Primjena FPS metode raspoređivanja na sustavu automatizacije upravljanog PAC uređajem

Goran Malčić*, Ivica Vlašić** i Danijel Maršić*
* Tehničko veleučilište u Zagrebu / Elektrotehnički odjel, Zagreb, Hrvatska
goran.malcic@tvz.hr
** Vjesnik d.d. / R.J. Održavanje, Zagreb, Hrvatska
ivica.vlasic@vjesnik.hr

Sažetak - Programirljivi automatizacijski kontroleri (PAC) su nova generacija industrijskih upravljačkih uređaja koji kombiniraju funkcionalnost PLC uređaja i PC računala. PAC uređaji slijede normu IEC 61131-3 koja između ostalog programeru daje punu kontrolu nad redoslijedom i dinamikom izvršavanja procesnih zadataka (*eng. task scheduling*). U većini slučajeva se koristi statičko raspoređivanje koje je tipično za upravljačke procese u industrijskim postrojenjima jer ono podrazumijeva prethodno poznavanje bitnih značajki procesnih zadataka kao što je učestalost zahtjeva za obradu. Iako se smatra nefleksibilnim jer se procesima dodjeljuju fiksni prioriteti (*eng. Fixed-Priority Scheduling, FPS)* ovo je najčešće korišteno raspoređivanje u takvim sustavima. Važna karakteristika ovako definiranog raspoređivanja je veća predvidljivost te mogućnost testiranja raspoređivosti koje se zasniva na ispitivanju iskorištenja procesa (metoda *Liu & Layland*). U ovom radu je uzet primjer upravljačkog programa gdje je primijenjen ISA S88 standard na sustavu distribucije cementa. Upravljački program ciklički izvodi programske module jedan iza drugoga što rezultira prevelikim ukupnim vremenom izvršavanja ciklusa od preko 260 ms. Kako program čita ulaze samo na početku ciklusa podaci se osvježavaju otprilike 4 puta u sekundi što je uzrok da se pojedina stanja periferije uopće ne registriraju. Rješenje ovog problema je da se kritične rutine izvršavaju periodički i osiguravaju neposredno ažuriranje stanja periferije.

I. UVOD

Proteklih desetak godina postavljaju se pitanja o prednostima i nedostacima programirljivih logičkih kontrolera (eng. ***P**rogrammable **L**ogic **C**ontrollers*, dalje *PLC*) u usporedbi s upravljačkim sustavima izvedenim pomoću PC računala (eng. *PC based control*). Analize su pokazale da je 80 % industrijskih aplikacija izvedeno tradicionalnim načinima upravljanja što na tržištu uzrokuje potrebu za jeftinim PLC uređajima [3]. To je potaknulo povećanu ponudu po cijeni pristupačnih mikro PLC uređaja koji sadrže samo digitalne U/I i osnovne instrukcije ladder logike. Jednako tako, došlo je do nesrazmjera u tehnologiji kontrolera jer 80 % aplikacija zahtijeva jednostavne i pristupačne PLC uređaje dok preostalih 20 % aplikacija uvelike premašuje mogućnosti klasičnih upravljačkih sustava. Projektantima modernih industrijskih aplikacija koje spadaju unutar tih 20 % postavljaju se zahtjevi za naprednim upravljačkim algoritmima i boljom integracijom sa poslovnim sustavima poduzeća. Vodeći proizvođači su povećanom opsegu zahtjeva modernih industrijskih aplikacija odgovorili razvojem industrijskih upravljačkih uređaja koji predstavljaju kombinaciju najboljih obilježja PLC uređaja i PC računala. Tako je nastala nova vrsta kontrolera koji su nazvani programirljivi automatizacijski kontroleri (eng. ***P**rogrammable **A**utomation **C**ontroller*, dalje *PAC)*. Skraćenicu PAC koriste kako proizvođači klasičnih PLC uređaja za opis svojih upravljačkih uređaja visoke klase (eng. *High end*) tako i proizvođači upravljačkih sustava izvedenih PC računalima za opis svojih industrijskih upravljačkih platformi. Time su spojene prednosti PLC uređaja u upravljanju procesima sa fleksibilnošću konfiguracije i sposobnošću integracije sa poslovnom razinom koje pružaju upravljački sustavi bazirani na PC računalima.

