Процесна информация и обработка

Лекции №13 и №14

ПРОМИШЛЕНИ КОНТРОЛЕРИ

Въпросът за автоматизация на производството е поверено на специфичен клас устройства и системи, наречени най-общо Системи за управление -Control Systems.

Въведение

Обект на системите за управление са машини, апарати, съоръжения, технологични процеси и др. Исторически първи се появяват чисто механичните системи за управление. В началото на 20-ти век се появяват електромеханичните системи за управление (релейно-контакторни), които господстват до средата на 20-ти век.

С появата и развитието на електрониката, се появяват и развиват електронни цифрови системи за управление, реализиращи алгоритмите за управление чрез „твърда" (непрограмируема логика). За пръв път идеята за програмируемите логически системи за управление възниква през 60-ти години на 20-ти век във фирмите производителки на автомобили. По същото време започва и работа по създаването устройства за логическо управление, използващи електронно-изчислителни машини.
Бурното развитие и възходът на логическите системи за управление започва през 70-те години на 20-ти век и продължава и до днес. То безспорно е свързано с появата и усъвършенстването на микропроцесорните системи. Въвеждането на микропроцесорната техника в системите за управление върви по два пътя. При първият от тях, те буквално заместват твърдото логическо управление с програмно, запазвайки общата структура и изградените вече принципи за логическо управление. При втория, микропроцесорните системи се допълват с необходимата апаратна част (предимно интерфейсна) и необходимо програмно осигуряване за изграждането на система за управление.
През 80-те години на 20-ти век, повлияно от взривното развитие на компютърната техника, започва изграждането на компютърни системи за управление. В момента е трудно да се предскаже дали компютърните системи ще изместят програмируемите логически контролери в системите за управление. Съвременните програмируеми контролери представляват сложни микропроцесорни системи, с изключителна функционалност и комуникативност. Процесът на развитие на системите за управление върви по пътя на интегрирането на програмируемите логически контролери с компютърните системи.

Терминът „Програмируеми логически контролери" - Programmable Logical Controller (PLC) се използва за обозначаване на програмно-управлявани електронни системи, проектирани за използване в качеството на промишлено оборудване за логическо управление на различни машини, съоръжения и технологични процеси през цифрови или аналогови входове и изходи.
Този термин - PLC се използва предимно от европейските производители на устройства за логическо управление. В американската литература този тип устройства са познати като Програмируеми контролери - Programmable controller (PC). В българската литература като синоним се среща понятието Промишлени контролери.
Съдържанието на понятието „Програмируеми логически контролери" е дадено през 1978 г. от NEMA (National Electrical Manufactures Association). Промишлените контролери са програмно управлявани устройства. Това означава, че връзката между входните въздействия (аналогови или цифрови) и изходните реакции на системата (също аналогови и цифрови) се осъществява чрез последователност от инструкции, реализиращи алгоритъма на управление. Програмната връзка се съхранява в програмируемата памет на промишления контролер и може да бъде променена в процеса на настройване на работата (при някои контролери и в процеса на самата работа).
Основните концепции на промишления контролер са формулирани още през 1968 г. от фирмата General Motors.

Те се изразяват в следното:
- лесно програмиране и препрограмиране (промяна в последователността на изпълняваните операции), по възможност при устройството за управление;
- лесно поддържане и замяна, с предпочитание към модулна конструкция;
- повишена надеждност и минимални габаритни размери;
- конкурентна цена.

Принципите са относителни и са били формулирани по времето когато все още е доминирала релейно-контакторната логика, но те остават в сила и важат и днес.

Обобщена структура на промишлен контролер

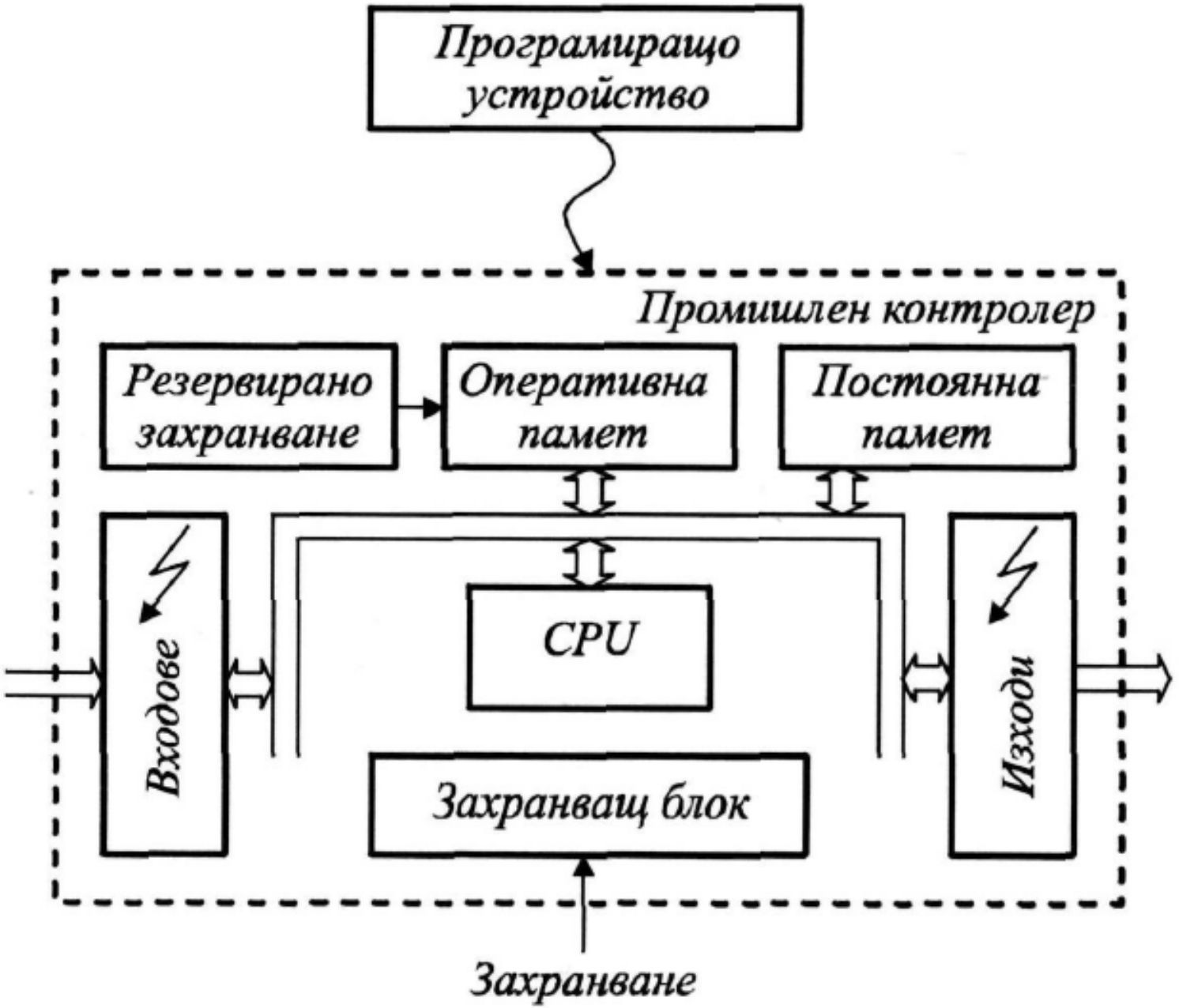
Типичната структура на един промишлен контролер се състои от описаните по-долу основни блокове.

- Захранващ блок. Основното предназначение на захранващия блок е да осигури вътрешно необходимите работни напрежения за контролера и неговите блокове. Най-често използваните вътрешни напрежения са +5V, ±12V и +24V. Съществуват два основни типа захранващи блокове - такива, които ползват входно напрежение от мрежата (например 220V AC) и такива, които ползват оперативно захранващо напрежение от управляемия обект (например 24V DC). Допълнително, захранващият блок може да осигури на контролера сигнал за отпадане на входното напрежение (Power Down).

- Процесорен блок (CPU). Процесорният блок съдържа процесорната част на контролера. При промишлените контролери се използуват микропроцесори или едночипови микрокомпютри от стандартен тип, т.е. такива, които се използуват в микропроцесорните системи с общо предназначение. Процесорът на промишления контролер определя основните характеристики на процесорния модул, като разредност, бързодействие и т.н. В първите промишлени контролери са използувани специализирани процесорни чипове, като например AMD2901, 2903 и др.

Към процесорният модул се включва и паметта на промишления контролер. Тя е разделена на постоянна и на оперативна памет. Постоянната памет се изгражда на базата на ROM, EPROM, EEPROM или FLASH. Цялата или част от постоянната памет може да бъде сменяема. Като оперативни памети най-често се използват памети от статичен тип. Нормално част от оперативната памет (а може и цялата) е с резервирано захранване - батерийно или акумулаторно.

- Входни и изходни блокове. Входните и изходните блокове осигуряват интерфейса между промишления контролер и външния апарат или процес, който трябва да се управлява. Обикновено входните и изходните блокове са с галванично развързване към входните и изходните устройства на управляемия обект.
Както се вижда от блоковата схема, структурата на един промишлен контролер не се отличава от тази на микропроцесорна система с общо предназначение. Това е естествено, тъй като се ползува една и съща елементна база, концепция и организация. Ако трябва да се търсят някакви схемотехнически разлики и принципи на които се изграждат промишлените контролери, те биха могли да се специфицират в следното:
- препоръчително галванично развързване на входовете и изходите на промишления контролер по отношение на управляемия обект, с цел повишаване на шумоустойчивостта и безопасността при работата му;
- резервирано захранване на част или на цялата оперативна памет, с оглед коректно и безопасно поведение на промишления контролер при аварийни ситуации на отпадане на захранващото напрежение.


Програмиращо устройство
Промишлен контролер
Резервирано захранване
Оперативна памет
Постоянна памет
CPU
Входове
Изходи
Захранващ блок
Захранване

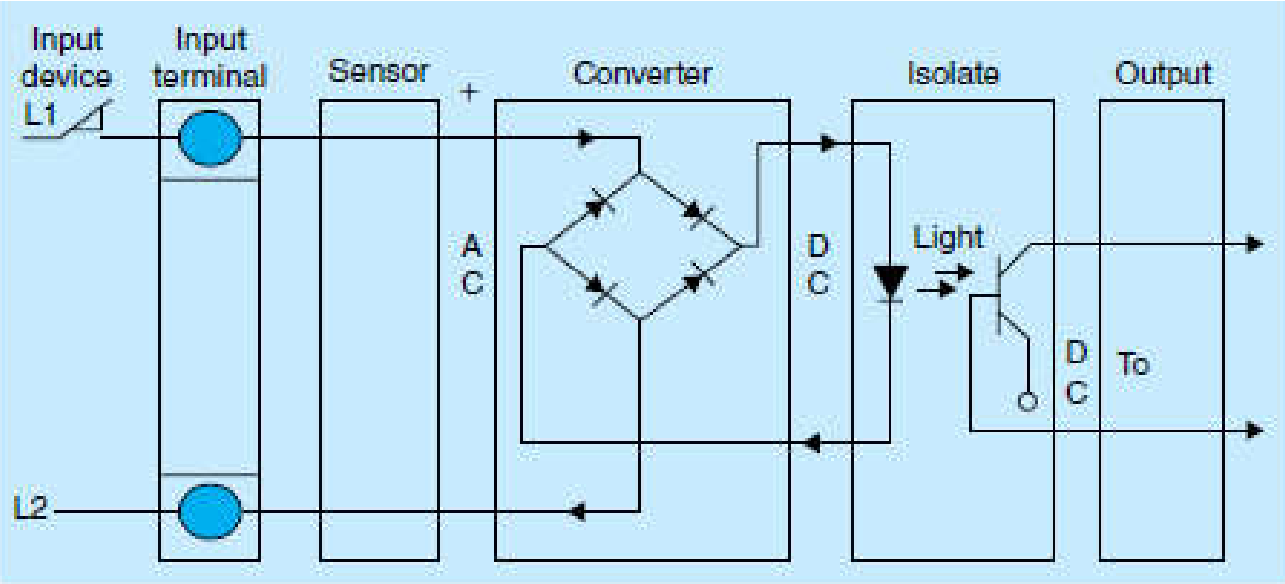
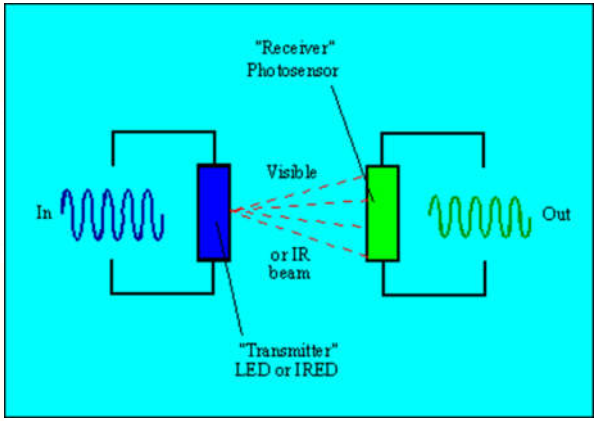
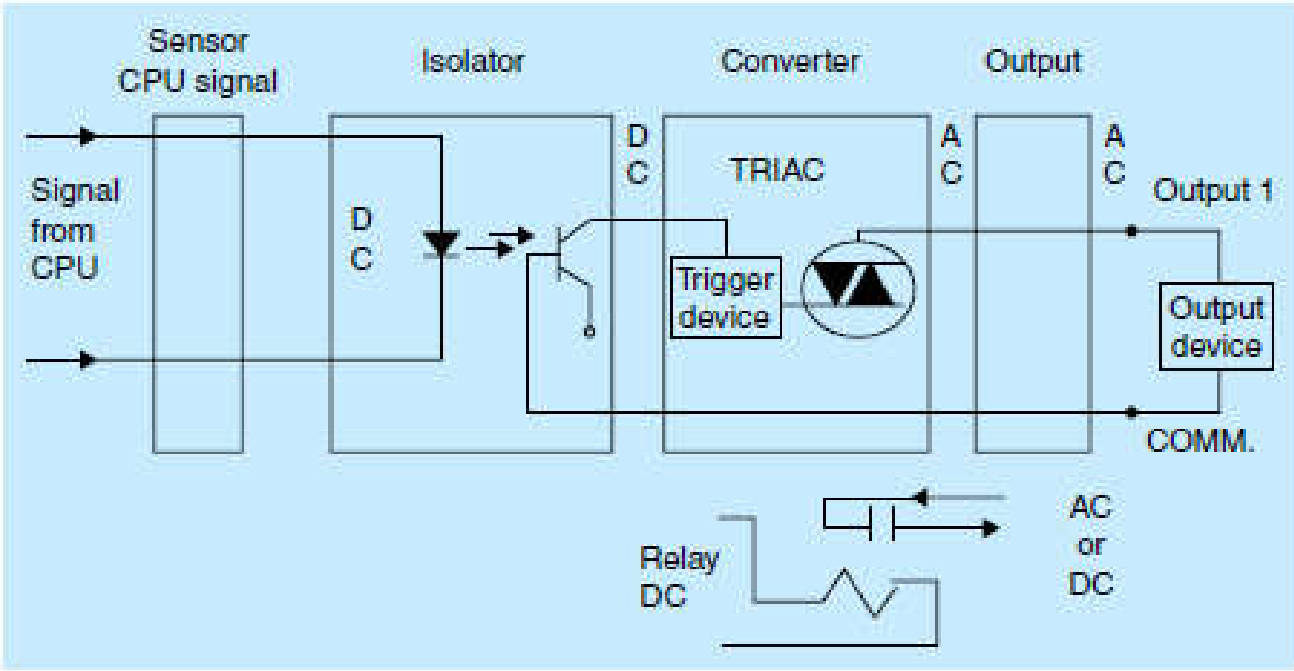
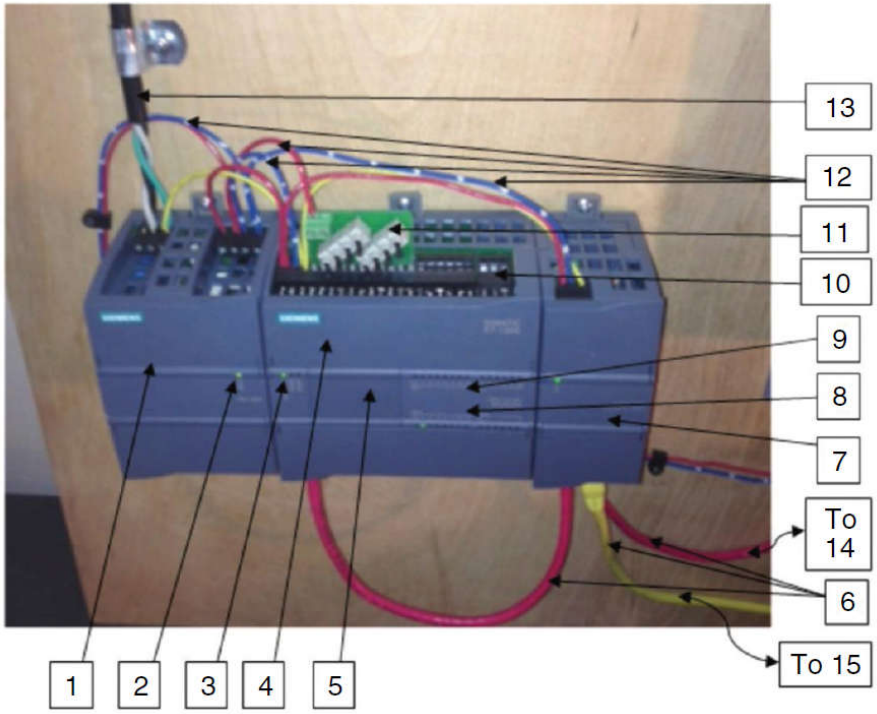


