Програмна и апаратна реализация на контролер за дискретното производство

маг. инж. Иван Ивайлов Димитров
(тел. 02 965-23-12, E-mail: ivand@vmei.acad.bg)
доц. д-р инж. Георги Славчев Михов
(тел. 02 965-32-81, E-mail: gsm@vmei.acad.bg)

Технически университет - София

Software and Hardware Architecture of a Controller for Discrete Manufacturing. In this paper the problem for the hardware components that are included in one controller is discussed. An analysis of existing systems for discrete manufacture control has been made as well as their structure, classification and features have been discussed. Using this analysis a structure of a microprocessor based industrial controller is proposed. It offers some advantages due to the specisfics of discrete manufacturing. The main features of the offered structure are: the controller contains an internal network used for communication with other slave controllers or intelligent sensors, the controller has the capability of connecting into a galvanic decupled industrial local area networks together with other controllers or central control computer in a low connected system. Also the hardware means for safety, diagnostics and self-test are taken into account. The aspects for developing the system software are discussed. These are as follows: service of the hardware equipment and drivers working in single or multitasking environment (preemptive and non preemptive dispatching).

I. Въведение.

Контролерите за дискретно производство, наричани още програмируеми логически контролери (PLC), се използват за управление на машини и дискретни технологични процеси. По същество това са програмно управлявани електронни системи, проектирани за използване в качеството на промишлено оборудване за логическо управление на машини, съоръжения и технологични процеси, посредством цифрови или аналогови входове и изходи. Базирани върху стандартна микропроцесорна архитектура, те позволяват лесно обслужване и програмиране. Програмното осигуряване (ПО) на контролера има за основна задача, при активирането на входен сигнал от логическо или аналогово входно устройство, да изработва съответстващ отговор – активиране на съответни изходи. Освен нея, ПО изпълнява и някой допълнителни задачи, свързани с диагностика, обработка на данни и др. Диагностиката се заключава в анализ на входящата информация, на изходите и самодиагностика. Входните сигнали се проверяват логически и параметрично. Изходите също се следят параметрично, като по този начин от една страна се предпазват от претоварване, а от друга се следи за повреда в изпълнителните механизми. Самодиагностиката е важен елемент от методите и средствата за повишаване на надеждността и отказоустойчивостта и включва диагностика както на апаратната част на контролера, така и на програмното му осигуряване. Задачите на ПО свързани с съби-

рането и обработката на данни са насочени основно към статистически и количествен анализ на обекта на управление и/или технологичния процес. Обикновено те или се предават по локална мрежа към централизирана система за събиране и обработка на информация, или се визуализират на дисплея на контролера.

процедурите и функциите на ПО се реализират като подпрограми, които не могат да бъдат прекъсвани – примитиви. Когато в примитивите се използват локални данни, те трябва да се реализират като повторно входими (т.нар. реентарни). В случаите, когато се използват общи данни или апаратни ресурси, примитивите трябва да се реализират и като критични секции – с програмни или технически средства. Разпространен подход е по време на изпълнение на примитивите да се забранят прекъсванията, за да се осигури непрекъснат интервал на работа. При програмното реализиране на подхода съществуват две важни особености. Първата е свързана с това дали прекъсването е било разрешено преди повикването на примитива. Ако не е било разрешено, след приключване на задачата то трябва да остане непроменено (забранено). В противен случай, то трябва да се забрани преди критичната секция и да се възстанови (разреши) след това. Апаратното реализиране на подхода се осъществява чрез извикване на програмно прекъсване. В този случай като особеност може да се посочи различният механизъм за извикване на процедурите както и за предаване на параметри. Процедурите се указват чрез индекс (динамично свързване), а аргументите се предават чрез регистрите на процесора. Удобно е константните аргументи да се разполагат в областта на програмния код, непосредствено след инструкцията за прекъсване. Типичен пример за тези примитиви са функциите свързани с преиндексирането и управлението на входовете и изходите.

