Inside Bosch Motronic

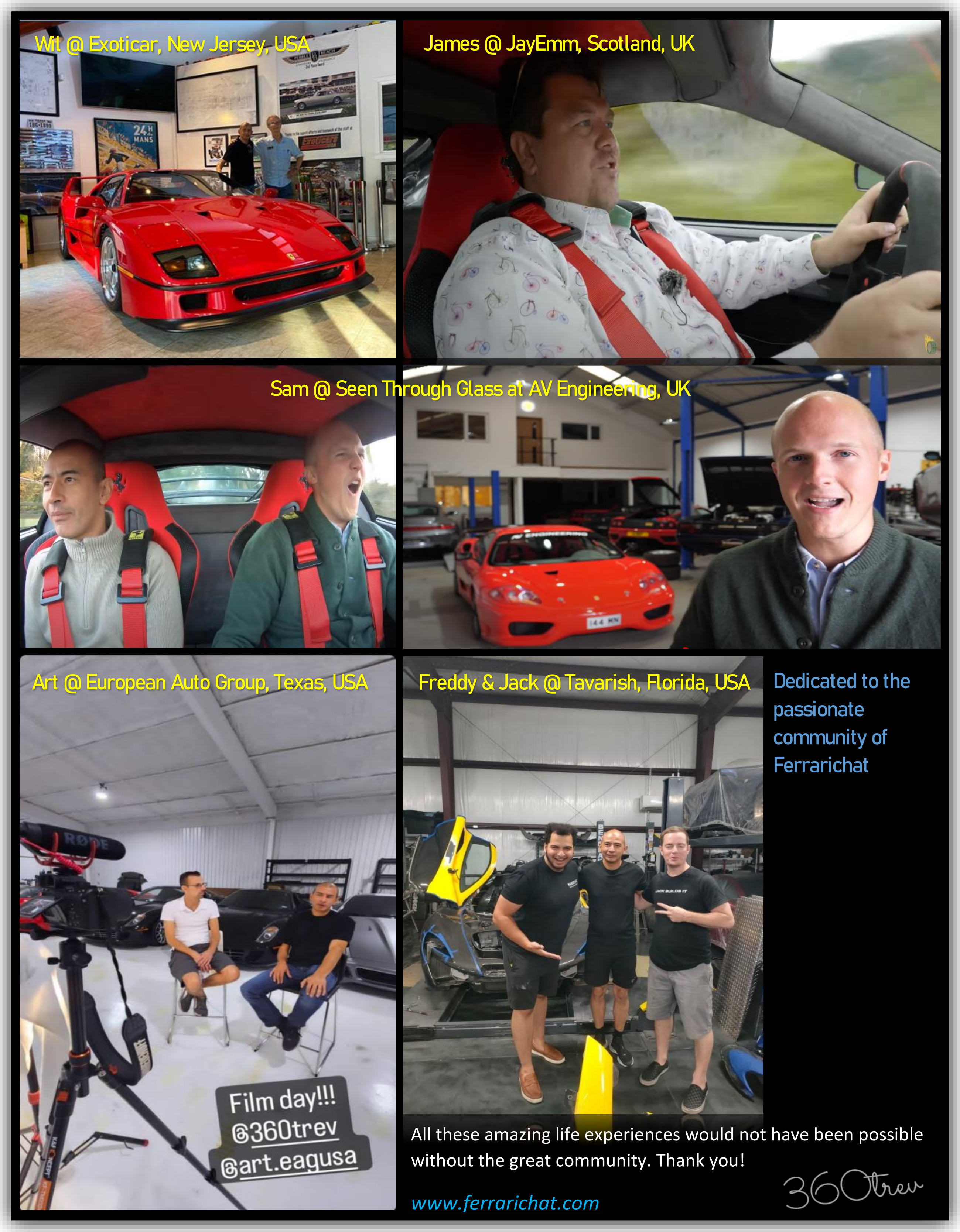
A Ferrari Perspective…

SENICA CONSULTING

First Edition – Part 1

8-28-2024

Castle Rock
Publishing Ltd.


Wit @ Exoticar, New Jersey, USA
James @ JayEmm, Scotland, UK
Sam @ Seen Through Glass at AV Engineering, UK
Art @ European Auto Group, Texas, USA
Film day!!!
@360trev
@art.eagusa
Freddy & Jack @ Tavarish, Florida, USA
Dedicated to the passionate community of Ferrarichat
All these amazing life experiences would not have been possible without the great community. Thank you!
www.ferrarichat.com
360trev

Forward

I am thrilled to share with you my work, Inside Bosch Motronic: A Ferrari Perspective.
The information is the culmination of many years working on real-time embedded systems in both hardware and software engineering. Having gained a wealth of “hands-on” experience in areas including detailed automotive engine calibration, research and development in a laboratory environment and specialized compression/encryption technologies.

As the founder of a technical consulting company, a former Telecommunications Chief Architect serving in several FTSE 100 companies, I have multiple decades of experience in real world large scale engineering projects. I am also an active member of the Ferrari community with “Project Master” as recognition for my contributions.

My journey on this path to writing this book began on Ferrarichat.com over two decades ago when I began to upgrade my own personal Ferrari (which I still own today). I discovered limitations that could not be resolved without going deeper into the software. The tipping point was when a friend of mine (also a member of the Ferrari community) had an issue with intermittent starting that could not be resolved at his local dealer. It turned out to be a fault on his immobilizer, which was eventually tracked down. I fixed it for him by modifying the software. All the work I have done and continue to do has been primarily focused on achieving the absolute best driver engagement and enjoyment, making each car I touch totally unique and tailored to the individual driver's demands.

I have written this book to provide a comprehensive resource for anyone who wants to learn more about this vast and rather complex subject. I hope to help readers understand why it takes years and spans multiple engineering disciplines to make it "just perfect" and also why economically it is just not viable for most companies to commit do doing things the right way on low volume production cars like Ferrari’s. Driving today some of the modern classics which have been upgraded in such a way is a "night and day" enlightening experience. I hope I can shed light here on how much effort goes into making each and every one a better version of itself.

The book is divided into three parts:

Part 1: A deep dive into the hardware of Bosch Motronic version 7 and 9 (a later addition to the series will include MED 17 which brings us right up to date), covering the subjects of hardware design, sensor basics, software architecture outline, and common faults.

This is an introduction to the topic, and while it may be quite heavy reading for the non-technically trained, I hope it at least gives a glimpse into the world of Bosch engine computers, beyond the manufacturers' very high-level overview that you would normally expect to see in a workshop manual.

In this part of the book, you will learn about:

- Background information on the Bosch Motronic System and the challenges it now faces
- A Deep dive into details covering the hardware design of Bosch Motronic version 7 and 9 systems, including the central processing unit (CPU), memory, and how sensors are connected to a design via various Integrated Circuit (IC) chips.
- Background on an ECU project, how it starts and what components you can typically expect to see in an engine computer.
- Software architecture outline, including a high-level overview of the design of the software that controls the Bosch Motronic system.
- Frequently Asked Questions, common faults in Bosch Motronic systems specific to Ferrari's, such as sensor failures, wiring problems, and software glitches.

Part 2: Engine Sensors & Actuators – the complete guide, this will do a deep dive into each and every sensor and actuator used in the vehicle and how they work.

Everything is covered from what they measure, their physical properties, and even some technical details to allow you to drive them outside of a car environment so they can be tested and understood even better. This may or may not yet include some sketches using an Arduino environment, so it is relatively easy to follow along and develop your understanding much further.

In this part of the book, you will learn about:

- Each and every sensor and actuator used in the vehicle, including how they work, what they measure, and their physical properties.
- Sensor basics, including the principles of operation of different types of sensors, such as temperature sensors, pressure sensors, and mass airflow sensors.
- Technical details to allow you to drive engine sensors and actuators outside of a car environment using an Arduino or similar microcontroller board for testing and troubleshooting purposes.
- A deeper understanding of the root causes of fault codes your make see appear and best way to resolve the issues.

Part 3: Bosch Motronic Software, this can be described as the "missing manual" and will cover the topics of how and what goes into the software.

In this part of the book, you will learn about:

- How and what goes into the Bosch Motronic firmware image.
- The bootloader, which is a small program that is responsible for transferring control to the main Bosch Motronic software.
- Communication protocols such as KWP2000 and CCP, which are used to communicate with the Bosch Motronic system.
- The ASAP/ASAM standards, which are a set of standards that define the data formats and communication protocols used in automotive software development.
- Checksums, which is a technique used to verify the integrity of data.
- Tabular calibration, which is a way of storing calibration data in a table based format.
- Interpolation, which is a technique used to estimate values between known points.
- Frequently Asked Questions, why it takes years to understand all the complex interwoven relationships between tables, how it all fits together and why so many tuners get things so badly wrong on Ferrari's.

This book is intended for a wide audience, including engineers, workshop technicians, students, and anyone else who is interested in learning more about the engine software, hardware, and sensors. I hope this book will be a valuable resource for anyone who wants to learn more about these topics.

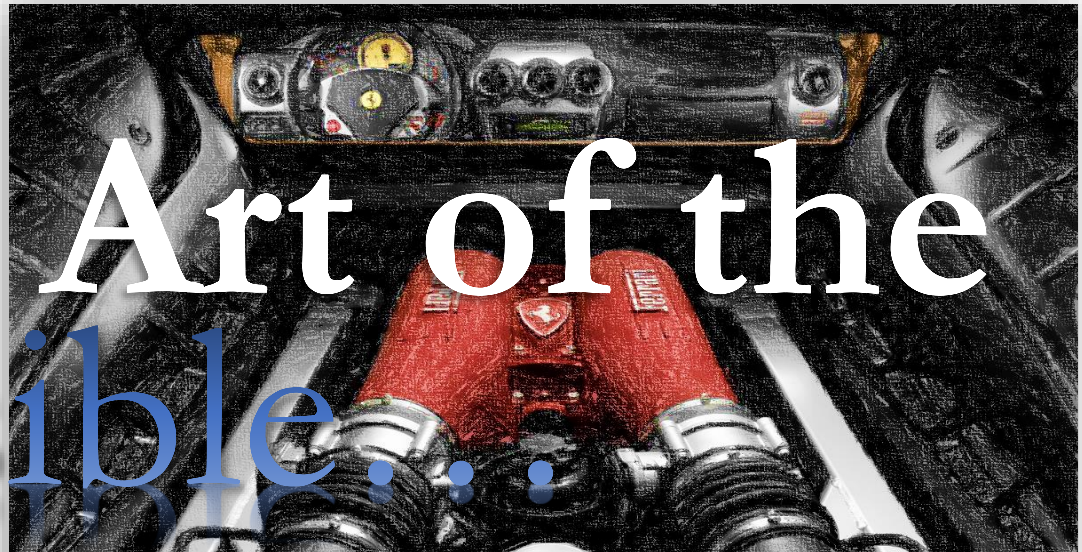
Sincerely,

Trevor (*AKA 360trev*)
Senica Consulting Ltd.

360trev


Corse Clienti

The Art of the Possible…



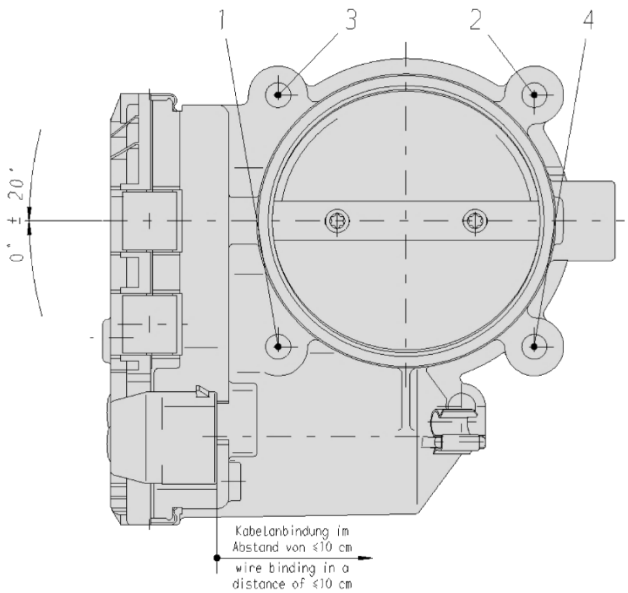
Let's dig deep into the hot topics of software surrounding the Engine Management Computers as used in Ferrari's. So much to cover, let's start with background first then hardware design, sensor diagnostics, software and Q & A regarding F1 to Gated conversions and much more…

Bosch Motronic Engine Management System continues to be a huge success story for the company but with the rise of pure Electric Vehicles (EV's) are its days now numbered? Bosch's market dominance appears to be threatened by the move to Electrification as it brings in an entirely new powertrain which requires a completely different approach and of course has no need to manage a traditional (ICE) internal combustion engine. That position changes with hybrid powertrain (i.e. ICE+EV combined) but no doubt engineers and senior management alike have an eye toward the future and what that means for Bosch. After many decades at the top therefore it is no surprise to learn they are actively working on what comes next, its future. In the meantime it is still the market leader selling up to250 million units per year. It can after all trace its origins right back to the beginning of the use of computers in vehicles in the late 1970's.

As you can imagine it has been produced in a wide range of different software and hardware configurations across many vehicle manufacturers' product lines and therefore needs to be highly flexible in both architecture and design. It's a huge project that spans hundreds of man-years worth of investment and millions of lines of source-code. The technical description manuals can span thousands of pages alone. The solution is still today the most popular combustion engine controller and used in both road & race applications, military equipment, plant and even marine. It is used pretty much everywhere. At its peak the software was under continuous development by a team of 100+ engineers but it is now believed that 'ICE' Motronic is currently under 'maintenance only' while all effort is re-directed to develop an entirely new solution with full autonomy and artificial intelligence for self-driving and more. As is the rest of the auto industry, they are playing catch-up to Tesla.

Historically, the design and development was originally done in Germany but eventually ended up spanning at least 3 different time zones. It has many thousands of adjustable parameters and calibrations which can be tailored by the OEM for every possible scenario. This flexibility costs millions to adapt and test to a specific car and its engines, it is just *that* complex. This is why it's a fallacy to believe replacing the factory ECU's with aftermarket would be better, the sheer amount of money and time spent is eye watering and no aftermarket ECU provider has spent even a fraction of the investment of Bosch on their solutions. This doesn't even take into account the hundreds of man hours individually spent on dyno's per engine project and the huge amount of testing to get everything working in all conditions correctly to ensure the engine runs within emissions standards for a hundred thousand miles and beyond. Tested to destruction and beyond your imagination, everything from sea level to up a mountain (where the air is thinner), to high humidity, to desert dry and even sub-zero conditions. Millions in R&D.

At the start of a new project the software could be adapted specifically to individual small batches of vehicles with custom project configurations driving 4 cylinders right the way up to 16, with or without superchargers or turbos (or both). Several major revisions across different hardware generations have happened over the years and the version used therefore depends entirely on the year of production. It became the "de-facto" solution for automotive manufacturers wishing to sell their cars globally due to significant efforts by Bosch to ratify international emissions standards and ensure their ECU software, hardware and proprietary sensors could adhere to them. Not only that, Bosch went on to develop sophisticated ABS, ASR and full ESP systems which communicate directly over CAN (Controlled Area Network – another Bosch invention). Ferrari continued this evolution with F430 adding sophisticated Electronic differentials.

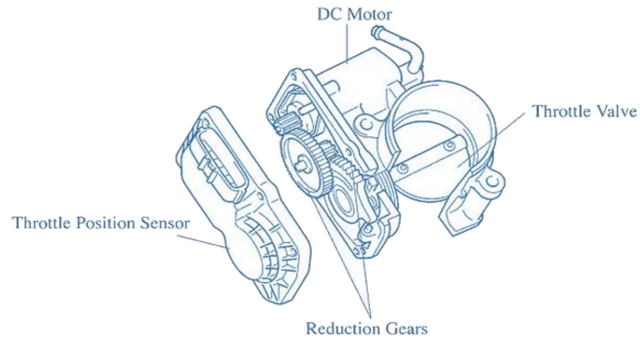


The Bosch ME7 is a good place to start, it is a system that was first used in production on the 1999 Model Year Ferrari 360 when it ushered in an entirely new 'torque based model' and used a modern DV-E (Electronic Throttle Body). It replaced the analogue 'throttle cable' originally connected the throttle pedal to a flap in the intake, as fitted to the previous models like the F355, 456, etc. (which would originally operate by opening and closing based on nothing more than the drivers pedal position).

The new electronic throttle pedal used potentiometers to send a signal to the engine software, also known as 'Drive by Wire'. This allows the ECU software to override the drivers request for torque and implement much more advanced torque management than ever before including faster responding anti-slip protocols.

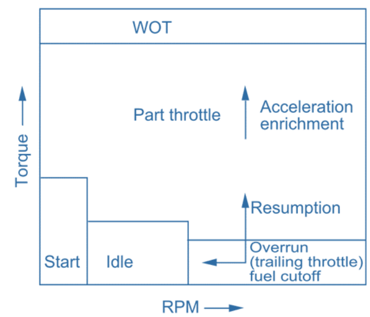
The ME7 architecture was used on many Ferrari models including the *Ferrari 360 Modena, 360 Spider, 360 Modena Challenge (race car), 360 N-GT (race car), 360 Challenge Stradale, F430 Coupe, F430 Spider, 430 Challenge (race car), 430 Scuderia, 16M, 575M, 575M Super America, 612, 612 OTO, 599 GTB and 599 GTO.*

On Ferrari's these all used a Master/Slave ECU pair and Siemens C167 microcontrollers (at various clock speeds and storage configurations which differed depending on year of manufacture). By the time the Ferrari 458 GTB was released, the technology used in the **ME7** ECU's was obsolete and needed to be replaced. At this point Bosch chose to use a new improved Microcontroller from manufacturer NXP. This was an MPC5xx series chipset and was about twice as fast as previous chipsets and boasted 32-bit CPU's (previous generation was 16-bit and could trace its origins back decades). This new generation was known as **MED 9**, with 'D' added for Direct Injection. The MED9 was an evolution of ME7, not really a revolution so carried on where the previous left off this time using an embedded PowerPC CPU. The models that MED9 was used in include the *Ferrari 458 GTB, 458 Spider, 458 Challenge, FF, F12, F12 TDF and California.*



ME7 also implemented the (at the time) latest innovations in the form of Bosch Wideband O2 Sensors (LSU 4.2) which replaced the original Narrowband O2's of ME5.2 (its older generation) which were both slower to respond and only flipped between 'rich and lean' whereas these new sensors could give an accurate reading of O2 and much faster too. Once up to operating temperature this allowed much more precise and accurate fuel management, not only where they quicker to respond they also gave the ability to adapt fueling 'on the fly' which was quite revolutionary at the time.

