

Ultra Low Emission Vehicle のための 高精度空燃比制御 ECU

High precision A/F Control ECU for Ultra Low Emission Vehicle

川 合 誠* 橋 本 英 樹** 赤 崎 修 介*
Makoto KAWAI Hideki HASHIMOTO Shusuke AKAZAKI
西 村 要 一* 大 保 慎 一* 上 田 伸 一*
Youichi NISHIMURA Shinichi DAIBO Shinichi UEDA

要 旨

ULEV（超低排出ガス車）の排気エミッション低減技術のなかで、現代制御を応用した高精度空燃比電子制御が特に重要な役割を果たしている。気筒間の空燃比ばらつきの補償にはオブザーバ理論が、過渡状態と外乱にはセルフチューニングレギュレータが適用され、制御性能の向上が図られた。CPU には高速演算能力が必要となり、演算速度が早くチップサイズの小さい車載用RISC型マイコンを開発し、ULEV制御ECUに適用し、C言語によりソフト開発を行った。これにより、高精度空燃比制御を実現し、コールドスタート時の排気ガス低減手法や、触媒活性化時間の短縮化とあわせ、初めのガソリン車によるULEVを実現した。

ABSTRACT

A high precision A/F control technology applied modern control theory is essential to decreasing emission for Honda ULEV. Usual Automotive CISC micro computers are not enough for high speed calculation in order to realize high precision A/F control for this purpose.

This paper described outline of the first gasoline engine ULEV system, especially the high precision A/F control by using newly developed 32bit 1chip RISC micro computer and C-programing.

1. まえがき

近年、自動車には社会・環境との調和が強く求められるようになり、低排出ガス・省燃費・安全性の向上についての要求が一層高まってきている。なかでも排出ガス規制はマスキー法以来次第に厳しさを加え、最近ではTable 1およびFig.1に示すように、CARB（California Air Research Board）の低排出ガス車計画に代表される達成困難といわれた程厳しい規制の超低排出ガス車（ULEV：Ultra Low Emission Vehicle）まで、要求されるようになってきた。

一方、機関の電子制御は排出ガス規制をクリアする過

程で発達をとげてきた。1975年に発表された空気流量と酸素濃度の測定による空燃比フィードバック制御技術は三元触媒の浄化効率をあげるために理論空燃比付近の極めて狭い領域内に電子制御システムを使用して、混合比を制御することによって画期的な効果をあげ、以後の機関制御システムの基本となった。その後、今日に至るまで、電子制御システムはマイクロエレクトロニクスの進歩を背景に著しく小型集積化され、機関の制御システムの高機能・高性能化に貢献してきた。

*和光研究所 **栃木研究所

Table 1 California low-emission vehicle standards

	NMOG	CO	NOx (g/mile)
Tier1	0.25 (NMHC)	3.4	0.4
TLEVs	0.125	3.4	0.4
LEVs	0.075	3.4	0.2
ULEVs	0.040 (0.055)	1.7 (2.1)	0.2 (0.3)

NMOG : Non-Methane Organic Gases
100,000 - mile standards in parentheses

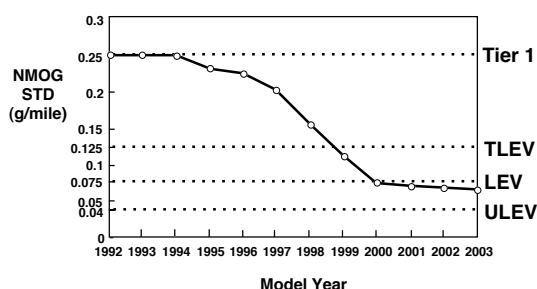


Fig.1 California NMOG fleet average standard

電子制御システムの特徴の1つとして、システムのハードウェアを大きく変更することなく、プログラムにより高度な制御を実現できることがあげられる。ULEV早期実現の課題に取り組むにあたって、機関の制御ハードウェアを大きく変更することなく量産可能な電子制御システムをめざした。その結果、VTEC (Variable Valve Timing & Lift Electronic Control System)によるコールドスタート時の排出ガス低減手法¹⁾や触媒活性時間の短縮化と並び、電子制御による新たな解決手法がULEV達成に、重要な役割をはたした。

従来の古典制御理論にかわり、気筒間の空燃比のばらつきを補償するためにオブザーバ理論を、また、過渡状態と外乱に対して、セルフチューニング・レギュレータ理論を適用することにより²⁾、高精度空燃比制御を実現した。

こうしたマトリックス演算を高速に行って空燃比を高精度に制御するためには、電子制御システムの頭脳にあたる電子制御ユニット (ECU: Electronic Control Unit) 内で使用するマイクロコンピュータ (CPU: Central Processing Unit) を変更する必要が生じた。すなわち、従来の16bit CISC CPU (Complex Instruction Set Computer)

では演算速度が著しく不足していたため、新たにワンチップ32bit RISC CPU (Reduce Instruction Set Computer)の開発を行った。

本稿ではCARBにおいて排出ガス試験を受け、ガソリン車でULEV規制を達成したプロトタイプULEVアコードについての技術要素概要ならびに高精度空燃比制御と、それを実現したECUについて述べる。

2. ULEV達成技術の概要

Fig.2に排出ガスの評価に使用されているLA-4試験サイクルとテールパイプより排出される炭化水素 (HC: Hydrocarbon) および各HC規制値の関係を示す。このように最初の505秒間 (CT-mode) のなかでも更に、エンジンを始動してから数分間にトータル排出ガスの大部分を排出しており、この数分間が極めて重要な期間となっている。更に、ULEVを実現するためには、この数分間だけではなく、その後のエンジン暖機後のサイクルにおいても排気エミッションを低減させることが必要である。

