 divide period by 2
01548 [02] 2778      1949          ade                     ; 3 add to period of last pulse
0154A [02] 2772      1950          nege                    ; 3 obtain 2's compliment for modulus counter
0154C [06] 377AF4C2  1951          ste     mcsm24cnt       ; 8 place number in MCSM 24 modulus latch
                     1952                                  ;   and initialise counter at same time
```

```
01550 [02] 7550       1953          ldaa    #$50            ; 3 enable interrupts (level 5)
01552 [06] 177AF4C0   1954          staa    mcsm24sic       ; 8
01556 [08] 3880F4C0   1955          bclr    mcsm24sic,#flag ;10 clear the overflow flag
0155A [12] 27F7       1956          rts                     ;14
                      1957  *
                      1958  *       Toggle port E-7 (test only)
                      1959  *
0155C [08] 3980FA13   1960  toggle1 bset    porte,#bit7     ; SIZ1
01560 [12] 27F7       1961          rts
01562 [08] 3880FA13   1962  toggle0 bclr    porte,#bit7     ; SIZ1
01566 [12] 27F7       1963          rts
                      1964  *
01568 [12] 27F7       1965  dummy   rts                     ;14
                      1966  *
0156A [12] 2777       1967  ctm_dummy rti                   ;14 should really clear the offending flag
                      1968                                  ;   but this is just test code
0156C [02] 274C       1969  BDM     nop
0156E [02] 37A6       1970          bgnd
                      1971
                      1972
```

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

freescale™
semiconductor

Freescale Semiconductor, Inc.