For emissions reasons Ferrari targeted 14.7 kg of air to fuel ratio which is known as the "ideal" combustion of 1 kg fuel to air at various points in the rev range and at specific requested loads. In engineering, this ratio of fuel to air is called the stoichiometric ratio.



Typically, during warm-up the engine will consume more fuel (older cars used to have a choke) and then as torque requests rise the target fueling will enrichen at higher rpm ranges and high rates of torque requested. The engine control unit needs exact information on the intake air flow in all operating situations so it can set the correctly adjust the ratio of fuel to air, so when ME7 was introduced the HFM (Hot Film Mass) Airflow meters where used, these allowed precise information to be derived and used in a feedback loop using the Wideband O2 sensors to verify and adapt fuel + or – up to 25% to achieve 'target fuel' to account for issues like small gasket leaks or other worn components, beyond this a CEL (Check Engine Light) will illuminate.

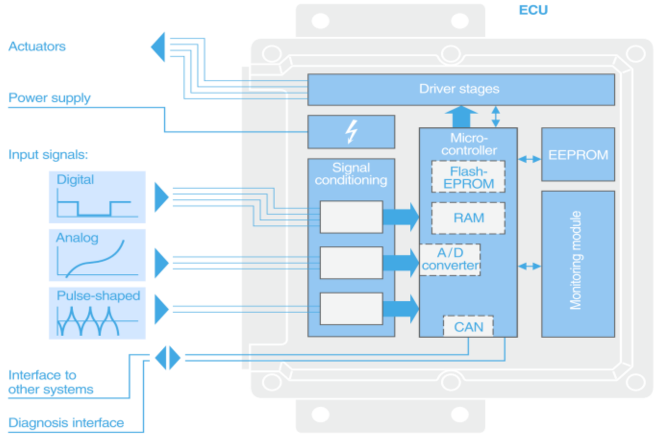
This is why it is critically important on Bosch equipped cars to have good working Primary O2 sensors and Air Flow Meters. Without which the ECU cannot target the correct air fuel ratios and as such the air ratios won't be accurate thus either running too rich or too lean.

*"The **575M** has huge untapped potential, with the right upgrades it can **shock and awe** you **with 595 rwhp (Enzo power!)** up from 430 rwhp"*

Advanced Sensor Modelling

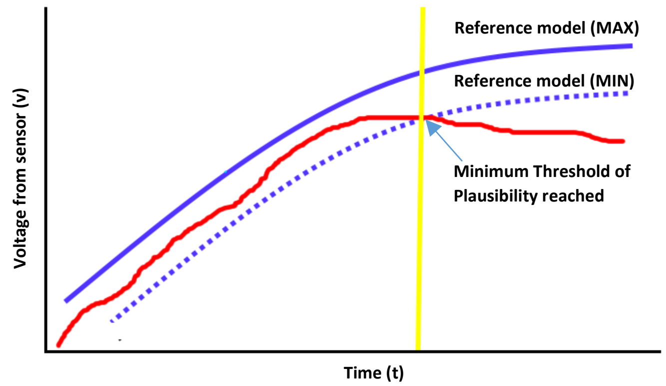
The Bosch ECU's have highly advanced diagnostics of their sensors and actuators. Detailed modelling and verification was incorporated into the design from inception. Every sensor that is connected either sends its signal in one of 3 ways, analogue, digitally or via CAN. (1) Digitally (as in an on or off e.g. for a switch). (2) A communications protocol (e.g. CAN bus). *[Typically originating from another computer on the bus such as the ABS computer]*. (3) Finally sensors are also directly connected to the ECU to measure an analogue voltage signal in the range of 0-5V (automotive sensor standard) as an analogue reading or pulse shaped.

Over time there has been a migration away from sending pure voltages over wires due to the tendency for the signal to be 'corrupted' by either noise from the alternator and other sources or wiring which has sub optimal resistance (bad earth's etc.), as such modern sensors today sample the signal "at source" and generate a CAN message which is transmitted across the bus as a digital signal.



In the generation of ME7/MED9 most sensors tended to have their analogue voltage signals sent directly to ECU and sampled within the ECU. The Motronic system has split out this requirement into various elements. The primary one being to receive at high speed (typically 1ms) multiple 'samples' and then average the signal (so as to eliminate noise) or put the signal through sophisticated 'noise' filters. The Analogue to Digital capabilities of the embedded microcontrollers are relied on here but require correct calibration to the type of signal envisaged and how it is impacted by delay or smoothing. These too have improved with newer generations of microcontroller getting higher resolution ADC capabilities for more accurate signal capture (more bits of resolution).

Before the 0-5V sensor readings can be used in calculations they require 'linearization'. That is they have a table of reference voltages and those points are mapped to a given 'arbitrary' value for what the sensor is monitoring. For example an Air Flow Meter will output a 0-5V based on the amount of airflow 'sampled'. In order to translate that into something 'meaningful' that signal needed to be understood. For example by putting the sensor under lab conditions and actually measuring the amount of air accurately going past the sensor a model can be built up and correctly modelled so we know at any given voltage how much air is flowing. Again this can get complicated when you add in air density changes the amount of oxygen content, hence use of an altitude sensors to provide even more accurate correction capabilities.



In the above chart we can see the expected signal traces between the min and max ranges vs the actual data sampled from the sensor (red). The yellow line shows the point at which the diagnostic function triggers the CEL, precisely after its signal falls below the minimum threshold expected.

The SD series of Ferrari Factory Diagnostic tools incorporate a communication with all the individual ECU's within them to be able to read these as fault-codes. Many diagnostic functions of course are application specific (Ferrari specific, defined in conjunction with Bosch Engineers and Ferrari) so the factory tools for example have an advantage here of being able to actuate tests and verify results that most generic OBD tools cannot understand (they are not published and the interpretation of manufacturer specific P-Codes is not available to those tool makers).

Ferrari used a 'Master/Slave' mode in their cars (8 or 12 cylinders with dual throttle bodies mandated this).

Unfortunately Bosch never implemented a 'start-up' verification check to validate that Bosch Master and Slave are running identical software versions.

The result is you can sometimes find 'hacked' Ferrari's with breaker purchased ECU's of 'unknown origin' fitted to the 'slave' ECU side. These are almost guaranteed to be running incompatible software (even if they have the same Ferrari part number as there are up to 4 revisions of software PER part number!). The Slave side is not immobilizer pairing verified (only the master) so it doesn't matter if it is locked to another car or not, it will still try to run.

Worst case is it doesn't even matter if its running an F1 and Gated part number from radically different assembly number ranges. The car will still start but of course not without consequences… and with no fault codes of course! This is primary reason why Ferrari would never sell an individual ECU. They must always be fitted in pairs!

What Bosch did was 'model' all sensors for their 'correct' operation. Under 'normal' operating conditions the minimum signal possible is derived and the maximum possible signal too. Once these 'thresholds' are known its relatively straight forward to derive plausibility. I.e. that the signal being received by the ECU is within the expected range of parameters. If it is not (in relation to other working sensors) then a fault CEL (Check Engine Light) can be illuminated to highlight precisely which sensor or actuator is misbehaving, either intermittently or failed. The CEL can be stored either in ram memory (which is erased once the power from the battery is removed) or persistently in EEPROM (which survives a battery power off) and a diagnostic tool can retrieve these codes over OBD II diagnostic interface.

Bosch was in a very unique position here to perform exceptional diagnostic modelling having developed and produced most of the sensors used in engine operation themselves, thus having full technical data available to be able to correctly model what sensors 'happy path' should look like relative to other sensors and programmatically encode such models. It took a while before the all Diagnostic routines where fully operational though, on the Ferrari 360's, early MY 99's have the least amount of working diagnostic functions but they improved over time.

> "Diagnostics in earlier '99 MY 360's didn't really work properly…"

Gearbox Control Strategies

Bosch implemented control strategies within the management control algorithms which are complicated and have extremely detailed algorithms behind them so it is not possible to use firmware designed to work with an F1 robotized without letting the software know about it. For starters when the software is running on an AMT gearbox it expects to see a TCU on the CAN bus sending messages at regular intervals. If no messages are received the engine control system will raise a CEL.

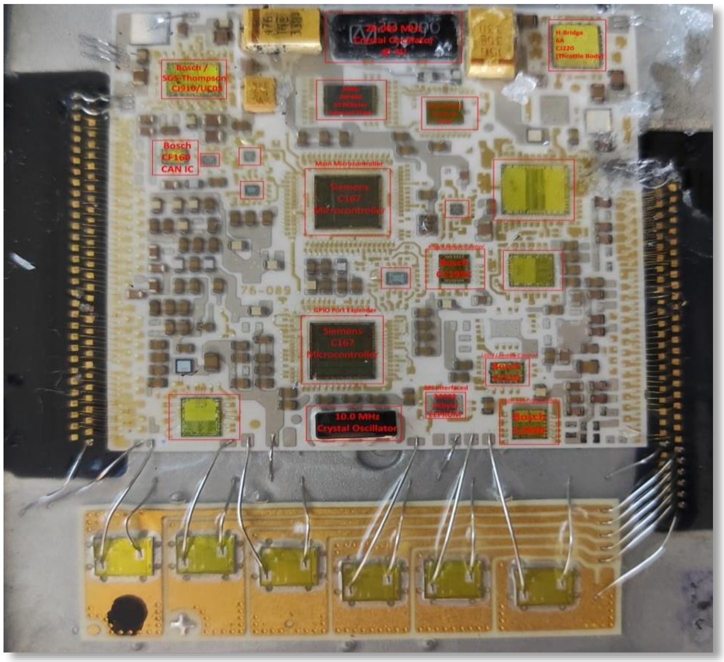
Ferrari also began to work with Magneti Marelli, their Transmission Control Unit supplier to request improvements to the system. These became known as SO-FAST. By around 2008 the F430 had seen big improvements and new sensors added to the gearbox which had allowed the TCU to perform automatic PIS (point of initial slip) re-calculation at all times. This meant the system no longer 'got out of sync' with the distance between the clutch engaging and slipping yielding much more accurate control of the clutch and therefore improved engagement and lifespan.

The 430 Scuderia the system was at SO-FAST III and by the time this was introduced the AMT was dramatically improved again. This time however it wasn't just software in the TCU that was impacted but also the Engine ECU's and even internals of the gearbox had to be improved for the faster shifting to work reliably.

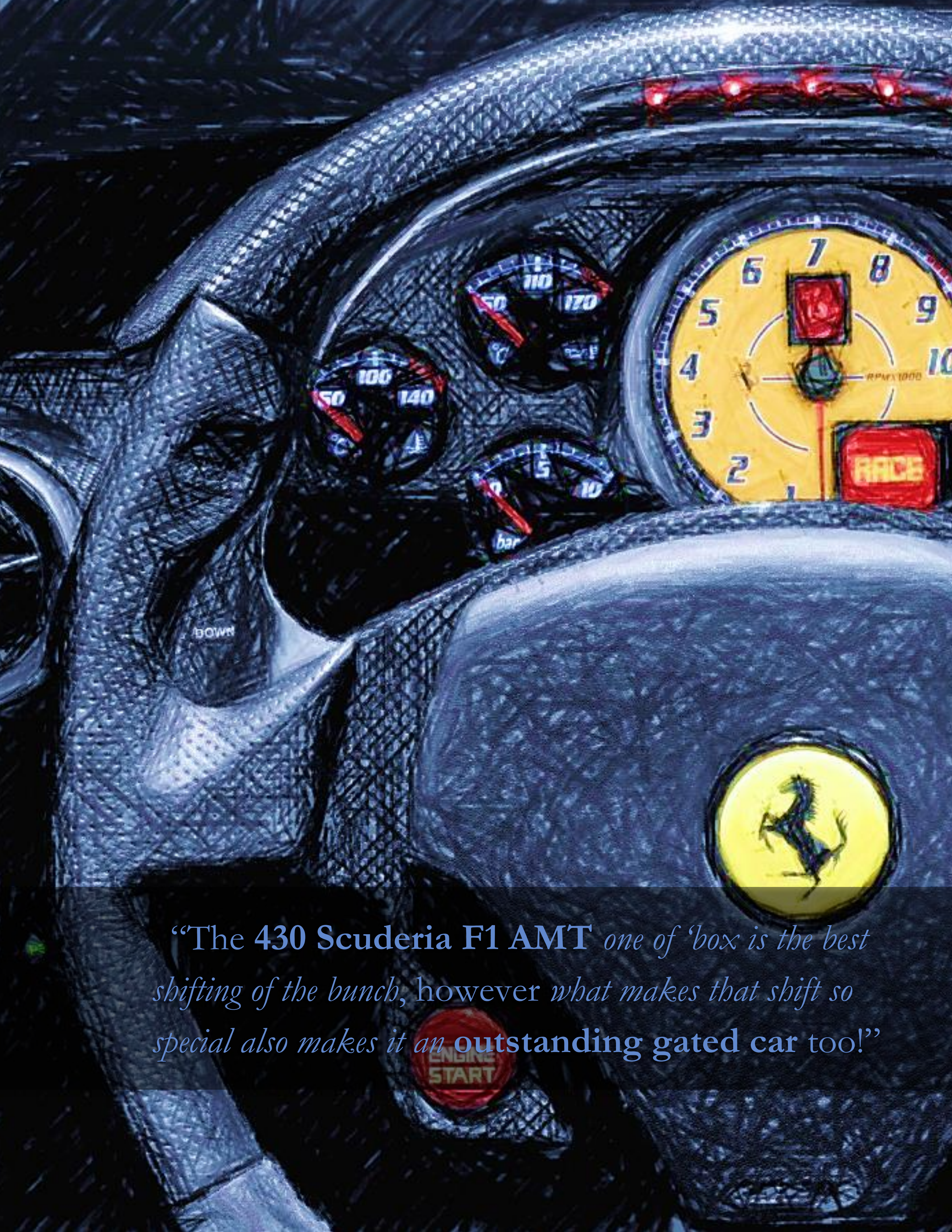
SO-FAST III Evolution

At this point we saw changes made to the engine software that made the manual gated cars essentially 'incompatible' with the baseline firmware and Ferrari was already leaning towards F1 only (After all over 80% of their cars sold at this point where ordered with F1 anyway so it made absolute sense from a cost saving perspective too).

This software evolution however gave Ferrari 'no way back' even if they wanted to produce a gated car and actually this bit them when they eventually decided to make a small production run of 599 gated cars which had been demanded by their high net worth clientele. Unfortunately due to the incompatibility in the software the result was those care drove very poorly. The revs would drop too fast when a human was engaging the clutch (which was perfectly matched to the robot and hydraulics shifting gear). With time and costs a major factor for such a small production run the 599 gated cars where never destined to correct these 'bugs' and they drove very disappointingly. Some even claimed the reason was that the car was designed with a very low inertia flywheel and therefore rate of rev drop was 'inevitable'. They all however missed the fact that rev control between gears has long been controlled by software (just as torque is managed too). It was indeed a software issue and of course we fixed it!



Ferrari 360 ECU's are tricky to repair due to Substrate. Typically the orange tantalum capacitors are the common failure points.



“The **430 Scuderia F1 AMT** *one of ‘box is the best shifting of the bunch*, however *what makes that shift so special also makes it an* **outstanding gated car** too!”

ME 7 ECU Hardware Design

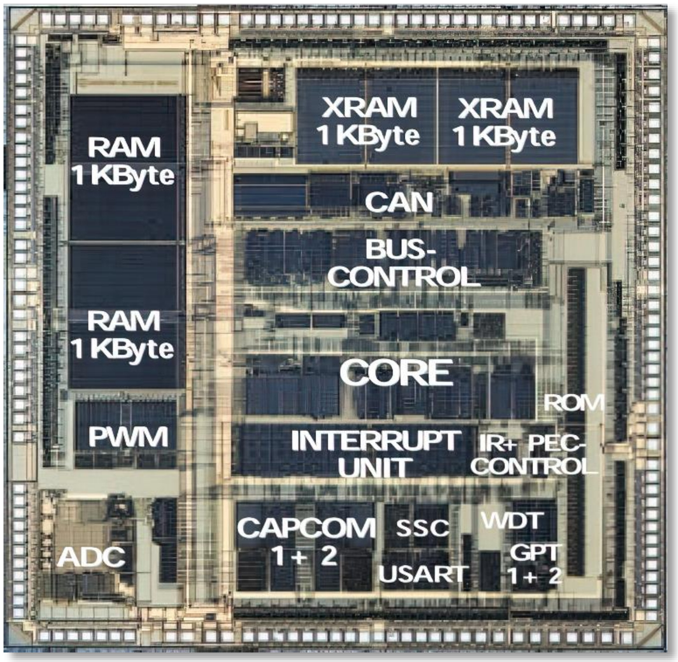
The automotive industry moves much slower than other hi-tech areas and Bosch still adopted early 1990's technology in cars that were produced in even as late as 2010 Model year vehicles. **ME7** ECU's where all designed around the capabilities of an inexpensive low-power RISC (Reduced Instruction Set) 16-bit microcontroller with a 144 pin package, the kind of thing you'd find in a coffee maker or toaster of the period. It was developed originally by another German company, *Siemens Semiconductors* (now Infineon) as a general purpose chip and was used in countless products of the era, not just engine management systems.