図中のそれぞれの棒グラフは、車輛の試験結果を示したものであり、厳しくなる規制値に対して量産の個体差を保証するためのマージンはどんどん縮小せざるを得ない。このため、製造ばらつき、エンジンの急な負荷変動や、経年劣化といった外乱を、できるだけ吸収できる安定した排気エミッション低減システムを開発することが重要である。ULEVを達成するという事は、目標規制値をクリアすることだけではなく、排気コントロールシステムの正確な制御性能を、常に維持することが大切な要素となってくる。

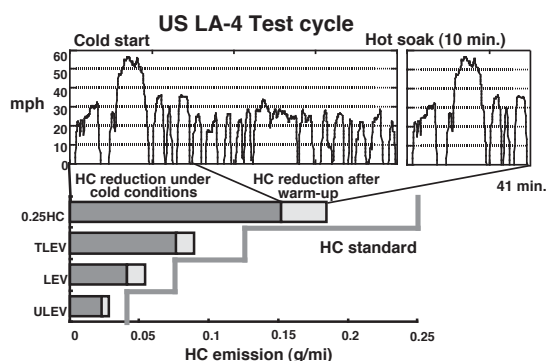


Fig.2 General concepts to achieve ULEV

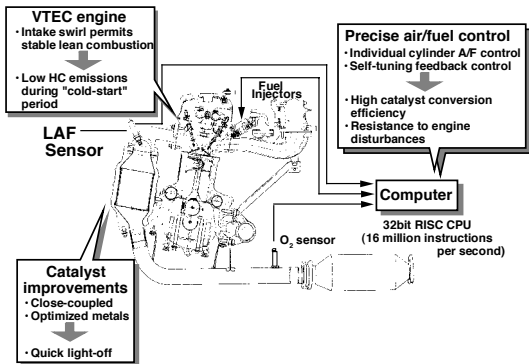


Fig.3 Honda ULEV technology

Fig.3は、プロトタイプULEVアコードで採用された技術の概要を示す。これは、エンジンハードウェアと触媒技術、コンピュータ制御技術の三つに大別できる。それらについて概説する。

エンジンはVTECにより、二つの吸入バルブの片側のリフトを抑制する機構を備え、適度なスワール比とタンブル比を与えられる。その結果、低温始動時およびエンジン暖機期間中の空燃比を理論空燃比近傍まで希薄化する事が可能となり、始動時および始動後に排出される未燃炭化水素(HC)排出量の大幅な低減を図ることができた。

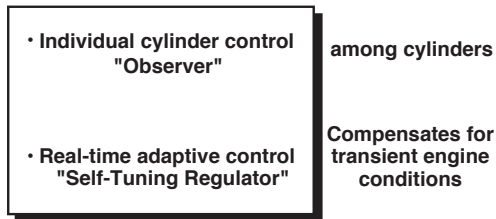


Fig.4 Technique for high precision A/F control

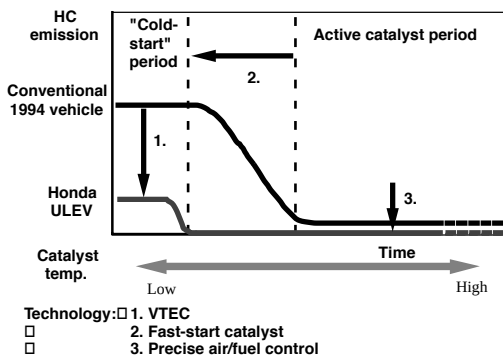


Fig.5 Three technologies work together to achieve ULEV

また触媒技術の分野では、直下触媒を追加しており、貴金属の担持量と担持割合、担体の構造を改良し、触媒の早期活性を図った。

高精度空燃比制御の目標とする性能は、空燃比の最適な制御による触媒浄化効率の向上と制御系のエンジン外乱に対する高い安定性であり、Fig.4に示すような次の2つの手法を基本にシステムを構築した。

- (1) 気筒別空燃比制御
- (2) セルフチューニングフィードバック

このシステムを実現するためには、リアルタイム演算機能が必要であり、複雑な計算と高速処理を可能にするためRISC型CPUを採用した。

Fig.5にそれぞれの区間におけるHCの低減効果を模式的に示す。二つの曲線は始動から始まる運転サイクルのHC排出量を示し、それぞれの矢印は(1)低温始動の期間でのVTECエンジンの効果 (2)触媒活性化時間短縮による、排出ガス低減効果 (3)精密な空燃比制御による最適触媒効率の維持による効果に対応する。

3. 空燃比の高精度制御

Fig.6に、一般的な三元触媒の空燃比に対する浄化特性を示す。排気三成分を同時に効率よく浄化可能なのは、理論空燃比付近のごく狭い領域だけに限定され、この領域から少しでも外れるといずれかの排気エミッションは急増する。触媒が活性化する領域内に空燃比を維持することは、非常に効果的である。空燃比をこの狭い領域内に制御するために、前述した (1)気筒別空燃比制御 (2)セルフチューニングフィードバックを用いた。

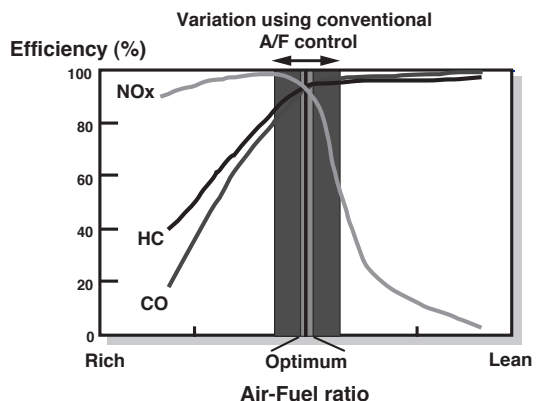


Fig.6 Catalyst efficiency vs. Air-Fuel Ratio

多気筒エンジンの場合、各気筒間に空燃比のばらつきが生じると排気管集合部または、触媒前の空燃比が変動し安定させることが難しい。そこで、気筒間の空燃比のばらつきを補償するためにオブザーバ理論を用いた気筒別空燃比推定を用いた。さらに、過渡状態とエンジンに対する外乱を抑制するために適応制御理論を適用した。この二つの制御技術は、現代制御理論を代表とする各制御理論を用いた数学的手法で求められ、排気マニホールドに取り付けられたリニア空燃比センサからの信号に基づいて高精度に燃料噴射量を制御している。