II. RASPOREĐIVANJE U RT SUSTAVIMA

U sustavima koji rade u realnom vremenu (*eng. **R**eal **T**ime*, dalje *RT*) raspoređivanje procesa (*eng. task*) može biti statičko i dinamičko. Statičko raspoređivanje je tipično za upravljačke procese u industrijskim postrojenjima jer ono podrazumijeva prethodno poznavanje bitnih značajki procesnih zadataka. To se u prvom redu odnosi na karakteristike kao što su učestalost (frekvencija) zahtjeva za obradom (procesiranjem) i računalna vremena (*eng. Computation times*). Raspored procesa kao rezultat ove metode je nepromijenjen tijekom izvođenja i omogućava procjenu mogućnosti izvođenja (*eng. Schedulability analysis*). Iako se smatra nefleksibilnom metodom jer se procesima dodjeljuju fiksni prioriteti (*eng. **F**ixed-**P**riority **S**cheduling*, dalje *FPS)* prije izvođenja ovo je najčešće korišteno raspoređivanje osobito u sustavima s tzv. „kritičnom sigurnošću“. Ovakvo raspoređivanje može biti izvedeno sa mogućnošću prekida (eng. *Preemptive*) i bez mogućnosti prekida (eng. *Non-preemptive*) koje predstavlja klasičan način izvođenja programa. Dakle, prvo spomenuto raspoređivanje karakterizira mogućnost prekidanja procesa i njihovo naknadno nastavljanje bez gubitka njihovih funkcionalnosti. Proces u trenutku izvođenja može biti prekinut samo od strane drugog procesa kojem je pridodjeljen viši prioritet. Prioritet procesa u takvom sustavu nije posljedica značaja dotičnog procesa nego vremenskih karakteristika. Metoda dodjeljivanja prioriteta procesima je veoma jednostavna i kaže da se prioriteti dodjeljuju monotono po učestalostima (*eng. **R**ate-*

***M**onotonic **P**riority **O**rdering*, dalje *RMPO*). To znači da se periodičnim procesima dodjeljuju jedinstveni prioriteti, monotono uređeni prema učestalosti procesa, tako da proces sa kraćom periodom ima viši prioritet.

Još jedna važna karakteristika ovako definiranog raspoređivanja je veća predvidljivost (*eng. Predictability*) te mogućnost testiranja rasporedivosti (*eng. Schedule Feasibility*). Kriterij provjere rasporedivosti skupa periodičnih procesa je zasnovan samo na ispitivanju iskorištenja procesora (*eng. Processor Utilization)*. Iako on nije egzaktan, jer predstavlja samo dovoljan, ali ne i potreban uvjet rasporedivosti, veoma je popularan zbog svoj jednostavnosti. Prema tome testu je skup periodičnih procesa rasporediv ako je ispunjen slijedeći uvjet:

$$\sum_{i=1}^{N}\left(\frac{C_i}{T_i}\right) \le N \cdot \left(2^{\frac{1}{N}} - 1\right) \qquad (1)$$

gdje je:

C_I - Vrijeme izvršavanja u najgorem slučaju (*eng. **W**orst-**C**ase **E**xecution **T**ime, WCET*)

T_i - Vrijeme između dvije susjedne aktivacije procesa (period procesa)

N - Ukupan broj procesa

Lijeva strana nejednadžbe predstavlja sumu svih iskorištenja procesora dok desna strana predstavlja graničnu vrijednost iskorištenja procesora za N procesa iznad koje uvjet više ne zadovoljava. Zbog toga što ova granična vrijednost teži brojci 69.3%, kada broj procesa teži beskonačnosti, zaključujemo da je svaki skup procesa koji ima ukupno iskorištenje manje od 69.3% sigurno rasporediv korištenjem ove metode.

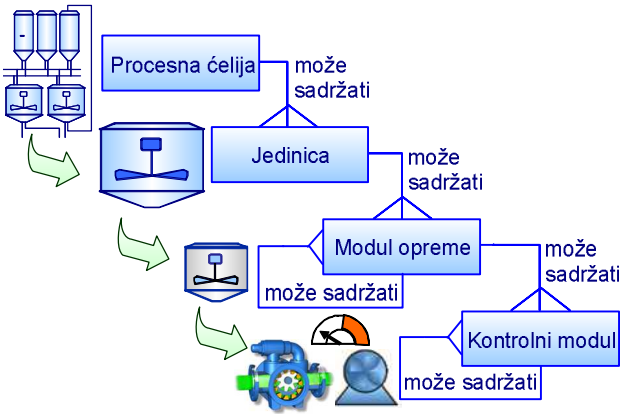
III. PRILAGODBA ISA-S88 STANDARDU

Industrijski procesi se definiraju kao slijed kemijskih, fizičkih i bioloških aktivnosti kojima se postiže pretvorba, prijenos ili pohrana materijala ili energije. Operacije u proizvodnji odnosno industrijski procesi se općenito mogu dijeliti na diskretne (eng. *Discrete*), kontinuirane (eng. *Continuous*) ili šaržne (eng. *Batch*).

Potreba za normizacijom i dugogodišnja iskustva mnogih stručnjaka u polju industrijske automatizacije, dovela su do razvoja ISA-S88 standarda za upravljanje šaržnim procesima, opće poznatijeg kao S88 standarda. Ovaj standard je svojim značajkama i sastavom uvelike priznat kao moćan alat u savladavanju prepreka prilikom planiranja, projektiranja, izvedbe i održavanja šaržnih procesa. Standard nudi precizno definiranu terminologiju i standardizirane modele koji su svim korisnicaima šaržnih procesa lako shvatljivi.