This chip really wasn't ideal for Bosch but it was inexpensive and that mattered when you were making tens of millions of them. The biggest hurdle Bosch engineers had to overcome was that the chip had relatively few GPIO (General Purpose Input Output Pins) which are used to connect all the sensors and actuators to it. In the end they came up with quite an unusual method to resolve this (no, not a pin multiplexer) but to use TWO microcontrollers and join them together via the SSC (a sort of high speed serial interface), with the other controller simply quite 'dumb', just capturing the signals and transmitting them to the "master" controller at frequent intervals.

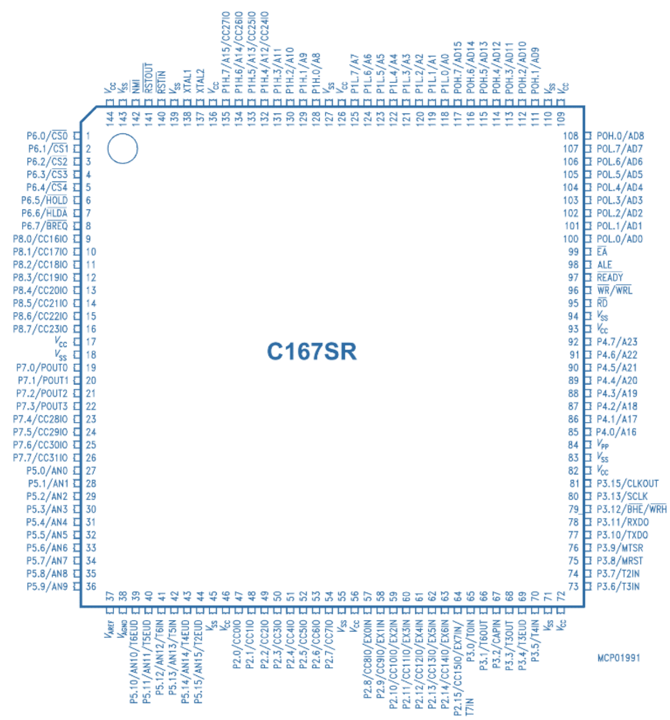
The master **C167** Microcontroller core was also designed to run 'standalone' but it only had a miniscule 4Kbytes of internal memory which was also insufficient for the type product Bosch had planned. Since the chip was designed to be extended they simply included an external 32Kbyte of memory (36Kbytes of memory total) and then depending on the generation and year kept expanding the external flash and EEPROM appropriately. The Ferrari 360 started out with a version of ME7 ECU equipped with just 512Kbytes of Flash (since it was 16-bit and used segmented paging its theoretical limit was actually up to 16Mb addressable flash) and 1Kbyte of EEPROM with the CPU running at 20MHz and 32Kbytes of main system ram. To give you some perspective by today's standards it now has less power than a 'door lock module' but don't let that fool you, the software was very complex and highly efficient.

By the time the Ferrari F430 came out those specs had been doubled, with the CPU now running at 40MHz, the flash at a whopping 1Mb (about the size of a single jpeg photo we take these days on our smart phones) and 2Kbytes of EEPROM. The 575M had to make do with a 33MHz CPU. One unusual feature of the Siemens chip was the use of a technology called Mask-able ROM. It was essentially ROM (Read Only Memory) burned into the chip at factory so you could order your chips (in large volumes of course) with your own software 'hardcoded' into it.

Today this concept sounds strange but the idea was entirely rationale back in the day when the re-writable flash was expensive this was a cost effective (if limited) alternative, embedded products really where designed to never be upgraded and having ROM was one sure fire way to ensure that. Even Bosch decided to skip this feature (for the most part by using it for its boot loader and some library functions only). By using external Flash connected via SPI (Serial Peripheral Interface) along with the EEPROM and memory chips software could be upgraded. This at least allowed the software to continue to be developed (and re-flashed) as and when Bosch where ready.



View of the Transistors (Internals) of the Siemens C167 Microcontroller.



More conventional view of the chip showing all the pins and their purposes.

ME 7 Hardware –Circuit Board Design & Layout

Let us 'demystify' this area a bit. As part of the work Bosch **hardware engineers** will have done, one activity is the actual design of the ECU printed circuit board hardware. Here they will have used software such as **EagleCAD** (commercial), **Altium Designer** (commercial) or **KiCAD** (free) to layout the board and place all the components required. To begin with, the bring-up of the Microcontroller would have been the priority and then adding any required peripherals such as external flash, ram and EEPROM. A good understanding of all of the chips, sensors and actuators we will be connecting is required at this point, however we haven't covered them yet so let us just assume that knowledge is already known and come back to that later. This chipset had 144 pins, today an equivalent chipset is running over 500 of them so while they have got a lot more capable (lots of new interfaces, more GPIO etc. fundamentally they do the same thing, so looking at earlier designs which are less complex still is a useful experience.

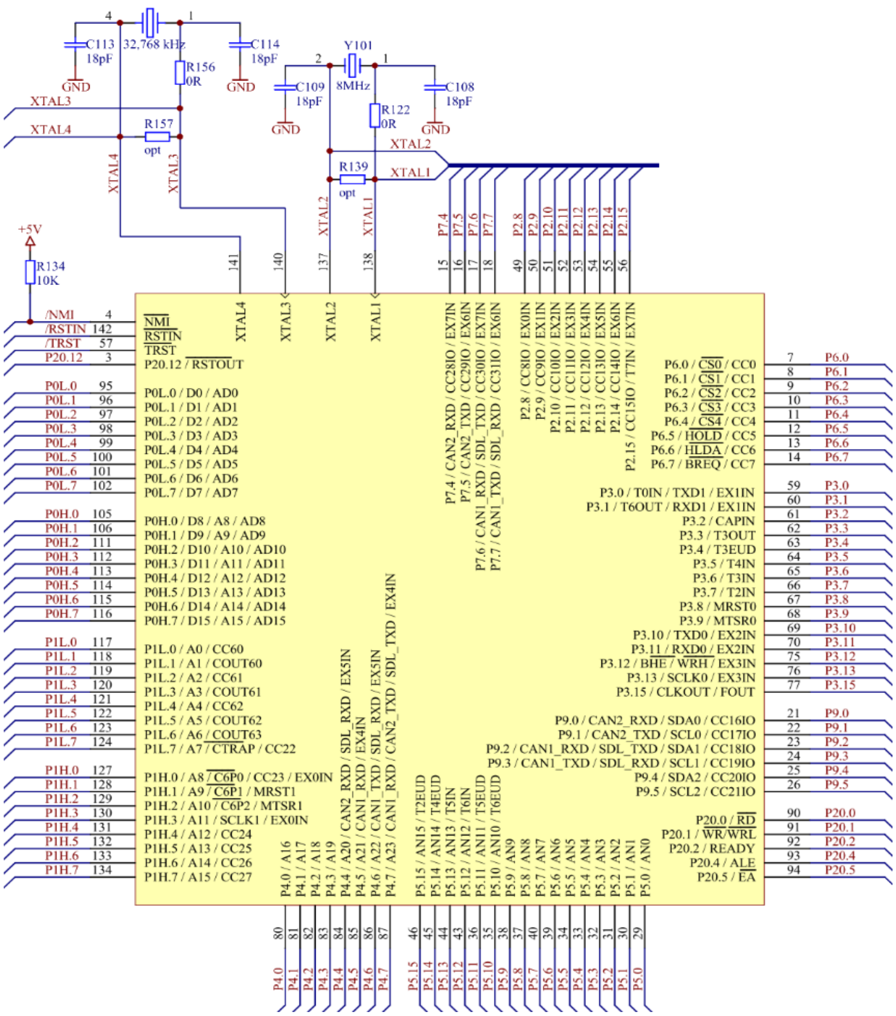
This looks complex but all we are seeing here is the view of all the wiring for the main microcontroller [or embedded CPU]. Grouped into logical areas called Ports, they are the ways to configure the GPIO pins so they can be joined to other things such as IC's, as either inputs or outputs. A full description of all of the Pins is available in the documentation called the **Datasheet.** Once you get familiar with viewing **Schematics** they are often quite easy to read at a glance.

A full description of all of the Pins is available in the documentation called the Datasheet. Once you get familiar with viewing Schematics they are often quite easy to read at a glance. Here you can see pins labelled D0 to D15 and A0 to A15 (left hand side of the chip) they are all the ADDRESS lines and DATA lines used to communicate with another chip like RAM or FLASH, the CS#0 (P6.0 through to P6.4) allow up to 5 in this case different SPI devices to be selected (Serial Peripheral Interface).

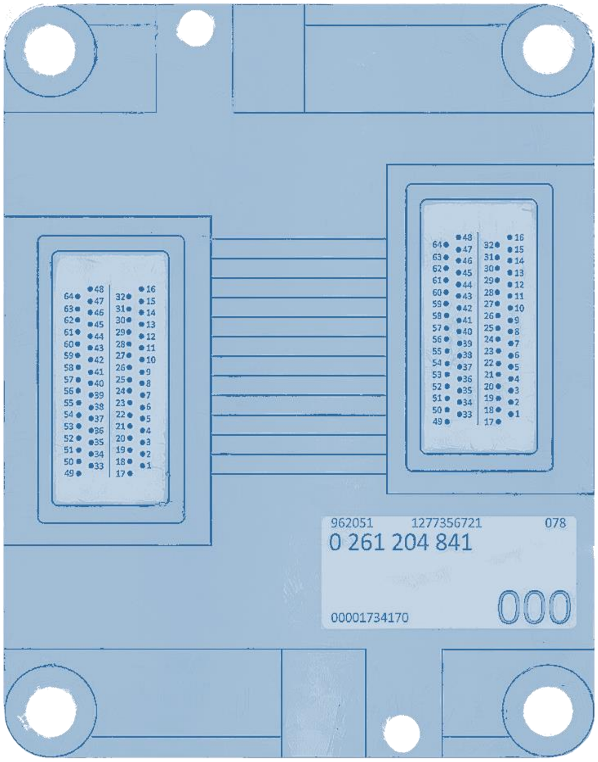
The CS lines are Chip Select and they allow you to choose (in software) which peripheral you are communicating with. So for instance Bosch on the F430 ECU's used CS#3 for the EEPROM. So P6.3 is used as the EEPROM chip select line. **the CS pin varies on the 'version' of the chip used hence why it is important to always refer to the datasheet.*

This is how you begin the process of designing the circuit board used for a new embedded project, reading the datasheets and working out how to add the various components such as a CAN bus transceiver, the microcontroller so communications can be done with the vehicles CAN network (for example).

Understanding the interfaces such as SPI, I2C, UART, SSC are key to utilizing a chips capabilities to the fullest and they give you the ability to extend its features way beyond the basics.

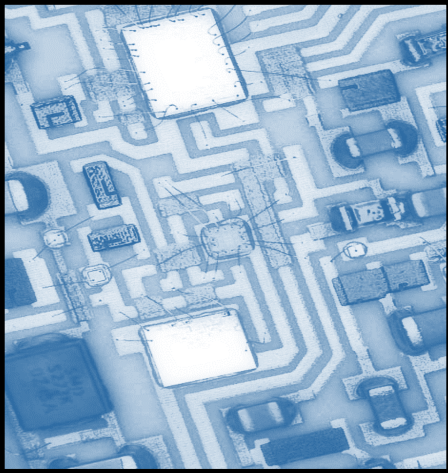


***Schematics** describe the individual connections at a Pin level and being able to read them is important for both Embedded Software and Hardware Designers alike.*

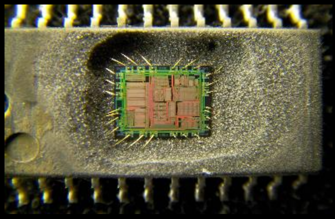


Back at the turn of the millennium when everyone was obsessed about the Year 2K (search 'Y2K bug' on Wikipedia for more information) Bosch engineers were busy looking at alternative ways to improve the weight and footprint in a vehicle of the Motronic ECU's.

Already Bosch had ventured into Bare Substrate manufacturing for their ABS computers, in an attempt to downsize them and give back space to the manufacturer.

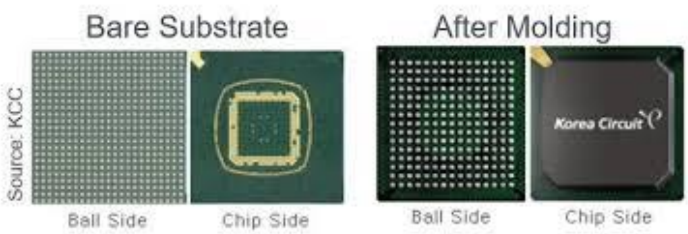


The techniques used "Place and Bond" machines. Tiny microscopic wires where bonded to 'die' assemblies, without the traditional package and pins (thus saving space). While spacing saving the technology wasn't without its issues….



ME7.3 Bare Substrate (using Bonded Wire)

MIL-STD-883 standards define MPT wire bonding processes. The Ferrari 360 used for its time, state of the art ECU technology which was heralded as 'next generation' for its ability to be placed in very harsh environments such as the engine bay. The ECU's in the 360 itself where in fact placed in the engine bay (behind the drip trays for this very reason) to space saving in the cabin and reduce wiring loom length. While this worked it in labs it quickly fell out of fashion when the apparent **failure rate** was proven to be *higher* than traditional circuit boards and components. Furthermore it also severely restricted the options to buy off the shelf parts and repair failed electronics economically which was always a mainstay of expensive electronics 'recycling'. The technology is so small it is very difficult without expensive machinery and techniques to repair these kinds of electronics. A kind of waterproof 'gloopy' substance which is transparent and makes repairs much more challenging for traditional electronic repair shops (also dying out).



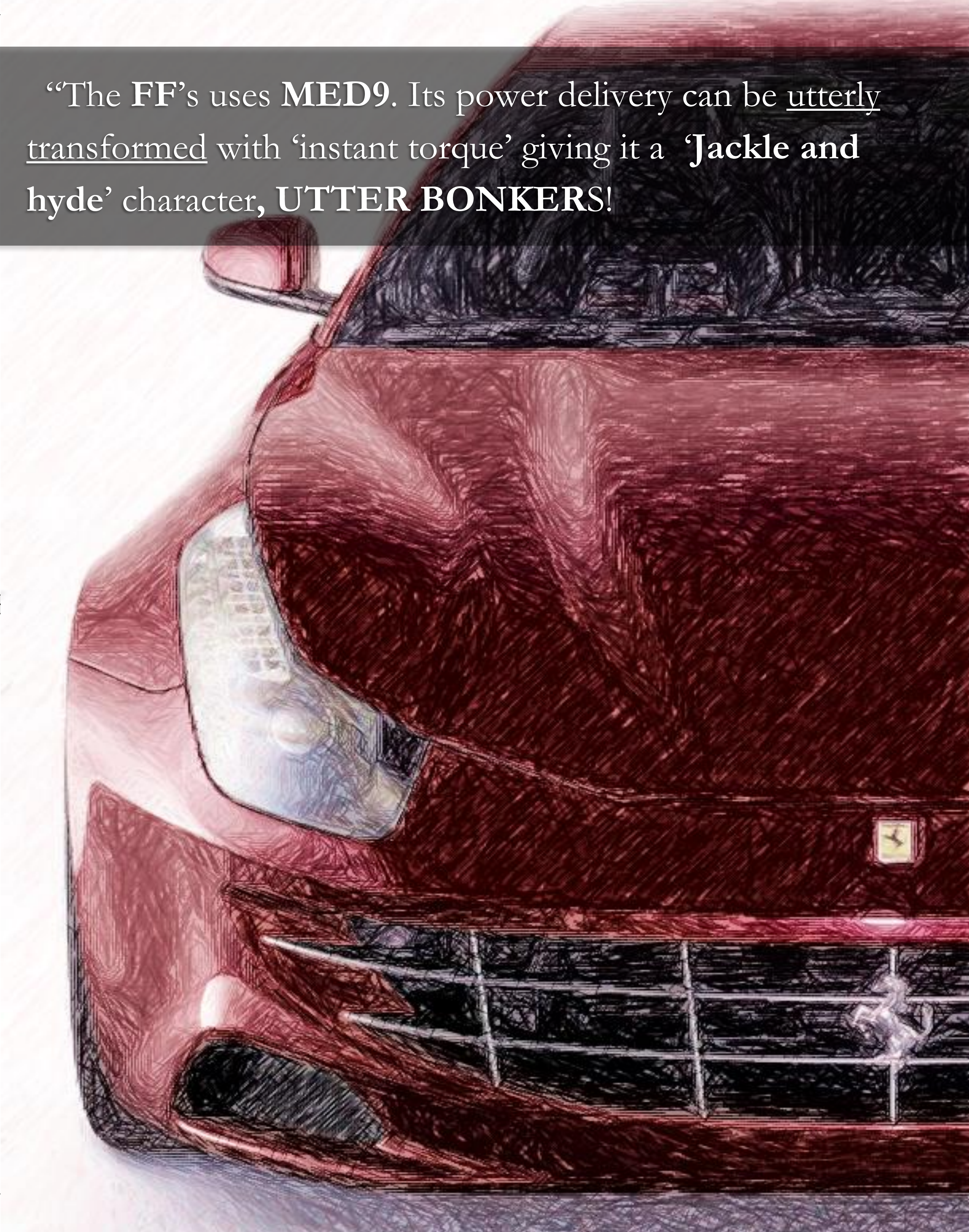
The removal of 'molding' material reduces the amount of space required on the circuit board because the wires which attach to the circuit can be thinner than human hairs if they are GPIO pins dealing with signals only.