Figure 2.4 Digital input module.





Siemens PLC
S7-1200 PLC



- 24-V power supply (1)
- Power-supply-ready light-emitting diode (LED) (2)
- Processor status LEDs (3)
- Siemens PLC processor; CPU 1214C DC/DC/DC, 6ES7 214-1AE30-0XB0 V2.0. (4)
- One-port integrated analog output module: QW80 (5)
- Ethernet/PROFINET cables; CPU, HMI, and programming computer (6)
- Ethernet/PROFINET four-port communication module (7)
- Processor-integrated 14-digital-output LEDs (8)
- Processor-integrated 14-digital-input LEDs (9)
- Two-port integrated analog input module: port 1 (IW64) and port 2 (IW66) (10)
- Processor plug-in input-switch module: eight ON/OFF switches (11)
- 24-Vdc power connections to processor, switch module, communication module, and HMI (12)
- 120 Vac (13)

Feature	CPU 1211C	CPU 1212C	CPU 1214C
Physical size (mm)	90 × 100 × 75		110 × 100 × 75
User memory			
• Work memory	• 25 Kbytes		• 50 Kbytes
• Load memory	• 1 Mbyte		• 2 Mbytes
• Retentive memory	• 2 Kbytes		• 2 Kbytes
Local on-board I/O			
• Digital	• 6 inputs/4 outputs	• 8 inputs/6 outputs	• 14 inputs/10 outputs
• Analog	• 2 inputs	• 2 inputs	• 2 inputs
Process image size	1024 bytes (inputs) and 1024 bytes (outputs)		
Signal modules expansion	None	2	8
Signal board	1		
Communication modules	3 (left-side expansion)		
High-speed counters	3	4	6
• Single phase	• 3 at 100 kHz	• 3 at 100 kHz 1 at 30 kHz	• 3 at 100 kHz 3 at 30 kHz
• Quadrature phase	• 3 at 80 kHz	• 3 at 80 kHz 1 at 20 kHz	• 3 at 80 kHz 3 at 20 kHz
Pulse outputs	2		
Memory card	SIMATIC memory card (optional)		
Real-time clock retention time	10 days, typical/6-day minimum at 40 degrees		
PROFINET	1 Ethemet communication port		
Real math execution speed	18 µs/instruction		
Boolean execution speed	0.1 µs/instruction		

Принципи на работата на промишлените контролери

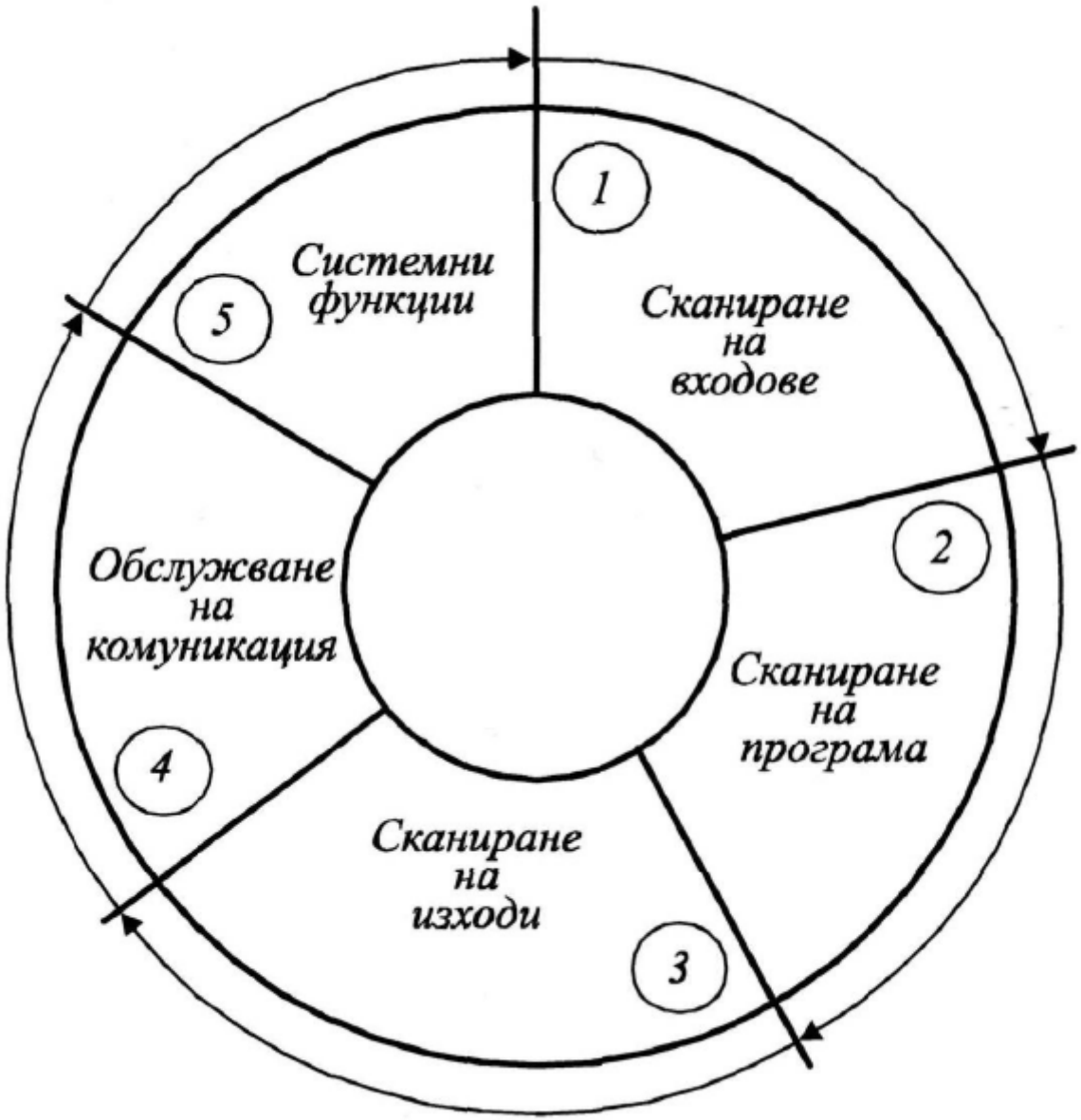
Основните различия между промишлените контролери и микропроцесорните системи с общо предназначение се състои в организацията на работата. Програмируемите логически контролери работят под управлението на операционна система за реално време (ОСРВ), която осигурява циклично изпълнение на логическата програма.

Операционната система осигурява последователното изпълнение на няколко основни дейности. Тази последователност, както и самото й изпълнение, се нарича оперативен цикъл (operating cycle) на промишления контролер. Основните дейности на промишлен контролер са 5 и са представени на фиг. Те се изпълняват последователно и се наричат още „фази на оперативния цикъл".

Фаза 1 се нарича сканиране на входове (Input Scan). Физическите места, където постъпват входните сигнали в промишления контролер се наричат входни точки (Input Points). За всяка входна точка е определен по един бит в оперативната памет - т.нар. входен бит (Input Bit). Дейността по сканирането на входовете извършва прочитане на входните точки и записването на информацията от тях във входните битове. С други думи, в оперативната памет се създава „образ" на входните въздействия към дадения момент (Process Image Inputs).

This is a list of real-time operating systems. An RTOS is an operating system in which the maximum time from an input stimulus to an output response can be definitely determined.

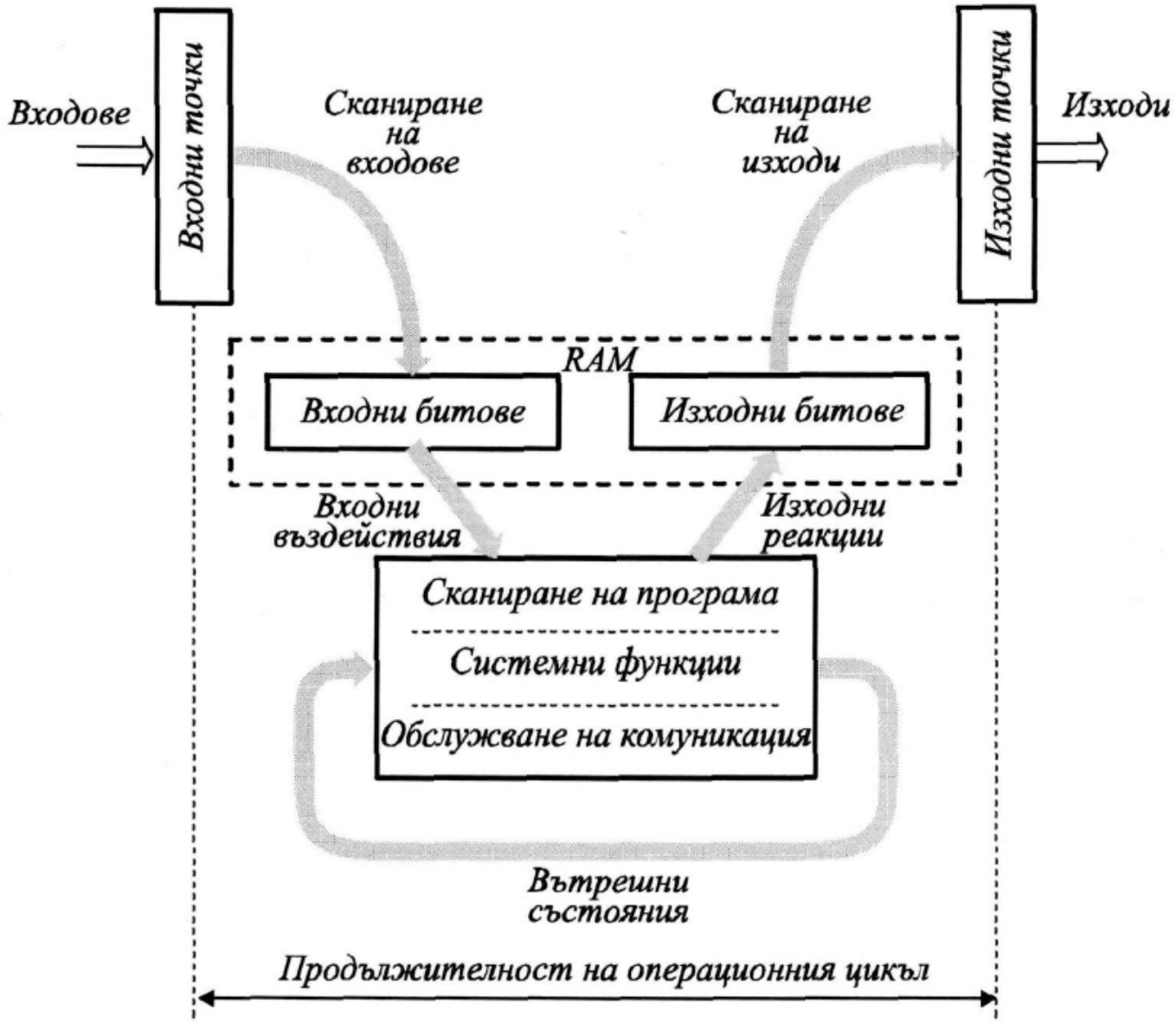
Name	License	Source model	Target uses	Status	Platforms
Abassi	Proprietary	closed	embedded	active	AVR32, ATmega, ColdFire, Cortex-A9, Cortex-M0, Cortex-M3, Cortex-M4, MSP430, PIC32, TMS320C2000, 80251, 8051
AMOS	Proprietary	?	commercial	closed	680x0, 683xx, x86 via emulation
AMX RTOS	Proprietary	closed	embedded	active	680x0, 683xx, ARM, ColdFire, MIPS32, PowerPC
Apache Mynewt OS	Apache 2.0	open	embedded	active	ARM Cortex-M, x86, MIPS
uKOS	GNU GPL	open source	embedded	active	Cortex-M3-M4-M7, 6833x, PIC, CSEM icyflex-1, STM32
ARTOS (Locamation)	Proprietary	closed	power-system automation	active	x86
ARTOS (Robotu)	Proprietary	?	embedded, robots	defunct	ARM9+
Atomthreads	BSD	open source	embedded	active	AVR, STM8, ARM, MIPS
AVIX	Proprietary	closed	embedded	active	Atmel AT91SAM3(U/S), EFM32, NXP LPC1300-1700, ST Micro STM32, Texas Instruments LM3S, Toshiba TMPM330, Microchip PIC32MX-PIC24F-PIC24H-dsPIC30F-dsPIC33F
BeRTOS	modified GNU GPL	open source	embedded	active	ARM, Cortex-M3, ARM ARM7TDMI, Atmel AVR, PowerPC (emu), x86 (emu), x86-64 (emu)
BOOS Core	GNU GPL	open source	embedded	active	ARMv5TEJ (TI AM18x), DSP (TI TMS320C64x)
BRTOS	MIT	open source	embedded	active	Freescale Kinetis (ARM Cortex-M4), Freescale Coldfire V1, Freescale HCS08, ST STM32F4xx (ARM Cortex-M4F), NXP LPC11xx (ARM Cortex-M0), NXP LPC176x (ARM Cortex-M3), Renesas RX600 (RX62N), Texas Instruments MSP430, Texas Instruments Stellaris LM3S8968 (ARM Cortex-M3), Texas Instruments Stellaris LM4F120H5QR (ARM Cortex-M4F), Atmel ATMEGA328/128 and Microchip PIC18
CapROS	GNU GPL	open source	embedded	active	IA-32, ARM9
ChibiOS/RT	Mixed, GNU GPL or proprietary	open source	embedded, small footprint	active	x86, ARM7, ARM9, Cortex-M0-M3-M4, PowerPC e200z, STM8, STM32, AVR, MSP430, ColdFire, H8S
ChorusOS	?	?	?	defunct	SPARC, x86, PowerPC
ChronOS	GNU GPL	open source	research, general purpose	active	x86, ARM
CoActionOS (now Stratify OS)	Modified GNU GPL or proprietary	open source	embedded	discontinued	ARM Cortex-M3, LPC17xx