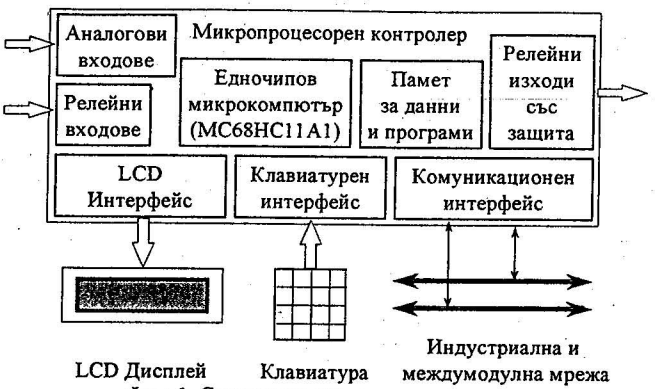
II. Постановка на задачата.

Поставяме за цел синтезирането на микропроцесорен контролер предназначен за нуждите на дискретното производство. Предвиждаме да разполага със следните ресурси: 8 релейни входа, 8 аналогови входа, 16 релейни изхода с електронна защита от претоварване, интерфейс за включване на допълнителни (отдалечени) входове и изходи, интерфейс за изграждане на разпределени системи за управление, локален дисплей и клавиатура, полупостоянна памет от тип EEPROM предназначена за нуждите на гъвкавото препрограмиране, микропроцесорна система изградена с MC68HC11, вградени средства за диагностика на входовете и изходите, библиотека с подпрограми, обслужващи ресурсите на контролера.

III. Апаратна реализация.

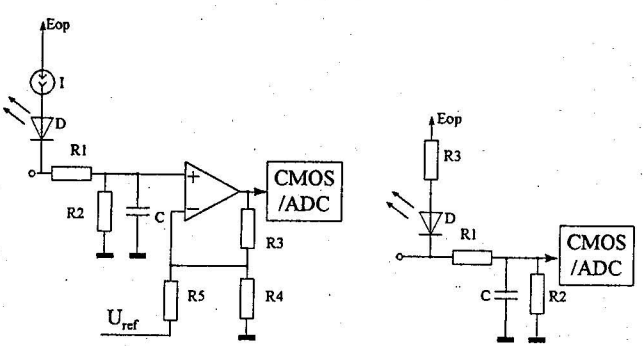
Структурната схема на контролера е показана на фигура 1. Контролера е изграден върху микропроцесорна система, състояща се от едночипов микрокомпютър (MC68HC11), постоянна памет EPROM с обем 16K и оперативна памет 32K. Предвидена е възможност за включване на две памети от тип EEPROM, със сериен достъп – съответно I^2C и SPI. За връзка с оператор е предвидена

течнокристална индикация и матрична клавиатура с 16 клавиша. Предвидени са две средства за връзка – индустриална и между модулна. Индустриалната е предвидена за връзки от вертикален тип (с централен компютър, концентратор и т.н.). За нея е характерно: галванична изолация, повишена шумоустойчивост, ниска скорост на обмена.



Фиг. 1. Структурна схема на контролера.

Междумодулната връзка служи за изграждане на хоризонталния слой (интелигентни датчици, отдалечени входове и изходи, подобни контролери и др.). Характеризира се с висока скорост на обмена, покрива къси разстояния, не е галванично изолирана, понеже се предполага, че служи за изграждане на силно свързана система в рамките на един агрегат. Контролера притежава 8 еднотипни аналогови и 8 еднотипни релейни входа (фиг. 2). Аналоговите входове са с възможност за работа с температурен датчик "платина 100". В този случай резистора R2 не се монтира, резистора R1 служи за защита на схемата. Датчика "платина 100" се захранва от източника на ток I, а адитивната съставна се компенсира от опорното напрежение U_{ref}. Съществува възможност за конфигуриране на аналоговите входове като релейни. В този случай P5 и P4 не се монтират, и ОУ работи като повторител. И в двата режима входната RC група повишава шумоустойчивостта на контролера. Релейният вход (фиг. 2б) работи аналогично. Характерно и за двете схеми е, че релейния сигнал освен към цифров вход може да се подаде и към АЦП. Тази възможност покрива изискването за диагностиката на нивото на входните сигнали.