3. 1. 気筒別空燃比制御（オブザーバ理論）

オブザーバ理論を用いた制御のコンセプトは、各々の気筒がすべて目標の空燃比になるように、噴射する燃料量を気筒別に制御することである。

我々は当初、各々の気筒を直接フィードバック制御することを検討したが、これでは気筒毎に4個のセンサが必要となり、システムのコストアップを伴ってしまう。そこで、一つのセンサで各々の気筒の空燃比が推定できる手法が求められた。

このため、排気マニホールド内の排気混合挙動の数学的モデルを検討し、その概要を Fig.7 に示す。

これは、センサ取り付け位置（例えば集合部の排気合流ポイント）での排気の混合する挙動を模式的に示したものである。図中の『V1からV4』は、ある時刻において、各気筒排気のそれぞれが混ざり合う様子を、排気の体積変化割合として展開を図った。またさらに、この空燃比を検出しているリニア空燃比センサの応答挙動を数

式化することも必要であった。これらの二つの数学モデルと、現代制御理論（この場合オブザーバ理論）をもちいて各気筒の空燃比の推定を行った。例えば『排気マニホールドモデル』として状態方程式にて表現すると次のようになる。

Exhaust manifold model

$$\begin{bmatrix} A/F(k-2) \\ A/F(k-1) \\ A/F(k) \\ A/F(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} A/F(k-3) \\ A/F(k-2) \\ A/F(k-1) \\ A/F(k) \end{bmatrix}$$

$$\text{Confluence Point } A/F(k) = [C_1 \ C_2 \ C_3 \ C_4] \begin{bmatrix} A/F(k-3) \\ A/F(k-2) \\ A/F(k-1) \\ A/F(k) \end{bmatrix}$$

State Equation

$$\begin{aligned} X(k+1) &= A \cdot X(k) \\ Y(k) &= C \cdot X(k) \end{aligned}$$

Observer configuration

$$\hat{O}(k+1) = (A - KC) \cdot \hat{O}(k) + K \cdot Y(k)$$

この数学モデルの C_1 から C_4 という係数は、センサ取り付け位置での空燃比挙動を正確に算出できるように、実機計測データでシミュレーションを行って同定した。こうして一本のセンサで検出された空燃比のみで、各気筒の空燃比推定が可能となった。

Fig.8 は、気筒別空燃比制御の効果を示す。ここでは、集合部空燃比制御を気筒別空燃比制御に切り換え、その前後の空燃比の変動を比較した。前者では気筒毎の空燃比のばらつきがあるが、後者では気筒間の空燃比のばらつきがなくなり、すべての気筒の空燃比が目標通りに制御できていることがわかる。

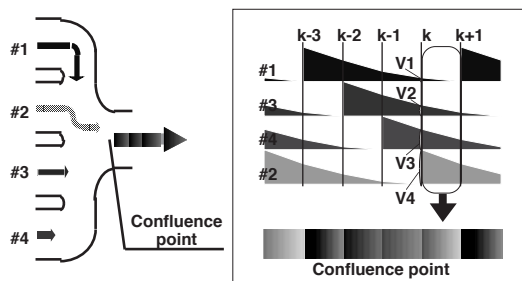


Fig.7 Individual cylinder A/F control observer

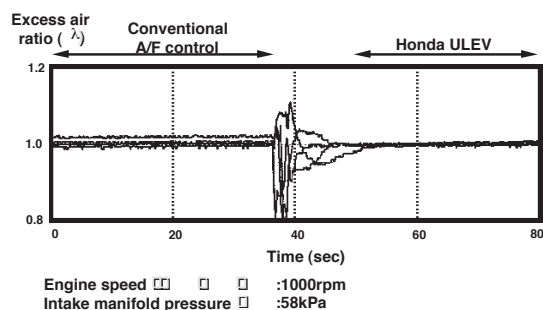


Fig.8 Effect of individual cylinder A/F control

3. 2. セルフチューニングレギュレータ (STR)

Fig.9 に、STRのブロック線図を示す。STRによる空燃比フィードバックシステムの効果は、スロットルによる加減速や、EGRの導入、キャニスタパージなどの急激な過渡状態であっても、常に安定した空燃比を維持できることにある。

従来型の燃料噴射制御システムと異なる点は、このSTR機構が真の意味で、適応制御システムとなっていることにある。例えば、在来型のシステムでは、空燃比を制御するために、エンジン運転条件に応じて、あらかじめ制御パラメータが設定されており、これに基づいて空燃比フィードバック制御を行っている。これに対してSTRは、常に目標空燃比とセンサによって検出された実際の空燃比と比較し、リアルタイムに最適な制御パラメータを算出している。このため、極めて高い応答性能と、様々な外乱に対する優れた安定性を合わせ持っている。しかしながら、ここで演算される制御パラメータは、複雑な計算を要するものであるため、設定データは少ないものの、CPUの演算スピードが従来よりも高速であることが望まれる。

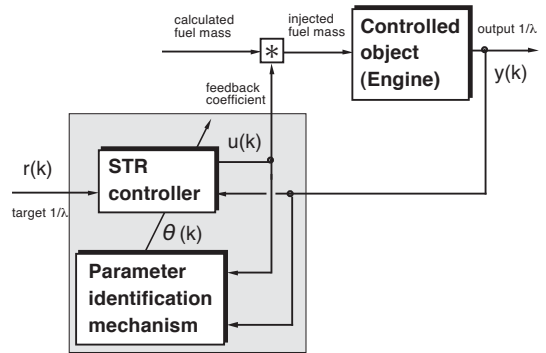
Fig.10には、在来型の制御方式を使用した場合の典型的なステップ入力に対する空燃比応答特性を示した。この場合、従来法では安定性を得るために応答性を失うか、逆に応答性を高めて安定性を犠牲にしなければならないかった。