1
Сканиране на входове
2
Сканиране на програма
3
Сканиране на изходи
4
Обслужване на комуникация
5
Системни функции

Фаза 2 е сканиране на програмата (Program Scan). През тази фаза се изпълняват всички инструкции от логическата програма по отработване на входните въздействия и формиране на изходните реакции. Програмируемите логически контролери са последователностни устройства, в които при формирането на изходните реакции освен входните променливи участват и вътрешни за контролера променливи (таймери, броячи, компаратори и т.н.). В много ограничени случаи, програмируем логически контролер може да изпълнява само комбинационни логически функции (изходните реакции се определят еднозначно само от входните въздействия).

Фаза 3 се нарича сканиране на изходите (Output Scan). Физическите места, през които промишленият контролер изпраща изходните сигнали към управляемия обект се наричат изходни точки (Output Points). Както при входните точки, така и за всяка изходна точка е определен по един бит в оперативната памет - т.нар. изходен бит (Output Bit). Дейността по сканирането на изходите извършва прочитане на изходните битове и прехвърлянето на информацията от тях в изходните точки. Отново, в оперативната памет се създава „образ", сега вече на изходните реакции в дадения момент (Process Image Outputs).

Фаза 4 е предназначена да обслужи комуникацията (Service Communication). В тази част на оперативния цикъл се извършва комуникация с други устройства, например програмиращо устройство, централен компютър, устройство за човеко-машинен интерфейс (операторски пулт, панел, терминал) и др. В повечето програмируеми логически контролери, комуникацията се обслужва главно от операционната система на контролера, според специални директиви, записани в управляващата програма. Някои контролери позволяват директно управление на комуникацията от потребителската програма чрез съответната системна функция.



Продължителност на операционния цикъл на промишлен контролер.

Фаза 5 е предназначена за изпълняване на системни функции (Housekeeping and Overhead). Към тази дейност влизат разнообразни действия, по-важни от които са:
- управление на паметта;
- управление на вътрешните регистри на микропроцесора;
- обслужване на вградените специални апаратно-програмни функции (Firmware Function) като броячни входове, импулсни входове и изходи, допълнителни комуникационни портове и др.;
- извикване на вградени логически функции, които имитират работата на хардуерни устройства, използвани за целите на логическото управление като устройства за времезадържане (таймери), устройства за отброяване на външни или вътрешни за контролера събития (броячи) и др.;
- тестване на апаратно-програмните функции на промишления контролер (Self Diagnostic Test).

Фигурата дава нагледна представа за дейностите, извършвани от промишления контролер в отделните фази на оперативния цикъл.

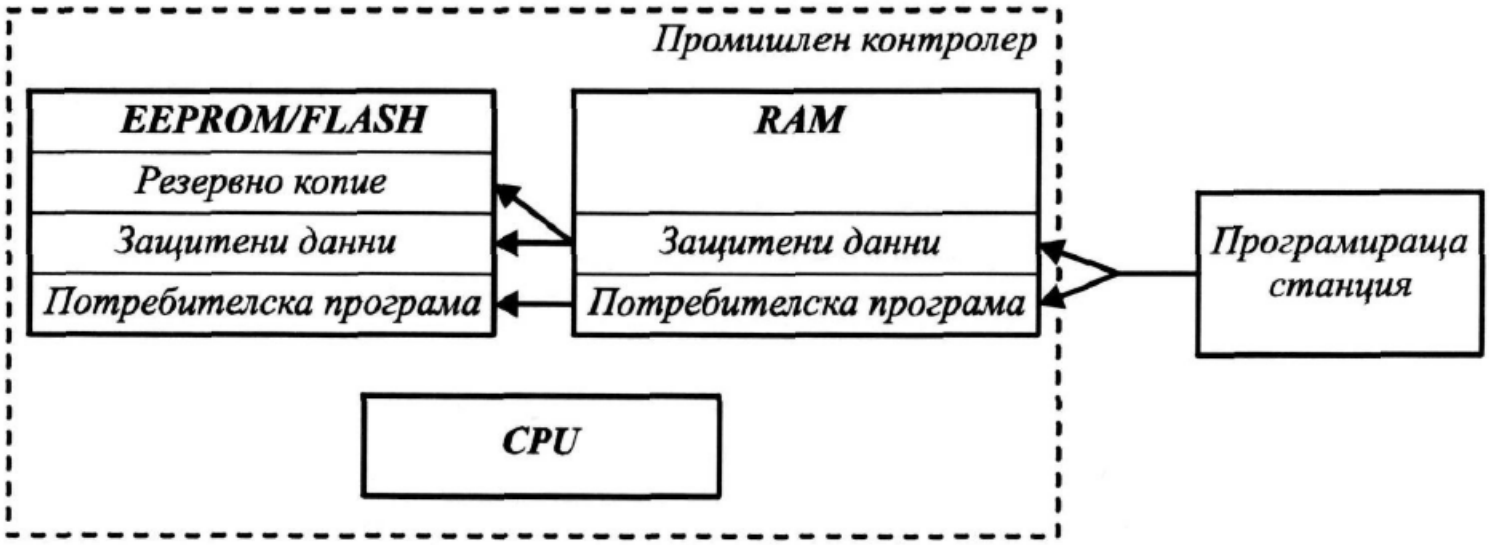
Организация на паметта при промишлените контролери

За разлика от микропроцесорните системи с общо предназначение, изграждането на паметта на промишлените контролери се подчинява преди всичко на режимите на работа, които те трябва осигуряват и които са съобразени със специфичните условия на работа и изпълняваните задачи. Основните режими, в които трябва да обезпечават промишлените контролери са 4:

- режим на програмиране на промишления контролер (Download);
- режим на нормална работа (Run Mode);
- режим на отпадане на захранващото напрежение (Power Down);
- режим на възстановяване на захранващото напрежение (Power Up). Постоянната памет на съвременните промишлените контролери е от EEPROM или FLASH тип. Една част от нея, която съдържа системните функции, може да бъде и от ROM или EPROM тип. Предназначението на постоянната памет е да съдържа потребителската програма и системните и потребител-ски данни (статични системни и потребителски даннови секции), които не се променят по време на работа. С използването на репрограмируеми постоянни памети по технологията FLASH не се затрудняват промените в програмното осигуряване при неговата настройка.

Оперативната памет в промишлените контролери съдържа т.нар. динамични даннови секции. Това са променливите данни на операционната система (системна информационна област) и данните (информационните области), осигуряващи обмена на информация между различните задачи в потребителската програма - образ на входовете и изходите, таймери, броячи, памет на променливите, вътрешни релета и др. В някои промишлени контролери, в оперативната памет се помества и работната програма, т.е тя се използва и като програмна памет.
Оперативните памети в промишлените контролери са от статичен тип. Тъй като информацията в RAM се изгубва при отпадане на захранващото напрежение (аварийна ситуация), се налага да се вземат специални програмно-апаратни мерки по възстановяване на работата на контролера след възстановяване на захранването. В зависимост от конкретния обект, управлението трябва да се възстанови от момента, предшестващия аварията или по друг указан алгоритъм.
Апаратното осигуряване за отработването на аварийните ситуации с отпадането на захранващото напрежение при повечето производители включва защита на част от оперативната памет с батерия или кондензатор, т.нар защитени данни (Retentive Data). Обикновено, в тази част влизат цялата област на системните данни, заявената за защита част от потребителските данни, част от областта на вътрешните релета и част от областта на броячите.
Задължително апаратно осигуряване е наличието на сигнал за отпаднало захранване, малко преди контролерът да го загуби напълно. Такъв сигнал обикновено се подава на апаратен вход за немаскируемо прекъсване на контролерния процесор и задейства процедурата за отработване на аварийната ситуация.

Програмният механизъм за отработване на аварийните ситуации при различните контролери се различава в някаква степен, но неговите принципи са еднакви.
На фиг. 1.5 е даден режимът на програмиране на промишления контролер. Потребителската програма (Program Files) и защитените данни се прехвърлят от програмиращото устройство в RAM паметта на промишления контролер. Оттам потребителската програма и защитените данни се запомнят в енергонезависима EEPROM или FLASH памет. При контролери, които не подържат батерийно резервирани данни в RAM, защитените данни се копират още веднъж в енергонезависимата памет като резервно копие (Backup Data).



1.5. Режим на програмиране на промишлен контролер.

В режимът на нормална работа, промишленния контролер изпълнява потребителската програма по управление от енергонезависимата памет. По време на работата настъпва промяна в областта на защитените данни. Тя се отразява единствено в RAM паметта, тъй като процесорът работи с нея. В този режим програмиращата станция може да чете и променя областта на защитените дан-ни в RAM паметта, т.е може да променя хода и параметрите на управляващата програма без да променя самата програма. Фиг. 1.6. илюстрира режима на нормална работа.
При някои контролери, изпълнението на потребителската програма се извършва от RAM паметта. В този случай програмиращата станция може да променя и самата потребителска програма. Последното не се препоръчва да се извършва в работни условия, но има приложение, когато управляващата програма се тества върху макет на съоръжението или процеса.

Програмиране на PLC

Основна роля за унифициране на езиците за програмиране на PLC изиграва споменатият вече Одо Щругер от Allen-Bradley. По-късно той става и вицепрезидент по технологиите в Rockwell Automation. Негова е водещата роля при разработката на стандарта IEC61131-3, на който езиците за програмиране на PLC отговарят днес.

Съгласно този стандарт, те са четири:

• Списък с инструкции (Instruction List) - изглежда по същия начин като асемблера на микропроцесорите. Много рядко се използва днес от разработчиците.

• Структуриран текст (Structured Text) - по синтаксис прилича много на Pascal. Използва, например, функции за условен преход от вида if... then... else... И той се използва много рядко.

• Ладер диаграма (Ladder Diagram) - графичен език за програмиране на PLC, пряк наследник на релейно-контактните диаграми. Използва символи като реле, контакт, функционален блок и се оформя във вид на мрежа. Това е най-широко използваното от инженерите по КИП и А средство за програмиране. До неотдавна в България беше много разпространен софтуерът Modsoft за програмиране на контролерите Modicon. Всички програми за PLC в средата на Modsoft са написани именно чрез ладер диаграми.

• Функционални блокове (Function Bloc Diagram) - графичен език, в който функциите са представени във вид на правоъгълници - отляво са входовете им, а отдясно - изходите. Има набор от библиотеки с предварително готови програмирани блокове, но потребителят може да създава и програмира свои собствени блокове.

обобщение

Начинът, по който се осъществяват много от индустриалните процеси днес, е резултат от дългогодишна изследователска и развойна дейност, насочена към подобряване на функционалността, управлението и организацията им. Нуждата от единно и категорично решение на множество разнородни проблеми и свързаните с тях разходи в производството е в основата на разработването и на програмируемия логически контролер (на англ. PLC - programmable logic controllers).

Съвременни приложения

Съвременните програмируеми логически контролери са базирани на микропроцесори. Днес те се използват буквално във всеки сегмент и подсектор на индустрията, а приложенията им са неизброими. Сред тях са: управление на машини и процеси, управление на задвижвания, управление на дискретно-непрекъснати (batch) процеси, диагностика и др. PLC устройства се произвеждат и продават по целия свят като самостоятелен продукт от множество глобални доставчици на контролно оборудване. Множество компании произвеждат контролери и за OEM приложения.

Ранна история

Ранната история на програмируемия логически контролер започва от 60-те години, когато системите за управление са релейни. По онова време в диспечерските зали има цели стени с шкафове, препълнени с релета, клемни блокове и неизброимо количество кабели. Тези системи срещат редица проблеми, по-съществените от които са следните: липса на гъвкавост при необходимост от разширение или изменение на технологичния процес; затруднено отстраняване на повреди, свързани със замърсени контакти, разхлабени връзки, несъответствия между остарелите планове на схемите и реалното им положение и т.н.

Идеята за "стандартен машинен контролер"

В далечната 1968 г. Бил Стоун от отдела за разработка и производство на автоматичните скоростни кутии на Дженерал Мотърс представя на конференция доклад за проблемите с надеждността на производствените системи в техния завод.

В доклада той посочва и критериите, на които трябва да отговаря един “стандартен машинен контролер”, а именно:

• да елиминира нуждата от скъпо излизащата подмяна на старата релейна логика при изменения в поточната линия, свързани с внедряването на нови модели, а също така и като цяло да замени ненадеждните електромеханични релета;

• да намали времето на принудителен престой на машините при възникване на някакъв проблем в системата за управление и да може лесно да бъде програмиран на място с вече възприетата тогава релейна ладер логика (relay ladder logic);

• да има възможност за разширяване на обхвата и функциите, което значи да бъде с модулна архитектура и да могат да се подменят/добавят компоненти;

• да може да работи в промишлени условия, т. е. в замърсена среда, при влага, електромагнитни смущения и вибрации.