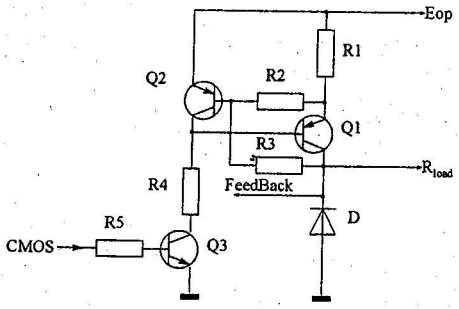


Фиг. 2. Входове на контролера, а) аналогов и б) релеен.

Реализирани са 16 еднотипни релейни изхода (фиг. 3) с защита от претоварване и предоставя товароспособност 24V/200mA. Схемата е така оразмерена, че при авария, резистора R5 да ограничи протичането на ток, който би повредил цифровата част. Диода на изхода служи за предпазване от отрицателни напрежения, възникващи при преходни процеси в индуктивен товар. Предвидена е обратна връзка за нуждите на диагностиката, която се подава на схема от типа показан на фиг. 2б. Благодарение на нея, контролера може да установи състоянието на товара и работоспособността на стъпалото.

IV. Програмно осигуряване.

Програмното осигуряване включва модули поддържащи апаратните ресурси на контролера, и библиотечни функции свързани с характерни елементи от работата на контролерите. Това са обекти (класове) включващи специфични структури от данни, съпроводени със съответни обслужващи функции (методи на класа).



Фиг. 3. Релеен изход на контролера с защита и обратна връзка.

Тъй като ПО се разработва предимно на процедурни езици (C, Асемблер), то ударението се поставя върху процедурите, чрез които се осъществява функционирането на алгоритъма. При това промяната на процедурите води веднага до съществени промени в структурата на данните. Процедурите са реализирани като примитиви, т.е. те са организирани като повторно входими и като критични секции, когато това е необходимо. Благодарение на това се дава възможност за работа в многозадачен режим, реализиран с различни механизми.

Повторната входимост се постига съответно за езика С по естествен път чрез вградените в езика средства – използването на автоматични променливи, и аналогично за асемблерните модули – чрез разполагането на променливи в стека. Типични за този тип примитиви са функциите за четене на служебна информация (напр. версия на ПО) и изчислителните функции.

Критични секции са реализирани като програмно се забраняват прекъсванията. За целта са реализирани два механизма. Първият се прилага при асемблерски модули с критично бързодействие. При него

Етикет	Мнемоника	Коментар
	...	Начало на критична секция
entry:	TPA	Прочитане на CCR(I флага)
	PSHA	Съхраняване на CCR
	SEI	Забрана за прекъсване
	...	Тяло на критичната секция
exit:	PULA	Прочитане CCR
	TAP	Възстановяване на CCR
	...	Край на критична секция

Фиг. 4. Реализиране на критична секция в асемблерна процедура.

непосредствено преди навлизане в критичната секция съдържанието на регистъра на състоянието CCR, включващ флага за маскиране на прекъсването

I, се запомня в стека. Съответно след края на критичната секция регистъра на състоянието се възстановява. Вторият механизъм е предвиден за С – функции. Реализиран е с помощта на две функции – съответно за четене и за установяване на флага за маскиране на прекъсването I.

При реализирането на критични секции съществува опасност от възникване на мъртва хватка (deadlock). Борбата в тази насока се обуславя от спецификата на процедурите на ПО – простота, висока ефективност, работа с "собствени" ресурси. Приложените стратегии за избягване на мъртва хватка са от тип "предпазване" и са свързани с предварителен анализ на възможните връзки между процедурите.

```
Етикет      Мнемоника     Коментар
GetIflag:   TPA           Прочитане на CCR(I флага)
            ANDA #16      Отделяне на флага I
            CLRB
            BEQ exit      Изход с 0
            LDAB #1       Изход с 1
            CLRA
exit:       RTS           Връщане от подпрограмата
SetIflag:   TSTB          Проверка на аргумента
            BEQ label1    Аргумент = 0
            SEI           Установяване на I
            BRA label2    Изход
label1:     CLI           Нулиране на I
label2:     RTS           Връщане от подпрограмата
```

Фиг. 5. Функции за четене и установяване на прекъсването.