A. *Fizički model*

Oprema je podijeljena po modulima pomoću S88 alata koji se naziva fizički model (eng. *Physical model*). Koristi za opisivanje materijalnih sredstava poduzeća i hijerahrijskih odnosa između različite opreme uključenje u proizvodnju šarža. Fizički model je sredstvo kojim se organizira i definira oprema korištena za upravljanje šaržnim procesom. Sastoji se od četiri glavne hijerarhijske razine prikazane na slici 1.



Slika 1. Fizički model ISA-S88 standarda

B. *Kontrolni modul*

Kontrolni modul (eng. ***C**ontrol **M**odule*, dalje *CM*) je temeljni element fizičkog modela kojeg S88.01 standard ga definira kao: „*Kontrolni modul je u pravilu jedan ili grupa senzora, aktuatora, ostalih kontrolnih modula, i povezane procesne opreme, koji sa stajališta upravljanja djeluje kao zaseban uređaj*“ [4]. Dakle, svaki kontrolni modul osigurava izravnu "poveznicu" sa procesom preko aktuatora i senzora te tako izvršava neke osnovne upravljačke funkcije.

Ovakva podjela opreme omogućuje jednostavan razvoj i standardizaciju programskih komponenata za upravljanje radom pripadajuće opreme koja služi za obavljanje šaržnog procesa. Kontrolni moduli, odnosno programske komponente, se standardiziraju prema najčešćim tipovima uređaja korištenih u industrijskim postrojenjima. Na taj način se jednom isprogramirana programska komponenta (npr. dvosmjerni ventil sa potvrdom položaja) može kasnije neograničeno primjenjivati u izradi programske podrške za razne šaržne procese u kojima se taj dio opreme koristi.

IV. RAZVOJ UPRAVLJAČKOG PROGRAMA

Razvoj upravljačkog programa počinje razradom procesnih dijagrama (eng. ***P**iping and **I**nstrumentation **D**iagram*, skraćeno *P&ID*) i detaljnom tehničkom specifikacijom svih komponenti procesnog sustava (ventili, motori, digitalni i analogni senzori, PID regulatori, pretvarači napona i frekvencije, itd). Nakon razrade, u posebno izrađenom *MS Excel* predlošku se upisuju podaci za svaku komponentu zasebno, koji su bitni za rad te komponente u sustavu.

Tako spremljene podatke jednostavnim postupkom učitamo u programski editor *RS Logix 5000*[1], u kojem se piše upravljački program *PAC* sustava distribucije cementa. Isti nazivi komponenti i tag-ovi kreirani u *MS Excel* tablici koriste se i u izradi nadzornog programa, odnosno SCADA sustava (eng. ***S**upervisory **C**ontrol **A**nd*

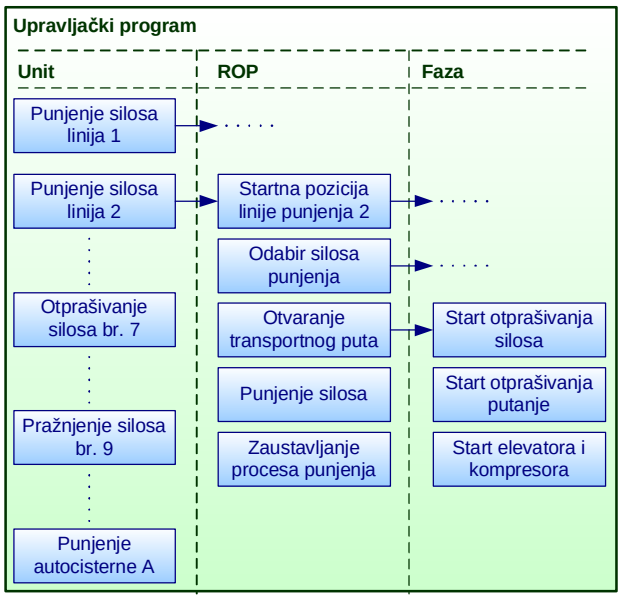
[1] Programsko okruženje za izradu programske podrške PAC uređaja serije *ControlLogix* tvrtke *Allen Bradley*

***D**ata **A**cquisition*, dalje *SCADA*), čime je postignuta apsolutna transparentnost u izradi programske podrške.

Nakon izrade liste komponenti, za svaku od komponenti postoji programski modul, odnosno gotov programski kod izrađen u jednom od jezika za programiranje PAC uređaja. Taj kod se konstantno ciklički izvodi za svaku komponentu koja je pozvana unutar upravljačkog programa. Na taj modularan način, nije potrebno pisati programsku podršku svaki put za pojedine komponente sustava, nego se programski kod prenosi iz projekta u projekt, a mijenja se samo tok programa i imena komponenti.