Надпреварата

Спецификацията с горните критерии е предоставена на четири фирми, които да разработят прототип на този “стандартен машинен контролер” - Allen-Bradley, DEC, Century Detroit и Bedford Associates. DEC предлага “миникомпютър”, който бива отхвърлен поради редица причини, но едно от най-сериозните му ограничения е статичната памет.

Allen-Bradley по онова време е сред водещите производители на релета и управления за двигатели и включвайки се в този проект, попада в положението да се конкурира със себе си в една от най-силните си области - електромеханичните релета. Предлага два прототипа, но и двата, макар и в различна степен, са били твърде обемни, твърде сложни и трудни за програмиране.

Печелившото предложение идва от страна на Bedford Associates. Екипът и дотогава работи по проект за такова устройство с модулна и достатъчно здрава, надеждна архитектура, работещо по зададен алгоритъм, а не само по външни прекъсвания, и използващо пряко изобразяване на адресите на основната памет в кеш-паметта на процесора.

Това устройство е наречено 084 - поредният номер на проекта в Bedford. Създава фирмата Modicon (от анг. MOdular DIgital CONtroller). Именно под наименованието “Modicon 084” през 1969 г. на Дженерал Мотърс е представен прототипът на програмируемо логическо устройство, изпълнено с полупроводникови елементи.

Modicon 084 има три обособени части – процесорна платка, памет и логическо устройство, което изпълнява основния алгоритъм, зададен по правилата на релейната ладер логика (ladder logic). Освен това е проектиран с изключително здрава и надеждна конструкция, без вентилатори за охлаждане, плътно затворен, дори без ключ за включване и изключване.

Ричард Морли обяснява: “Нямаше вентилатори, не се допускаше външен въздух да влиза в системата, защото се бояхме от замърсяване или корозия. Представяхме си, че програмируемият контролер ще се монтира под шасито на камион, през цялото време ще е на открито и ще пътува от Тексас до Аляска. Искахме да оцелее при тези условия.

От друга страна, стоеше изискването той да обслужва някакво устройство, което да не зависи от климата и въобще да не се нуждае от поддръжка.” Именно Modicon 084 се смята за първия програмируем логически контролер, макар и създателите му да са го наричали просто “програмируем контролер” (programmable controller - PC).

Едно решение с много имена

Наименованието “програмируем логически контролер” (PLC) се въвежда малко по-късно от Одо Щругер от Allen-Bradley, където работата по това направление продължава, въпреки първите не особено успешни и неприети от Дженерал Мотърс прототипи. През 1971 г. Одо Щругер и Ернст Дамермът създават първия контролер на Allen-Bradley под наименованието Bulletin 1774 PLC.

Най-точен превод на български би трябвало да бъде “контролер с програмируема логика”, но отдавна в България е възприето понятието “програмируем логически контролер”. Има държави, които използват собствено название поради ред причини, включително и вследствие на оригинални собствени разработки.

В немскоезичното пространство могат да се срещнат със съкращението SPS (нем. speicherprogrammierbare Steuerung – управление чрез програмируема памет). Във Франция ги наричат “програмируем индустриален автомат” (съкр. API, от фр. automate programmable industriel). Но във всички случаи същността на устройството е една и съща.

Опитът на Modicon

Modicon набира опит и през 1973 г. пуска следващия много успешен модел PLC, Modicon 184. В първите години PLC се приемат много трудно. Дик Морли си спомня: “В началото имахме големи проблеми да убеждаваме хората, че една кутия със софтуер, макар и изработена от здрава стомана, би могла да свърши същата работа като 15 метра шкафове, натъпкани с релета и кабели”.

Той припомня и как през 1969 г. “всички компютри изискваха чиста, климатизирана среда, и въпреки това често се повреждаха”. “При това положение, макар че PLC бяха и са специализирани компютри с конкретни функции, полагахме специални усилия те да не бъдат отъждествявани с компютрите поради тяхната слаба надеждност и факта, че не са направени за работа в производствени условия”, добавя Морли. За разлика от компютрите по онова време, програмируемите логически контролери са проектирани така, че да бъдат надеждни.

След поредица от фирмени прехвърляния днес Modicon е собственост на Schneider Electric, като нейните контролери, наследници на разработките от 70-те, продължават да се предлагат под търговската марка Modicon.

Предизвикателства

Сериозен проблем при използването на ранните модели PLC е фактът, че за тяхното програмиране са необходими специализирани терминали. Тези терминали представляват голямо предизвикателство за хората, програмиращи PLC. В това направление Скот Зиферер, съосновател на софтуерната фирма ICOM, и Нийл Тейлър, начело на собствена фирма за промишлен софтуер, започват да развиват програмирането за PLC и съответното документиране, като по този начин оказват огромно влияние върху промишлената автоматизация, каквато я познаваме днес.

Скот Зиферер съсредоточава усилията си единствено към продуктите на Allen-Bradley. “Исках да използвам компютър за програмирането на PLC и за съставяне на съответната документация, вместо специализирания терминал, който в Allen-Bradley наричаха Терминал Т-3. Те самите разработваха подобен проект, но го правеха твърде бавно”, споделя той.

Постепенно потребителите на Т-3 започват да работят много по-удобно с усъвършенствания потребителски интерфейс, разработен от ICOM. Този подход допринася за по-доброто възприемане на PLC на Allen-Bradley от страна на инженерите по КИП и А и техниците по поддръжката и така разширява полето на приложение на контролерите.

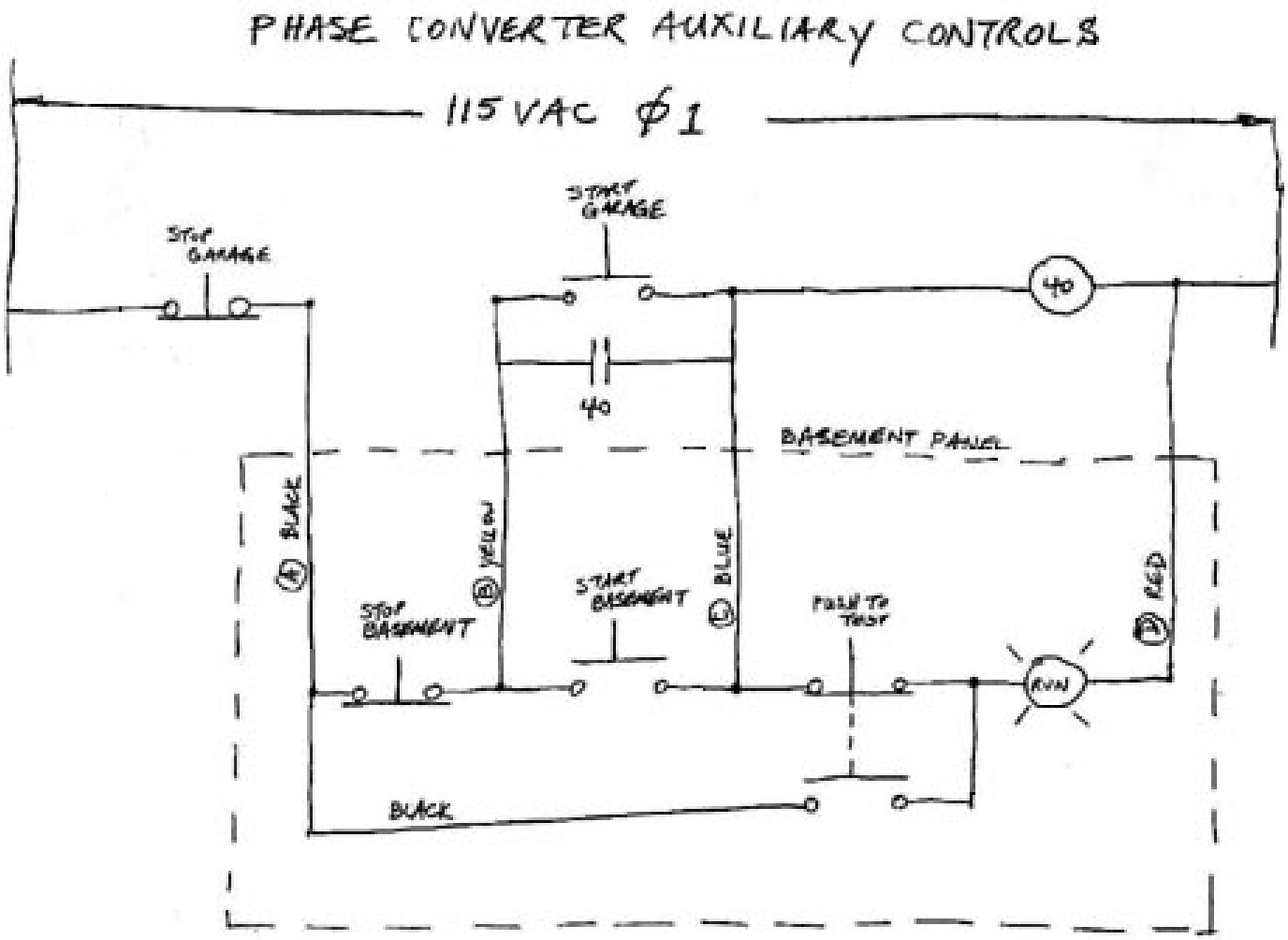
Фирмата на Зиферер е погълната от Rockwell Automation през 1993 г. Това става и с Allen-Bradley още през 1985 г., но произвежданите от Rockwell Automation програмируеми контролери продължават да се предлагат под същата търговска марка Allen-Bradley и до ден-днешен.

Нийл Тейлър се насочва към контролерите Modicon и по-точно към проблемите, свързани с тяхното програмиране. "Давах консултации по тази тема и виждах нуждата да се осъвремени създаването на диаграмите с ладер логика на чертожната дъска, което беше трудно и времеемко", разказва Тейлър.

Първоначално той се захваща с документирането и създава множество възможности за генериране на доклади и формати, които, макар и офлайн, спомагат за отстраняване на проблеми при работата на PLC. По-късно Тейлър създава и средства за онлайн наблюдение на изпълнението на програмния код от контролера.

Programmable Logic Controller

The concept: a machine that can be started from two remote places



Inputs

Program

Outputs

Inputs and Outputs Devices

Inputs

- Push Buttons
- Proximity switches
- Photoelectric sensors
- Temperature sensors
- Pressure sensors

Outputs

- Motors
- Solenoids
- Indicator lamps
- Resistive loads
- Contactors

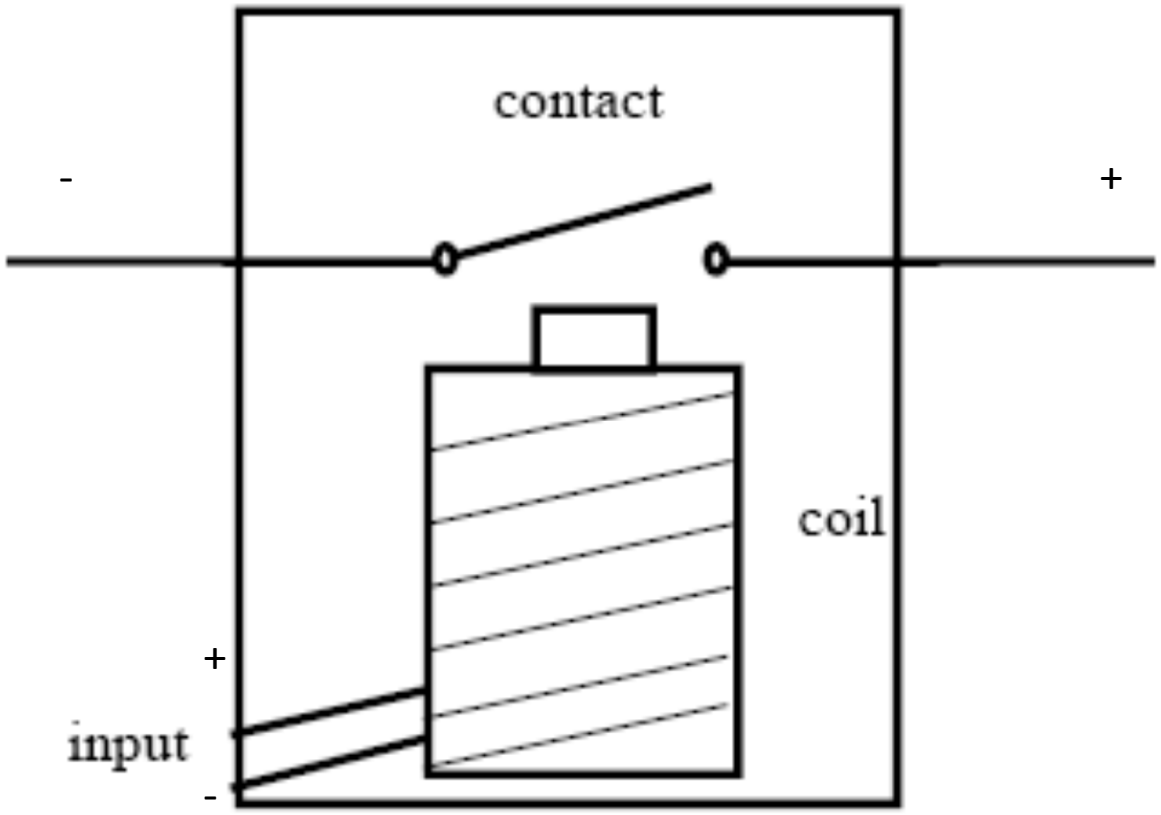
Push button

Photo Sensor

Pressure Sensor

Motor

Relays



Programmable Logic Controllers

Те обикновено имат от 4 до 40 входа и изхода. В зависимост от размера и функционалността те могат да струват между $ 100 и $ 1000.

Обикновено има от 20 до 100 входа и изхода и нагоре. Тези обикновено започват от $ 500 и могат да станат доста скъпи

Box Type

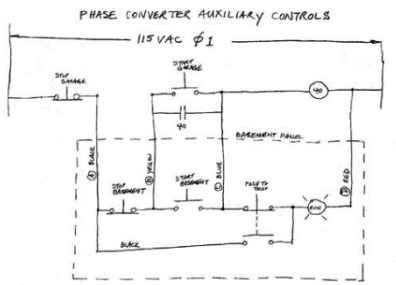
Modular or Rack Type

Typical PLC Applications

- Coin-Operated Carwash
- Conveyor Diverter Control
- Greenhouse Irrigation
- Lumber Mill Operation
- Oil recovery systems

Тези малки компютри са предназначени да бъдат здрави.

