**CHAPTER 4: DRIVER INTERFACE SYSTEM**

**Design Targets**

1. **Vehicular Interaction with the driver**

Showing vital information to the driver such as rpm, engine temp, oil pressure, speed and fuel used.

1. **Ergonomically placed steering system and dashboard**

The way a driver interacts physically with the car is also important. This means the location of the steering wheel, the dashboard and shifter link play major role in driving feasibility. Thus an ergonomically designed driver interface system is a must.

1. **Compactness and vehicular integration**

The use of CAD models during PCB design to formulate a well-designed and compact system

**System Design**

The electronics system was designed along with the 3d model of the car which proved to be very reliable. The electronics systems are placed within the secondary and the primary firewall. This assured the temperatures never reached more than 60 degrees. The compactness of the secondary firewall provided a safety factor for the systems.

Secondly a high speed microcontroller was used to achieve refresh rates 67Hz as compared to 1Hz in our previous design. Sensor calibration was achieved using 5 point calibration technique to prevent linear hysteresis.

Electronics hardware integration within the steering wheel made the design and handling very user-friendly. The dashboard was made ergonomically and acquire minimum space and the cause least driver interference.

CAD / 3d part files and assembly was done on Solidworks to achieve accurate integration with other systems and it also provided an overlay for the wiring harness. The use of CAD was done for the first time and it proved really helpful in design and manufacturing.

**Dashboard**

The dashboard consist 3 switches; the cockpit kill switch, the cooling switch and the ignition switch. These are ground activated switches and are used to turn on relays. The cockpit kill switch is a push to break switch and disconnects the power supply to the ECU M400. This in turn turns off the supply to the fuel and ignition system thus stopping the engine.



**Fig 4.1**

**External Kill Switch**

The cooling switch turns on and off a PCB mountable relay which in turn switch on or off the cooling centrifugal pump or the cooling fan.

  
 **Fig 4.2**

**Cooling Switch**

The ignition switch is ground connected to the automotive starter motor relay. The switch is a momentary switch and pressing the switch turns on the stator motor which help the engine to crank and synch the crankshaft and the flywheel.

 **Fig 4.3**

**Ignition Switch**

**Steering Display System**

The design phase started with what parameter was to be displayed on the screen.

The steering wheel is one of the most important parts of a formula race car. It needs to show vital engine parameters to the drivers during a long endurance run. The steering wheel is made of 7 series aluminum and the electronic system consists of a 7 segment display for showing gear. Two 4 digit 7 segment displays to show different engine parameters of the engine. There is a 16 bit RPM/GSL LEDs to tell the driver when to shift to get maximum torque in the next gear.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | **Output Shaft Torque** |  |  |  |
| RPM |  | Engine torque | |  |  | 1st | 2nd | 3rd | 4th | 5th | 6th |
|  |  |  |  | HP | Ratio :1 -> | 2.833 | 2.062 | 1.647 | 1.421 | 1.272 |  |
| 2000 |  | 13.27 |  | 5.1 |  | 38 | 27 | 22 | 19 | 17 | 0 |
| 3000 |  | 19.914 |  | 11.4 |  | 56 | 41 | 33 | 28 | 25 | 0 |
| 4000 |  | 25.814 |  | 19.7 |  | 73 | 53 | 43 | 37 | 33 | 0 |
| 5000 |  | 33.19 |  | 31.6 |  | 94 | 68 | 55 | 47 | 42 | 0 |
| 6000 |  | 37.614 |  | 43.0 |  | 107 | 78 | 62 | 53 | 48 | 0 |
| 7000 |  | 42.778 |  | 57.0 |  | 121 | 88 | 70 | 61 | 54 | 0 |
| 8000 |  | 44.327 |  | 67.5 |  | 126 | 91 | 73 | 63 | 56 | 0 |
| 9000 |  | 42.778 |  | 73.3 |  | 121 | 88 | 70 | 61 | 54 | 0 |
| 10000 |  | 37.615 |  | 71.6 |  | 107 | 78 | 62 | 53 | 48 | 0 |
| 12000 |  | 31.715 |  | 72.5 |  | 90 | 65 | 52 | 45 | 40 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |
| NOTE : The chart at left will help you find what RPM's each gear will jump to. It takes the gear ratios you entered, then calculates what RPM the next gear gives you. In the default scenario, if you shift at 5600 in each gear, then going to 2nd will drop you to 3300, from 2 to 3 (at 5600rpm) gets you at 3771, and so on.  To navigate, using the TAB key will produce the best results, and data entry areas are in blue.  This worksheet is protected, but no password is required . The BASELINE and DISCLAIMER are password protected.   |  | | --- | |  | |  |  |  |  |  | Shift RPM | RPM after shift | | | | |
|  |  |  |  |  |  |  | 1 - 2 | 2 - 3 | 3 - 4 | 4 - 5 | 5 - 6 |
|  |  |  |  |  |  | 2000 | 1456 | 1597 | 1726 | 1790 | 0 |
|  |  |  |  |  |  | 3000 | 2184 | 2396 | 2588 | 2685 | 0 |
|  |  |  |  |  |  | 4000 | 2911 | 3195 | 3451 | 3581 | 0 |
|  |  |  |  |  |  | 5000 | 3639 | 3994 | 4314 | 4476 | 0 |
|  |  |  |  |  |  | 6000 | 4367 | 4792 | 5177 | 5371 | 0 |
|  |  |  |  |  |  | 7000 | 5095 | 5591 | 6039 | 6266 | 0 |
|  |  |  |  |  |  | 8000 | 5823 | 6390 | 6902 | 7161 | 0 |
|  |  |  |  |  |  | 9000 | 6551 | 7189 | 7765 | 8056 | 0 |
|  |  |  |  |  |  | 10000 | 7279 | 7987 | 8628 | 8951 | 0 |
|  |  |  |  |  |  | 12000 | 8734 | 9585 | 10353 | 10742 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | |  | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| |  | | --- | |  | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**Fig 4.4**