Fig.11には、STRを使用した時の空燃比応答性を示している。このように、素早く目標値に応答し、そのまま安定性を維持している。従って応答性も安定性も犠牲にする必要がなく、最適な制御性能を得ることができる。さらに、従来の制御器では、エンジンの様々な運転条件を想定し、パラメータを設定しなければならないが、こういった作業はほとんど必要がなく、大幅な開発効率の向上が可能である。

4. シミュレータによる制御系設計

空燃比制御の高精度化のためにオブザーバ理論とセルフチューニングレギュレータを用いたエンジン制御ECUの制御系設計の為に、本研究では設計ツールとしてMATRIX-X、AUTO-CODEを用いた。

Fig.12に開発環境の概要を示す。ワークステーション上では制御対象となるシステムのモデル化、作成された



F/B type STR

Fig.9 Block diagram of STR

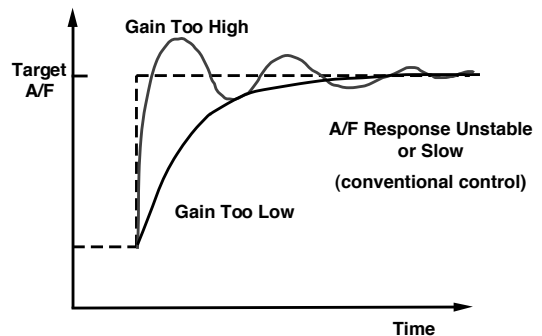


Fig.10 Responsibility of conventional control

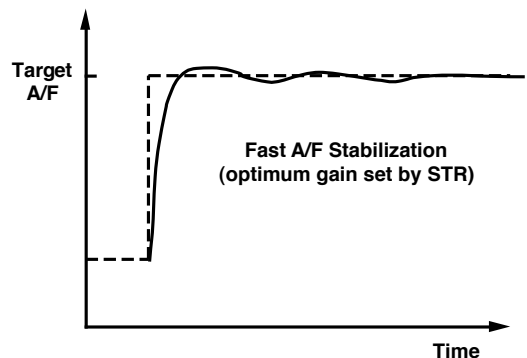


Fig.11 Response of adaptive control
"Self-Tuning Regulator"

システムモデルに対して効率的なコントローラの設計、設計されたコントローラよりプログラムの自動生成を行う。汎用ECU上では生成されたプログラムによる実際のエンジンを用いた制御性能の確認となる。

Fig.13に設計フローを示す。システムのモデル化では制御対象の特性を把握し、次に作成したモデルがシステム本来の特性を表しているかを検証し、それと同時にモデル内の実際の係数を同定した (Fig.13 *1)。制御対象と

なるシステムのモデル化において重要なことは、システムの特徴を表現するできるだけ簡単なモデルを作成することである。より簡素化することによりシステムの特徴を把握しやすくすると同時に、効率的な制御が可能となる。

続いて制御目的と要求を明確にし、作成したモデルに対して各種制御理論の適応を検討し、コントローラの設計を行う。コントローラ設計の上で重要なことはシステ

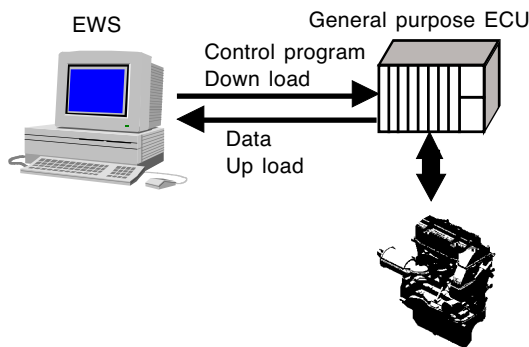


Fig.12 Development system

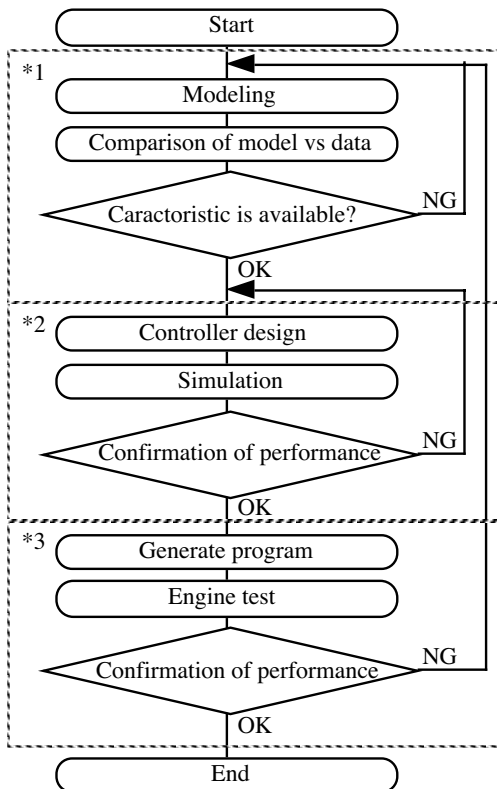


Fig.13 Design flow

ムのモデル化と同様に要求を満足しうるできるだけ簡単なコントローラを設計することである。簡素化することで各コントローラの動作、干渉を把握しやすくできる。各コントローラはシミュレーションにより制御定数の設定および制御性の確認を行う (Fig.13*2)。

最後に実際のエンジンを用いた制御性能の確認は、作成されたコントローラからプログラム自動生成ツール (AUTO-CODE) によりプログラムを生成し、Fig.14 に示す様な汎用 ECU 上で行う。汎用 ECU は制御を実行する Main CPU board、Main CPU board と入出力をつなぐ Sub CPU board、アナログ入力用の A/D Conv.board、パルス入力用の Pulse input board、インジェクタを駆動する Output board からなる。自動生成されたプログラムはデータ演算に浮動小数点を使用しているため整数型に変換後、ワークステーションから汎用 ECU 上にダウンロードして実際にエンジンを制御する。実際のエンジンでの確認時には入力データ、出力データ、内部制御データを汎用 ECU でロギングしてシミュレーションに使用できるようにしてあり、期待した制御特性が得られなかった場合、ロギングデータを使用して解析、シミュレーションを行い、システムモデル、コントローラのリファインを行う (Fig.13*3)。このようにシステムモデルおよびコントローラに対してフィードバックを行うことで、制御系の完成度を高めた。完成したコントローラの一例を Fig.15 に示す。

