INDUSTRIJSKI MIKRORAČUNALNIŠKI KRMILNIK

M. Colnarič, M.Gerkeš, J.Györkös, P.Kokol, K.Rizman, I.Rozman, B.Vukelič, A.Zorman, V.Žumer

Ključne besede: mikroračunalniški krmilniki, industrijski krmilniki, mikroprocesor, večprocesorski sistemi, sistemska programska oprema, razvojna programska oprema

POVZETEK: V članku je podan pregled sistemske programske opreme in programskega okolja industrijskega mikroračunalniškega krmilnika. Namesto klasičnega pristopa - zamenjave relejske logike z mikroračunalnikom - smo razvili delovno postajo, ki uporabniku omogoča razvoj aplikacije na osebnem računalniku s pomočjo grafičnega in mnemoničnega zapisa, sodobnih orodij, kot so simulacije v nadrejenem sistemu ali v objektu in podobno.

INDUSTRIAL MICROCOMPUTER CONTROLLER

Key Words: microcomputer controller, industrial controller, microprocessor, multiprocessor systems, system software, development software

ABSTRACT: In the article, a review of system software and programming environment of the industrial microcomputer controller is given. Instead of the conventional approach based on the simple substitution of relay and contactor logic with the microprocessor, we designed an engineering workstation, which anables the user to develop the application on a personal computer, using mnemonic and/or graphic notation, modern tools like on-and offline simulation etc.

1. UVOD

Mikroprocesorji tudi pri nas vse bolj prodirajo v proizvodne hale. Na začetku so bili to sistemi, razviti posebej za določene aplikacije, kar pomeni, da so bili relativno dragi, saj so predstavljali unikate. Pozneje so začeli razvijati univerzalno aparaturno opremo, ki so jo programirali ročno, v zbirnem jeziku, kvečjemu s pomočjo operacijskega sistema in knjižnic rutin za delo v realnem času.

Število aplikacij, podprtih z mikroračunalnikom, raste iz dneva v dan. Vse bolj se pojavlja potreba ne le po univerzalnem krmilniku, temveč tudi po okolju, ki bo omogočalo enostavno in hitro programiranje s čim manj možnosti vnosa napak v sistem. Da bi ustregli tem zahtevam, smo se na Tehniški fakulteti, VTO Elektrotehnika, računalništvo in informatika v sodelovanju z Metalno, Maribor, lotili programske opreme za sodobni industrijski krmilnik.

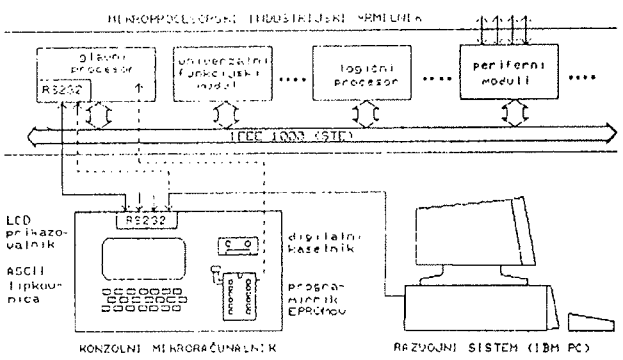
2. ARHITEKTURA SISTEMA IN APARATURNA OPREMA

Programska oprema, ki smo jo razvili, je namenjena za krmilnik, katerega aparaturno opremo je izdelala Metalna. Za lažje razumevanje delovanja podajamo njen kratek opis.

Arhitektura sistema je prikazana na sliki 1. Sistem je sestavljen iz

* samega krmilnika
* konzolnega računalnika, ki omogoča uporabniku komunikacijo s krmilnikom in njegovo aplikacijo, shranjevanje podatkov na digitalno kaseto, programiranje EPROM vezij in omejene funkcije razvojnega sistema. Izdelan je na osnovi M68008 in je bil prav tako razvit v Metalni.
* razvojnega sistema; to je računalnik, kompatibilen IBM PC XT ali AT, na katerem teče uporabnikovo programirno okolje.

Osnova krmilnikove aparaturne opreme je vodilo STE (IEEE 1000). Sistem je koncipiran kot večprocesorski. Poleg glavnega procesorja imamo na vodilu še univerzalne funkcijske module (prav tako na osnovi M68008), ki izvajajo različne funkcije, kot so števci, časovniki (timerji), registri, programatorji in podobno), ter logične procesorje, ki izvajajo logične funkcije. Te predstavljajo večino programa, zato uporaba logičnega procesorja, ki je zgrajen diskretno, bistveno pospeši delovanje sistema.



Slika 1: Zgradba sistema

Za aparaturno opremo je značilna relativno uniformna struktura, ki je bila dosežena z abstraktnim modeliranjem strukture, funkcij in kontrole sistema. Ta način gradnje omogoča večjo zanesljivost, saj so skupne lastnosti večkrat testirane na abstraktnem modelu. Načrtovanje posameznih modulov se tako prevede na izdvajanje posebnih funkcij iz splošnega modela. V realizaciji so bile uporabljene VLSI, ASIC in integrirane močnostne komponente ter podatkovni in signalni pretvorniki. Na ta način je bilo možno vgraditi vse funkcije v 19-palčno ohišje.