**RPM v/s Horse Power**

The microcontroller used to control the display unit is a PIC18F6720 microcontroller. The microcontroller has timer circuits to detect RPM and analog channels to measure engine sensor values. This microcontroller was chosen based on its ability to provide a range of input output ports.

**Microcontroller Specifications**

**High-Performance RISC CPU:**

• C compiler optimized architecture/instruction set:

- Source code compatible with the PIC16 and PIC17 instruction sets

• Linear program memory addressing to 128 Kbytes

• Linear data memory addressing to 3840 bytes

• 1 Kbyte of data EEPROM

• Up to 10 MIPs operation:

- DC – 40 MHz osc./clock input

- 4 MHz – 10 MHz osc./clock input with PLL active

• 16-bit wide instructions, 8-bit wide data path

• Priority levels for interrupts

• 31-level, software accessible hardware stack

• 8 x 8 Single Cycle Hardware Multiplier External Memory Interface (PIC18F8X20 Devices Only):

• Address capability of up to 2 Mbytes

• 16-bit interface

**Peripheral Features:**

• High current sink/source 25 mA/25 mA

• Four external interrupt pins

• Timer0 module: 8-bit/16-bit timer/counter

• Timer1 module: 16-bit timer/counter

• Timer2 module: 8-bit timer/counter

• Timer3 module: 16-bit timer/counter

• Timer4 module: 8-bit timer/counter

• Secondary oscillator clock option – Timer1/Timer3

• Five Capture/Compare/PWM (CCP) modules:

- Capture is 16-bit, max. Resolution 6.25 ns (TCY/16)

- Compare is 16-bit, max. Resolution 100 ns (TCY

- PWM output: PWM resolution is 1 to 10-bit

• Master Synchronous Serial Port (MSSP) module with two modes of operation:

- 3-wire SPI™ (supports all 4 SPI modes) 2-I C™ Master and Slave mode

• Two Addressable USART modules:

- Supports RS-485 and RS-232

• Parallel Slave Port (PSP) module

**Analog Features:**

• 10-bit, up to 16-channel Analog-to-Digital Converter (A/D):

- Conversion available during Sleep

• Programmable 16-level Low-Voltage Detection (LVD) module:

- Supports interrupt on Low-Voltage Detection

• Programmable Brown-out Reset (PBOR)

• Dual analog comparators:

- Programmable input/output configuration

**Special Microcontroller Features:**

• 100,000 erase/write cycle Enhanced Flash program memory typical

• 1,000,000 erase/write cycle Data EEPROM memory typical

• 1 second programming time

• Flash/Data EEPROM Retention: > 40 years

• Self-reprogrammable under software control

• Power-on Reset (POR), Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)

• Watchdog Timer (WDT) with its own On-Chip RC Oscillator for reliable operation

• Programmable code protection

• Power saving Sleep mode

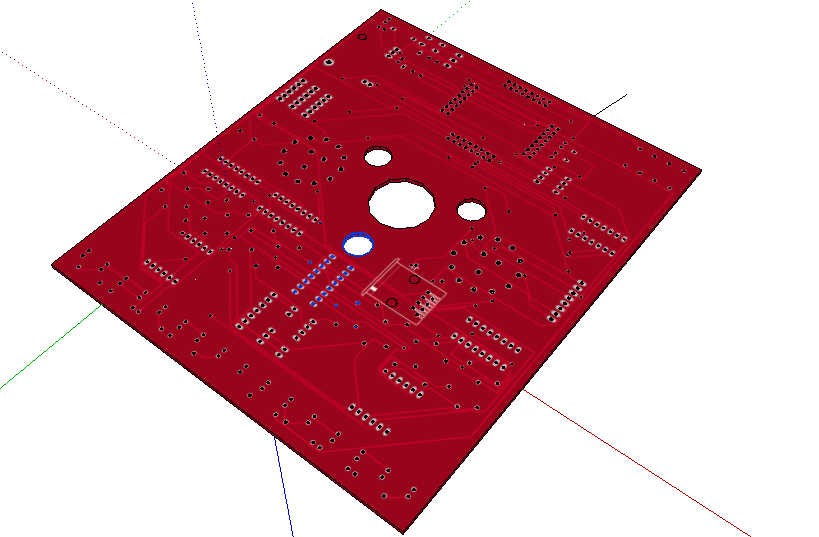
• Selectable oscillator options including:

- 4X Phase Lock Loop (of primary oscillator)

- Secondary Oscillator (32 kHz) clock input

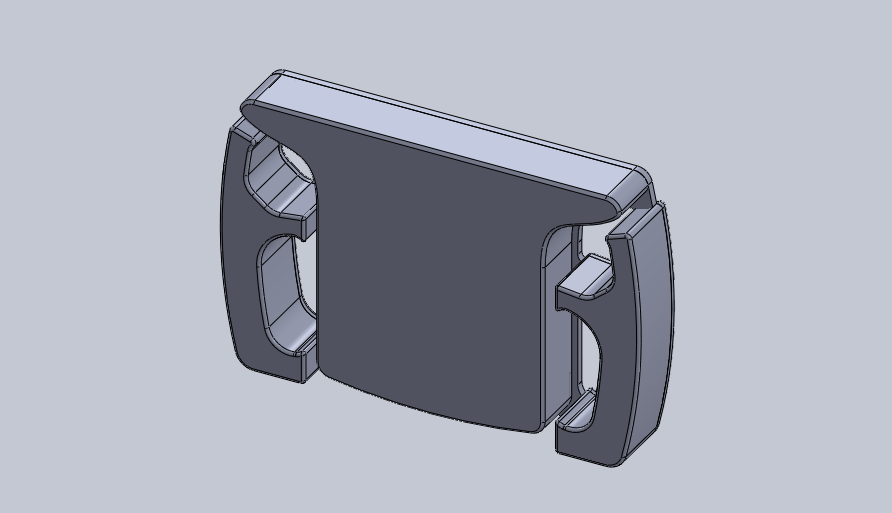
• In-Circuit Serial Programming™ (ICSP™) via two pins

• MPLAB ® In-Circuit Debug (ICD) via two pins



**Fig 4.6**

**Driver Interface system**



**Fig 4.7**

**Steering Wheel**

**Coding and Testing**

The code was written using Mikroc along with simulation in Proteus

Code:

//Decleration and Initialization

#include <built\_in.h>

char ur,tr,hr,thr,ul,tl,hl,thl; //segment choping variables

int count,i,j,value,rpm,gearp;

char teeth;

int volt,etmp,as,ts;

int seg7(int num)

{

int val[]={0x3f,0x06,0x5b,0x4f,0x66,0x6D,0x7D,0x07,0x7F,0x6F,0x77,0x7C,0x39,0x5E,0x79,0x71};

return(val[num]);

}

int gear(int num)

{

int val[]={0x54,0x06,0xDA,0xCE,0x66,0xEC,0xFC,0x86,0xFE,0xEE};

return(val[num]);

}

int fun(int num)

{

int val[]={0x7f,0x3f,0x1f,0x1e,0x1c,0x18,0x10,0x18,0x1c,0x1e,0x1f,0x3f,0x7f,0x00,0x7f};

return(val[num]);

}

int bar(int num)

{

int val[]={0x81,0xc3,0xe7,0xff,0xe7,0xc3,0x81,0x81,0xc3,0xe7,0xff,0xe7,0xc3,0x81};

return(val[num]);

}

int gps()

{

j=ADC\_READ(3);

if (j>=0 &&j<88)

{PORTB=gear(1);

return 1;}

// else if (j>=151 && j<159)

// {PORTB=gear(0);

// return 0;}

else if (j>=89 && j<284)

{PORTB=gear(2);

return 2;}

else if (j>=285 && j<499)

{PORTB=gear(3);

return 3;}

else if (j>=500 && j<700)

{PORTB=gear(4);

return 4;}

else if (j>=700 && j<897)

{PORTB=gear(5);

return 5;}

else if (j>=898 && j<1020)

{PORTB=gear(6);

return 6;}

}

int et()

{

j= ADC\_Read(0); //engine temp begins

return etmp= (6.4\*j\*j/100000 +-0.19\*j + 126);

}

speed()

{

value=TMR1H;

//use timer 1 for speed anf fuel used calculations

}

fuelused()

{

//use timer 1 for speed anf fuel used calculations

}

rpmlights(int num)

{

switch(num)