5. ソフトウェアのシュリンクと C 言語の採用

制御系の設計・実車確認を性能・汎用性に優れた汎用 ECU によって実施することを先に述べたが、最終的に量産車で適用できるかどうかの確認は量産 ECU の形状に近

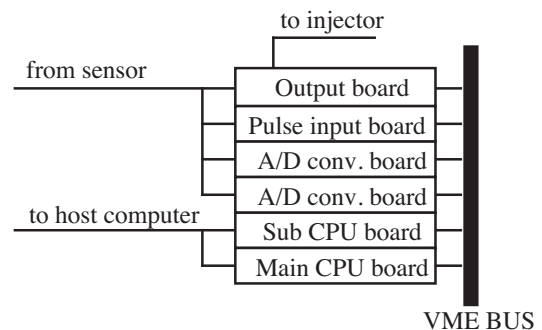


Fig.14 General purpose ECU

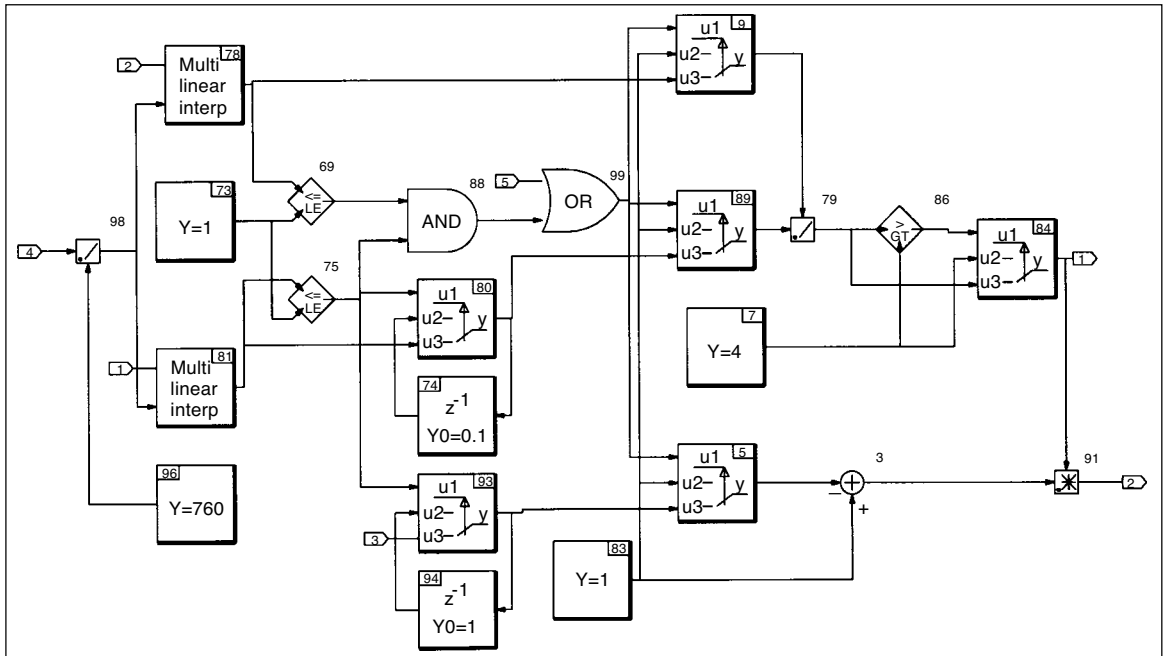


Fig.15 Example of controller

い小型 ECU により実施する必要がある。汎用 ECU ではハード資源の制約がないことにより、汎用の高速 CPU や大容量のメモリを搭載しているが、コスト・耐環境性能の面で量産適用実現不可能であり、いかに性能を落とすことなく同様の制御系を構築するかが課題となる。小型化を実施する上で、最大の課題はソフトウェアである。MATRIX-X, AUTO-CODE で出力されるソフトウェアは C 言語記述であるが冗長があり、プログラムサイズが大きいという欠点がある。我々は C 言語を使いつつ、ソフトウェアのシュリンクという課題を達成しようと考えた。

車載用の量産 ECU の場合、大きさ・コスト・消費電力の制約からインテル 86 系やモトローラ 68 K 系の汎用高性能マイコンに比べ、機能・性能の限定されるワンチップマイコンを使用することが多く、このワンチップマイコンの性能上の制約からソフトウェア記述言語はアセンブラが主流であり、エンジン制御量産 ECU への C 言語の採用がなかった。しかし、本研究のように数値演算を多用した現代制御理論ベースで、制御ロジックの設計・確認を短期間で何度も繰り返し実施して目標性能を達成する開発手法をとり、ソフトウェア規模が大きい制御システム開発の場合、C 言語のように論理を自然な形で表現できる高級言語の採用が必要であった。

C 言語の特徴をまとめると下記ようになる。

- (1) 数値論理式がそのまま記述できる為、数値演算ソフト開発に有効。
- (2) シミュレーションによる制御ロジック検証容易。
- (3) マイコン依存度が低く、選択自由度が大。
- (4) 最もポピュラーなコンピュータ言語の一つであり、教育・習得が容易。
- (5) メモリ使用量が少なく、コンパクト。
- (6) 割り込みの記述や、ハードウェアへの直接アクセス可能。