Three Phase Converter

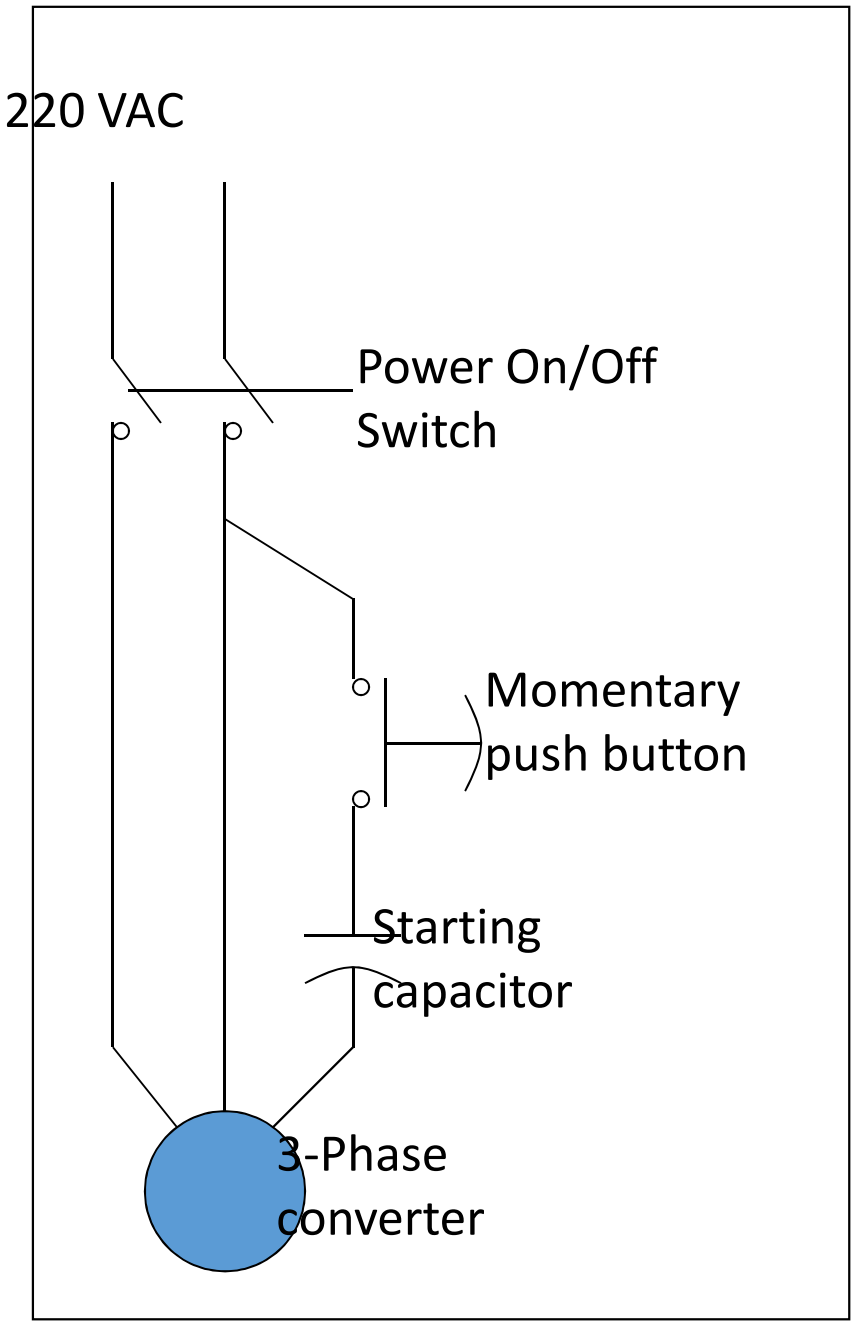


- A machine requires three phase power to operate but only two phase power is available.
- A power converter must be built using a three phase converter.

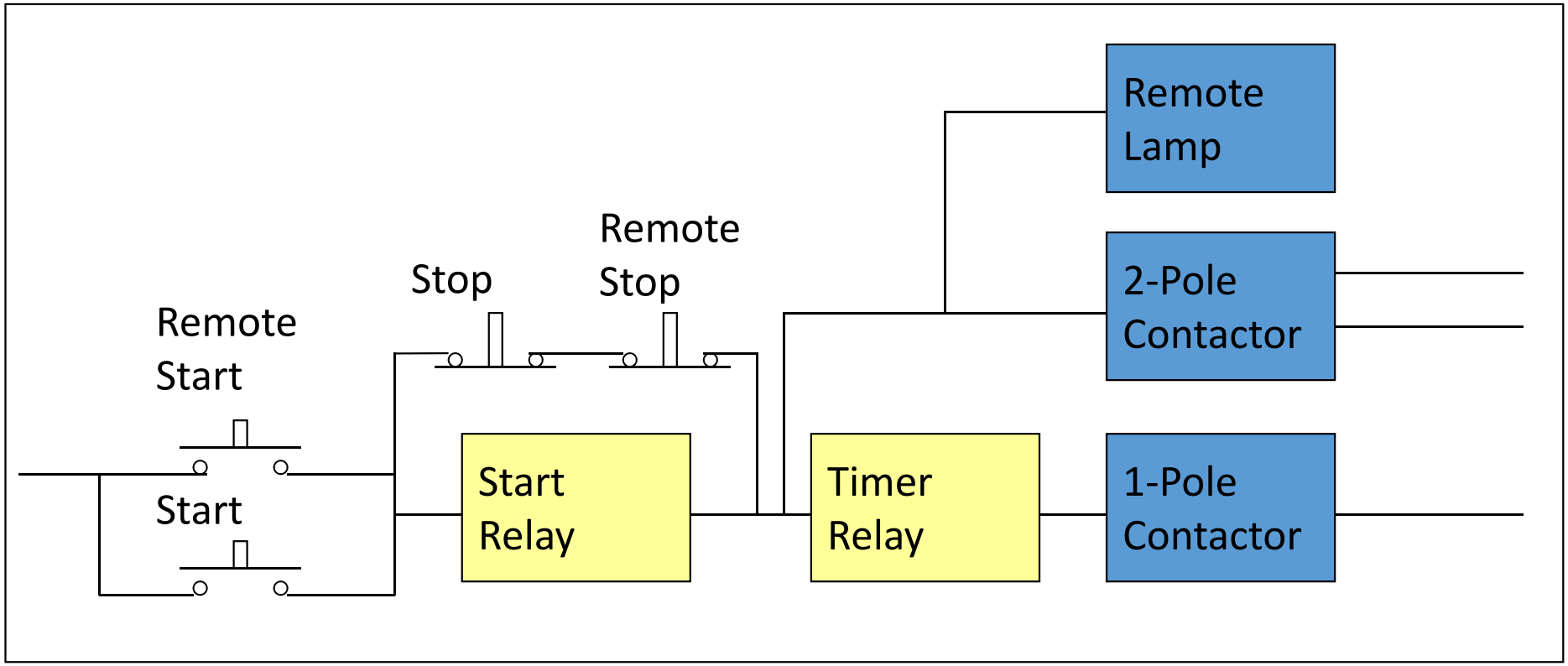
Three phase converter

Converter Operation

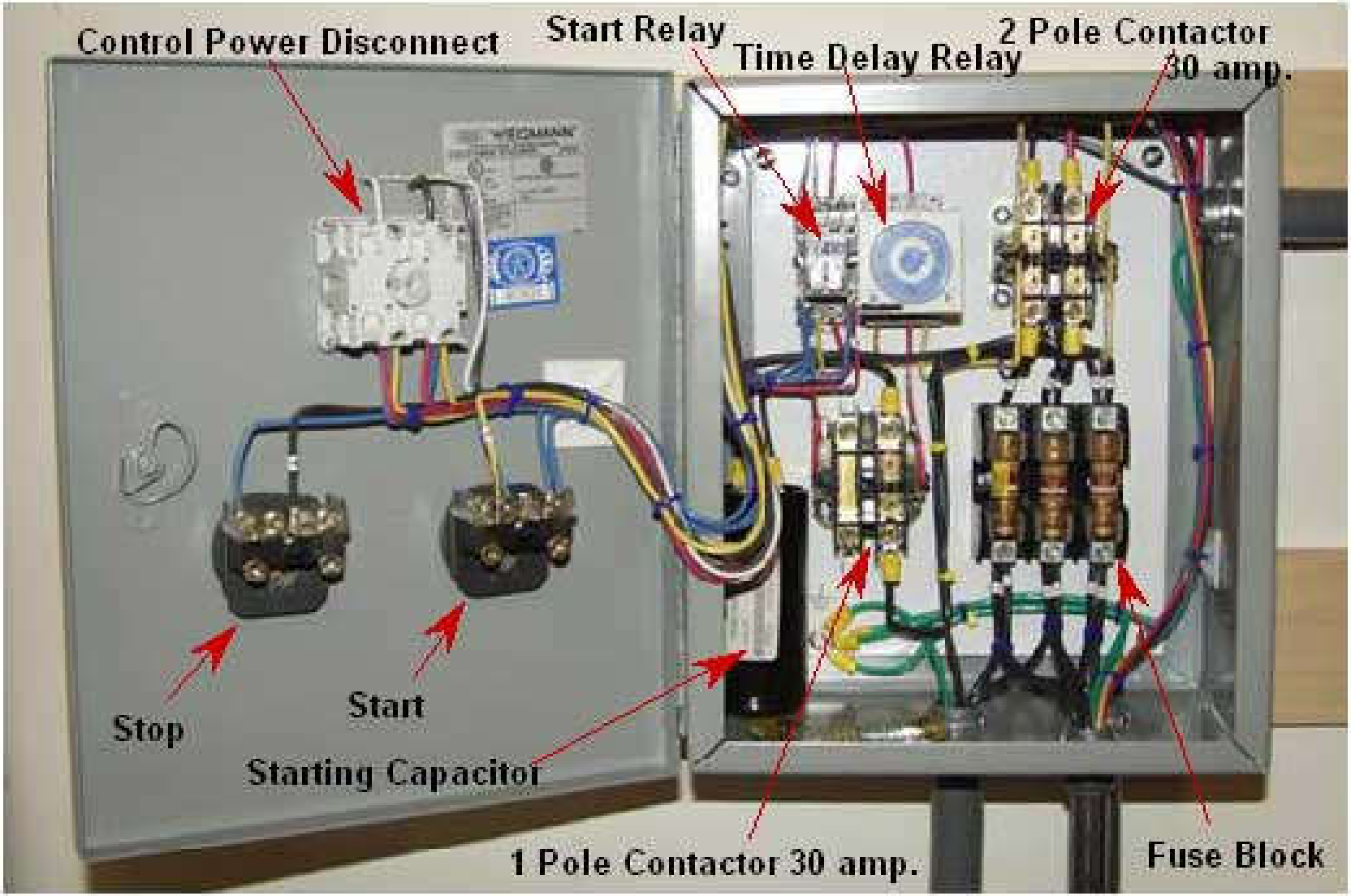
- The power is switched on
- To start the converter's motor the push button is held for 1 to 2 seconds