{

case 1: PORTF=0x01; break;

case 2: PORTF=0x01; break;

case 3: PORTF=0x01; break;

case 4: PORTF=0x03; break;

case 5: PORTF=0x07; break;

case 6: PORTF=0x1f; break;

case 7: PORTF=0x3f; break;

case 8: PORTF=0x7F; break;

case 9: PORTF=0xff; break;

case 10: PORTF=0x1c; break;

case 11: PORTF=0x1C; break;

case 12: PORTF=0x1C; break;

}

}

segr()

{

thr=value/1000;

hr=value/100%10;

tr=value/10%10;

ur=value%10;

if(value>999)

{

RC7\_bit=1;

RC6\_bit=0;

RC5\_bit=1;

RC4\_bit=1;

PORTD=seg7(thr);

delay\_us(1000);

}

if(value>99)

{

RC7\_bit=0;

RC6\_bit=1;

RC5\_bit=1;

RC4\_bit=1;

PORTD=seg7(hr);

delay\_us(1000);

}

if(value>9)

{

RC7\_bit=1;

RC6\_bit=1;

RC5\_bit=1;

RC4\_bit=0;

PORTD=seg7(tr);

delay\_us(1000);

}

if(value>=0)

{

RC7\_bit=1;

RC6\_bit=1;

RC5\_bit=0;

RC4\_bit=1;

PORTD=seg7(ur);

//delay\_us(2500);

}

}

segl()

{

thl=value/1000;

hl=(value/100)%10;

tl=value/10%10;

ul=value%10;

if(value>999)

{

RG3\_bit=1;

RG2\_bit=0;

RG1\_bit=1;

RG0\_bit=1;

PORTE=seg7(thl);

delay\_us(1000);

}

if(value>99)

{

RG3\_bit=1;

RG2\_bit=1;

RG1\_bit=1;

RG0\_bit=0;

PORTE=seg7(hl);

delay\_us(1000);

}

if(value>9)

{

RG3\_bit=1;

RG2\_bit=1;

RG1\_bit=0;

RG0\_bit=1;

PORTE=seg7(tl);

delay\_us(1000);

}

if(value>=0)

{

RG3\_bit=0;

RG2\_bit=1;

RG1\_bit=1;

RG0\_bit=1;

PORTE=seg7(ul);

//delay\_us(2500);

}

}

void coolon()

{

RG3\_bit=1;

RG2\_bit=0;

RG1\_bit=1;

RG0\_bit=1;

PORTE=0x39;

delay\_us(4000);

RG3\_bit=1;

RG2\_bit=1;

RG1\_bit=0;

RG0\_bit=1;

PORTE=0x3F;

delay\_us(4000);

RG3\_bit=0;

RG2\_bit=1;

RG1\_bit=1;

RG0\_bit=1;

PORTE=0x54;

delay\_us(4000);

}

void cooloff()

{

RG3\_bit=1;

RG2\_bit=0;

RG1\_bit=1;

RG0\_bit=1;

PORTE=0x39;

delay\_us(4000);

RG3\_bit=1;

RG2\_bit=1;

RG1\_bit=0;

RG0\_bit=1;

PORTE=0x3F;

delay\_us(4000);

RG3\_bit=0;

RG2\_bit=1;

RG1\_bit=1;

RG0\_bit=1;

PORTE=0x71;

delay\_us(4000);

}

void engoff()

{

RC7\_bit=1;

RC6\_bit=0;

RC5\_bit=1;

RC4\_bit=1;

PORTD=0x79;

delay\_us(4000);

RC7\_bit=1;

RC6\_bit=1;

RC5\_bit=1;

RC4\_bit=0;

PORTD=0x3F;

delay\_us(4000);

RC7\_bit=1;

RC6\_bit=1;

RC5\_bit=0;

RC4\_bit=1;

PORTD=0x71;

delay\_us(4000);

}

void main() {

T0CON=0XF8; // timer0 on, 8 bit mode, counter, falling edge, with no prescalar

T1CON=0x0B;

TRISA=0xFF;

ADCON1=0x00;

TRISF = 0;

TRISE = 0;

TRISB = 0;

TRISD = 0;

TRISC0\_BIT=1;

TRISC1\_BIT=1;

TRISC2\_BIT=1;

TRISC3\_BIT=1; // 1's inputs

TRISC4\_BIT=0;

TRISC5\_BIT=0;

TRISC6\_BIT=0;

TRISC7\_BIT=0; // 0's outputs

TRISG = 0x00;

for(i=0; i<=14; i++)

{

PORTF=bar(i);

PORTB=fun(i);

PORTD=PORTE=fun(i);

PORTG=0X00;

RC4\_bit=RC5\_bit=RC6\_bit=RC7\_bit=0;

delay\_ms(60);

}

PORTF = 0x00;

while(1)