Tok programa određuju tehnolozi proizvodnog sustava, a opisuju ga tekstualno i tablično u tzv. *FDS* dokumentima (eng. ***F**unctionally **D**escription **S**oftware*, *FDS*). Prema njemu i prema procesu, programeri određuju najmanje samostalne programske cjeline (eng. *Unit*) za koje pojedinačno izrađuju potprograme i pozivaju prema potrebi (npr. punjenje silosa, kotao, filtar i sl.). Te cjeline, kao dijelovi procesa, sastoje se od jedne ili više komponenti, koje se u potprogramu pojedine cjeline pozivaju prema potrebi procesa. Više takvih cjelina koji se međusobno pozivaju i istovremeno i/ili zasebno izvode, čine upravljački program.

Slika 2. prikazuje blok shemu toka izrađenog upravljačkog programa sustava distribucije cementa primjenom S88 standarda.



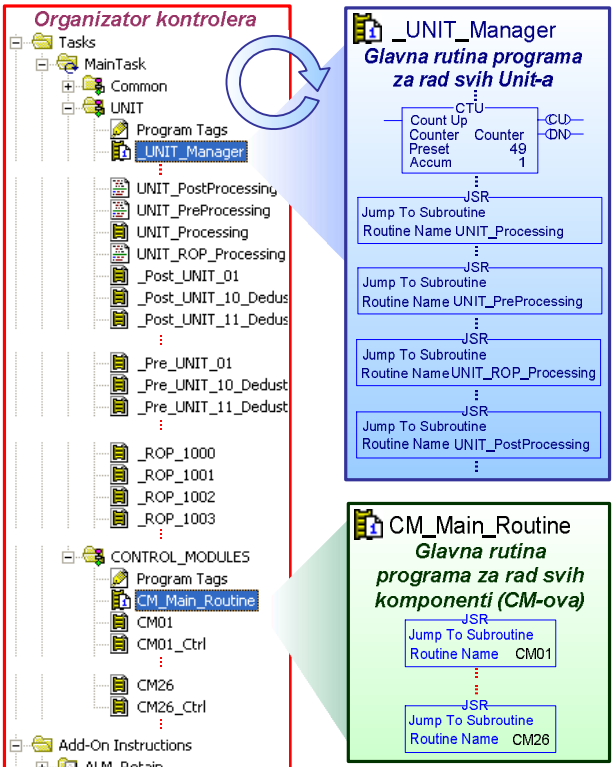
Slika 2. Blok shema toka izrađenog upravljačkog programa

Izrađeni upravljački program sustava distribucije cemeta ima ukupno 23 procesne cjeline koje su slične i ponavljaju se:

- *Punjenje silosa linija 1* te još takve dvije za liniju punjenja 2 i 3;
- *Otprašivanje silosa br. 6* i još takvih 7 po strukturi programskog koda identičnih za ostalih 7 silosa;
- *Pražnjenje silosa br. 6* i još takvih 7 identičnih strukturi programskog koda za ostalih 7 silosa;
- *Punjenje autocisterne A* i još 3 takva identična za punjenje autocisterni na utovarnim mjestima B, C i D.

Svaka cjelina ima svoju proceduru toka izvođenja koja se sastoji od jednog ili više *ROP*-ova (eng. ***R**ecipe **O**peration **P**rocedures*, *ROP*) koje se zasebno programiraju i izvode sekvencijalno prema receptu izvođenja danom od operatera nadzornog programa [5].

Upravljački program poziva zasebno svaki kontrolni modul i prema dobivenim statusnim varijablama upravlja izvršnim elementima procesa (Slika 3.).



Slika 3. Principna shema rada sustava na *Rockwell software* platformi prema S88 standardu

Problem koji je prisutan kod ovako napisanog upravljačkog programa je ukupno vrijeme izvršavanja cijelog ciklusa preko 260 ms. Iz razloga što upravljački program čita svoje ulaze samo na početku ciklusa ispada da se ulazi osježavaju samo 4 puta u sekundi što je izuzetno sporo. Teoretski se može dogoditi da pojedini signal traje npr. 200 ms, a vrijeme između dva čitanja ulaza je 260 ms, te je moguće da sustav uopće ne registrira taj signal.

V. PRIJEDLOG RJEŠENJA

Kao što je na početku spomenuto, PAC koji se koristi kao procesno računalo podržava IEC 61131-3 standard što znači da omogućava izvršavanje programskog koda kako u kontinuiranom vremenskom procesu tako i od korisnika definiranim procesima.

Postoje dvije vrste takvih procesa: periodički (eng. *Periodic*) koji se izvršavaju u zadanim vremenskim intervalima i procesi potaknuti događajem (eng. *Event*) koji se aktiviraju na promjenu stanja sklopovske opreme ili programskih varijabli.