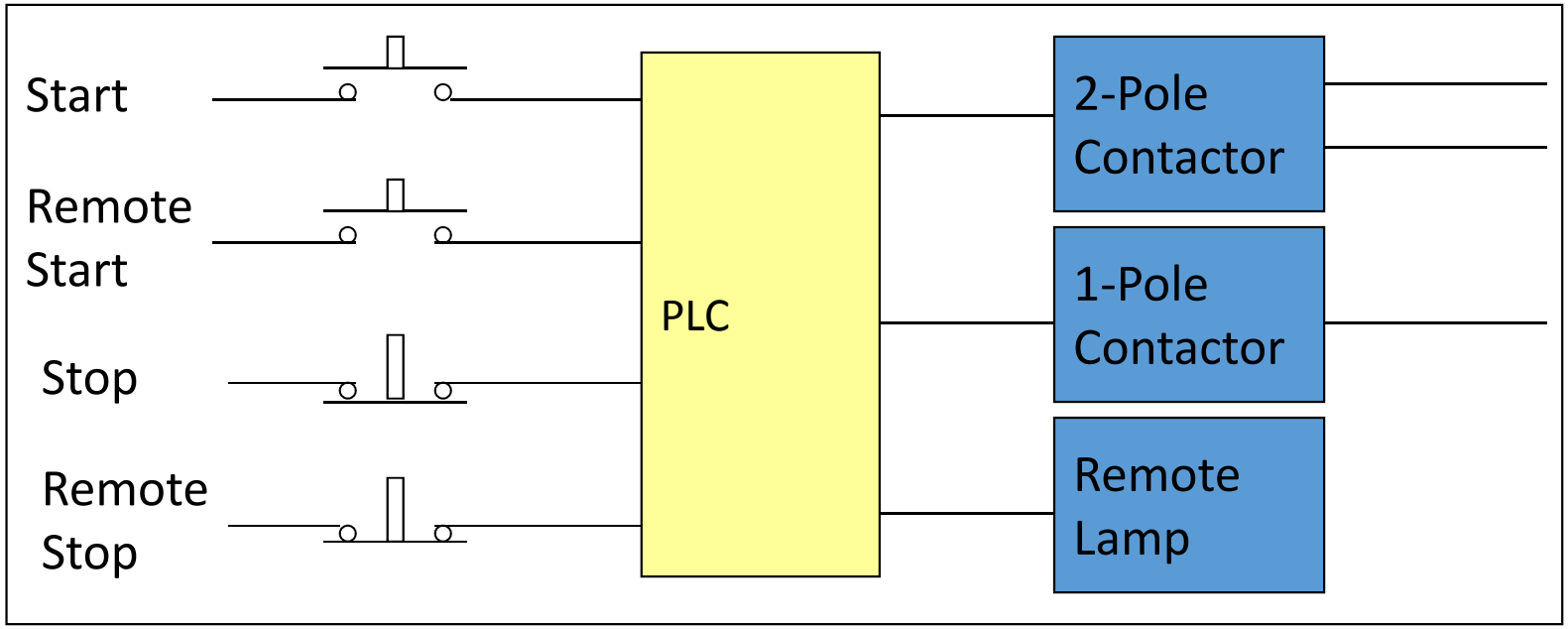
3-Phase Converter



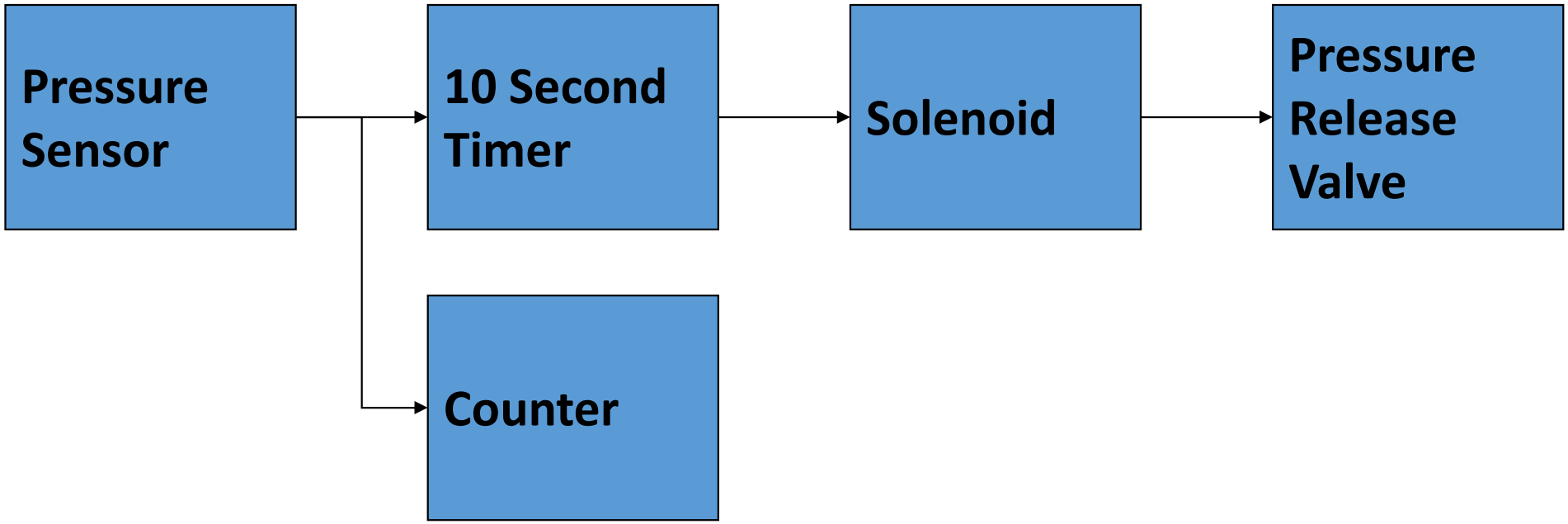
3-Phase Converter



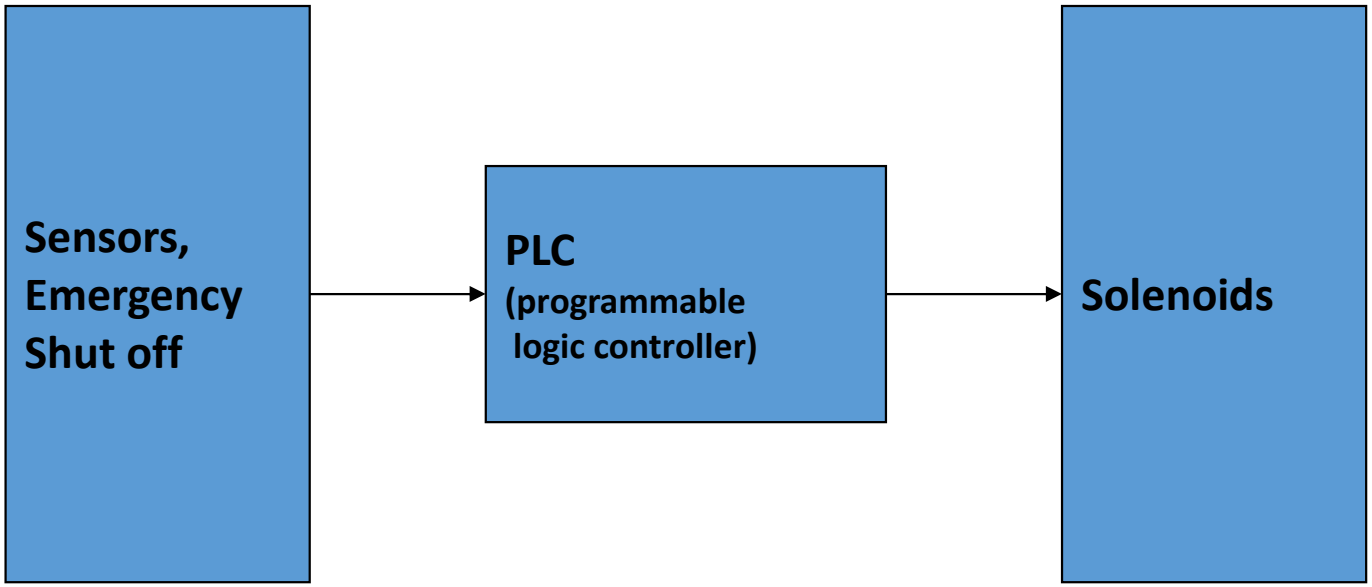
3-Phase Converter



Example: Process Safety Pressure Emergency Release

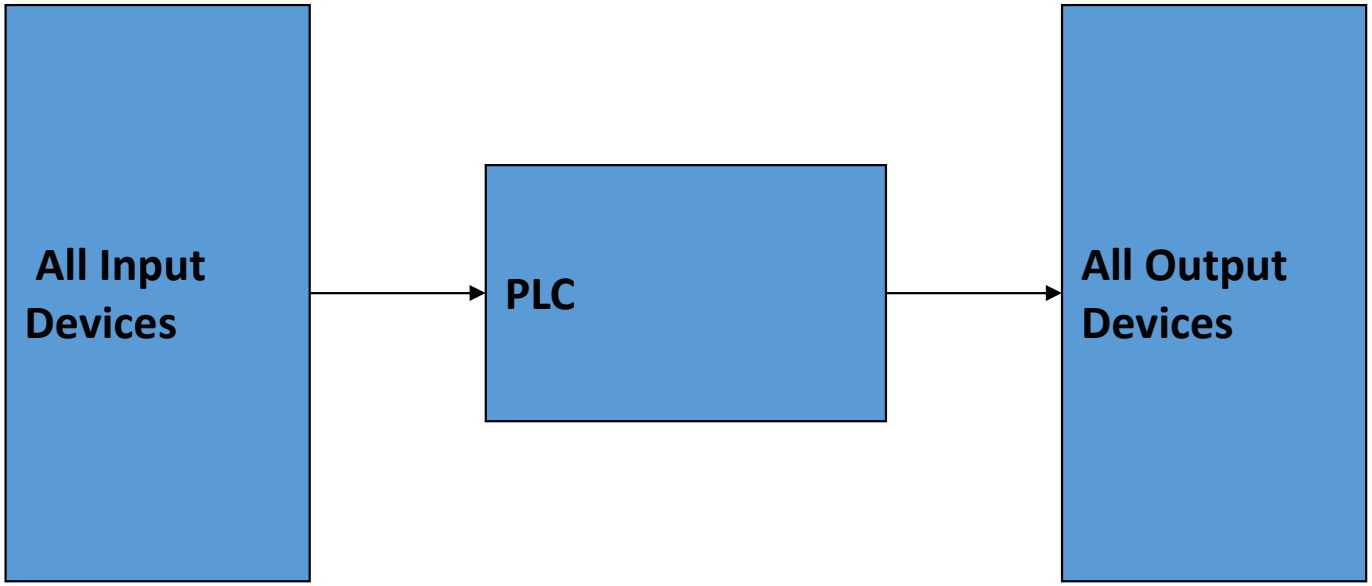


Example Process



A PLC replaces the wiring between input and output devices. Instead of being wired together, all equipment is wired to the PLC.

The logic implementd through wiring is now a program inside the PLC

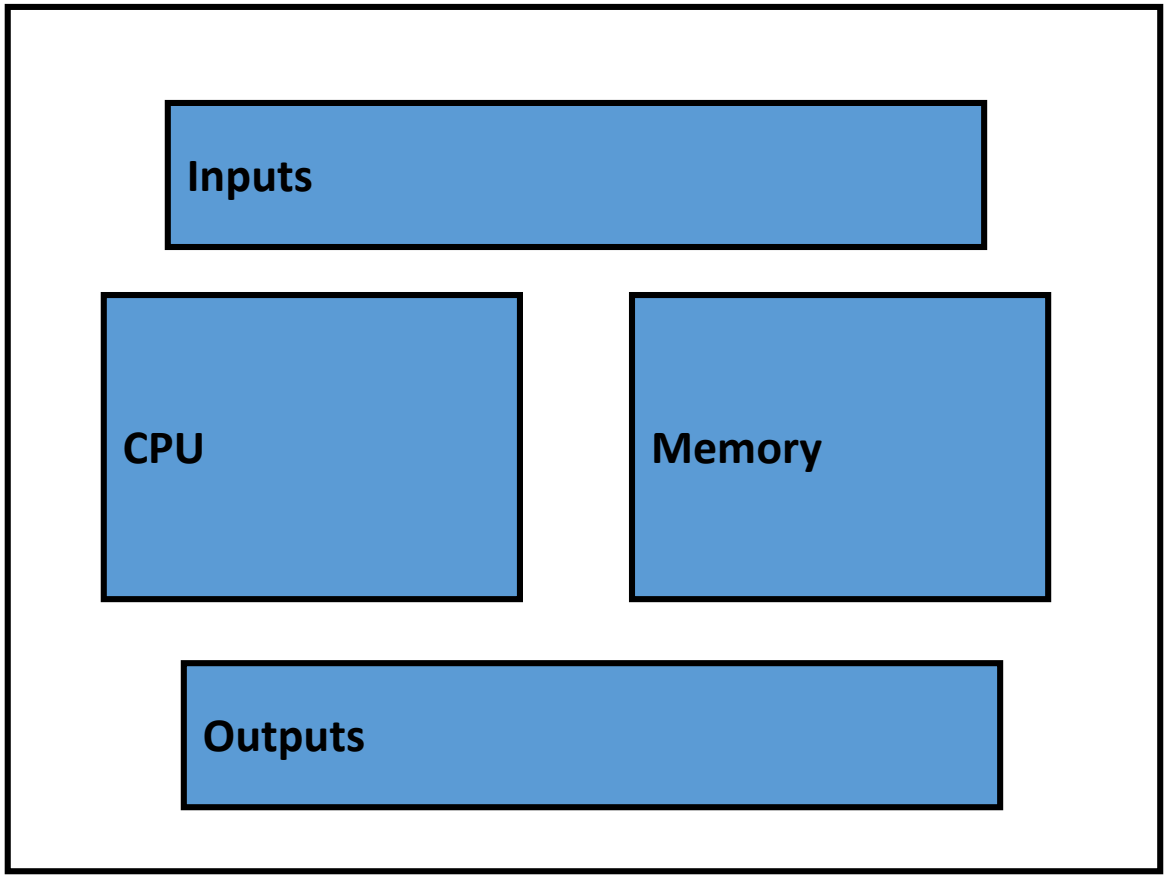


Programmable Logic Controllers

- PLCs (Programmable Logic Controllers) is a miniature industrial computer performs control functions [4]
- The first PLCs can be traced back to 1968 and became popular in the 1980’s [4]
- PLCs are rugged and designed to withstand the industrial environment.

Components of a PLC

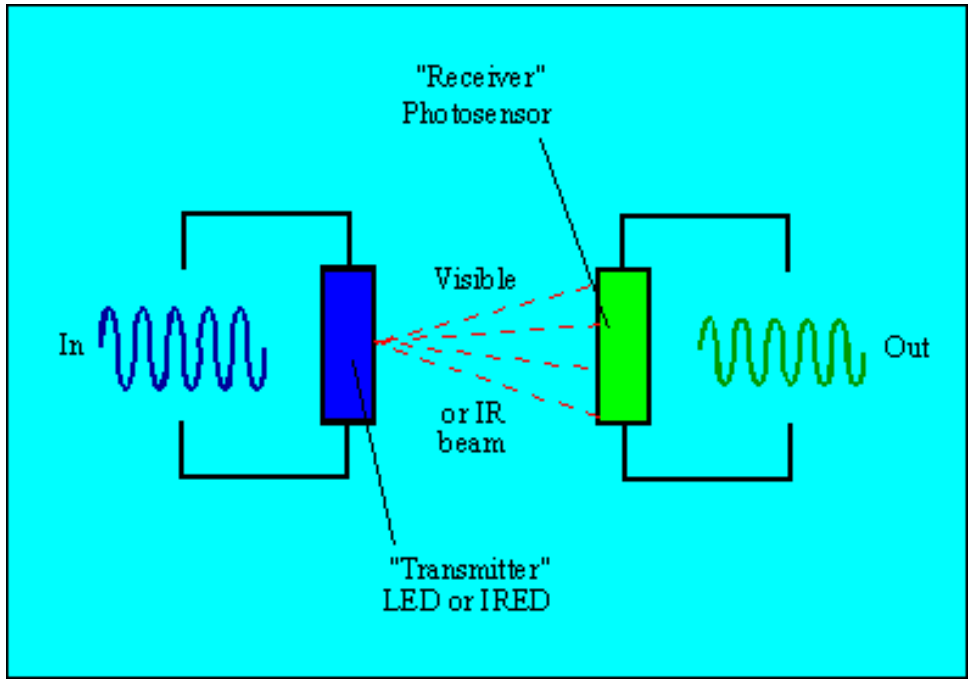
- Inputs
- CPU
- Memory
- Outputs



PLC Inputs and Outputs

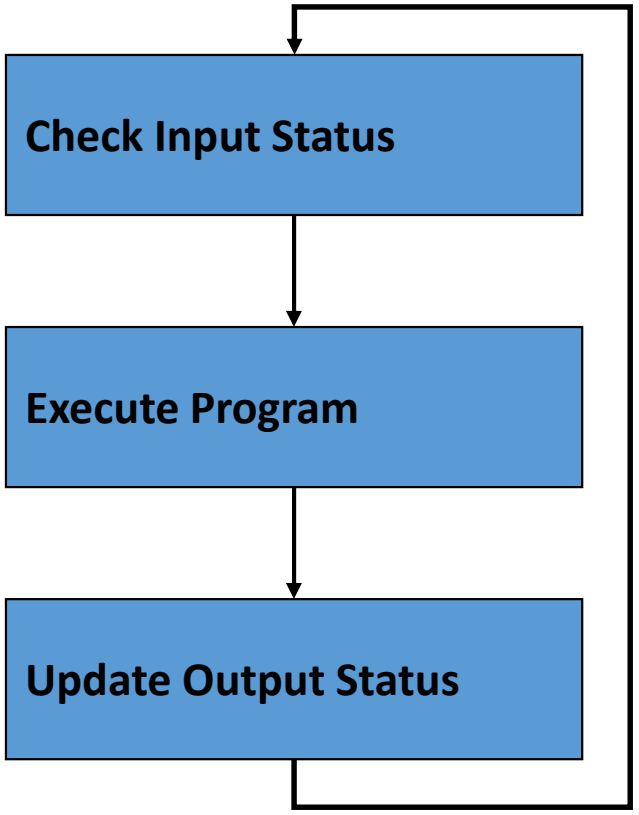
- Analog, Discrete or Digital
- Protected by Optoisolators
- Sourcing and Sinking
- Communications

Optoisolator



PLC Operation

- Scanning is the process of running the PLC program
- Cycle time is the total time for one loop.