よって、C 言語に適してかつ性能も十分なマイコンさえあれば、制約の多いアセンブラ言語に比べ開発効率を上げることができる。

Fig.17 に本研究の制御仕様例、Fig.16 に AUTO-CODE 出力ソフトウェア例、Fig.18 にシュリンクした C 言語ソフトウェア例を示す。

AUTO-CODE 出力ソフトウェアは C 言語記述であるが冗長かつ理解しにくい構造になっていることが分かる。これに比べ Fig.18 のシュリンクした C 言語記述ソフトウェアは直感的であり、設計変更に強い記述形態となっている。

AUTO-CODE 出力ソフトウェアのシュリンクは下記の

手法により実施した。

(1) 変数データサイズの見直し

C言語ではデータ型を利用でき用途に応じ使い分けるが、AUTO-CODE ではすべての変数を同じ型としてしまうので、各変数を最適な型に変換する。

```

/*----- Elementwise Product */
/* {ZETA_GAMMA_ZETA..67} */
BUS[1] = U[0] * U[5];
/*----- Elementwise Product */
/* {ZETA_GAMMA_ZETA..65} */
BUS[2] = U[1] * U[6];
/*----- Quantization */
/* {ZETA_GAMMA_ZETA.QTZ 1 65535.95} */
BUS[3] = 0.0000153 * (RT_FLOAT)((RT_INTEGER)( BUS[1] /0.0000153 ));
/*----- Quantization */
/* {ZETA_GAMMA_ZETA.QTZ 1 65535.92} */
BUS[4] = 0.0000153 * (RT_FLOAT)((RT_INTEGER)( BUS[2] /0.0000153 ));
/*----- Summing Junction */
/* {ZETA_GAMMA_ZETA..68} */
BUS[5] = BUS[3] + BUS[4];
/*----- Elementwise Product */
/* {ZETA_GAMMA_ZETA..66} */
BUS[6] = U[3] * U[8];
/*----- Elementwise Product */
/* {ZETA_GAMMA_ZETA..64} */
BUS[7] = U[2] * U[7];
/*----- Quantization */
/* {ZETA_GAMMA_ZETA.QTZ 1 65535.94} */
BUS[8] = 0.0000153 * (RT_FLOAT)((RT_INTEGER)( BUS[6] /0.0000153 ));
/*----- Quantization */
/* {ZETA_GAMMA_ZETA.QTZ 1 65535.93} */
BUS[9] = 0.0000153 * (RT_FLOAT)((RT_INTEGER)( BUS[7] /0.0000153 ));
/*----- Summing Junction */
/* {ZETA_GAMMA_ZETA..98} */
BUS[10] = BUS[9] + BUS[8];
/*----- Quantization */
/* {ZETA_GAMMA_ZETA.QTZ 1 65535.88} */
BUS[11] = 0.0000153 * (RT_FLOAT)((RT_INTEGER)( BUS[10] /0.0000153 ));
/*----- Quantization */
/* {ZETA_GAMMA_ZETA.QTZ 1 65535.87} */
BUS[12] = 0.0000153 * (RT_FLOAT)((RT_INTEGER)( BUS[5] /0.0000153 ));
/*----- Summing Junction */
/* {ZETA_GAMMA_ZETA..99} */
BUS[13] = BUS[12] + BUS[11];
/*----- Elementwise Product */
/* {ZETA_GAMMA_ZETA..91} */
BUS[14] = U[4] * U[9];
/*----- Quantization */
/* {ZETA_GAMMA_ZETA.QTZ 1 65535.96} */
BUS[15] = 0.0000153 * (RT_FLOAT)((RT_INTEGER)( BUS[14] /0.0000153 ));
/*----- Quantization */
/* {ZETA_GAMMA_ZETA.QTZ 1 65535.85} */
BUS[16] = 0.0000153 * (RT_FLOAT)((RT_INTEGER)( BUS[13] /0.0000153 ));
/*----- Summing Junction */
/* {ZETA_GAMMA_ZETA..90} */
BUS[17] = BUS[16] + BUS[15];
/*----- Quantization */
/* {ZETA_GAMMA_ZETA.QTZ 1 65535.89} */
BUS[0] = 0.0000153 * (RT_FLOAT)((RT_INTEGER)( BUS[17] /0.0000153 ));
/*----- Collect Ext. Outputs into Y vector */
Y = BUS[0];

```

Fig.16 AUTO-CODE output

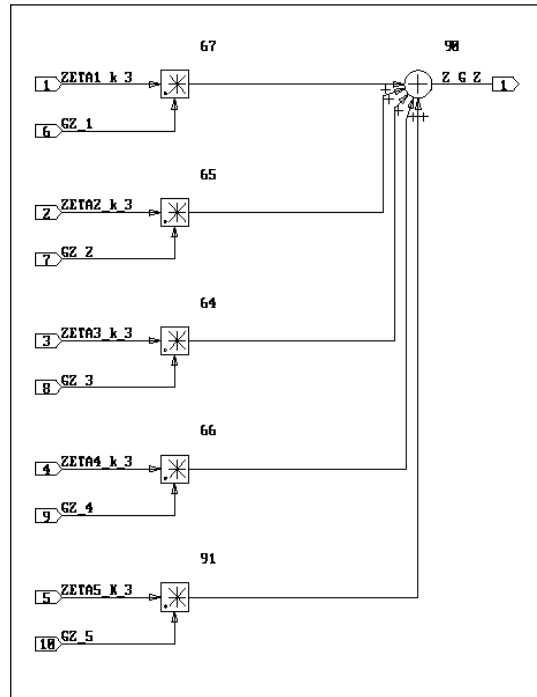


Fig.17 Example of control spec.

$$Y = ((Zeta1 * GZ1) + (Zeta2 * GZ2) + (Zeta3 * GZ3) + (Zeta4 * GZ4) + (Zeta5 * GZ5)) >> 16;$$