{

DELAY\_MS(16);

gearp=gps();

teeth= TMR0L;

value=teeth\*60;

rpm=value/1000;

while(value <1000)

{engoff();

teeth= TMR0L;

value=teeth\*60;

if(ADC\_Read(1)<= 350)

cooloff();

else

coolon();

}

rpmlights(rpm);

value = et();

segr();

fuelused();

speed();

segl();

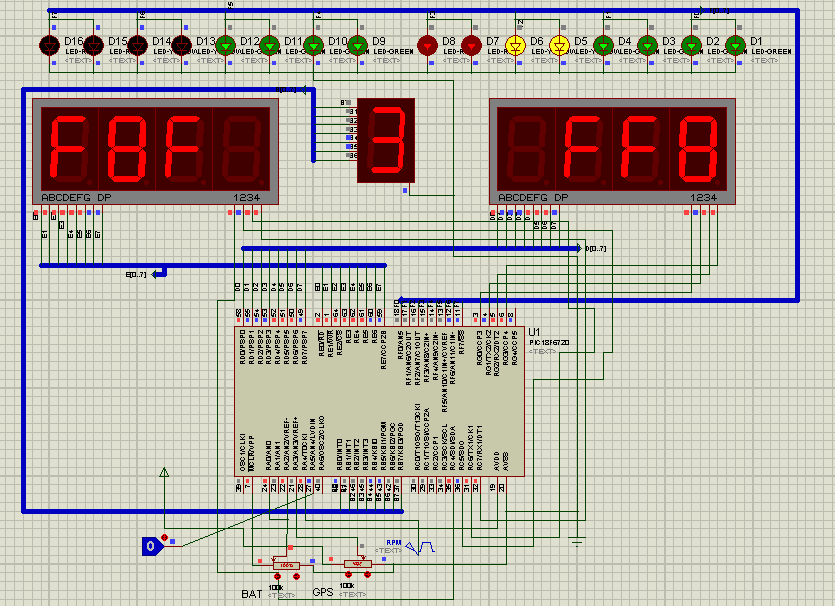
// write code for right rotary switch//

TMR0L=0x00;

TMR1H=0x00;

}

}



**Fig 4.8**

**Driver Interface System Simulation**

**Gear Change Ignition Cut**

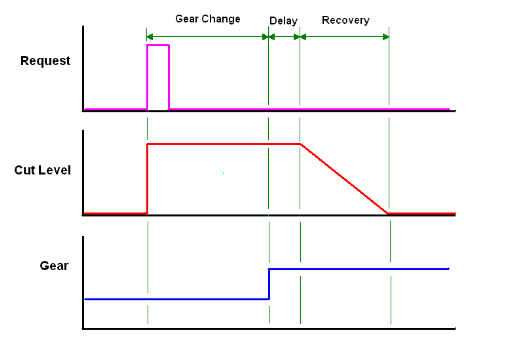
The MoTeC GCIC function is used to allow the driver to change gears without lifting the throttle or (in most cases) the clutch, it is also often referred to as “Flat Shift” or “Shift without Lift”. The system allows a short ignition or fuel cut and/or ignition timing retard to reduce the engine’s load on the transmission so that the gear can be changed. It is also possible to have the position of an Electronic Throttle (Drive by Wire) reduced for a short amount of time as well.

The function is designed with both H pattern and sequential, dog engagement gear boxes in mind, it should be noted that this function is not completely suitable for synchromesh gearboxes.

**Cut Modes**

**Mode 1, Delay from next gear stable**

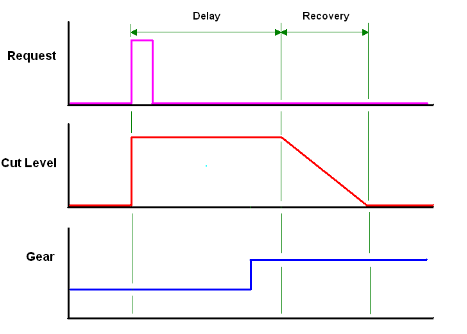
This mode is only suitable when the ECU has a gear position input. The ignition cut starts as soon as a valid shift signal is detected and will hold for however long it takes for the gear position pot or gear calculation to detect the next gear has been reached or the Minimum RPM or Minimum Throttle Position have been reached. The Delay time starts when the new gear is detected to allow time for full engagement then the ignition is phased back in linearly over the recovery time.



**Fig 4.9**

**Mode 2, Delay from Cut Signal**

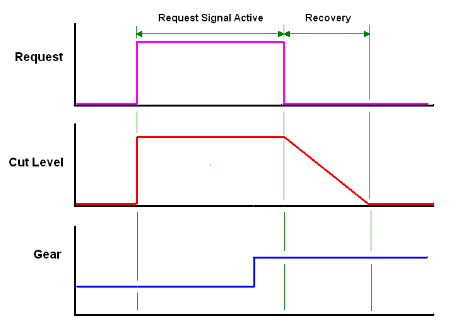
The Delay from Cut Signal mode starts the ignition cut when the cut source becomes active and runs for the user defined Delay time (as set in Delay Table). After the delay has timed out the ignition Recovery time starts.