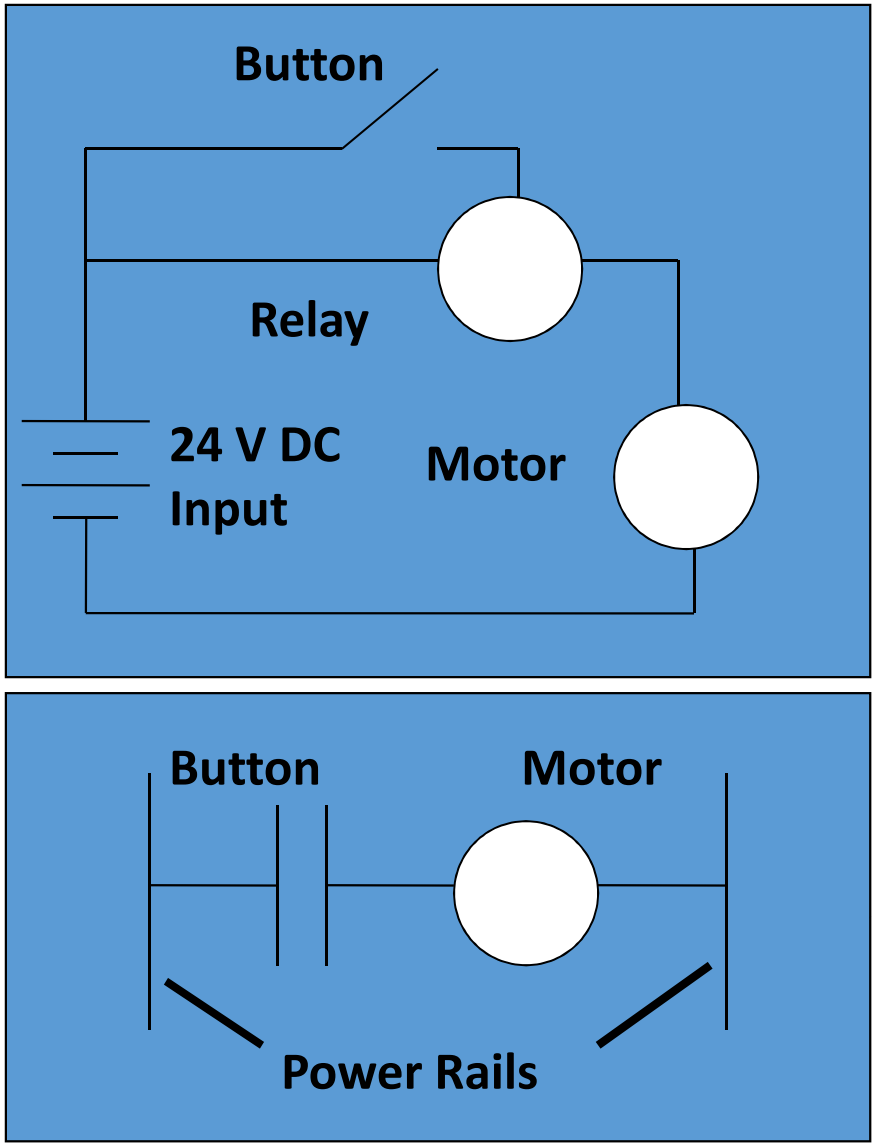


PLC Programming

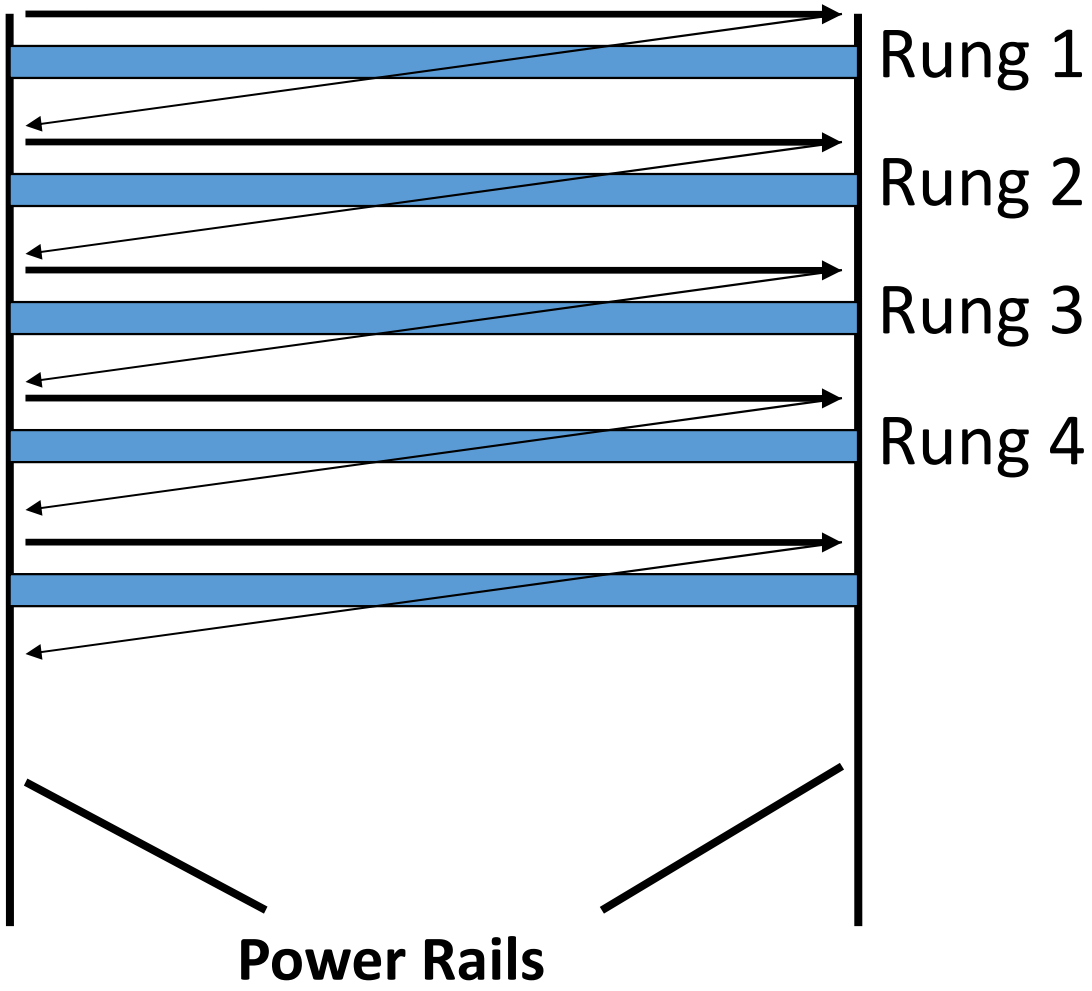
- Ladder logic is the main programming method used for PLCs [39]
- It is a visual and symbolic programming language that resembles relays logic diagrams
- Ladder logic has been developed to mimic relay logic to reduce amount of retraining needed for engineers and trades people [39]

Ladder Logic

- Compare a circuits diagram to a ladder logic diagram



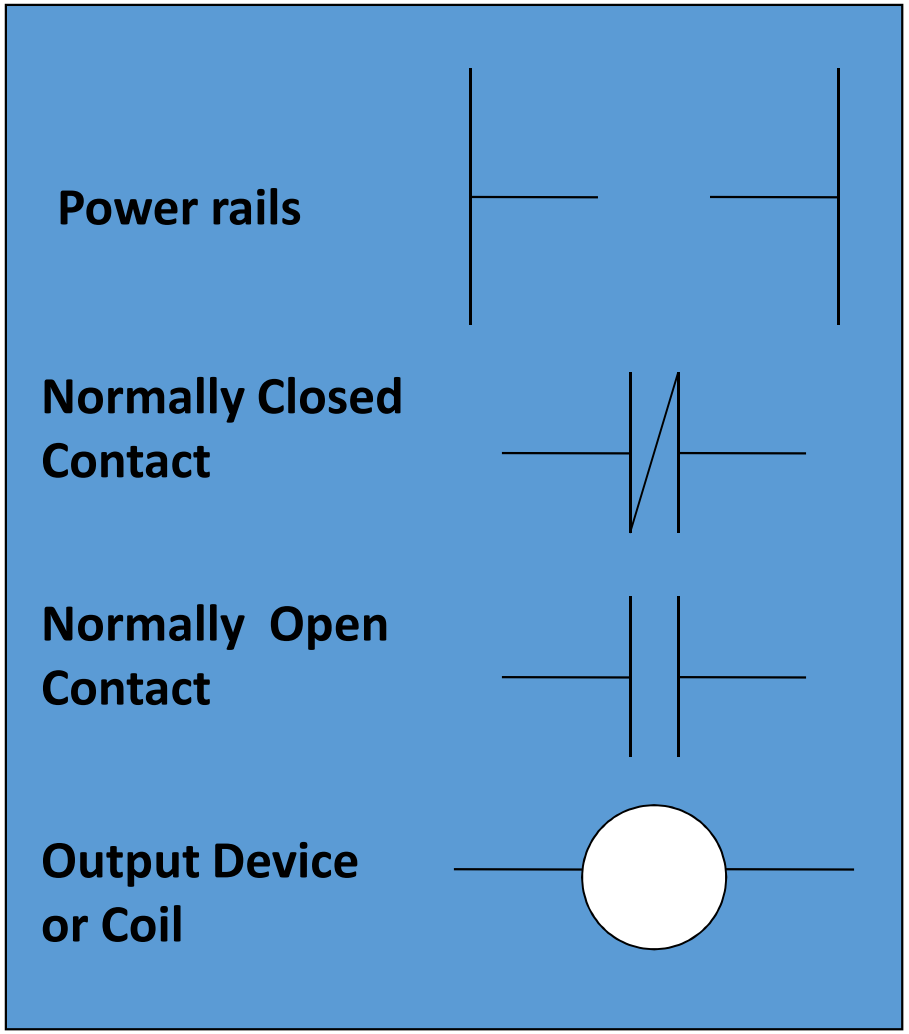
Ladder Logic е наречена така, защото диаграмата изглежда като стълба. Всяка стъпка в програмата се нарича "стъпало". Вертикалните линии отляво и отдясно са силови релси. Всяко стъпало определя една операция в процеса на управление. Структурата на стълбата се чете от ляво на дясно и отгоре надолу. Всяко стъпало започва с един или повече входа и завършва с отдаване под наем на един изход.



SG1

Ladder Logic

- Ladder logic uses a variety of programming symbols
- Power always flows from left to right
- Output devices are in the ON state if power flows through them

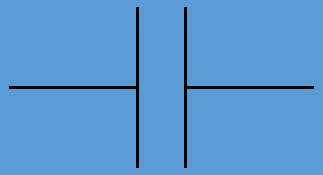


Slide 52

SG1 Stefan Garabitov; 18.5.2017 г.