Prijedlog rješenja opisanih nedostataka uključuje definiranje nove strukture programa gdje se upravo u

periodičkim procesima pokreću kritične rutine. Samo periodično pokretanje odabranih CM-ova nije dovoljno jer u polju ulazno-izlazne memorijske procesne mape *AB_I[]* i *AB_Q[]* u tom trenutku nisu ažurni podaci pa je njih potrebno osvježiti. Da bi se zadovoljili ovi uvjeti napisane su nove rutine koje osvježavaju polje procesne mape prije pokretanje kritičnih CM-ova.

A. *Rutine*

Za svaki CM napisana je vlastita rutinu za osvježavanje podataka nazvana *CMxx_RefreshData* koja je prikazana slikom 4. Ona provjerava sve upotrijebljene CM-ove koristeći status *CMxx_CONFIG[n].CM_Enable* te čita ulazne signale i zapisuje izlazne.

Ulazni parametar rutine *CMxx_RefreshData* je modus rada *_RefreshMode* koji određuje što će se odraditi:

- 1 – čitanje podataka sa periferije PAC i spremanje u *AB_I[x,y]*
- 2 – čitanje iz *AB_Q[x,y]* i zapisivanje na periferiju PAC-a
- 3 – obadvije navedene operacije u jednom prolazu

Sva svojstva CM-a imaju filtar upotrebljenosti *IOpar_<name>Exist* i ne osvježavaju se ako se ne upotrebljavaju da bi se maksimalno smanjilo vrijeme potrebno za izvršavanje rutine. Za čitanje sa perifernog sklopovlja procesnog računala koriste se rutine *Get_BitFromSlot* i *Get_AnalogFromSlot*.

Za digitalne signale koristi se rutina *Get_BitFromSlot* a njezini ulazni parametri su sljedeći:

- *_SlotNumber* – definira s kojega će se slota čitati signal
- *_ByteNumber* – definira se adresa kartice ako se koristi distribuirana periferija
- *_BitNumber* – definira koji bit predstavlja željeni signal

Za analogne signale koristi se rutina *Get_AnalogFromSlot* a njezini ulazni parametri su sljedeći:

- *_SlotNumber* – definira s kojega će se slota čitati signal
- *_ByteNumber* – definira se kanal analoge kartice

Analogno tome za zapis na izlazno sklopovlje PAC-a koriste se rutine *Set_BitOnSlot* i *Set_AnalogOnSlot* koje imaju identične ulazne parametre kao i rutine za čitanje.

TABLICA I. POPIS NAPISANIH PROGRAMSKIH RUTINA

Red. br.	Rutina	CM tip
1	*CM01_RefreshData*	Valve On/Off (air/spring)
2	*CM02_RefreshData*	Motor/pump
3	*CM04_RefreshData*	Motor variable speed
4	*CM05_RefreshData*	Proportional valve
5	*CM06_RefreshData*	Digital Switch
6	*CM07_RefreshData*	Motor 2 speed
7	*CM08_RefreshData*	Motor 2 directions
8	*CM09_RefreshData*	Analog measurement
9	*CM13_RefreshData*	Valve double solenoid (air/air)
10	*CM14_RefreshData*	Motor Variable Speed 2 dir
11	*CM15_RefreshData*	Two way flap
12	*CM16_RefreshData*	Totalizer
13	*CM17_RefreshData*	Digital output
14	*CM25_RefreshData*	Motorized 2 way flap
15	*CM26_RefreshData*	Two way flap with middle position

Gore opisane rutine su univerzalne i mogu se koristi u svim projektima ne narušavajući opisani način generiranja programskog koda. Rutine za pristup sklopovskoj opremi *Get_BitFromSlot*, *SetBitOnSlot*, *GetAnalogFromSlot* i *SetAnalogOnSlot* moraju se pisati posebno za svaku sklopovsku konfiguraciju. Ovo je zbog toga što razvojno okruženje *RSLogix 5000* ne dopušta simbolično adresiranje periferije nego se ono mora egzaktno pozivati u kodu.