Even with BGA and ball mounting the molding on a chip can take up considerable footprint. This is significantly reduced when using Bare Substrate so packages can be much closer together.

Ball bonding uses round gold wire (typically 0.001" in diameter). An electric discharge creates a ball at one end of the wire which is bonded to a conductive surface with a thermosonic ball bond on a heated stage. The second bond is a wedge bond, which can be topped with a security ball bond. Ball bonds are able to bond to a smaller bond pad than wedge bonds

Wedge bonding can use either gold or aluminum wire or ribbon and is created by ultrasonically welding the wire between a rectangular wedge tool and the bond pad. An oval shaped bond results, which is very strong and is well suited for larger wires and higher power. Gold wire bonding requires a heated stage, while Aluminum wedge bonding occurs at room temperature, which can avoid thermal damage to some sensitive components. Most IC's include Aluminum metalized bond pads, so Aluminum wire bonds also avoid any intermetallic formation between gold and aluminum which can lead to reliability issues over time particularly at higher temperatures.

“The **FF**’s uses **MED9**. Its power delivery can be utterly transformed with ‘instant torque’ giving it a **‘Jackle and hyde**’ character**, UTTER BONKER**S!



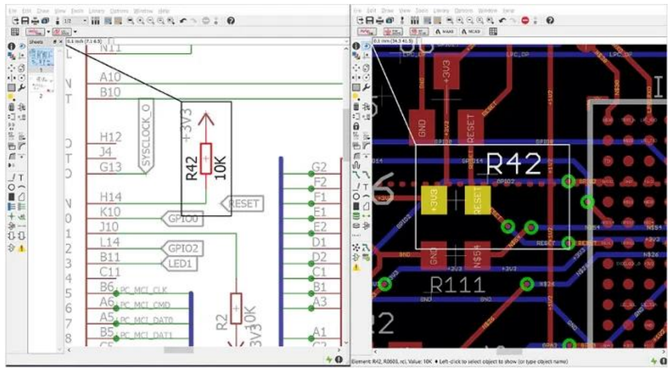
Once the work of deciding all of the peripherals to use is completed the task of doing 'circuit board layout' would commence and eventually you will end up with a layout which resembles pretty much what we end up with at the final stage, a circuit board and a BoM (Bill Of Materials). That's the full complete list of all of the components required to build the board and the cost of each one.

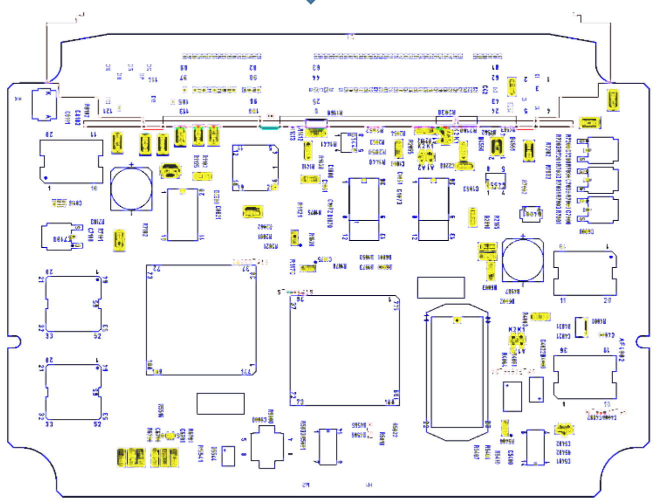
These days a company such as *JLCPCB* can manufacture you a small production run board inexpensively. From this you end up with a final hardware design in the physical world. Of course this is just a summary of the activities (high level) but the aim here is to give you an appreciation of the steps and what goes in to producing a hardware design. It is beyond the scope of this guide to teach you how to design and layout circuits but if you so desire there are plenty of free training resources out there on YouTube and so forth now that you can do more of your own research if you are interested to pursue this further. **Arduino**'s and **Raspberry Pi's** are a good learning resource to get started in the area.

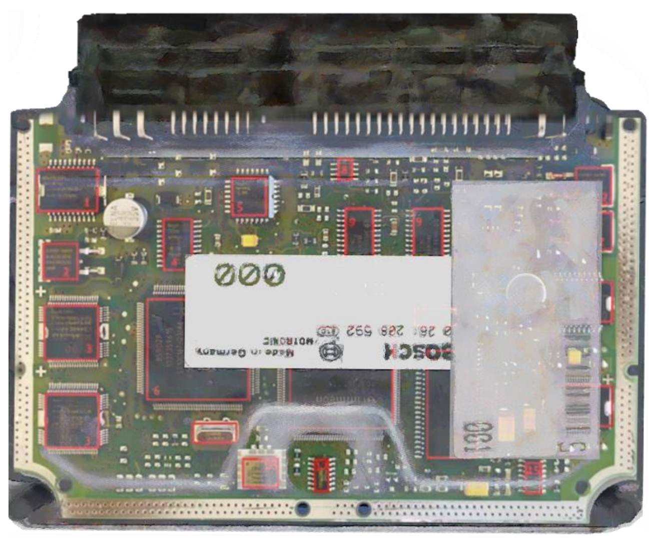
These ancient microcontrollers are now so old that they are obsoleted and no longer available to buy new. What has not changed however is that the same exact concepts are still relevant today as they ever where, just like they were some 25 years ago now when these activities where being planned by Bosch for these old ECU designs. The biggest difference is the capabilities are much higher for very low end chips and they come equipped with much richer hardware interfaces, it is also often much easier to add wireless and network connectivity to designs too.

Bosch Motorsport use a 'hybrid' hardware design in their race based ECU's integrating an Field Programmable Gate Array (FPGA) into the design to give them total flexibility to re-design the hardware 'with software' – it's a skill they inherited from their learnings from ETAS (the company has used the same technology in many of their products including diagnostic tools).

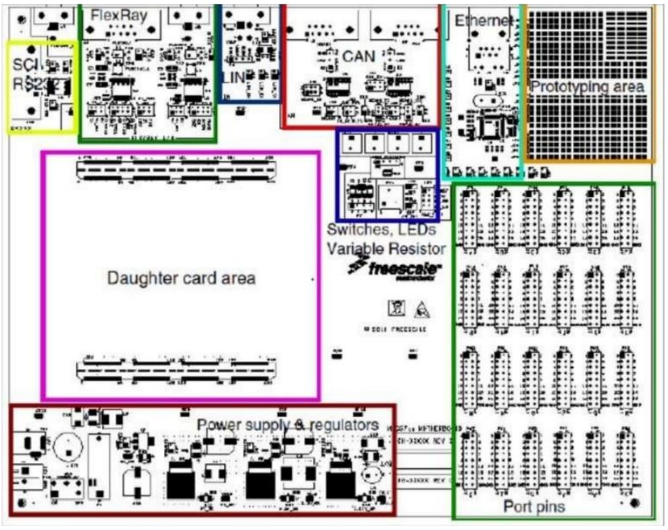
> "Understanding how to read a **Datasheet** is a *must* know skill in the embedded world…"







A 'Period' 1996 vintage Siemens C167 Reference Design board from Rigel Corp which allowed to get familiar with the hardware before designing your own.



A Later Reference design by Freescale for use of MPC5xxx Series cards includes a prototyping area, access to all Port Pins, swappable daughter cards to try different designs, LIN, FlexRay, CAN, etc.

Reference Designs

The **software engineers** will have already started their work long before a first circuit board would have been ready. They typically use something called a 'reference design'. That is a system that is provided by the original designer of the microcontroller to 'demo' the features and capabilities of the chip. Usually there are a lot of pin headers to connect all the additional peripherals to the system, typically on a 'breadboard' during the first experiments.

This '**rapid prototyping**' approach allows you to 'test drive' the system long before the task of making tailored hardware is taken on or completed.

Reference designs now are so complete and comprehensive they often act as a good 'starting' point for the hardware designer to work from, showing all the finer points of getting things up and running and examples of the power circuits and tips to ensure low noise, etc.

> "Software development gets well underway before the final hardware arrives…"

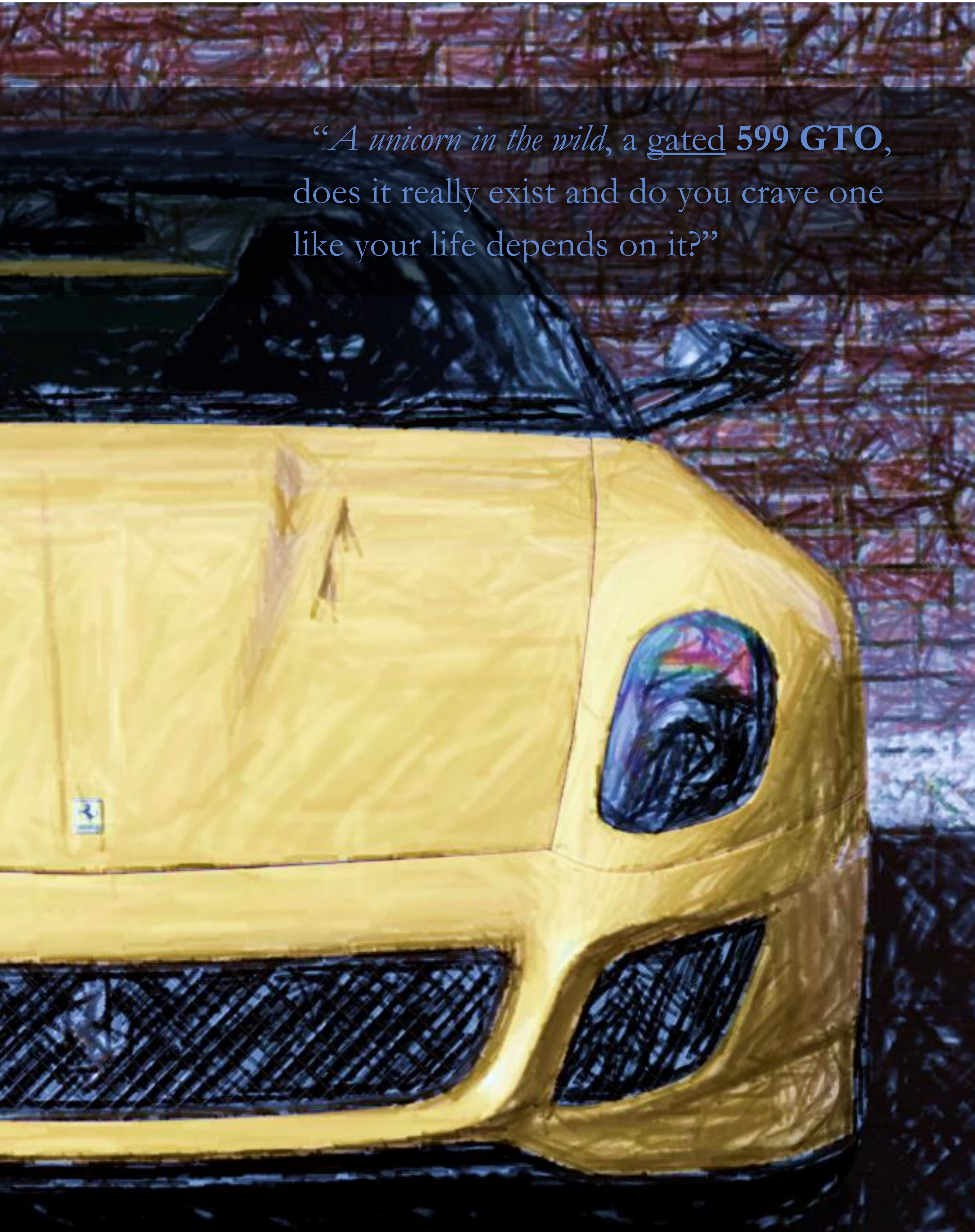
Typically a good reference design also comes with good software examples too of using all the features of the chipset. This is again an excellent way to get quickly up to speed on new architectures and capabilities added by the IC manufacturer.

More and more we are seeing the move away from simple Microcontrollers to fully blown SoC's (System on a Chip), a SoC includes a processor (or more than one) but also integrates other IP blocks too such as integrated Wifi/Bluetooth or even Video Decoding and encryption engines and additional security for booting the chip using only trusted source firmware, to attempt prevention of tampering or slow down the process to reverse engineer the designs.

What is a SoC?

A system on a chip is an IC (integrated circuit) that combines many elements of a computer system into a single chip, usually to reduce size and costs. Mobile Phones use a SoC.

A SoC always includes a CPU, but it might also include system memory, peripheral controllers such as graphics processing units (GPUs), specialized neural network processing (NPUs) and security features like secure boot.



"*A unicorn in the wild*, a <u>gated</u> **599 GTO**, does it really exist and do you crave one like your life depends on it?"

ME7.3 Printed Circuit Board – Decoding the chips

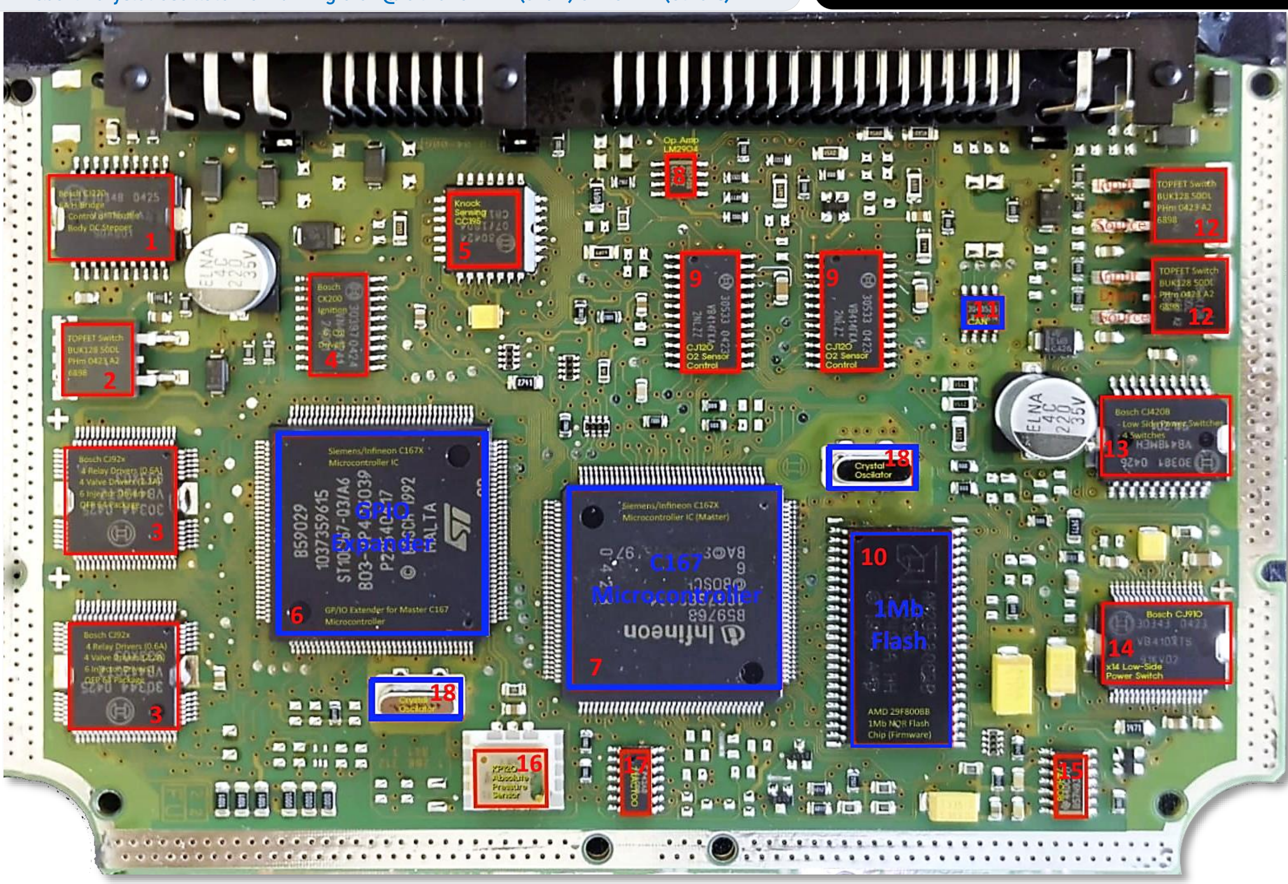
The Bosch **ME7.3** was developed both as a die-substrate board (just like their ABS Computer) as well as a traditional PCB (printed circuit board) so they are very similar in features. The traditional PCB is much better from a repair perspective because it is far easier to replace components when they fail so it has a longer service life and is much more robust too. Here we can see the same basic components as in the Die substrate version, the main difference being that this is a multi-layer PCB so the underside also has components too which is for the *External RAM (32Kbytes), EEPROM (ST95P08 2Kbytes)* and a further **Bosch CK200 Coil driver** (for up to more 6 coils). Bosch decided to try to 'hide' the purposes of the chips by using code numbers, let's look at what they really are here;

1. Bosch CJ220 : 6A H-Bridge – Used to drive the DC-Stepper motor of the Throttle Body
2. TOPFET Switch BUK128 50DL (general input/drain/source power control)
3. Pair of Bosch CJ92x: 8 Relay Drivers, 8 Valve Drivers, 12 Injector Drivers (for the pair)
4. Bosch CK200: 6 Ignition Coil Drivers (= 12 Ignition Coil Drivers with one at back of PCB)
5. Bosch CC195: Knock Sensing Controller
6. General GPIO Expander Chip (with Mask rom)
7. *Infineon C167 - Main Micro-controller (with Mask rom)*
8. Operational Amp (OpAmp)
9. Pair of Bosch Wideband O2/Lambda Sensor Controllers
10. AMD 29F800BB NOR Flash (1Mb) Chip
11. CAN Bus Transceiver
12. Bosch TOPFET Switch (general)
13. Bosch CJ420B Low Side Power Switch
14. Bosch CJ420B Low Side Power Switch
15. 44DC37T Quad 2-input AND Gate
16. Ambient Pressure Sensor (Used for Air Density Calculations)
17. 74ACT00 Quad 2-Input NAND Gate
18. Quartz Crystal Oscillator for running C167 @ either 32MHz (575M) or 40MHz (Others)

REPAIR TIPS

These driver IC's can be diagnosed and replaced (use a thermal heat scanner to see which IC is giving off a lot of heat – it is usually the one that's died). If someone welds on the car with the power connected to the ECU, it is typically the O2 Sensor IC's (9) that get wiped out. Injector driver fail (3 and rear of board same chip), if a Coil driver fails its (4).