3. SISTEMSKA PROGRAMSKA OPREMA

Sistemska programska oprema je na prvi pogled neznačilna za sistem vodenja procesov v realnem času. Izdelana je na ideji, da je v takšnem sistemu vse vnaprej definirano in da prestavljajo izjeme le napake in izredna stanja. Zaradi tega imamo v osnovi en sam proces, prožen s pomočjo ure, ki daje osnovni ciklus aplikacije. Znotraj tega procesa tipamo vsa predvidena stanja in jih po potrebi obdelujemo.

Osnovni ciklus glavnega procesa je sestavljen iz aplikacijskega dela, ki ga uvede, proži in zaključi sistemski del. Ta na začetku cikla prečita vse aktualne vhodne podatke. Nad njimi operirajo aplikacijski del in pomožni paralelni procesorji, ki jih le-ta proži in ki prek svojih funkcij nastavljajo ustrezne izhodne spremenljivke. Te sistemski del na koncu ciklusa vpiše na izhodne. Neposrednega dostopa do vhodov in izhodov aplikaciji ne dovolimo iz bojazni do nestabilnosti in možnih rekurzivnih pojavov ter oscilacij.

Od zaključka dela v posamezni iteraciji do začetka naslednjega ciklusa mora ostati še nekaj časa. Če tega časa ne bi bilo, bi pomenilo, da je sistem preobremenjen, oz. da je nastavljen prekratek čas za ponovitev zanke. V tej performančni rezervi se izvajajo časovno manj pomembne funkcije. Prva med njimi je spooling - enkratne dele aplikacij, ki časovno niso vezani na proces, kot so na primer izpisi, arhiviranje ipd. izločimo iz nje in jih izvedemo, ko je procesor prost. V ta namen se formira posebna čakalna vrsta, v kateri takšni procesi čakajo na obdelavo.

Kadar je tudi ta vrsta prazna, teče v performančni rezervi minimalni monitor. Ta omogoča pregled in vpis registrov in pomnilnika, nalaganje programov v RAM prek serijske zanke, njihovo testiranje (debugging), postavljanje prekinitvenih točk, koračno izvajanje, prikaz in forsiranje vhodov in izhodov za testiranje aplikacije ter nekaj nevidnih funkcij, ki omogočajo prenos informacij za izvajanje simulacije v objektu v povezavi z nadrejenim sistemom.

Poleg teh so v sistemski programski opremi še nekatere transparentne funkcije, potrebne za delovanje sistema. Ena izmed njih je upravljanje s pomnilnikom. To podpira aparaturno opremo, vgrajeno na procesorskem modulu in prikazano na sliki 2. Osnova upravljanja s pomnilnikom je tabela, vpisana v posebnem hitrem RAM pomnilniku, prek katerega se preslikajo vsi naslovi, ki jih generira procesor. V tej tabeli so podatki o atributih pomnilniških blokov ter o njihovih logičnih naslovih. Ob prihodu logičnega naslova s procesorja vezje preveri privilegij dostopa do ustreznega bloka ter generira fizični naslov ali javi napako naslavljanja.

Naslednja transparentna funkcija je upravljanje z vhodno-izhodnimi napravami. V pomnilniku je shranjena slika vhodov ter dve sliki izhodov. Ustrezna funkcija ob začetku aplikacije preslika vhode v njihovo pomnilniško sliko, izhode pa iz slike aktualnih izhodov na fizične. Hkrati zamenja sliki izhodov, tako da imamo vedno sliko izhodov iz prejšnje iteracije in sliko, v katero aplikacija in paralelni procesorji vpisujejo svoje rezultate v tekoči iteraciji. Sliki vhodov in prejšnjega stanja izhodov sta relevantni za izračun izhodov v novi iteraciji.

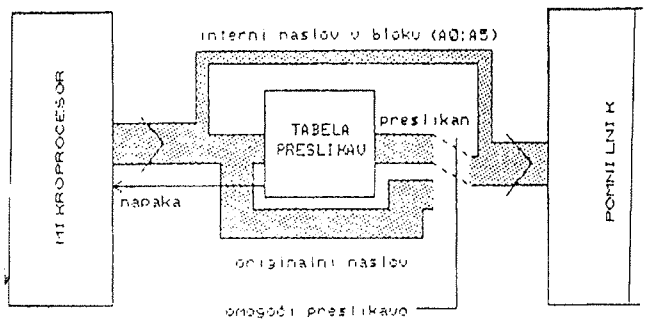
4. OKOLJE ZA RAZVOJ APLIKACIJE

Programska oprema za razvoj aplikacije je izdelana v PASCAL-u in lahko paralelno teče na razvojnem sistemu, ki ga predstavlja IBM PC XT/AT ali kompatibilni računalnik ter na konzolnem računalniku, razvitem v okviru tega projekta. Funkcije, ki so implementirane na konzolnem računalniku, so omejene zaradi njegove narave (nepopolna tipkovnica, LCD prikazovalnik itd.).