B. *Izbor kritičnih zadataka i definiranje procesa*

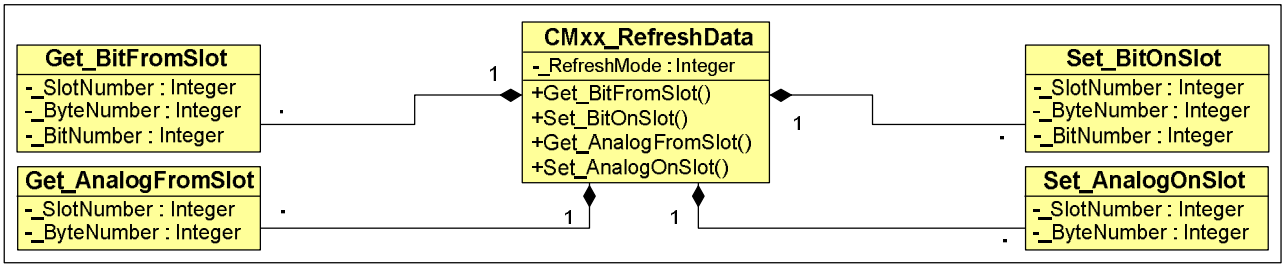
1. Isklop u nuždi

Jedan od kritičnih zadataka je detekcija zahtjeva za isklop u nuždi koji se obrađuje s kontrolnim modulom *CM06:Digital Switch*. Kako bi se zaštitili životi i imovina, a u skladu s IEC normama trajanja napona dodira, sustav u najgorem slučaju mora reagirati 70ms nakon što je stisnut taster isklopa u nuždi (eng. *Safety switch*). Iz tog zahtjeva proizlazi da se CM06 mora aktivirati najmanje svakih 70ms pa slijedi:

$$C = C_{CM06\_Re} + C_{CM06} = 889 + 35140 = 36029$$

$$T = 70000\mu s$$

$$U = \frac{C}{T} = \frac{36029}{70000} = 0{,}5147$$



Slika 4. Blok shema *CMxx_RefreshData* rutine

2. Klapna sa srednjim položajem

Tijekom rada postrojenja kritične su se pokazale klapne sa srednjim položajem. Radi se o elektromehaničkom sustavu pokretanom elektromotorom čiji se radni status dojavljuje pomoću tri senzora položaja (početni, srednji i krajnji). Kako bi se osigurala sigurna detekcija položaja klapne period procesa koji čita njezin status mora biti manji od vremena prolaska preko senzora položaja. Na taj način osiguravamo da dva puta detektiramo aktivan senzor položaja. Iz mehaničkih karakteristika sustava smo izračunali da je vrijeme kada je klapna na senzorima položaja 110ms. Iz ovoga proizlazi da se očitanje položaja kontrolnim modulom *CM26 - Two way flap with middle position* mora periodično izvršavati svakih 100 ms pa slijedi:

$$C = C_{CM\,26\_\mathrm{Re}} + C_{CM\,26} = 1950 + 7144 = 9094$$

$$T = 100000$$

$$U = \frac{C}{T} = \frac{9094}{100000} = 0{,}09094$$

3. Provjera raporedivosti po Liu & Layland

Nakon što smo definirani zahtjeve za periodične procese provjeriti ćemo njihovu rasporedivost odnosno mogućnost da se procesi korektno izvršavaju.

$$\sum_{i=1}^{N}\left(\frac{C_i}{T_i}\right) \le N \cdot \left(2^{\frac{1}{N}} - 1\right)$$

$$\sum_{i=1}^{2}\left(\frac{36029}{70000}, \frac{09094}{100000}\right) \le 2 \cdot \left(2^{\frac{1}{2}} - 1\right)$$

$$0{,}5147 + 0{,}09094 \le 2 \cdot (1{,}41421 - 1)$$

$$0{,}60564 \le 82842$$

Test je pozitivan što znači da ova dva kritična zadatka možemo potjerati u periodičkim procesima a prioritete ćemo odrediti monotono po učestalosti:

- Proces 1: CM06 svakih 70ms, prioritet 5
- Proces 2: CM26 svakih 100ms , prioritet 6