**Blue components are everything required to boot strap the environment and communicate with outside world over CAN. All others are used for 'drivers'.*



Why does a V8 or V12 Ferrari need a Master/Slave ECU when there are plenty of coil & injector drivers? It comes down to lack of inputs for the 2 of everything required. Maserati made do with a Single Throttle Body/Air Flow Meter setup *which helped them squeeze it to use a single ECU (saving valuable costs!)*

As always in any design there will be compromises for *economic reasons*, the Bosch ECU's are no different in this regard. As mentioned previously, there was a severe limitation on the number of pins required that needed to be overcome. These where required for all the inputs and outputs as the humble C167 Microcontroller didn't have enough of them. Instead of moving to a more capable Microcontroller a second one was used as a **Port Expander**. Furthermore due to not having enough outputs in the chips selected either it became required to use a second ECU too meaning it wasn't possible from a single ECU since the original design brief didn't mandate it because majority of cars didn't need so many power I/O controls.

The high level schematic to the right shows an example of a design integrating External Flash storage, External Memory and EEPROM added via SPI interface (Serial Peripheral Interface), in exactly the way Bosch did it and pretty typical of a device of that era which required more space than the original 4Kbytes of ram afforded. SPI allows sharing of the address and data pins which in order to connect up all these devices was essential. Where it differs from your 'typical' design is in the need to power quite a few things (and this list excludes all the CAN messages sent too);

Inputs & Controls:

- Throttle Position (Dual Potentiometer) Sensors
- Primary O2 & Secondary O2 Sensors (input & control)
- Hot Film Air Flow Meters (HFM)
- Crankshaft Position Sensor
- Camshaft Position Sensor
- Oil Temperature Sensor
- Coolant Temperature Sensor
- Exhaust Thermistor Temperature Voltage (from Cat ECU)
- Pre-ignition (Knock) Detonation Sensors
- Ambient Pressure Sensor
- Crash Signal (for engine off)

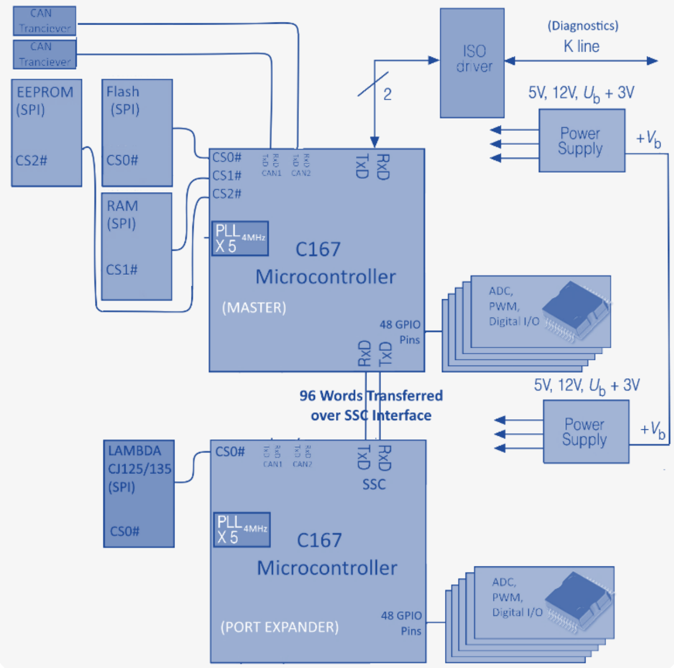
Outputs:

- Fuel Injectors (Via IC)
- Coil Packs (Via IC)
- Throttle Body DC-Motor Flaps (via H-Bridge IC)
- Engine Exhaust valve variator's (on/off)
- Exhaust system loud/quiet mode solenoid
- Modular Manifolds (2 control stages)
- EVAP Solenoid Valve (PWM Output)
- Air conditioning control (on/off)

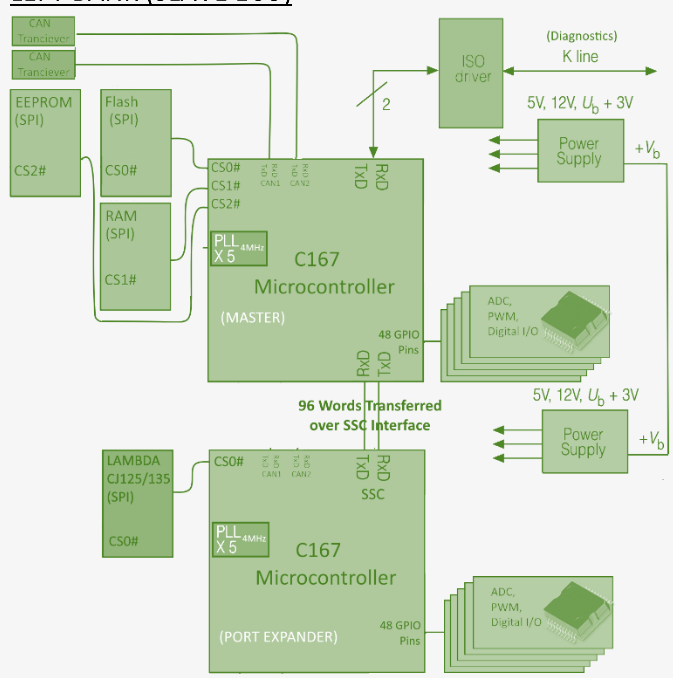
To drive these demands custom IC's where used such as Low and High Side Drivers, H-Bridge IC's and customized Bosch IC which controlled the O2 Sensor requirements (not depicted on this example high level schematic to keep the example clearer but *see previous page*). So while its origins are pretty simple it ended up utilizing a 4-cores but not purely for performance reasons.

By the use of dual ECU's (Master and Slave) and the use of two micro controllers PER ECU we actually ended up with a design that dramatically extended the original scope of the humble C167! In all it use x4 of them to drive 8 or 12 cylinders and x6 for 16 cylinders (Bugatti Veyron Anyone?)!

RIGHT BANK (MASTER ECU)



LEFT BANK (SLAVE ECU)



Due to the fact that Bosch only placed enough driver IC chips (depicted simplified in diagrams above) on the original designs a Master/Slave ECU configuration was required for ALL Ferrari's. This meant use 2 ECU's with each ECU operating 4 cylinders and a complete duplicate of all the sensors.

Now you know the exact reason why Ferrari's use dual ECU's!

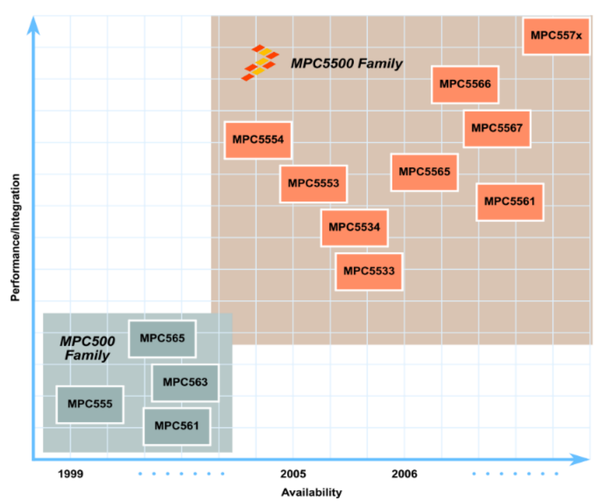
"The **458 GTB** was the *first of the <u>new generation</u>* to get **MED9** which brought with it **Direct Port Fuel Injection** as well as **Ion-Sensing** (*first seen on the 430 Scuderia)*"

MED 9 ECU Hardware Design

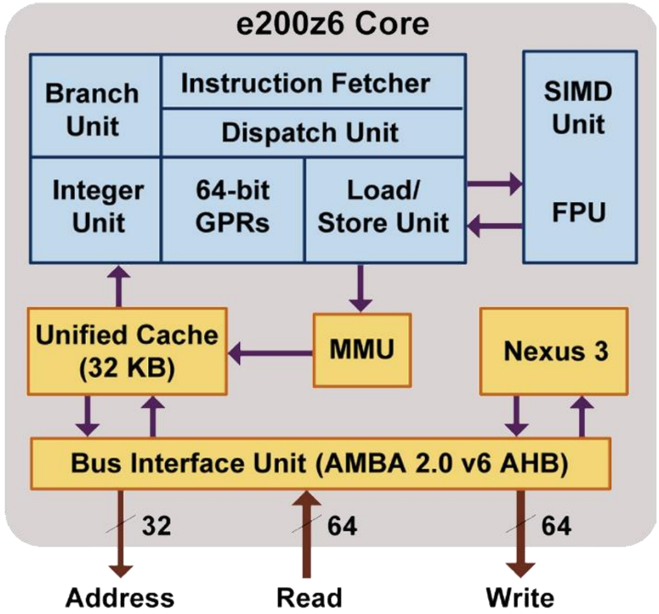
The Bosch **MED version 9 series** ECU's where all based around a more modern MPC5xx family architecture (a decade newer technology (!) and from this century – but still *old* by today's standards). It used **32-bit** embedded (e200z6 core) PowerPC core with Unified 32-KB, 8-way set-associative cache memory and 7-stage pipeline (with in-order execution).

This time they were from (US-based) Freescale Semiconductors (Motorola) and it was a departure from the earlier 16-bit C167's which was finally de-commissioned from its long service life in ME7. Processing speeds improved with a similar base 40Hz clock speed out of the box (optional 56 or 66MHz was possible and indeed used in Ferrari applications). Also code generation quality improved, the C compiler improved things significantly because PowerPC had also been used in old **Apple** computers, **PlayStation III** and **Nintendo** consoles and so had much more advanced GCC compiler optimizations. The result was faster code than the clock speed would suggest, plus with improved superscalar design and VLE (Variable Length encoding) it was also memory efficient too. In also came 64-Bit Floating Point math (in hardware as an internal FPU module). The Ferrari's like the 458 and beyond used the cost optimized **MPC56x series chipset.**

Clearly Freescale's technical department had been studying the C167 architecture too as many features used by Bosch where present and correct including, this time internally a 32-Kbyte of static RAM (no external chip required) and optional 512Kbytes of internal re-writable flash (gone was the Mask-Rom from earlier generations). If Bosch had prioritized to optimize their internal code base they could have used this chip 'as is' with no external flash or ram, reducing their costs for the Bill of Material. Furthermore it added Three internal CAN modules (vs 2 of the previous generation) and the pin count *ballooned* to **388 -PBGA** (Plastic ball grid array) package with more than enough GPIO pins thus reducing costs further still, meaning Bosch had no need to go the route of Port Expander with this chipset, even so Motorola added its equivalent of SSC in its own way (QSMCM – Queued Serial Multi-Channel Module) just in case. Clearly they were taking no chances here and technically left nothing out that could be missed.

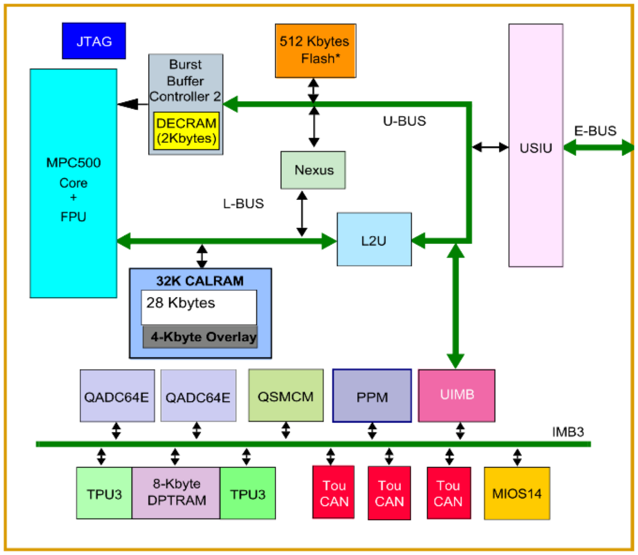


Roadmap presented in 1999 showed a path to increased performance eventually ending up at MPC5777C of circa 2015



e200z6 CPU Core is a SIMD, Single Instruction, Multiple Data. It's available as licensable synthesizable Verilog source-code for custom ASICs designs

"PlayStation III's and Nintendo's share common PowerPC heritage with cores found in the Ferrari 458's"

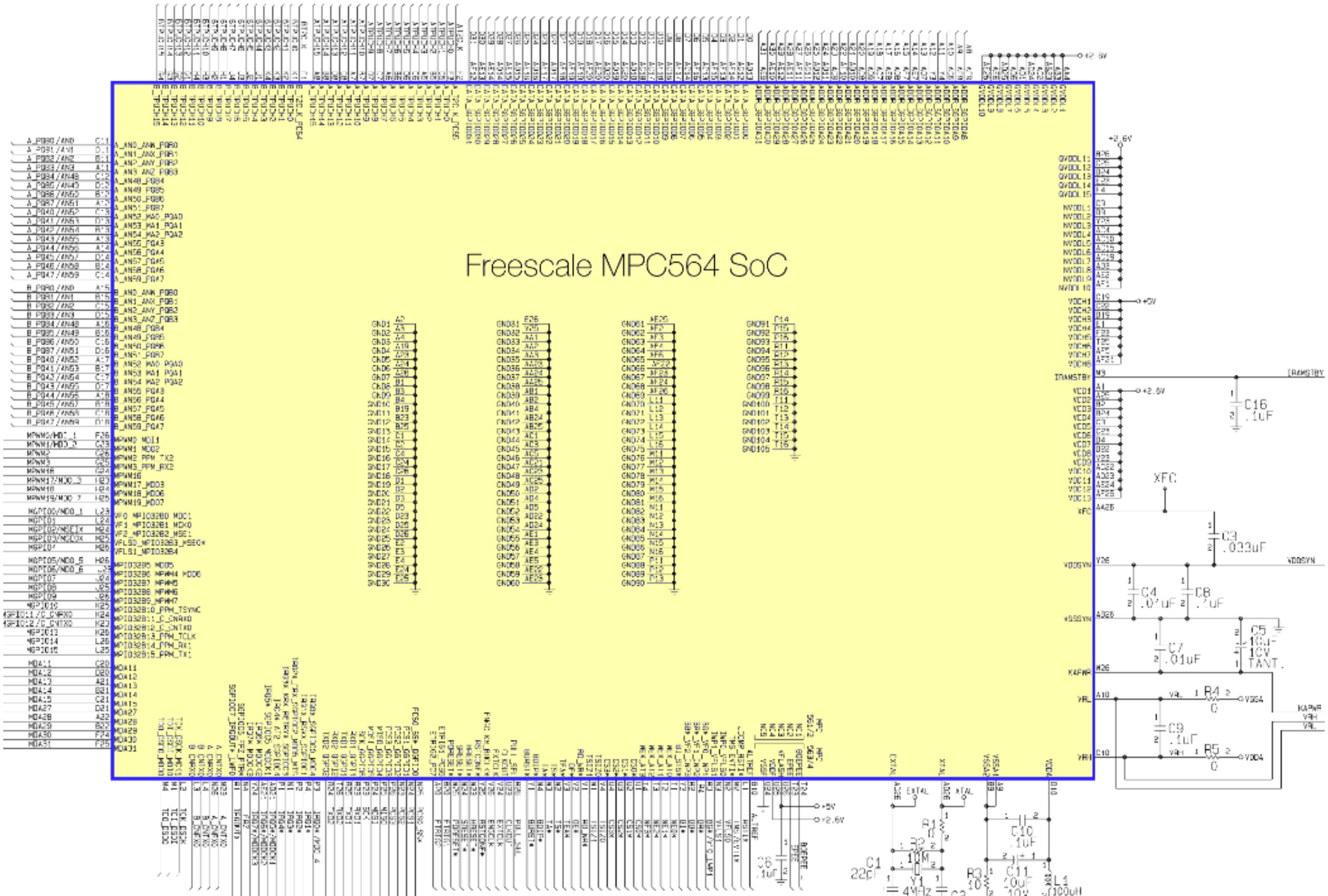


MPC500 series architecture had 32Kbytes of internal CALRAM and 512Kbytes of Internal Flash as well as a CPU (e200z6) and FPU math in hardware.

MED 9 – Driver Chips vs GPIO Pins

The biggest call out from the new chip was the <u>vast amount of GPIO</u> available internally, absolutely no problems to support all the sensors and actuators a project could throw at it. They also used the opportunity to improve efficiency with low power modes, enhanced security features through the use of hardware encryption and keys. This was no surprise considering the chip was more than a decade newer, this time from circa 2000's vs the 1990's. Ferrari's still however (frustratingly) had to make do with Master/Slave ECU's due to the fact the board layout did not accommodate enough space for the additional "driver" chips required to run dual sensors, e.g. throttle bodies, air flow meters, o2 sensors and so forth. This quite easily could have been achieved with a de-populated PCB but they decided not to do this for some reason.

Freescale had a full 'glide path' of options to much higher performing chipsets in the form of MPC5500 Family which continue to this day, however to the surprise of many people Bosch took Motronic back to its roots for the replacement of MED9, going back to its ***German Supplier***, Infineon for its **MED17** replacement chipset, in the form of the **Infineon Tri-core** series chipset which continues the story to this date (As used on the Ferrari 488, 812, SF90, etc.). More will be covered on **MED17** in part 2 of the series.