**Fig 4.10**

**Mode 3, Cut While Signal is active**

This mode will simply cut the ignition for as long as the shift signal is active. No delay times are entered for this mode. Mode 3 is mainly designed for external gear box control systems that are able to send a GCIC signal to the ECU; the cut time is calculated in the external controller.



**Fig 4.11**

It can be seen from the previous two diagrams that the gear position does not affect the cut at all.

**Cut Source**

To make the GCIC system work the ECU needs a request that the driver wishes to make an up shift, there are a number of options for this. It is possible that race class rules may only allow certain methods for GCIC so the rule book will need to be checked.

1. A simple momentary (on/off) switch the driver activates when an up shift needs to be made

2. A momentary switch on the clutch pedal

3. The request can be started when the RPM Limiter is reached

4. A strain gauge may be installed on the gear lever

5. The request can be started based on a rate of reduction of throttle position

6. The up shift request can come from the request linked to the Gear Shift function

(“semi-auto” or controlled gearboxes)

How the gear request device or condition is setup can have a very large effect on the performance of the GCIC function. One fairly important aspect of a gearbox that cannot be ignored is the mechanical free play from moving the gear shift lever to moving the dog ring. Another factor is how much force should be used to “preload” the gearbox shift mechanism to get the best shifts. To make a positive shift the driver needs to move to gear lever enough so that all the free play is eliminated and then apply enough force so that when the GCIC starts the gearbox will move almost instantly. Some problems that can arise from incorrect setup of the shift request are:

1. If the request system does not allow the driver to move the gear lever far enough to remove all the free play in the gearbox the GCIC will start too early. Starting the GCIC too early will mean that the total cut time will have to be longer to compensate.

2. If the driver does not have enough “preload” force on the gear lever before the

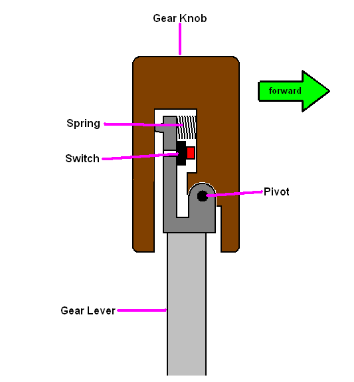
GCIC starts the gearbox mechanism will not accelerate quickly, again needing a longer cut time to compensate.

3. If too much force is required to start the GCIC the driver may become fatigued or the gear box may be damaged.

1. **Digital (Switched) Input**

A switched input can be located almost anywhere within reach of the driver but the best logical solution is to have it located on the gear lever. The switch may also be located on the clutch pedal but this may defeat the purpose of GCIC where clutch less shifts are desired. Some racing classes specify a switch location, e.g. steering wheel but this, like a clutch switch, means that the drive must co-ordinate the state of GCIC and the moving of the gear lever correctly and repeatable to get the best performance.

The diagram below shows a basic example of a “double jointed” gear knob for a sequential gearbox where GCIC is only needed on the up shift. The knob is able to pivot for a short distance on the gear lever when pulled towards the rear; this activates the switch giving the ECU its GCIC request. It is recommended that the spring be adjustable for preload so that the correct amount of force can be applied before the GCIC function is started.



**Fig 4.12**

**GCIC Switch**

If an H pattern system is needed or GCIC is required for both up and down shifts on a sequential gearbox a dual switch setup can be made.

The switch can be wired to either a Digital input or a Switched (Analogue Temperature) input. A switch wired to a Digital input must connect the input to 0v to pull it low and can be normally open or normally closed, whether the request is active when the switch is open or closed can be selected with the Logic Polarity parameter.

A switch that is wired to a Switched (Analogue Temperature) input can be connected to a voltage higher than 5v as the trigger levels can be defined by the user but it is most common practice to connect the input to 0v via the switch the same as the Digital input. Again a normally open or normally closed switch can be used.

1. **RPM Limit**

The RPM limiter can be used to request GCIC. The GCIC function request is active when the driver reaches the RPM limit. This method does not require any extra sensors to be added to the car but the driver must be aware that any time they reach the limiter they will get prolonged engine cutting if a gear change is not desired at that time. Delay from next gear stable may not be suitable for this method.

**3. Gearshift Force - Strain Gauge**

The Strain Gauge works by being bonded onto the surface of an item, its electrical properties change as it is deformed by the small amounts of tension and compression in the item. For GCIC the strain gauge is bonded onto the front or back of the gear lever. As the driver puts force on the gear lever it bends slightly causing a change in signal form the strain gauge.