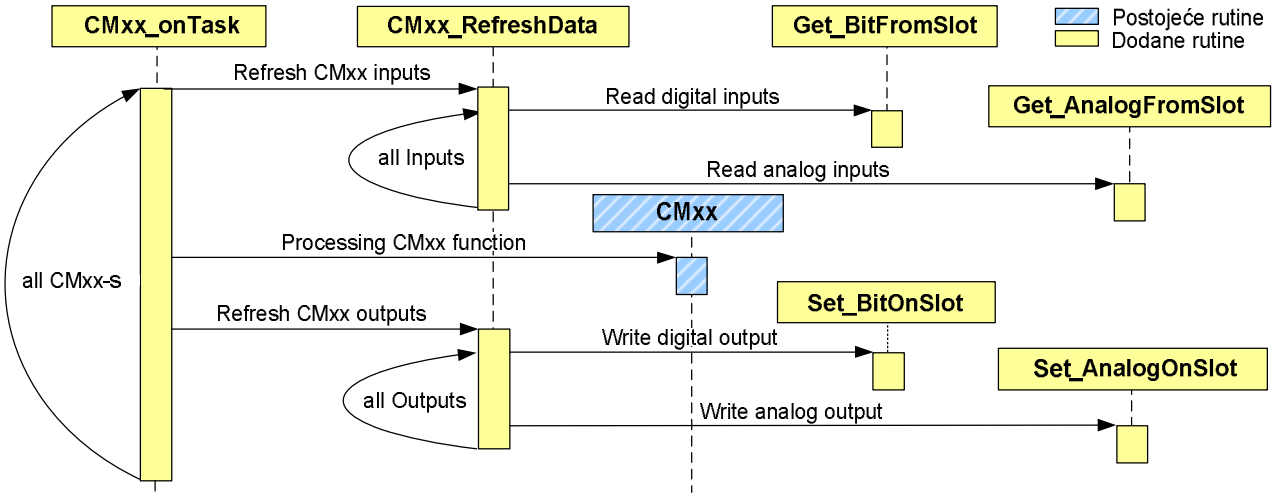
C. Implementacija

Implementacija opisanih rutina u program je relativno jednostavna. U početku u stablastu strukturu *Tasks* dodamo nove periodičke procese *PeriodicalTask_70ms i PeriodicalTask_100ms* te im dodijelimo prioritete. U njima su rutine *CM_OnTaskMain* za poziv kritičnih CM-ova. U njemu pozivam odabrane kritične rutine *CM06_onTask i CM26_onTask* na isti način kao i u kontinuiranom tasku *CONTROL_MODULES/ CM_Main_Routine*. Spomenute rutine osvježavaju podatke prije i nakon poziva originalnog CM-a i tako osiguravaju aktualna ulazna i izlazna stanja industrijskog postrojenja (Slika 5.). Ovdje pozvane odabrane komponente ne pozivamo u kontinuiranom procesu.

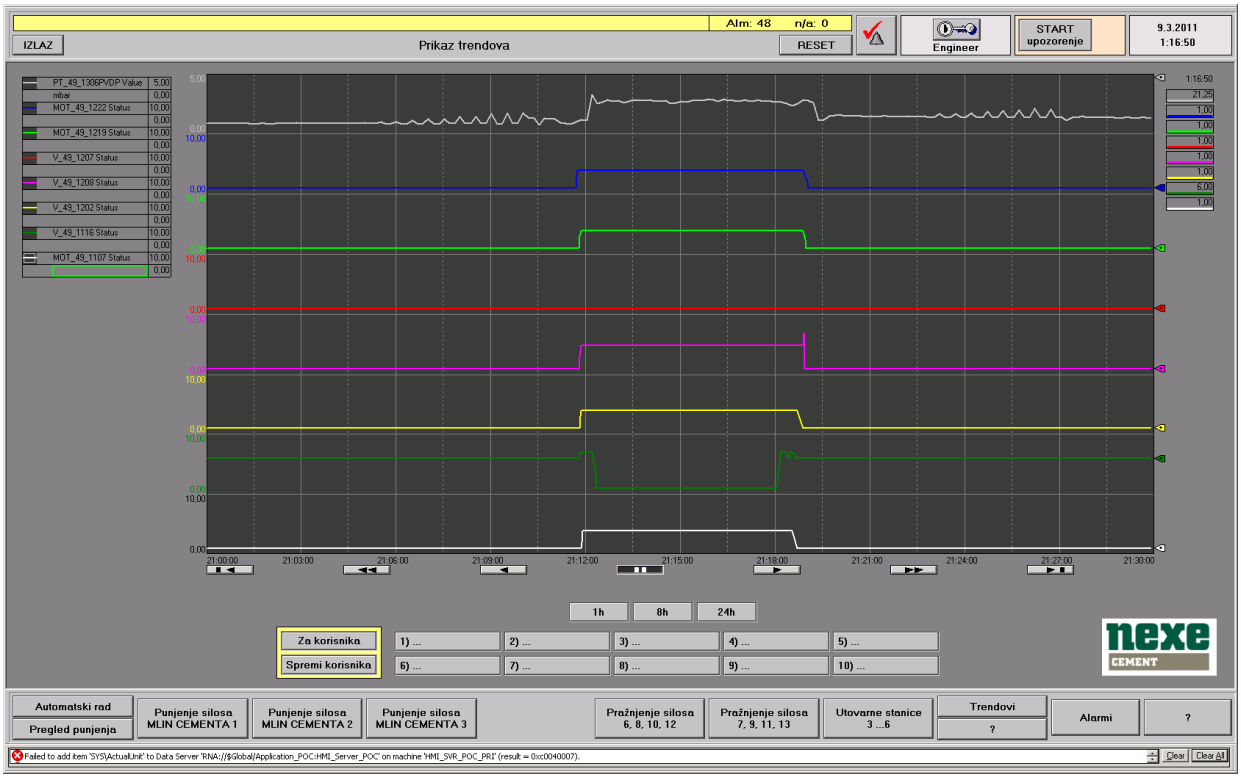
VI. EKSPLOATACIJSKE KARATKERISTIKE

Kao primjer eksploatacijskih karakteristika dan je trend koji prikazuje postupak pokretanja i zaustavljanja komponenti transportnog sustava kod punjenja autocistene preko mobilnog punjača (Slika 6.). Uvjet da se autocisterna počne puniti je da u silosu ima dovoljno cementa prema količini cementa koja je na vagi zadana za istovar i da sustav razrahljivanja cementa silosa iz kojega vršimo punjenje radi. Kako se količina utovarenoga cementa približava zadanoj, zadnjih 700 kg pneumatska zaklopka se ponovo zatvara na 50% tako da se tok cementa uspori. Kada vaga pokaže da preostaje još 200kg, pneumatska zaklopka se potpuno zatvara i preostala količina isteče u autocisternu. Točnost sa kojom se računa je ±40 kg.