Fig.18 Shrunked C-program

(2) 中間変数の削除

AUTO-CODE ではブロックごとにソフトウェア展開するので、ブロックを接続するだけの中間変数定義が多いので、これを削除する。

(3) 機能ブロックの見直し

変数変換のブロックや、デバッグ用のブロックなど小型 ECU には不要なブロックは削除する。

(4) その他最適化

最適化という観点でみると、自動生成されたソフトウェアには冗長な部分が多いので、丹念にソフトウェアを解析、修正する。

Fig.19に燃料量算出部のAUTO-CODEで自動生成されたソフトウェアのサイズとシュリンク後のサイズを示す。編集後約 30%にシュリンクできたことがわかる。

また、ソフトウェア品質作り込みの為、設計の標準化を行った。Fig.20に「ソフト設計ルール」の一例を示す。

6. ワンチップ RISC マイコンの開発

半導体技術の進歩により、年々マイコンの性能機能向上が行われ、現在エンジン制御ECU用として使用しているマイコンは 16bitCPU コアに 64 バイト以上のメモリを内蔵したワンチップマイコンが主流となっている。汎用マイコンとの違いは大容量の ROM (Read Only Memory) と RAM (Random Access Memory) を内蔵していること、多チャンネルの AD コンバータとインテリジェントタイマを内蔵していることである。特にエンジン制御用の場合、高精度のインジェクタ、イグナイタ制御が必要となるため、キャブチャ機能や一致判定機能を複合した高機能のタイマを専用設計し、ソフトウェアの負担を軽減している。

本研究で適用したオブザーバおよびセルフチューニングレギュレータは多数の行列演算で構成されている。この行列演算は言い換えると乗算と和算の集合であり、乗算和算を高速実行するプロセッサが必要になる。我々のプログラムを解析した結果、従来のマイコンに比べ約 8 倍から 10 倍程度演算速度の早いマイコンが必要であることがわかった。

一般的に乗算和算を高速実行するプロセッサとして DSP (Digital Signal Processor) があり、オブザーバおよびセルフチューニングレギュレータの演算を担当するという構成も考えられるが、ECU の大幅なコスト上昇と大型化を伴う為、マイコン自体の演算処理能力を向上する方法を考えた。更に、数値演算制御に適した C 言語で開発を進めることから C 言語プログラムに適したアーキテクチャのマイコンが必要となった。

従来の自動車用マイコンはアセンブラベース開発に最適な、少ないメモリでより多くの仕事ができる、いわゆる CISC (既出) 型 CPU コアベースで開発してきた。これは各演算命令に対してアドレッシング (メモリアクセス手法) を多数用意することによって、プログラムの開発を容易にするアーキテクチャである。CISC 型の場合、命令数が多く複雑になるため、CPU の機能向上によりチップサイズが増大し、コストが上がり、当初の狙いから相反する事象がでてきた。更に、CPU 命令形態が複雑であるため C コンパイラの開発が難しく、かつこれ以上の良いものが作成しにくい状況になった。

これらの打開策として、我々はマルチメディア用とし

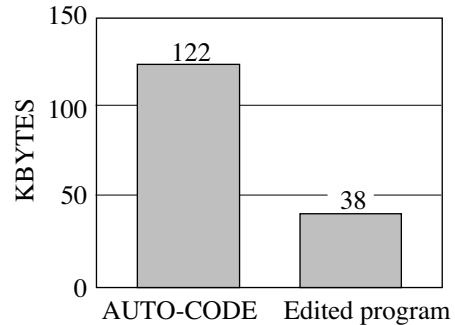


Fig.19 Program size of the fuel volume calculation part

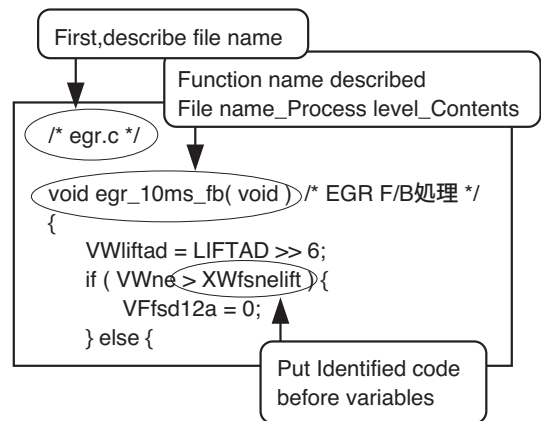


Fig.20 Example of design rules

て開発されてきた組込用 RISC アーキテクチャに着目した。C 言語開発を前提に C 言語であまり使用しない CPU 命令を大幅に削減し、チップサイズを限界まで小さくしたアーキテクチャである。

CPU の高性能化にはクロックの高速化が有効であり、従来からクロックアップの改良が続けられているが、自動車用クリスタルは 32MHz が物理的に最高速度であり、ラジオノイズ対策からもパソコンで使用しているような超高速化は限界がある。そのため同じクロックでも高性能となる CPU アーキテクチャの採用が必要である。

このアーキテクチャのひとつとしてパイプライン手法がある。Fig.21 にパイプラインの構成例を示す。CPU の

処理機能をいくつかに分け、平行処理することによって、あたかも1クロックで命令が実行しているかのように見える手法である³⁾。実際には演算の対象を内部レジスタに限定し、ハード構成をシンプルにしてパイプラインを実現している。このRISC型CPUコアをベースに自動車用32bitワンチップRISC型マイコンを開発し、高性能化とコスト化の両立をはかった。Fig.22に開発した自動車用マイコンの構成を示す。RISCの場合必要なメモリ容量が大きくなるという課題があるが、CPUコアチップ面積が小さくなった分メモリを大量に搭載し、さらに性能向