The voltage signal for a strain gauge is very small and must be amplified to become useful to the ECUs input. When considering bonding a strain gauge onto a gear lever a suitable strain gauge amplifier is needed, this can add a considerable amount to the cost of the system.

The major benefit of the strain gauge is that it is an analogue input with a voltage reading proportional to the drivers force on the gear lever. With the analogue input to the ECU the tuner is able to program the level of force needed to trigger the GCIC request. Another benefit is to be able to data log how the driver is using the gear lever with much more information made available when compared to a simple on/off switch.

The Strain Gauge amplifier’s output can be connected to either an Analogue Voltage input or an Analogue Temperature input although it must be noted that the signal will be slightly non-linear when connected to the Analogue Temperature input due to its internal 1K ohm, 5v pull-up.

When calibrating the Gear Shift Force channel there is no need for the reading to be completely correct as the force level for the request will be set later. As a basic setting the calibration for 5v = 100N will be acceptable, if a more accurate reading is desired a custom calibration (using a fish scale for example) may be required.

**4. Throttle Position Reduction**

The rate at which the throttle is reduced can be a request for the GCIC function. The request is active when a user defined rate of reduction (TP% per second) of the Throttle Position Channel is reached or exceeded.

This method does not require any additional sensors to be added to the vehicle. Again,

driver education and correct setup is needed as the function will not be able to determine if the driver has lifted the throttle for a gear change or if it is a correction for a loss of traction for example. Again, the delay from next gear stable mode may not be suitable for this method.

Thinking outside the square for experienced users some conditions could be used where no cut is applied (from the cut level table) if there is any wheel slip, allowing the driver to quickly lift the throttle without inadvertently requesting GCIC.

**5. Gear Shift Function**

The gear box control function will already have a request for an up-shift, so it is possible to link the two functions in software so that the GCIC is automatically triggered when an up-shift request is made by the driver.

**6. Minimum RPM**

A lower level of RPM is set so that the function cannot be requested. For example, if the car is being driven through the pit lane it would be unnecessary to have GCIC active for the slow shifting needed, the driver would most likely want to use the clutch.

The Minimum RPM can work in different ways in different modes so care must be taken in setup. For the Delay From Cut Signal and Cut While Signal Active modes the Minimum RPM is only a condition that needs to be met at the start of the request and will not affect the cut event once it has been started. For example: If the Minimum RPM is 5000rpm and the engine is at 5050rpm, the GCIC can start. Clearly the RPM will drop as soon as there is any cut. If the engine drops below 5000rpm while the GCIC event is going cut WILL NOT be stopped, the GCIC event will continue for its programmed time.

For the Delay from Next Gear Stable mode the minimum RPM is used in two ways, as a condition to allow the GCIC request to be valid and also as a safety measure for the RPM dropping to low. Because the Delay from Next Gear Stable mode can cut for a much longer time than the other two modes it is possible that the engine may stall if the driver makes a mistake.

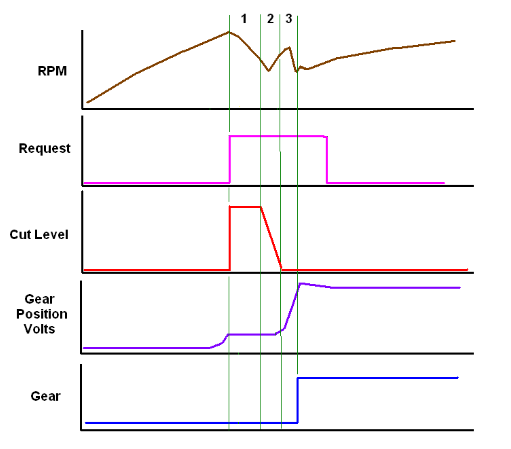
**7. Minimum Throttle Position**

As per Minimum RPM there is also a parameter for a Minimum Throttle Position below which a request is ignored by the ECU. The different mode selected will affect how the Minimum Throttle is use in the same way as the Minimum RPM, for the Delay from Cut Signal and Cut While Signal Active modes the Minimum Throttle is used as a condition to allow the request and start the GCIC event, if the throttle position drops below this value during the GCIC event the cut will not be stopped, for the Delay From Next Gear Stable mode the Minimum Throttle is used as a condition to accept the request and if the throttle drops below this during the GCIC event the cut WILL be stopped.

**Delay Table**

The Delay Table sets the amount of time the ECU must cut the engine power so that the driver can make a successful shift. The Delay table works in two different ways depending on which Cut Mode is used.

For Mode 1, Delay from Next Gear Stable the request starts the cut and the ECU will continue to cut until the next gear is acquired or the Time Out period has elapsed. The Delay Table sets extra cut duration once the next gear has been reached; this allows time to ensure that the next gear has been successfully engaged. This time is generally short when compared to the Mode 2, Delay from Cut Signal and is often left as 0msecs.