За реализирането на някой системни функции е предложен абстрактния клас **Infinite**. Той включва единствен тип данни и няколко метода за работа с тях. Данните са дефинирани като тип **Inf**, който представлява обединение от стандартен двубайтов целочислен тип (**int**) и елемента ∞ (безкрайност). По този начин на **Inf** отговаря крайното изброимо множество {0, 1, ..., 65534, ∞}. Методите на класа са шест: прочитане и присвояване, събиране и изваждане, увеличаване и намаляване с единица. Възможно е да се включат и другите аритметични операции, но засега не е възникнала потребност от тях. Основна причина за създаването на този клас се явява това, че операциите върху стандартните типове данни са реализирани по модул.

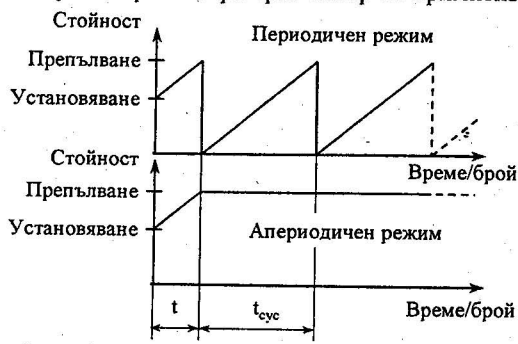
Като обекти за реализиране на системните функции брояч и таймер, са предложени съответно класовете **Timer** и **Counter**. Съчетани със целочислен тип данни или с класа **Infinite**, с тях могат да се реализират броячи (таймери) съответно с периодичен или апериодичен режим на работа.

Предвижда се примитивите на ПО да работят независимо от организацията на потребителската програма. Предвидена е работа както в многозадачен с и без изтласкване, така и в линеен режим. За реализиране на многозадачни режими съществуват различни среди, например MCX11 (с изтласкване) [] и MICROS (без изтласкване) []. Всяка една от тях се характеризира с определени предимства и недостатъци. За постигането на един добър баланс между специализираност и универсалност на операционна система за този тип контролери, е предложен подходящ механизъм. Той позволява съвместната работа на няколко (високоприоритетни) задачи без изтласкване, извиквани от прекъсване (най–често от таймер), с тази на други (ниско приоритетни) задачи, работещи фоново, т.е. подлежащи на изтласкване. Предимството на тази

организация е простото синхронизиране на процесите с външно събитие или по време, без да се изисква сложна многозадачна операционна система. Реализира се чрез два списъка, съответно за високоприоритетните и за нископриоритетните задачи.

Достъпа до входовете и изходите се осъществява от съответните примитиви (четене на входове и изходи, установяване на изходи). Предвидени са два метода за четене на изходите. При първият, те се прочитат от обратните връзки, а при вторият се прочитат сигналите подадени към изходите. Освен състоянието на входовете и изходите от контролера, могат да се четат и дистанционно включените към междумодулната мрежа.

Обмена на информация с други модули се реализира чрез набор от примитиви. Основните функции са изпращане и приемане на съобщение.



Фиг. 6. Режими на работа на таймерите и броячите.

V. Заключение.

Направен е анализ на апаратните и програмните компоненти съдържащи се в микропроцесорните контролери за дискретно производство. Въз основа на това е предложена апаратна и програмна реализация на такъв контролер. Благодарение на локалната и междумодулната система за връзка е изгражда нето на силно и слабо свързани разпределени системи. В програмното осигуряване освен достъпа до ресурсите на контролера (собствените и на подчинените модули), са предложени и няколко специфични абстрактни обекта, облекчаващи разработването на алгоритми свързани със специфичната работа на контролера.

Литература.

1. Белев, С. Промишлени контролери. Русенки университет, 2000.
2. Луканчевски, Л. Системно програмиране за едночипови микрокомпютри. С. Техника, 1993.
3. A.T. Barrett & Associates. Microcontroller Executive for the Motorola MC68HC11. Version 1.3. Motorola, Inc., April 1990
4. MOTOROLA. MC68HC11 Reference Manual. USA, Banta CO., 1998.
5. Николов, Л. Операционни системи. С., ИК Ciela, 1998.
6. Gentleman, W.M. Message passing between Sequential Processes: the Replay Primitive and the Administrator concept. Software Practice and Experience, Vol.11, 1981, p.435-466.
7. Шоу, А. Логическое проектирование операционных систем. М., Мир, 1981