Schematics for the SoC as used in the Ferrari 458 and others.

Bosch CY315	: *Power supply, Monitoring, CAN, Small signal driver, Serial IO*
Bosch CAN	: *Additional CAN Communications*
Bosch Ambient	: *Ambient Pressure Sensor*
Bosch CC196	: *Knock detection*
Bosch CJ945	: *18x low side, power amplifiers, serial control, programmable, current limitation*
Bosch CK110 / BIP306	: *Ignition coil driver/transistor*
Bosch CJ230	: *Throttle Body Control H-bridge*
Bosch CJ840	: *BDE valve output stage module*
Bosch CJ125 + FET	: *02 Sensor controller (LSU type) + heater*
Bosch CY370	: *Fuel Injector final stage*

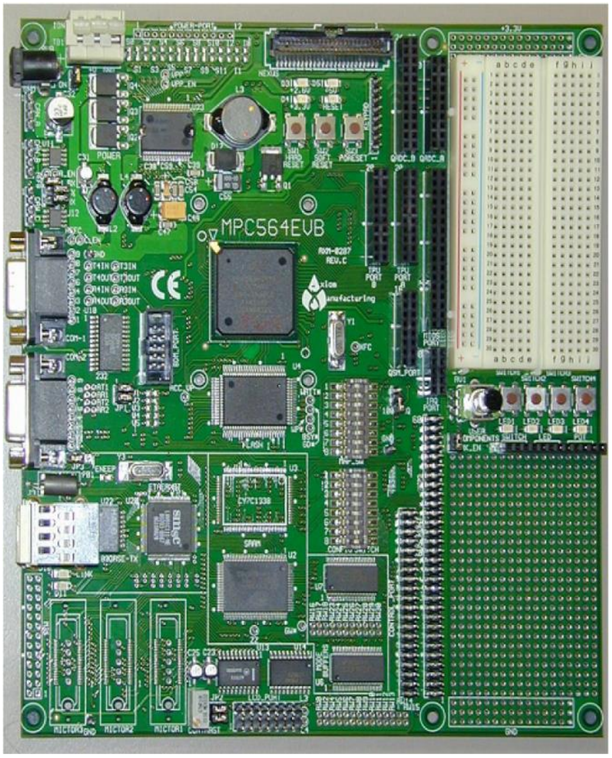
<u>To the left</u> *we can see the typical additional IC chip "Drivers" which are used in MED9 applications - beyond the microcontroller to manage all the sensors and actuators. Bosch developed or licensed these to work with their own proprietary sensors for full <u>horizontal integration</u>*

MED 9 - MPC5xx Series Hardware – Reference Design Kits

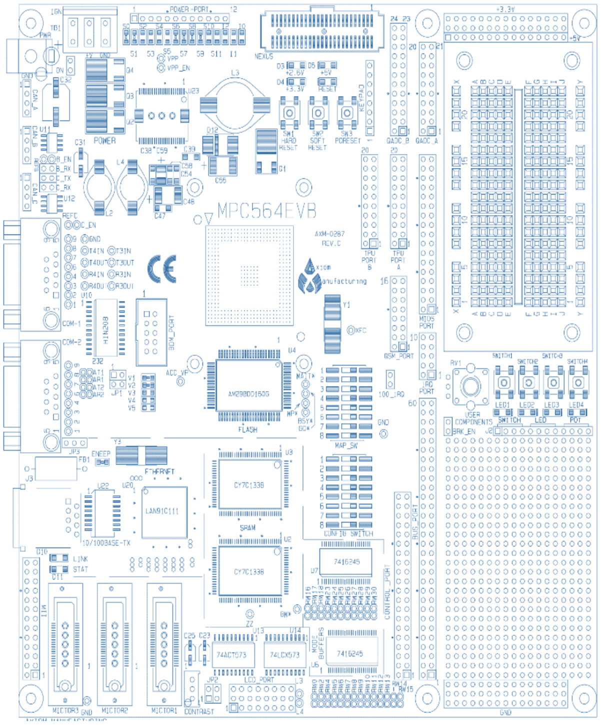
Just like for previous hardware designs, it is all focused around the main SoC (System On a Chip), the Freescale **MPC562/564**. You'll instantly note that there is a lot more pins in this chip than for older generations, it's **a PBGA388 (388 pins)** so significantly more expansion potential than previous chipsets. As before there is a XTAL (Crystal Oscillator) providing the stabilized frequency to operate the chip at **66MHz**, internally there is a multiplier which is configured in software.

The development board (as did the ECU used in Ferrari MED9's) came equipped with 2Mb External Flash (up from 512kbytes in a MED7.3 and 1Mb in 7.3.1) and additional 512Kbytes of external memory too – so pretty significant increases across the board in speed, memory and storage. It was a big step up from ME7 hardware specs.

As before there isn't a huge amount to say here than looking at the Datasheet to understand all of the chipset capabilities gives you a deep insight into how to begin designing an ECU which incorporates this chip (or variants of it). These days it wouldn't be advisable to use it in modern designs as the reference design dates all the way back to 2003 (!) some 20 years ago now!



Breadboard to allow fast rapid prototyping, allowing placement of extra components directly on the board



The Reference design is where it all starts from…

Faster = Better right?

If Bosch had decided not to switch back to its German supplier, Infineon (To use the Infineon Tri-core), and kept using the Motorola (Freescale) products what would they be using today?

Q. How much faster are equivalent ICE Engine SoC's in the past 15 years gotten?

These days they have upgraded the number of pins again to BGA516 (516 pins! Yikes) and the CPU cores are now **Dual Cores** running **@ 300MHz each** (combined throughput is around **x10 to x15** faster than the 66MHz of the MED9) with bigger CPU data and instruction caches so probably **x30 to x40** faster than a humble C167 @ 20MHz (ME7 ECU) under real operating conditions (*optimizing PowerPC compilers are much better too). The question remains, how much more speed is really needed to operate a combustion engine? Laws of diminishing returns?!?

Not so fast, as with everything done in software these days, optimization and efficiency is much more of a 'dying art' and you are lucky if engineers even profile their code to verify for bottlenecks. So this can dramatically eat into that performance advantage, particularly if larger more complex software is introduced and generic open source adopted (designed for flexibility, not efficiency). Performance is a bit like highway capacity, the more that's available the more that is used. Unfortunately that doesn't necessarily equate to better overall results, just less of the engineers job is focused on ruthless efficiency…

CCP is a protocol used for calibration and data acquisition from electronic control units (ECUs) which was used on MED9+.

It is defined by ASAM and is a top layer (layer 7) according to the OSI model. CCP does not describe how bits and bytes are created but uses the CAN 2.0B protocol physical, data link, and network layer.

MED 9 ECU Emulator Piggybacks

Modern chipsets have advanced tremendously in their capabilities and now include standardized **debugging interfaces** (JTAG is one of the most common) for instance rather than manufacturer proprietary standards like Motorola's BDM. They also have substantially more memory than earlier attempts so it is much easier now to copy calibration maps from flash into memory (where they can be edited in runtime, especially on later 2019+ platforms like MPC777C)

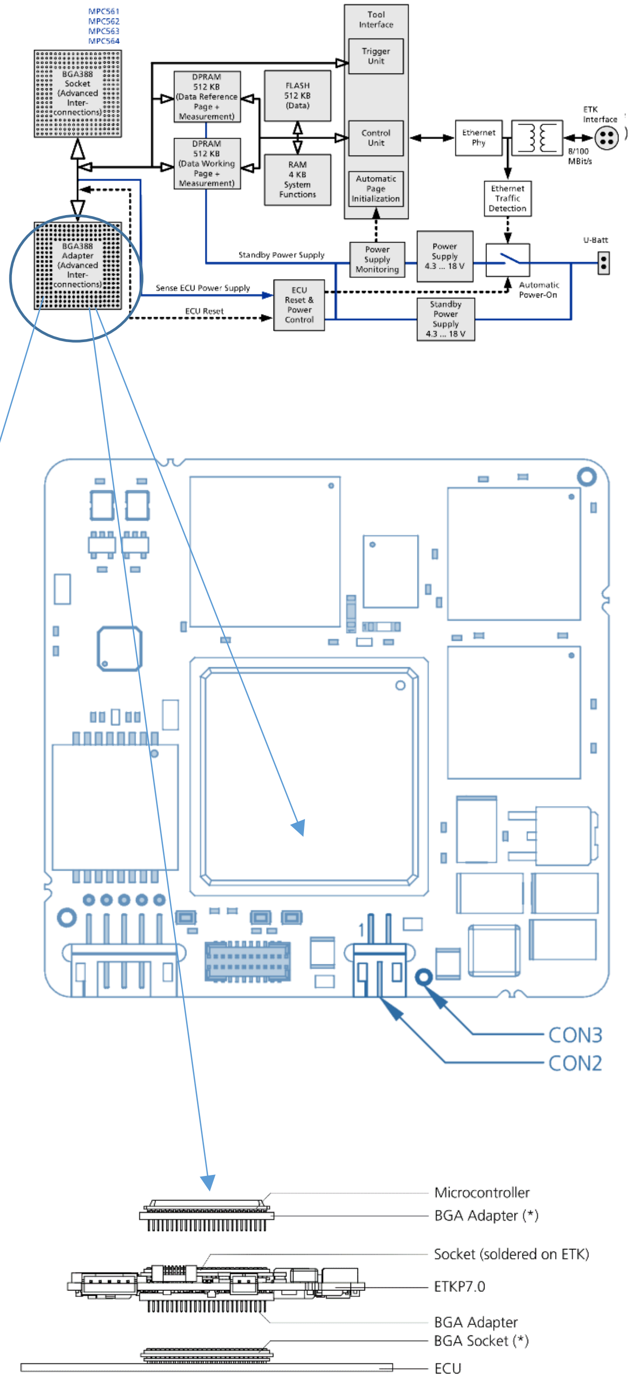
ETAS (a subsidiary of Bosch) used the direct **PBGA388** socket of the MPC5xx chipset and produced an ***emulator piggyback card***.

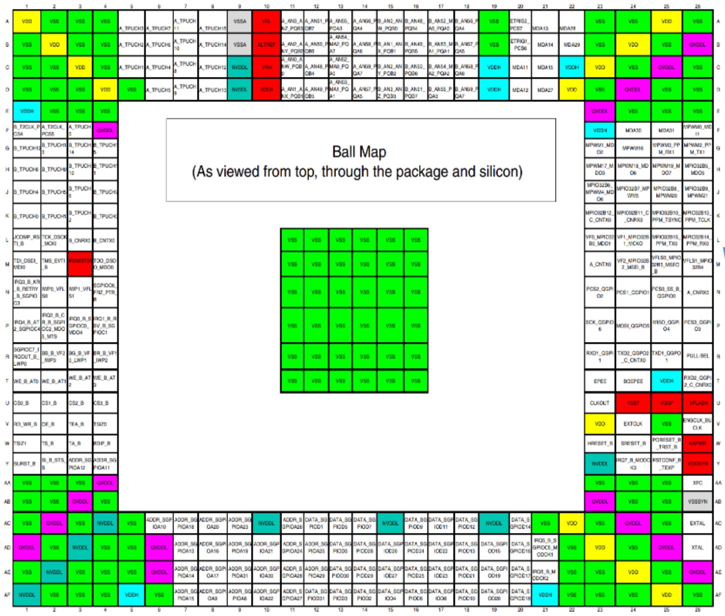
Q. *What is it and what does it do?*

A. It emulates the 'flash' and 'ram' of the original chipset and even can clone external ram and flash too. The purpose of which so you can do live calibration changes (real time) without re-flashing every time and without having to write any custom code.

While these capabilities are useful they are not 'essential' because you can always reserve some 'ram' for Live Tuning and add a capability into the software to copy a map one at a time into ram (if memory is limited, a few hundred bytes per map is the typical cost) then save the changes back later. On newer platforms like MED17.x this a well-understood way to do things are now. This is why emulators have become less and less used, they simply aren't really needed as much as before – as there is plenty of ram available to mirror the most often changed calibration maps.

" Flash/Ram Emulators really have become a bit pointless… "





“As insanely quick as *the* ***F12*** *and* ***F12 TDF*** *are, they run essentially a de-tuned* ***812 Competizione*** *engine but can the chassis handle yet* *<u>more power</u>* and does it really need it?”

Motronic Software Architecture

RTOS

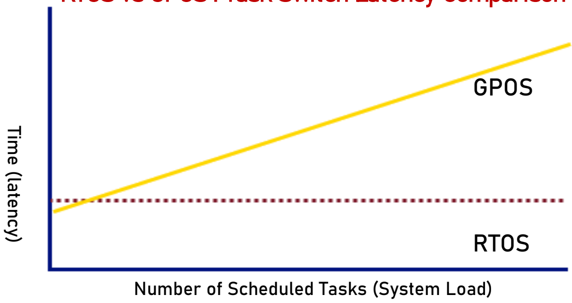
- Deterministic : No random execution patterns
- Predictable Response Times
- Time Bound
- Preemptive Kernel

GPOS

- Dynamic Memory Mapping
- Random Execution Patterns
- Response Times not guaranteed.

*This matters when you are dealing with events that need to be responded to within milliseconds and they cannot wait. Fuel injection and ignition timing fit precisely into these definition, wait too long and you miss the window and it is why it is important to use an RTOS.

RTOS vs GPOS : Task Switch Latency Comparison



Highly efficient, deterministic software is developed using an RTOS

FreeRTOS RT-Linux VxWorks eCos

Examples of various RTOS's available …

The designation M(E)(D) x.y.z stands for:

M = Motronic
E = Electronic throttle (Version 7+)
D = Direct injection (Version 9+)
x. = Version
y.z = Development Level

Here we will only cover the basics as the topic is vast and way beyond the scope of this guide to present here, usually described as part of software engineering degree or PhD masters course.

The system software was built upon a traditional *embedded solution* and utilized a proprietary and compact **RTOS** (Real-time Operating System).

> RTOS definition: The main responsibility of an operating system is to manage hardware resources and activities in the system: scheduling application programs, writing to storage, sending data across a network, and so on. When the OS must handle multiple events concurrently and ensure that the system responds to those events within predictable time limits, it is called a real-time operating system, or RTOS

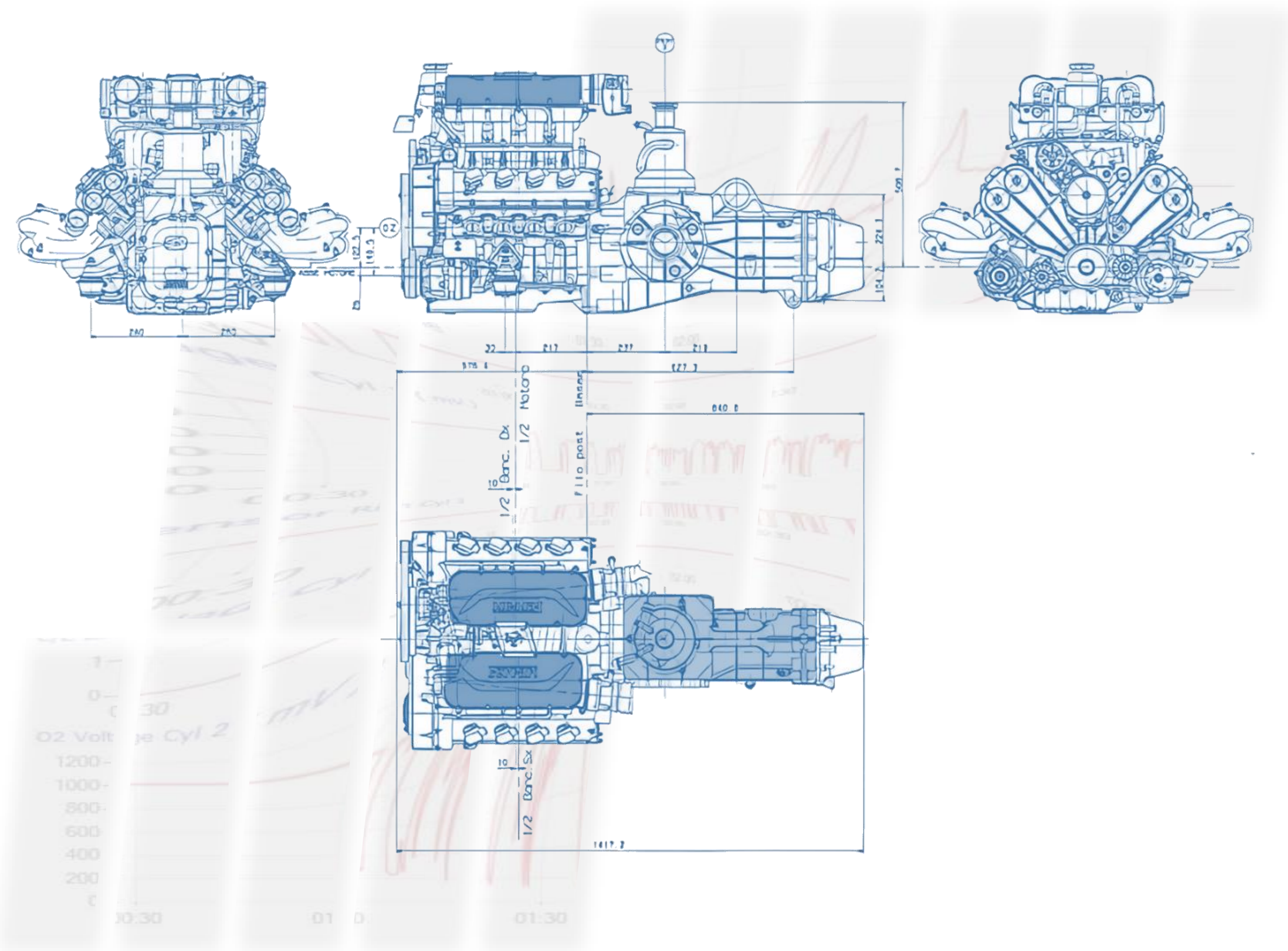
Bosch acquired a company called ETAS who provided an automotive certified RTOS (to ISO standard DSIL 262626) called **ERCOS**. This tiny little piece of software written mostly in C language and some assembly (4k bytes in size) implemented *a scheduler, messages, semaphores, interrupt management and more*. It allowed precisely timed control of all of the software functions required to be executed at very strictly controlled intervals, unlike a traditional General Purpose OS (GPOS) like Windows (see left information boxes to find out why this is important).