Ključno kod ovoga dijela procesa je da se pokretanje pojedinih motora za uzdužno, poprečno i vertikalno gibanje punjača autocisterni vrši preko kutije sa tipkalima. Između ostaloga, do rješavanja ove problematike je upravo došlo iz razloga što sustav nije svaki puta mogao ustanoviti i pravovremeno reagirati na pritisak tipkala, te je u najgorem slučaju radnik na punjaču morao držati tipkalo pritisnuto dulje od pola sekunde. Naravno, takva situacija je nedopustiva. Kasnije se iz tog problema pokazalo da je sustav manjkav u pogledu brzine izvršavanja i u radu sa ostalim komponentama koje su vremenski kritične. Primjenom periodičkog izvršavanja pojedinih komponenata ova se problematika u potpunosti uklonila.



Slika 5. Dijagram izvršavanja *CMxx_onTask* rutine



Slika 6. Pokretanje i zaustavljanje komponenti transportnog sustava kod punjenja autocistene preko mobilnog punjača

VII. ZAKLJUČAK

Uvođenjem i razvojem ISA-88.01-1995 standarda za izradu programske podrške šaržnih procesa dobila se ujednačena struktura upravljačkih programa koja je standardizacijom pružila mogućnost jednoznačnog opisa i izrade. Možda najvažniji dio za inženjere u praksi je vrlo jednostavno prepravljanje postojećeg upravljačkog programa i njegova nadogradnja prema jasnom obrascu koji je uvjetovan S88 standardom. Prikazani sustav je temeljen na unaprijed isprogramiranim komponentama koje se nalaze u upravljačkom programu i unaprijed izvedenim komponentama *SCADA* sustava koje se prilikom izgradnje aplikacije za zadani proces povezuju u cjelinu.

Zbog sporosti rada računalnog sustava, koji nije bio u stanju registrirati pojedine kritične signale u trajanju i do pola sekunde, pristupilo se promjeni upravljačkog programa tako da se se pojedine vremenski kritične komponente izvode u posebno definiranim cikličkim procesima. Pri tome su se prije primjene definirali zahtjevi te izabrale frekvencije procesa kritičnih zadataka. Ograničenja primjene ovakvih rješenja je mogućnost raspoređivanja pa je neophodno testiranje istoga. Nakon izvršene provjere rasporedivosti (metoda *Liu & Layland*) zaključeno je da je ona moguća. Potom su napisane nove programske rutine koje su se uspješno implementirale u postojeće programsko rješenje i tako riješile problem.

Naravno, ovdje se može postaviti i teoretsko pitanje što ako je veća količina komponenti kritična u pogledu vremena izvršavanja tako da procesor ne može izvršiti kod u zadanom periodu odnosno ciklusu pozivanja. Ukoliko bi došlo do toga, moralo bi se nabaviti brži procesor, što je opet relativno loš odabir ukoliko se sustav planira proširivati, ili bi se nabavila dva procesora koji bi djelili komponente. Svakako, uvijek ostaje mogućnost optimiranja koda na uštrb gubljenja pojedinih mogućnosti komponente (simulacijski mod rada, testni mod rada).

Izrada ovakvih aplikacija pokazuje kako je za automatizaciju procesa potrebno poznavati i upotrijebiti znanja iz raznih područja tehničke struke.

LITERATURA

[1] G. Malčić: *Automatizacija sustava distribucije cementa primjenom S88 standarda*, Magistarski rad, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Zagreb, 2011.

[2] J. Radej: *Sustavi za rad u stvarnom vremu*, skripta s predavanja, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Zagreb, 2009.

[3] National Instruments: *PACs for Industrial Control, the Future of Control*, Tutorial, Feb 28, 2012.

[4] M. Barker, J. Rawtani: *Practical Batch Process Management*, ISBN: 0-7506-6277-8, Newnes, Elsevier Science & Technology Books, 2004.

[5] D. Koren, Z. Himelreich, G. Malčić, L. Padovan: *Upravljanje sustavom distribucije cementa*, Computers in Technical Systems (CTS), MIPRO 2009.

[6] D. Milićev: *Programiranje u realnom vremenu*, skripta, Beograd, 2002.

[7] ***: Logix5000 Controllers Execution Time and Memory Use, Reference manual, Rockwell Software, Publication 1756-RM087J-EN-P - August 2010.

[8] ***: Logix5000 Controllers IEC 61131-3 Compliance , Programming, Rockwell Software, Publication 1756-PM018B-EN-P - July 2008