**Buna ziua,**

Numele meu este Buzuriu George-Lucian si astazi am sa va vorbesc despre optimizarea timpului de reactie in sisteme TCU multicore.

**Ce este un TCU?**

* TCU sau unitatea de control a cutiilor cu transmisie automată, este un dispozitiv care controlează transmisiile electronice moderne automate. Un TCU foloseşte, în general, senzorii de la vehicul, precum şi datele furnizate de către unitatea de control a motorului pentru a calcula cum şi când să schimbe vitezele în vehicul pentru performanțe optime, economie de combustibil şi de calitate a schimbării treptelor.

**Ce se intelege prin timp de reactie?**

* Prin termenul de timp de reactie, m-am referit la perioada in care TCU-ul incepe sa comunice cu celelalte componente, adica cu unitatile de control electronice din masina. Deci, timpul de reactie este de fapt timpul de initializare al unitatii de control a transmisiei. Astfel, dupa cum reiese si din titlul lucrarii, imi doresc sa reduc acest timp de initializare.

**De ce este necesara imbunatatirea?**

* Dorinta imbunatatirii a iesit la iveala in momentul in care Hardware-ul proiectului s-a schimbat. De la un microcontroler cu un singur core, TC1784, s-a trecut la un microcontroler cu 3 core-uri, Aurix TC277. Bineinteles, odata cu schimbarea, s-au preluat in totalitatea functionalitatile dezvoltate pe generatia veche.

In continuare, o sa ilustrez o schema succinta de comunicare a TCU-ului cu celelalte componente electronice care la randul lor manipuleza: ESC -> ESP-ul si anume viteza rotilor, EFP -> acceleratia, QSP diferentialul blocabil pentru masinile sport, Motorul si Gateway...

**Slide urmator**

**Componentele de initializare ale TCU-ului**

Datorita faptului ca microcontrolerul are spatiul pentru data flash, Dflash, avem EEPROM emulat. Bineinteles, chipul de EEPROM este la locul lui. In plus, avem si functionalitatile de safety. Asdar, pentru ca un TCU sa se initializeze, este necesar sa citim datele din EEPROM-ul emulat, EEPROM-ul fizic si sa realizeze Switch off patch checkul.

Deoarece este o memorie din care doar putem citi, este clar ca avem nevoie de un element intermediar. Acesta il presupune oglinda RAM. Aici, ne vom salva datele de la ultima folosire a masinii, atat date din EEPROM fizic, cat si date din EEPROM emulat.

In continuare, prezint o mica schema care ilustreaza intreg procesul de stocare si folosire al datelor. : Astfel,

* Se actioneaza KL15, datele din EEPROM de la ultima functionare a masinii sunt necesare
* Atat cele din EEPROM fizic cat si cele din EEPROM emulat le stocam in oglinda RAM
* De aici, le putem accesa pentru folosirea locala
* Totodata, putem altera aceste date si rescrie valorile.
* La KL15 OFF, vom memora datele in EEPROM, deoarece informatiile din RAM se vor pierde.

Cand se scriu date in RAM:

* La KL15 ON
* Pe durata functionarii
* La momente de timp bine stabilite

**Slide urmator**

Revenind la procesul de imbunatatire al timpului, prima idee logica si intuitiva a fost sa masor durata de citire a datelor din EEPROM-ul extern. Tin sa precizez ca in acel moment, nu cunoasteam celelalte elemente necesare initializarii. Aveam cunostinte doar despre chipul EEPROM. Asa ca am sapat dupa informatii. Am descoperit ca citirile datelor din EEPROM-ul extern sunt realizate de 2 functii. Pana la urma, solutia 1 a fost sa iau tot datele citite de prin cele 2 functii si sa le pun pe core-ul 0, lasand operatiile aditionale pe core-ul1. Beneficiul nu a fost cel scontat, astfel ca am continuat cautarile.

Apoi, dezvoltand cautarile am ajuns la SOPC, Switch off patch check.

Prin SOPC, trebuie să fim capabili să dovedim că mecanismele de safety funcționează. În această categorie intră toate funcțiile care pot duce la răniri. Acestea din urmă, trebuie să fie tratate special conform ISO 26262.

* se verifică Question<->Answer între microcontroler și unitatea de monitorizare. Dacă vin prea multe răspunsuri greșite, se va cere starea de siguranta, safe state.
* microcontrolerul poate să mai ceară starea de siguranță pe o linie digitală supervizată de către unitatea de monitorizare
* după ce s-au efectuat verificările că se poate intra în safe state, se resetează și se face self test dacă motorul poate răspunde la schimbări

EEPROM-ul emulat este Dflash-ul microcontrolerului, de dimensiuni reduse prin care se citesc doar 4 canale de EEPROM.

**Slide urmator**

Asadar, am pornit de la situatia initiala, cu toate elementele citite pe un core. Astfel, am dispus tot calupul de date din EEPROM-ul fizic pe core-ul0, iar pe core-ul1 am pus SOPC + citire din EEPROM emulat.

Totodata, in situatia implementata, interfata SPI-ul este necesara sa aiba 2 job-uri separate. Unul sa vina cu intrerupere mereu pe core-ul 0 pentru citirea datelor din EEPROM-ul extern, iar celalalt sa vina cu intrerupere pe core-ul 1, pentru realizarea SOPC-ului.

Prin aceste modificari, s-a adus o imbunatatire considerabila a timpului de initializare.

Astfel, de la o durata de aproape 200 milisecunde, s-a redus la o durata undeva la 111milisecunde.

O imbunatatire considerabila si acceptata de catre client.

Cum am facut sa testez? La mom. De actionare al KL 15 ON, am initializat o variabila cu 0. Apoi, am intrat in task-ul de init si acolo am citit datele din EEPROM extern si emulat, dar si functionalitatile de safety in conf. Cu metoda implementata. Cand aceste elemente s-au terminat, culmineaza cu executia primului task prioritat, task de 2.5 ms. Atunci schimbam variabila in „1” logic.

In viitor se doreste trecerea de la EEPROM-ul extern la cel emulat in totalitate. Principalul element in aceasta schimbare ar fi costul.

O imbunatatire ar fi si implementarea blocurilor