The software was architected to execute in a deterministic way which included utilizing different "raster's" operating at different speeds to execute all of the activities that needed to be completed. These operated as a background raster, crank raster, 1ms, 2ms, 5ms, 10ms, 50ms, 100ms, 200ms, 500ms and 1000ms. If you needed a longer time to wait for example you could use a counter and since you know exactly how long your raster takes you could derive custom time periods from that.

As is common on embedded systems a "**Watchdog**" timer is active at all times and if it doesn't respond in a timely fashion the hardware will force a reboot (to prevent a deadlock) resulting from a software bug causing it to crash indefinitely. Obviously the faster they operated the less they should be doing 'as not to overload the CPU'. So typically the 1ms activities for instance would be to read data from sensors, slower raster's would be used to do real-time analysis of sensor information for example to raise a CEL if a threshold was lost or react to other conditions, implement algorithms and so forth. Bosch uses runtime checking of both memory and flash to ensure various static and runtime '**checksums**' are performed. All of these are done to ensure that any hardware bit failures don't cause disastrous impacts (one badly corrupted byte could destroy an engine if you were unlucky!)

Each "task" had a small initialization function (called ini) which would execute at the start, also a function that would execute at a precise given interval (e.g. every 10ms), optional software off function, which would be executed on power off stages. This was used to save information into EEPROM for example before the ECU powers down. The software is split (as is common on embedded systems) into a boot loader and a main firmware. The boot loader passes over control to the firmware image once it is initialized the basic hardware, to allow ram to function for example. As a failsafe the boot loader on Bosch systems had a recovery mechanism so that it could be recovered over KPW2000 if firmware gets corrupted.

Q&A



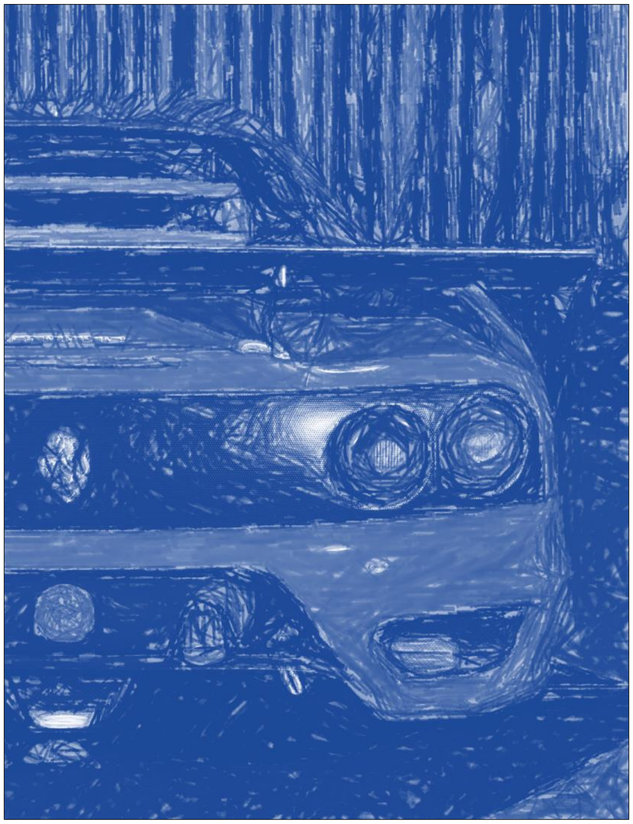
Q. Can the software in the Engine computers be upgraded?

A. Yes, the original Bosch ME system was never really designed nor intended to be upgraded 'in the field' and as such it had for instance no 'OTA' or over the air mechanism to self-upgrade as you would find and expect today in every smart phone and cheap tablet or say a modern EV like a Tesla. In fact there are no wireless or Bluetooth interfaces anywhere on the hardware at all, even today.

Having said that they can still be backed up with and re-programmed, verified and therefore re-cycled. The tools I developed can be used to remotely (over the internet) convert the ECU's while attached to the official affiliate support tablet. There are also other generic aftermarket tools available too but the majority of them are either stolen 'pirated' software which are clones of commercial tools or they are genuine tools which cost several thousand to acquire and have yearly support licenses to pay too.

Depending on the generation of hardware used generally Bosch has left in production units the mechanisms to upgrade the software too. These however are not intentional and really due to its origins of being a solution that typically borrowed its core components from the 'generic' microcontroller category. These where there for developer use during the software R&D phases and for testing at manufacture so not all ECU's even had the wires pinned out to the public wiring loom connectors. Regardless, it can be still be accomplished but for reliability reasons it is done on the bench connected to a clean DC-DC power source that's both regulated and appropriately fused.

"The Ferrari 360 has an astonishing **108+** *unique* firmware revisions…"



Some version of the software did support upgrading over either CAN bus or K-Line via either KWP2000 (Keyword 2000 Protocol) or on later generations CCP over CAN. Again it differs on support from generation to generation but as a general rule of thumb, earlier ME7 generation all used K-Line serial for the communication protocol and diagnostics. Some diagnostics information is however also broadcast over CAN messages such as the immobilizer lock status and even the state of the sports switch!

Q. Is it a good idea to get a 360 Transmission Computer upgrade from a Challenge Stradale for my 360 Modena or Spider?

A. It depends, it can make a considerable improvement to gearbox shift speeds on the AMT F1 system but to do this properly you <u>must</u> upgrade both the Transmission computer and the Engine Management Software too. If it is not done the calibration is out of spec (the engine and gearbox software need to share some parameters which must match to work correctly).

If you use CS TCU with Modena firmware for example you 'Normal' driving mode it will still give the impression of 'working' but it is far from perfect and you will experience increased slop and clutch wear vs stock TCU, it works better in Sport mode (called Race on the CS). To benefit properly from the improvements Challenge Stradale shifting strategy needs alignment with the engine software too.

If you looking to purchase a vehicle that has had a CS TCU fitted be on the look out to verify it the car has also had the Engine ECU's also upgraded as without it the car won't perform as well as it should.

Q. Can you use Engine ECU's previously fitted to another Ferrari of the same type and year or even recycled ones from a breakers yard?

A. This is more challenging than first meets the eye. The most obvious first obstacle is that Ferrari along with all other manufacturers where required by law (a regulatory mandate) to fit an immobilizer function in every car produced. This was paired to each car's engine computers (master ECU) and it is unique to the vehicle. It really depends on the model year and the generation of Motronic used about which methods are supported but the simple use case of 'just using' a used ECU is generally not going to work. It can still be done but not without work to replace the software with the exact same version that the car came with and have all its immobilizer pairing reset back to virgin or cloned across from the original ECU (if available). Later generations of Bosch ME got ever more secure including password locked and signed flash and even to the point that they used 'write once' memory (which would actually use a kind of pyrotechnic fuse to set the bits to a 0 or 1 forever burned in). This was in an attempt to lock the ECU to the vehicle forever and for example was used to set the Serial number to be unique (done at the factory).

Of course many of these limitations can be overcome with software and specialist tools so while it is not 'easy' it is not impossible to re-use and re-cycle breaker yard parts.

Many people don't appreciate is that low volume manufacturers like Ferrari continue to do things very differently (due to production number differences) and therefore often self-imposed strict budget restrictions with regards to development and calibration of ECU's. Therefore issues discovered have to continuously be revised during the lifecycle of the vehicle, typically done to fix in the field issues discovered after production started or after feedback from customers. Large volume producers like VAG group or Mercedes can afford to spend a lot more on testing when production volumes are orders of magnitude more so they tend to be impacted much less by issues requiring software updates. As a result most often the Ferrari firmware versions are *not fully compatible with each* other and they were <u>never</u> intended to be. This means it is critically important to ***never use firmware designed for a different assembly number*** group or from another car, it might not work exactly like you would expect, even if the engine runs there can be issues waiting to bite unsuspecting owners and mechanics alike.

As such you should *never install* the wrong software version in a different car without specialist knowledge which is way beyond the scope of your average tuner or ECU repair specialist. End result is it can be dangerous, even if the car appears to run OK many issues can be present which won't even show up on a diagnostic tool, these can make the car unsafe or fail emissions or cause long term damage to the engine.

Diagnostic and Re-flashing over the internet using 360trev developed Software and failsafe methods developed over many years …

Q. Why does my OBD scan tool only find one ECU?

A. This is usually a software bug/issue in the 'generic' scan tool software and how much of the standard the developers implemented correctly. On earlier cars like 360, F430, 599, 612 and so forth the diagnostics are all done over K-Line (serial) which is a single wire communications interface and uses KWP2000 (Keyword Protocol) to interrogate various types of controller. It can also be due to immobilizer issues so ensure it is correctly deactivated before scanning.

If Immobilizer is not the issue, missing an engine computer is probably because most cars don't have a Master/Slave arrangement so limited testing was done by the manufacturer of their tools on cars with multiple ECU's of the same group, so as their software is performing 'discovery' they likely assume there should be only 1, so it could be seen really as sloppy coding. Why? Well, if the software in the OBD tool confirms to the standard it should be able to automatically detect the controllers and number of them based on their Identification codes as there is a published standard for this.

Powertrain Controllers	*Hex*
Integration/Manufacturer Expansion	00-0F
Engine Controllers	10-17
Transmission Controllers	18-1F
Chassis Controllers	***Hex***
Integration/Manufacturer Expansion	20-27
Brake Controllers	28-2F
Steering Controllers	30-37
Suspension Controllers	38-3F
Ferrari Specific Engine Controllers	***Hex***
Master Engine Controller:	11
Slave Engine Controller:	14

Body Controllers	*Hex*
Integration/Manufacturer Expansion	40-57
Restraints	58-5F
Driver Information/Displays	60-6F
Lightning	70-7F
Entertainment/Audio	80-8F
Personal Communication	90-97
Climate Control (HVAC)	98-9F
Convenience (Doors, Seats, Windows, etc.)	A0-BF
Security	C0-C7
Future Expansion:	C8-CF
Manufacturer Specific	D0-EF
Off-Board Tester/Diagnostic Tools:	F0-FD
All Nodes	FE
Null Nodes	FF

In fact you can see in the above charts from the standard that not only can it detect the number of engine controllers (basically any controller which responds with a hex identifier of 10 through to 1. Theoretically the standard assumes up to 8 engine computers could exist on the bus in a single car, it is debatable whether sufficient testing (even with simulated computers) is ever done by the cheap OBD clone makers to see that they fully comply.

"The dealer level **Systema Diagnosi SD2 / SD3 and SDX tools** use proprietary extensions to allow them to do things that not all generic ones are capable of doing…"

SDX software and VCI controller is vastly more capable than your typical generic scan tool and used by dealers and specialists alike

Q. Can a gated firmware be simply "cross flashed" onto a car that originally came with F1, what about if it's the same Model year?

It is not recommended as it is risky without expert advice due to both hardware (sensors, actuators and wiring differences) and software compatibility issues across assembly number ranges, not just model years.

The Ferrari 360 for example has an astonishing 108+ different official factory firmware revisions of the software spread over its production cycle which are often not fully backward compatible. Just think about that for a moment… that works out as a different software version every couple of hundred cars produced, or put it this way, every couple of weeks of production cycle.

There are equally many cases of F430 owners having had some less knowledgeable (but still well known) companies attempt to 'cross flash' with disastrous consequences. Basically a non-start! It gets even worse when the said companies didn't even backup the original software and therefore leave you without a working car. This has happened more times than you would believe and we have even personally helped fix some Ferrarichat 'non start' cars which have had this very thing happen to them.

> "Next generation diagnostics requires new thinking… We will enhance ECU software to enable support for many more features than factory tools…"

Prototype of new OBD dongle and tablet/phone apps which can support 'beyond' factory level diagnostics and real-time telemetry for Ferrari enthusiasts. Watch this space!

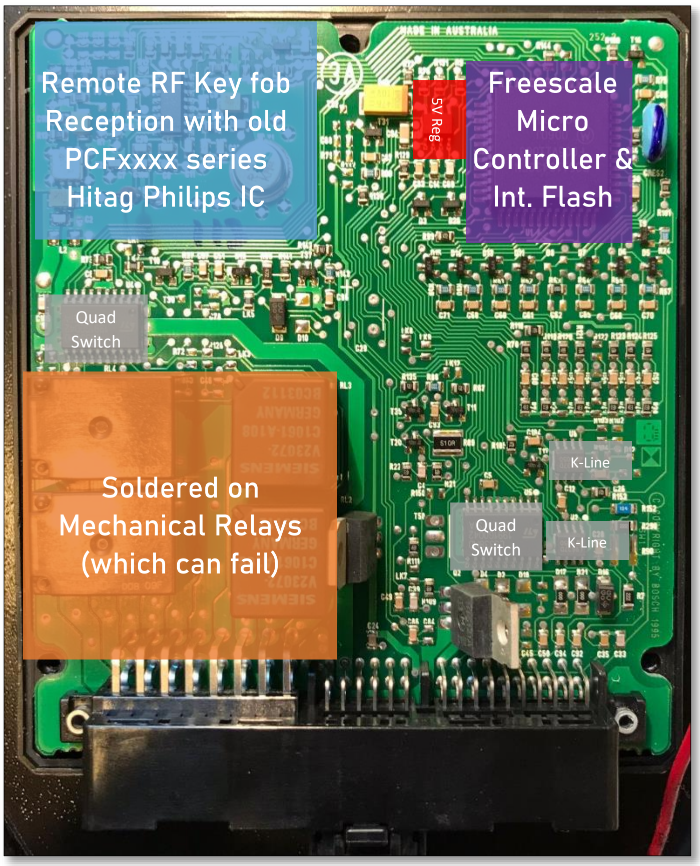
Q. Can the Vehicle Immobilizer be deleted on a 360 / F430 / 575M, 612, etc.?

A. After investing a lot of time to analyze all the different configurations a solution has been found to entirely remove the immobilizer for cars with a faulty system but it varies depending on the model and also whether it is a good solution or not (which depends on your own personal circumstances). For instance on the earlier cars like 360 and 575M the immobilizer also contains the wireless fob interface so if the immobilizer is removed you will lose wireless (remote) central locking function. The key in the door will still work as central locking but not over the air if the unit is completely removed from the vehicle.

It is also worth bearing in mind if such a bypass is done its vital to inform your insurance company and also recommended to fit additional immobilization circuitry such as an aftermarket 2 or 3 stage immobilizer with a keycard or fob.

Later cars like F430 got a newer style of VIM (Vehicle Immobilizer) which didn't require mechanical relays and was standalone from key fob reception. This can be bypassed in software without fear of losing remote central locking which was a combined design.

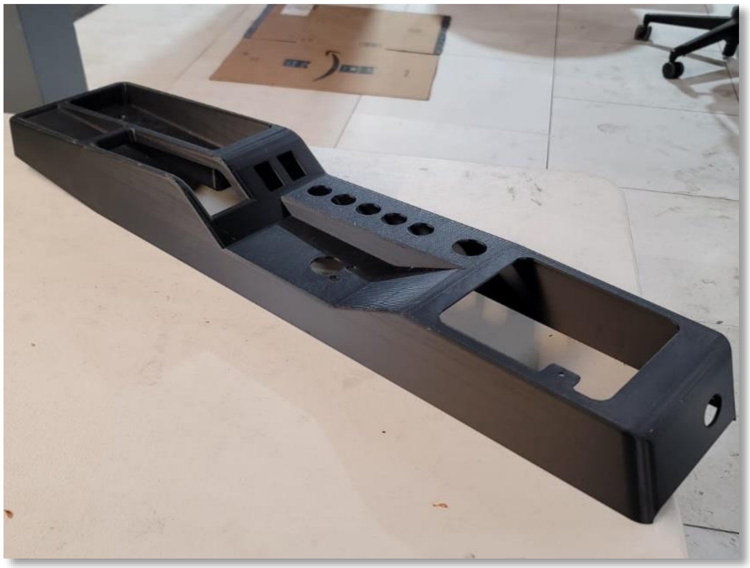
Also design is smaller and has much lower power consumption for improved battery drain vs older IC's used in earlier design from 1990's.



Ancient circa 1995 Bosch design for a combined immobilizer & fob reception with soldered on 12V Mechanical Relays which control Door Locks, Starter & Flash Indicators. Used also on even older cars like 456, 550, 575M, 360, etc. Can be repaired, relays can be replaced, etc. K-Line's are used for communication

Old Microcontroller is quite power hungry by today's standards so contributes significantly to battery drain if left for long periods of time.

"The Ferrari 612 was **de-tuned** *from the factory to be the* GT cruiser, *however with a* little brain update *it becomes much more focused on driver enjoyment and it's a* real gem…*"*



Rapid prototyped central tunnel replacement unit was 3D printed to quickly evaluate different designs before being productionized.