Okolje je povezano v celoto s povezovalnim programom, ki je izdelan na principu drevesnih menijev. Izberemo lahko operacije s knjižnico, grafični editor kontaktnih shem, mnemonični editor, grafični Grafcet editor, prevajalnik, generator objektne kode, simulator in prenos podatkov.

Knjižnica je razdeljena na tri dele: delovno področje, glavna knjižnica in uporabniška knjižnica. Nad datotekami lahko izvajamo operacije, kot so brisanje, kopiranje, preimenovanje, iskanje in izpis posameznih aplikacij ali njihovih delov.

Aplikacije lahko pišemo v standardni grafični obliki relejnih shem, razširjeni s funkcijami, ki je je vajena večina aplikacijskih programerjev, ali pa v posebnem jeziku



Slika 2: Upravljanje s pomnilnikom

IMCL, ki je bil razvit v okviru tega projekta in ki omogoča programiranje kompleksnejših aplikacij. Gre za jezik, ki podpira sodobne koncepte strukturnega in modularnega programiranja, tanjšanje (stepwise refinement) ter navzdoljnje in navzgornje programiranje (top-down in bottom-up design). Program je sestavljen iz:

* deklaracijskega modula, ki vsebuje podatke o funkcijskih modulih (časovniki, števci, programatorji...), vrednosti konstant, definicijo okolja itd.,
* glavnega programa, najvišjega nivoja aplikacije,
* podprogramov, mehanizmov preprostejših enot,
* funkcij, pravil za kompleksnejše obdelave,
* sekvenc, pogojnega izvajanja

Generatorju objektne kode je vhod vmesna koda, ki jo dobi iz prevoda relejnih shem ali programa v IMCL. Program predela v binarno kodo, primerno za vpis v EPROM vezja ali za prenos v RAM v krmilniku. Glede na to, ali so prisotni paralelni koprocesorji, jih program vključi, ali pa vključi njihove simulacijske podprograme v glavnem procesorskem modulu.

Za hiter in preprost razvoj aplikacij je v okolje vključena tudi simulacija. Ta omogoča preizkus delovanja aplikacije že v delovni postaji in tako zmanjšuje čas in stroške, potrebne za implementacijo. Uporabnik podaja svojo aplikacijo v grafični ali mnemonični obliki. Simulator s svojo drevesno menijsko strukturo omogoča nadzorovano izvajanje programa z opcijami več ciklov, en cikel, izvajanje po korakih ali zvezno delovanje s prekinitvenimi točkami ali brez. Podatke lahko prikazuje na zaslonu ali tiskalniku, spreminjamo pa jih lahko z dvema namenskima editorjema.

Ko je aplikacija gotova, lahko generirano kodo prenesemo v RAM krmilnika, kjer jo lahko preizkusimo v objektu, ali pa jo vpišemo v EPROM vezja v konzolnem računalniku, ko smo v delovanje res prepričani.

5. ZAKLJUČEK

Opisani krmilnik je že prestal nekaj implementacij na precej zahtevnih objektih. Pokazalo se je, da so zmogljivosti sistema zadovoljive, da pa je zaradi kompleksnosti razvojne programske opreme njegova šibka točka počasnost prevajalnikov in generatorja objektne kode. Zaradi tega bo v prihodnjih verzijah treba razmisliti o drugačnih rešitvah v okolju za razvoj aplikacije, še posebej pri prevajanju ter generiranju objektne kode.

6. LITERATURA

1. M.Colnarič, M.Gerkeš, I.Rozman, B.Stiglic, M.Kolarič: Nadzorni program za industrijski mikroračunalniški krmilnik, MIPRO87, Opatija, maj 87, pp.4-78/4-82

2. J.Gyorkos, M.Gerkeš, I.Rozman, V.Žumer, P.Kokol, B.Vukelič: Generiranje objektne kode mnemoničnega jezika 16-bitnega industrijskega krmilnika za mikroprocesor MC68000, MIPRO87, Opatija, maj 87, pp.4-73 / 4-76

3. P.Kokol, M.Mernik, V.Žumer, I.Rozman, B.Vukelič: IMCL - Industrial Microcomputer Controller Language, ICS, Taipei University, Taiwan, 15.-17.12.1988

mag. Matjaž COLNARIČ, dipl.ing.
Jozsef GYÖRKÖS, dipl.ing.
mag. Peter KOKOL,dipl.ing.
Krista RIZMAN, dipl.ing.
dr. Ivan ROZMAN, dipl.ing.
Brane VUKELIČ, dipl.ing.
Anton ZORMAN, dipl.ing.
dr. Viljem ŽUMER, dipl.ing.

vsi Univerza v Mariboru
Tehniška fakulteta,
VTO Elektrotehnika, računalništvo in informatika
Smetanova 17, 62000 MARIBOR
dr. Maksimiljan GERKEŠ, dipl.ing.,
METALNA Maribor,
Zagrebška 20

Prispelo: 23.11.1988 Sprejeto: 9.4.1989