For Mode 2, Delay from Cut Signal the Delay Table sets the time that the ECU cuts the engine power for, this time is started as soon as the cut request becomes active. Enough time must be allowed so that the driver is able to get to the next gear; this mode will not allow for mistakes and will reinstate the engine power after the Delay plus Recovery Time regardless of what position the gear box is in, see example below:

**Fig 4.13**

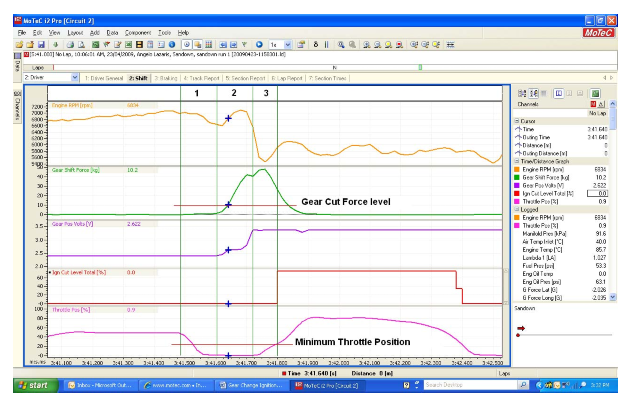
**Driver Education**

For drivers who have spent most of their racing career quickly lifting the throttle to change gear it can be quite difficult to convince them not to lift once GCIC function has been switched on. This can cause problems for the tuner who needs to set minimum and maximum values, especially throttle position, to make the function work. There are many cases of the driver not obeying the instruction to keep the throttle wide open on a shift and this can cause many problems for the logic of the function, below is a common example.

1. The driver has lowered the throttle position until it is well below the Minimum Throttle position.

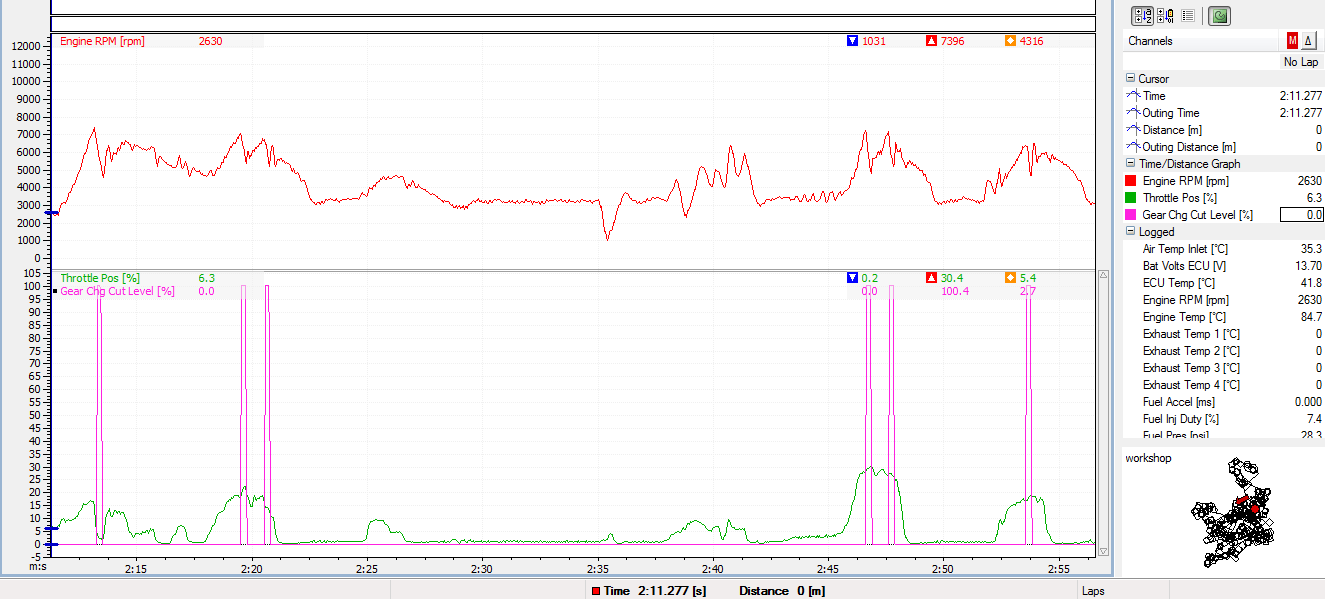
2. The driver then pulls on the gear lever and is able to change gear with ease because there is no engine torque.

3. Because the driver is still holding enough force on the gear lever as the throttle is reapplied the cut starts. In this example the Mode was Delay from Next Gear Stable meaning that the cut is applied for the full Time Out because the driver triggered the GCIC request after the shift was made and not before.



**Fig 4.14**

**Gear Shift Timing**



**Fig 4.15**

**Gear Shift Logged Data**