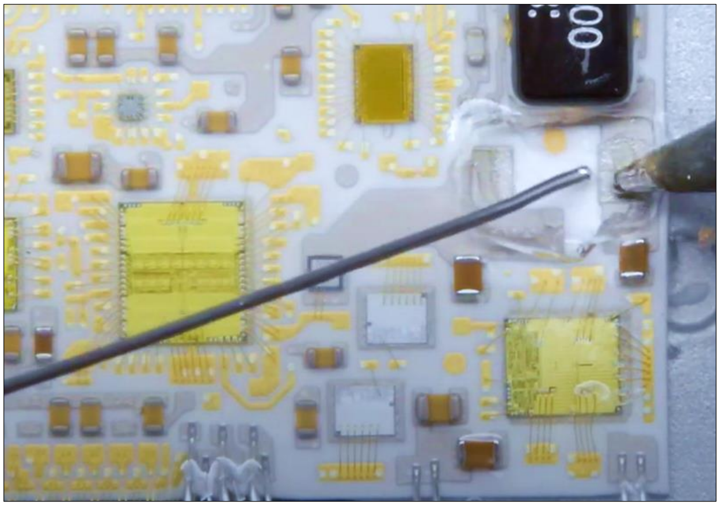
Challenge Stradale with a re-manufactured 'OEM Inspired' gated central tunnel for that factory look in Carbon exactly like the original F1 version.

FlashMagic software and tools by Senica Consulting (AKA brainchild of 360trev) to allow software conversion over the internet for their affiliates

The Interior of a rare 430 Scuderia converted famously exclusively by EAG for one of its lucky clients. 1 of only 5 known to exist in the world.

Q. What causes the dreaded 'Ticking Throttle Body' problem on the 360 ECU's and can it be fixed?

A. Ferrari 360's have seen a spate of **ECU failures** in recent years and this has been even more common on the 360 Spider than the coupe. The root cause for the failures has been identified as the solder of the orange tantalum capacitors on the die substrate board, it exceeds a temperature threshold and melts. When this happens the only thing holding the capacitor in place is the clear goopy gel leading to intermittent (temperature dependent) engine start issues.



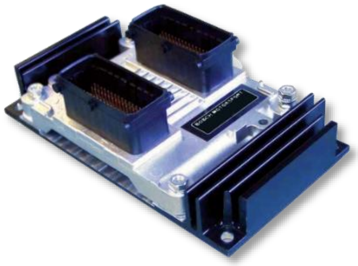
The failure condition is that you get a 'crank no start'. The ASR/ABS light illuminates and the car will not run. This is because the capacitor in question is used for the power circuitry to drive a Bosch (H-Bridge) chip with enough current to drive the up to **5 AMP** DV-E Electronic Throttle Bodies. It requires more power to open the flap up at high speeds than at idle due to air resistance. During the series of self-tests (which activate upon key-on) the ECU's wake up procedure is initiated. The drives the throttle body (with the engine off) by moving the Throttle blade from 0% to 100% to allow it to re-calibrate itself every time you key on. If the circuit cannot generate enough power to drive the throttle you get a ticking throttle (it keeps retrying and failing indefinitely).

REPAIR TIP

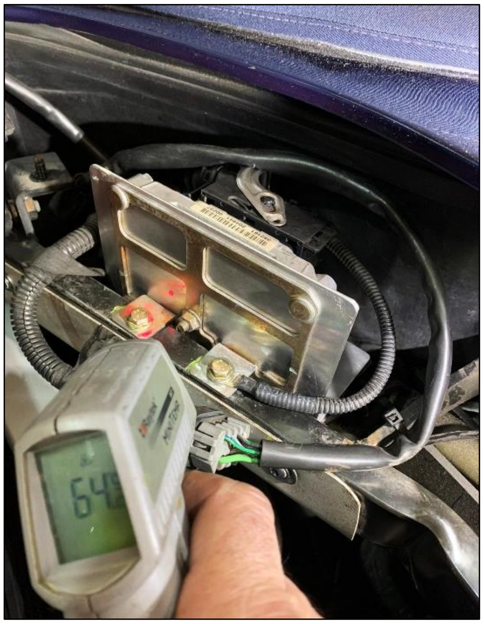
The repair to the right isn't pretty but in this example (on a similar but different skew of a die-substrate Bosch ECU) they decided to relocate the tantalum SMD capacitor because of the difficulty to solder directly on the pads. The better solution is to heat up the ECU case and this makes act less like a 'heat sink' and makes soldering much easier.

While it is possible to scrape away some goop and re-solder the component it could still fail again. The best solution is preventative medicine. I.e. prevent it happening in the first instance. Just like a CPU on a PC uses a Heatsink and you can buy improved heat sinks for different clocked CPUs (under extreme workloads for example) Motorsport Bosch knew about these shortcomings and fitted a Motorsport heat sync to their race applications. Some gains have been had with simple home-made shields with reflective exhaust wrap to try to reduce heat soak from the heat exchanger, another more extreme solution is to air duct cool air in and use and a professional Motorsport Grade heatsink.

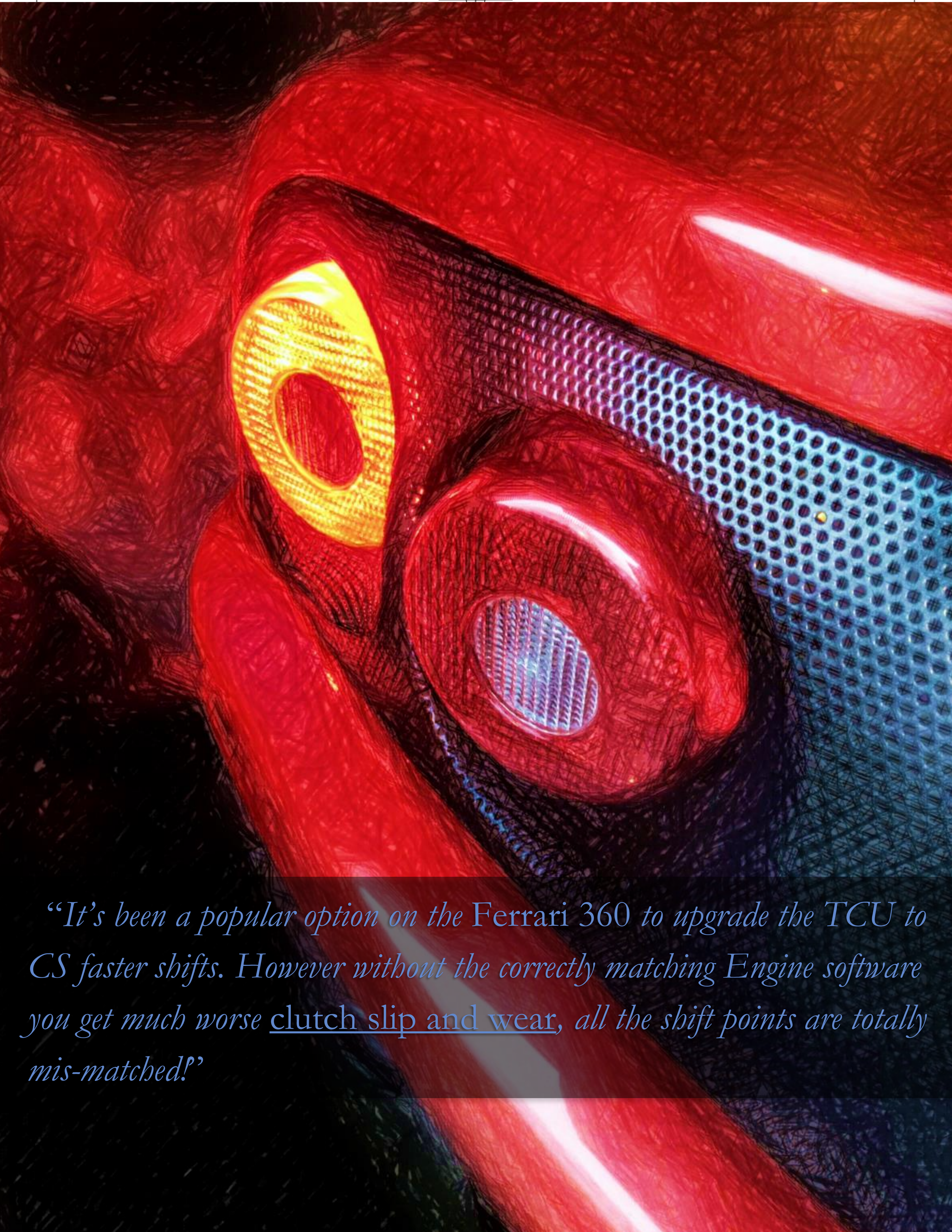
Improved cooling with a beefed up Heatsink dissipating the heat should help reduce heat soak issues considerably. In testing so far this has reduced the temperatures enough to prevent the ECU failure occurring in the first place, i.e. going over the tipping point. This is now particularly of importance because the Ferrari factory has ran out of replacement ECUs (due to the problem being so widespread – it's a *design flaw*) and as such it is far better to prevent the failure and deal with it later which is then much more costly. As the Throttle Bodies age they begin to consume more power which only adds to the heat problem



*Bosch Motorsport knew about the problems of Heat Soak and used a substantially beefed up **CNC'd Billet Heatsink** for their Racing Application customers which used a Die Substrate ECU similar to fitted to a Ferrari 360. As such we are exploring options here for a solution for the 360 Market*



Eddie Blasi, Lead Master Ferrari Technician of AV Engineering UK, here seen testing the effectiveness of his experimental design for a heat shield.



"It's been a popular option on the Ferrari 360 *to upgrade the TCU to CS faster shifts. However without the correctly matching Engine software you get much worse* clutch slip and wear*, all the shift points are totally mis-matched!"*

Q. Are you doing anything in diagnostics space?

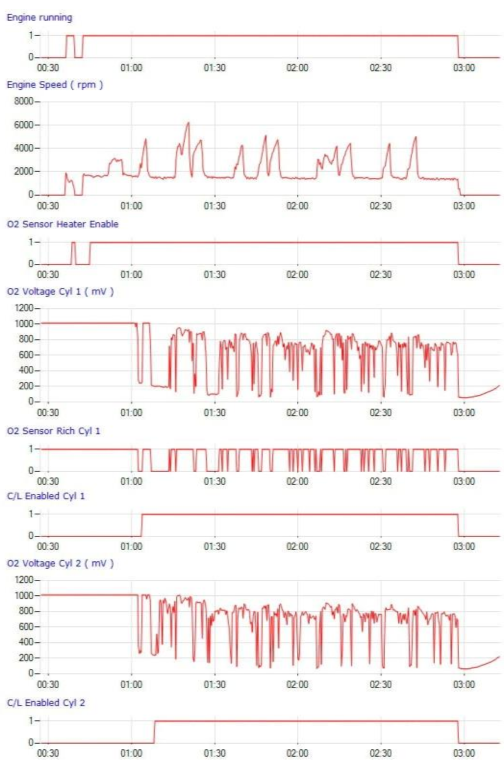
A. Not today, however as part of our commitment to help the market with specialist tools to support Ferrari's (not just engine ECU's but others too) we have been working painstakingly on getting useful kits produced which help to ensure (initially) ME7 and MED 9 ECU's (circa 1999-2014) can be supported and maintained into the longer term future. These already support the ability to clone, upgrade, repair, backup, restore and clean back to virgin a Ferrari ECU's of the period. Great for re-cycling!

More will be announced when the time is right on this but suffice to say we want to ensure there is a long term support for these cars beyond the point that the factory all but 'given up' on the models.

There has been a lot of work underway recently on extending much more advanced data logging (see panels) and diagnostic capabilities with improved tests which we are extending in the ECU firmware. This will allow technicians and enthusiasts the ability to much more what's going on and get a much better understanding of any faults that arise as well as to make improvements. There are thousands of parameters that even the factory diagnostic tools have no access to and those which are most valuable will be exposed.

We believe our expertise and advanced software technologies are 'unrivaled' in the Ferrari space and totally unique so it will be exciting to reveal more as and when it becomes available.

```
nmot_w        ; engine speed
tmot          ; engine temperature
tans          ; intake air temperature
ml_w          ; air mass flow (filtered)
zwout         ; ignition angle
ti_w          ; injection time
rl_w          ; load (relative air charge)
wdkba_w       ; throttle angle
mshfm_w       ; air-mass air-flow HFM
ub            ; battery voltage
vfzg_w        ; vehicle speed
usvkk_w       ; primary O2 Lambda probe LSU voltage (corrected)
ushk_w        ; primary O2 Lambda probe voltage
nsol          ; idle setpoint speed
tateout       ; duty cycle purge control valve
tvu_w         ; battery voltage
upwg1_w       ; throttle pedal voltage PWG potentiometer 1
upwg2_w       ; throttle pedal voltage PWG potentiometer 2
wped_w        ; normalized angle acceleration pedal
fr_w          ; lambda controller output
frao_w        ; multipl. mixture adaptation factor higher load
frau_w        ; multipl. mixture adaptation factor of the lower mult. section
fra_w         ; multiplicative mixture adaptation factor
rkat_w        ; additive correction (per time) of the mixture adaptation
rkaz_w        ; additive correction (per ignition) of the mixture adaptation
rka_w         ; additive adaptive correction of the relative fuel amount
lrnstep_c     ; learning step counter for throttle self-learning
gangi         ; gear indicator
dmvad_w       ; delta motor torque from loss torque adaptation
mdverl_w      ; resistant torque of the engine
fho_w         ; correction factor: altitude
pu_w          ; ambient pressure
lamsoni_w     ; lambda actual value
lamsons_w     ; required lambda referred to lambda sensor fitting location
lamsonh_w     ; pseudo lambda actual value measured w/nernst probe behind cat
lamsolh_w     ; pseudo lambda setpoint behind cat
wnkwas_w      ; Angle of camshaft to crankshaft in the working cycle
wnwue_w       ; camshaft overlap angle of inlet and outlet valve opening
fnwue         ; weighting factor of camshaft overlap
rlshk         ; (load) desired relative air charge altitude corrected
msdk_w        ; air-mass flow through throttle valve
dlahi_w       ; I-portion of the LRSHK
miist_w       ; indexed engine torque high pressure phase actual value
mifa_w        ; indexed engine torque driver request
mrfa_w        ; relative driver request torque from FGR and pedal
miopt_w       ; optimal indexed moment
mizsol_w      ; indexed resulting target torque for ZW intervention
mimax_w       ; maximum achievable indexed moment
miasrs_w      ; indexed engine target torque ASR (for rapid intervention)
miasrl_w      ; indexed engine target torque ASR (for slow intervention)
mimsr_w       ; indexed engine target torque MSR
nskup_w       ; target speed F1 gearbox (CAN signal)
dmar_w        ; delta torque anti-jerk
dmllri_w      ; desired torque change from the idle speed control (I-)
dmllr_w       ; desired torque change from the idle speed control (PD-part)
evtmod        ; modelled temperature at inlet valve
ftead_w       ; charcoal canister charge
msndko_w      ; norm leakage air mass flow through throttle blade
redist        ; actual cylinder cut off step
lamsoni2_w    ; actual lambda value (upstream cat) bank 2
```



To assist workshops and technical owners kits have been produced with new features being added all the time, specifically for Ferrari's. Watch this space!

Q. What is the concern with "cross-flashing" ECU's?

Ferrari 360 ECUs are challenging to repair due to Die Substrate manufacture…

The amount of software in a car is growing year on year and runs into tens of millions of lines of code. Software compatibility is a vastly complex subject and not simply limited to the Engine management control alone, it extends its reach into many other areas and interweaves its software versions against other software such as those found in the Instrument Cluster, E-Diff (F430, 430 Scuderia, 599, etc.), ABS, Transmission TCU, Engine ECU, Body Computer (if fitted), Alarm system and even the Immobilizer.

"The risks of 'dumb' cross-flashing cannot be under-estimated…"

For example if you flash the wrong software it may appear to work but you may not get the correct values sent to for instance the instrument cluster or the engine will start and then stop at random intervals. An example is fuel level sender messages are interpreted differently so they will not show correctly and other issues such as in an emergency situation the ABS may fail to operate because they are not talking the same versions of the ASR compatible messaging!

Do you really want to take this liability risk on just chancing a 'cross-flash'? Your typical non-Ferrari 'tuner' or refurbishment company may be able to re-flash computers using some aftermarket tools designed to support the mainstream aftermarket but they mostly have absolutely no clue about any of this and cannot guarantee they will not mess up the car. If they screw it up they will not be able to help you and you'll be in a much worse situation trying to recover if an ECU's data has got corrupted.

Due to the unique way Ferrari built its cars of this era, in many aspects there are quite subtle differences across small batches, everything from wiring loom changes, differences in sensors and actuators and how they operate, not to mention emissions requirements which differ from year to year too. If for instance a different sensor is used on say the brake pedal and its misconfigured for your exact assembly number you can get into all kinds of potentially dangerous situation despite the car appearing to drive 'fine' but it's not - it can be unsafe.

After investing several years in developing custom software to 'forensically' analyze all the different software versions with highly sophisticated algorithms and techniques a huge pool of knowledge has been gained which has been used to ensure the work we do is perfect and works in all model years and with all cars. We use the original firmware to 'derive' its original sensors, actuations, wiring and more, from this we can 'generate' bespoke firmware that's guaranteed to work reliably and safely.

After using our bespoke software to analyze all Bosch Ferrari firmware versions including 'aftermarket' tunes it became clear that the vast majority of people (even those who claim to be experts in the field) have very little deep understanding of how to make changes to OEM levels of quality. For instance we have encountered tuners who have deactivated all detonation sensing (!), also known as 'knock control' which is a safety critical system designed and calibrated to ensure you engine survives under various conditions and with external environmental conditions such as old or low octane fuel. This is borderline criminal. Same thing for fault codes, we have seen people 'carpet bomb' them to be OFF because they don't know what each of them does so instead of investing the time to work it out they just turn them all off. This is clearly a disaster waiting to happen. It is no wonder the tuning industry gets such as bad name, in many cases it is warranted.

Bosch used hvbrid die substrate on the Ferrari 360 ECUs'…

SENICA CONSULTING - CONTACT DETAILS

Senica Consulting Ltd.
2 Cruchetts Lane, Gibraltar

Tel. +350 (0) 5400 1431
Email: trevor@senicaconsulting.com
www.senicaconsulting.com