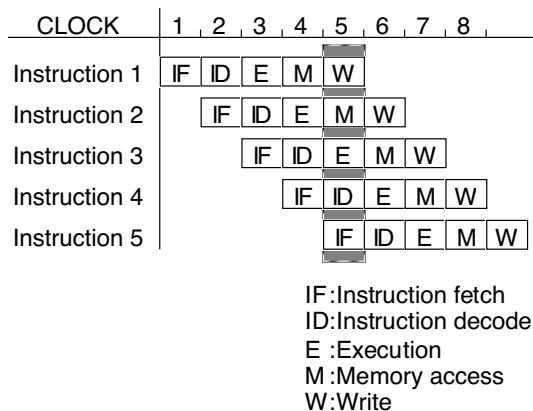


Fig.21 Parallel execution by pipe lining

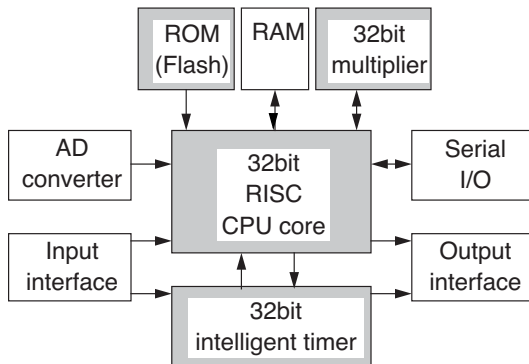


Fig.22 Newly developed automotive 1chip 32bit microcomputer

Table 2 Newly developed micro computer spec

	Newly Developed CPU	Conventional
Architecture	RISC	CISC
Instruction set	62	100 ~ 200
Minimum instruction size	2 byte	1~6 byte
Instruction cycle	1~4	2~49
Minimum execution times	50nsec	200nsec
16bit signed multiplier	0.15 μ sec	2.5 μ sec
16bit signed divider	4.0 μ sec	39 μ sec
CPU Core size	6.6mm ²	13.0mm ²
ROM size	128k	64k ~ 94k
RAM size	6k	2k ~ 4k
ADC	10bit 16ch	10bit 16ch
Serial I/O	3ch	2ch
Timer	32bit CAP 4ch 16bit Oneshot 8ch 16bit PWM 4ch 16bit CAP/CMP 16ch	16bit 4ch 4ch 4ch 4ch

at 20MHz

上のため32bit長の乗算器および32bit長データを扱えるインテリジェントタイマを新規設計し搭載した。

Table 2に新規開発マイコンと従来マイコンの性能比較を示す。同じ20MHzクロックを付けた条件で基本実行命令時間4倍、行列演算を多用する本ソフトウェアでの実行時間で従来比8倍の性能向上がはかれた。

このCPUを用いたULEV用ECUにより、高精度空燃比制御を実行した。

7. 実機テスト結果

7. 1. 空燃比出現頻度

オブザーバ理論とSTRを用いた空燃比フィードバック制御技術を併用した場合の相乗効果についてFig.23に示す。図に示すように目標値 $\pm 0.2A/F$ の非常に狭い領域内に、従来のシステムでは49%程度にすぎなかった空燃比の時間出現頻度が、約85%にまで向上している (LA-4 試験サイクル)。

7. 2. 排出ガス試験結果

Fig.24には、プロトタイプULEVアコードに対して10万マイル以上にわたって行われたCARBの排出ガス試験の結果とNMOG劣化特性を示す。この車輛の排出ガスレベルがULEV規制値に対して十分に低いレベルであることがわかる。また触媒装置は10万マイルまで劣化進行するが5万マイルの基準値以内である。

8. まとめ

- (1) 制御系シミュレーションシステムを構築し、現代制御による機関の制御設計を最適化した。
- (2) オブザーバとセルフチューニングレギュレータを車載用小型 ECU として具現化するため、数値演算の高速実行可能なワンチップ RISC マイコンを開発した。
- (3) 数値演算記述に適した C 言語のメリットを生かし、ULEV 用 ECU のプログラムを作成した。
- (4) 高精度空燃比制御を実現した本 ECU を用いた制御によって、 $\pm 0.2 \text{ A/F}$ の出現頻度を従来の 49% から 85% に高めることができ、コールドスタート時の排出ガス低減手法や触媒活性化時間の短縮化と組み合わせることにより ULEV を可能とした。

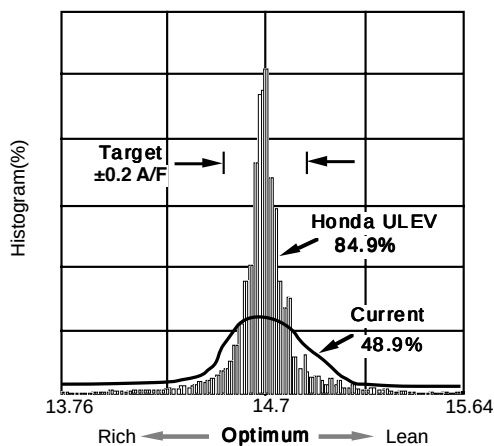


Fig.23 Effect of precision A/F control

	NMOG	CO	NOx (g/mile)
Honda	0.035	0.527	0.054
Standard	0.055	2.1	0.3

— Measured at ARB Laboratory, using California Phase-2 reformulated gasoline
 — 100,000 mile aged Catalyst
 — 100,000 mile standards

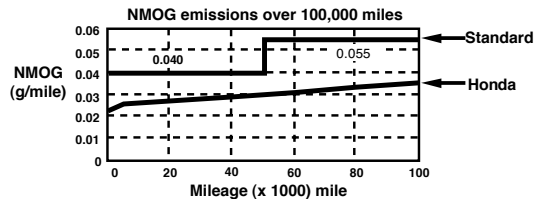


Fig.24 Honda ULEV test results

参考文献

- 1) K.Horie et al, 「Emission Reduction During Warm-Up Period by Incorporating a Wall-Wetting Fuel Model on the Fuel Injection Strategy During Engine Starting」, SAE 952478, 1995
- 2) H.Maki et al, 「Real Time Engine Control Using STR in Feedback System」, SAE. 950007, 1995
- 3) D.A.Patterson, J.L.Hennessy, 「Computer Architecture: A Quantitative Approach」, 1990

■ 著 者 ■



川 合 誠



橋 本 英 樹



赤 崎 修 介



西 村 要 一



大 保 慎 一



上 田 伸 一