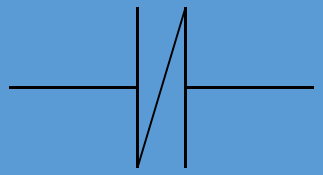
Contacts

Contact programmed normally open



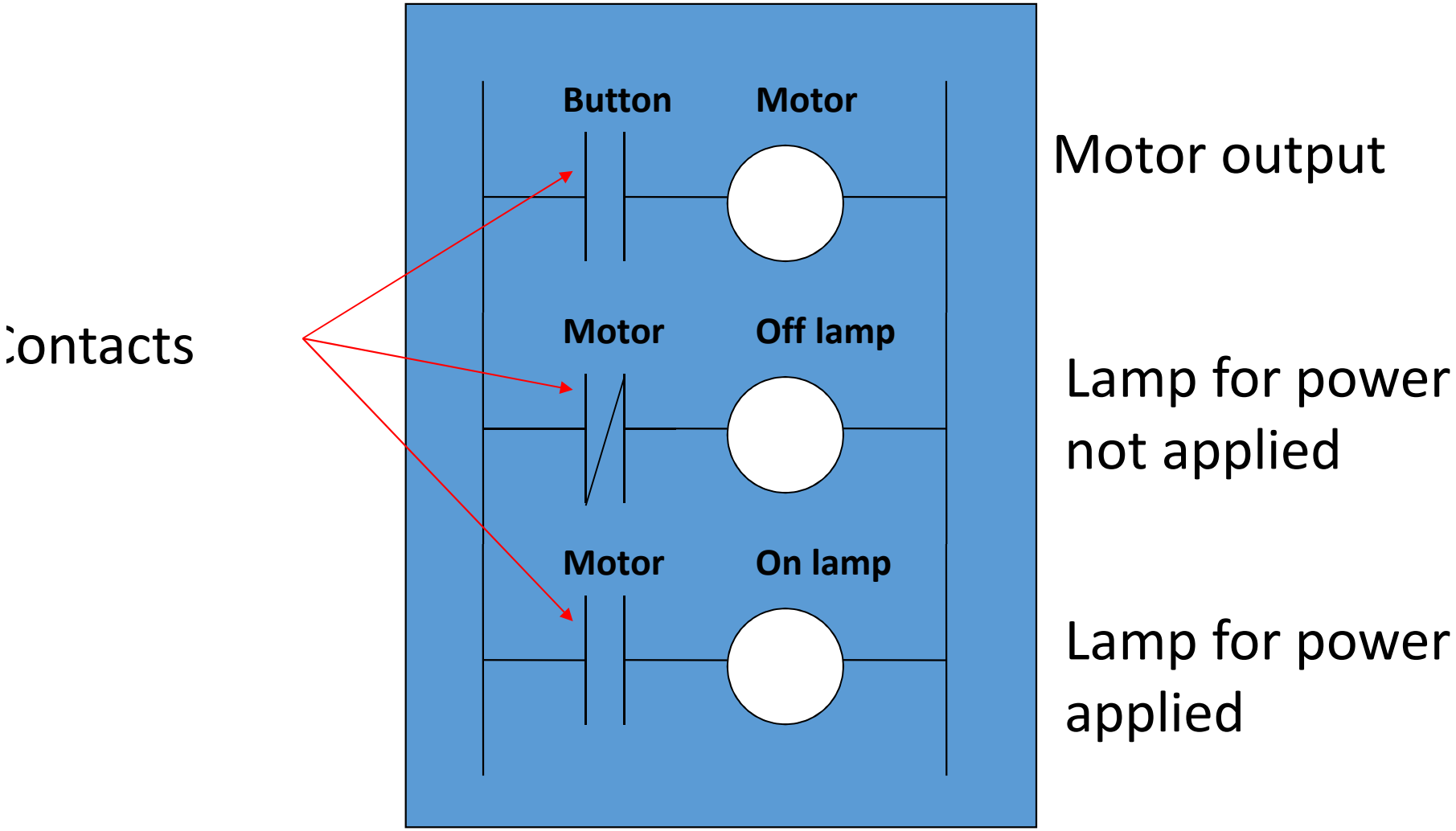
Power flows when the input device is on

Contact programmed normally closed



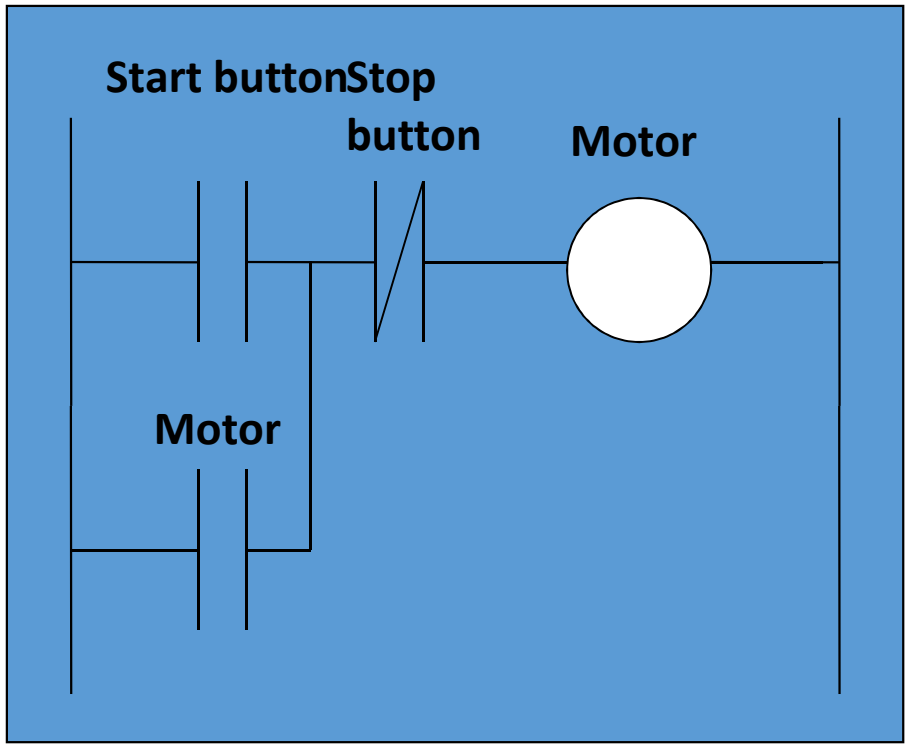
Power flows when the input device is off

Ladder Logic

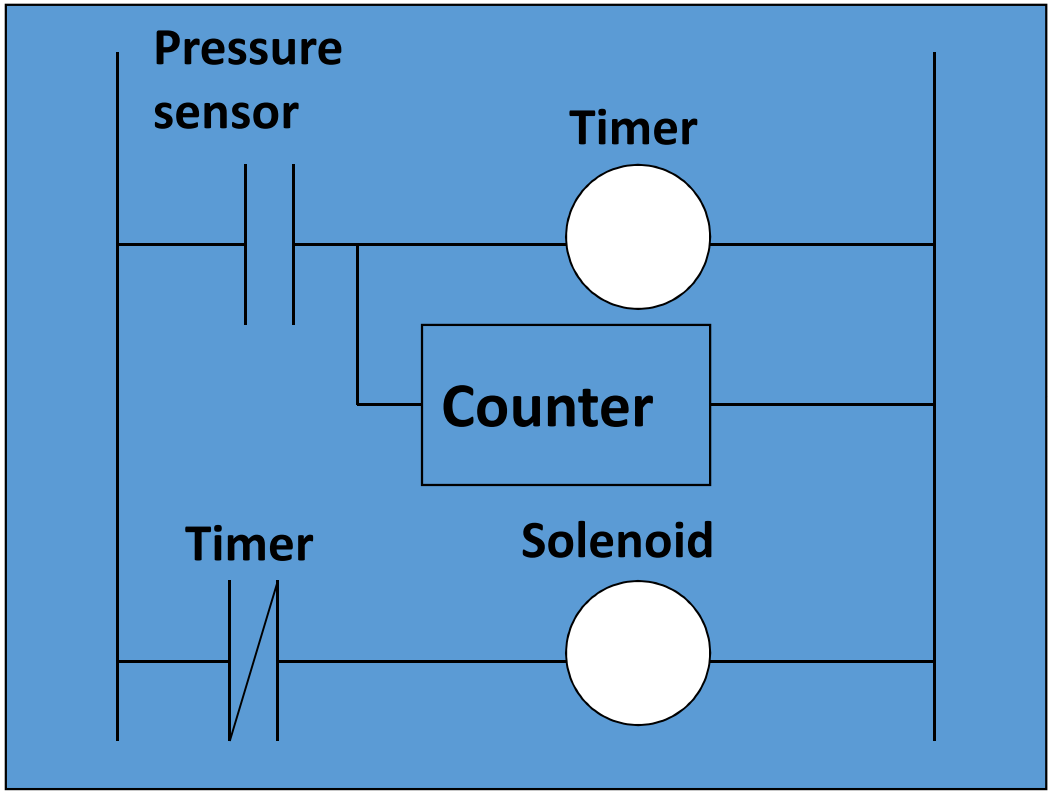


Ladder Logic

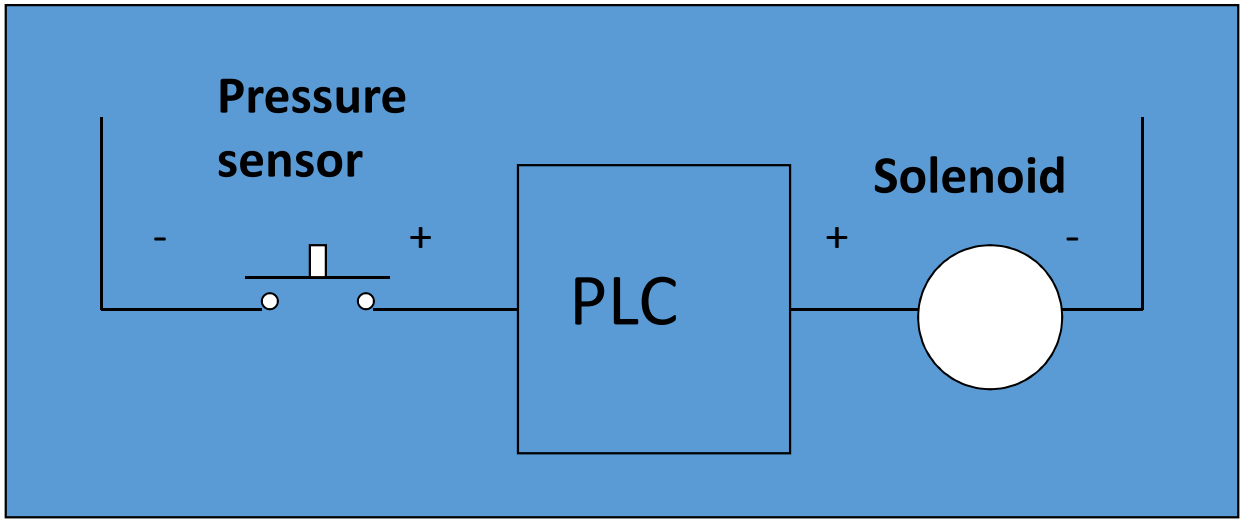
- Latching is the term for a self-maintaining circuit.



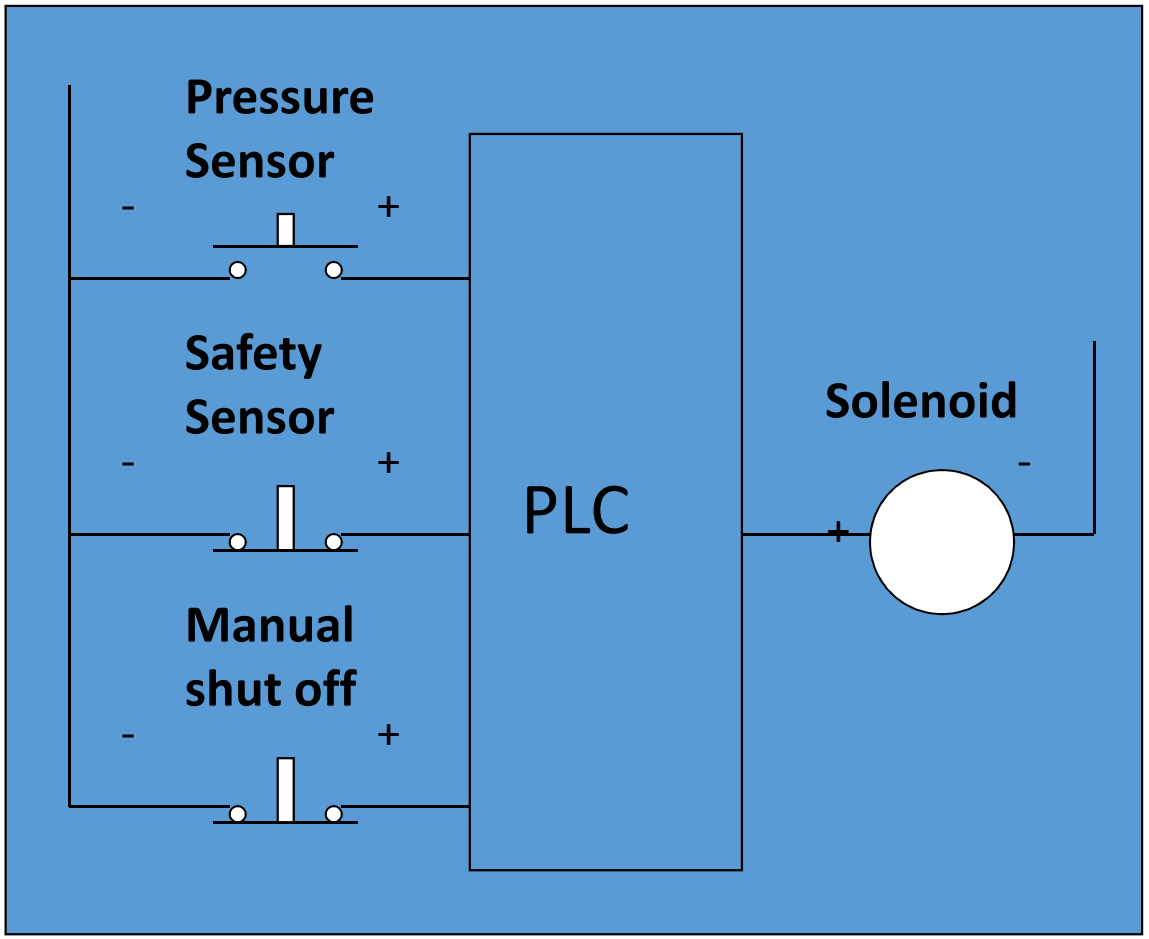
PLC ladder logic program



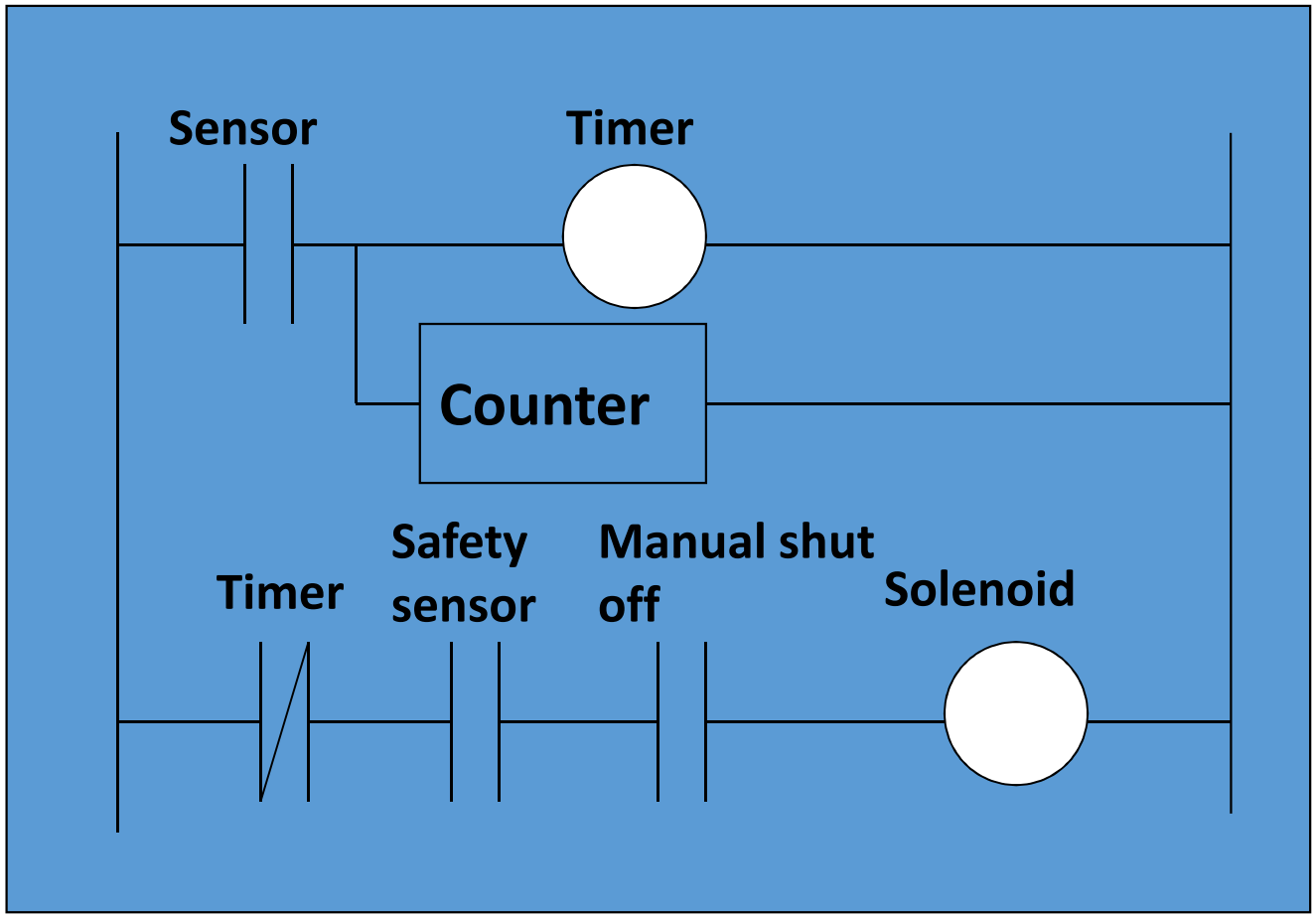
Wiring diagram for PLC



Wiring diagram for PLC



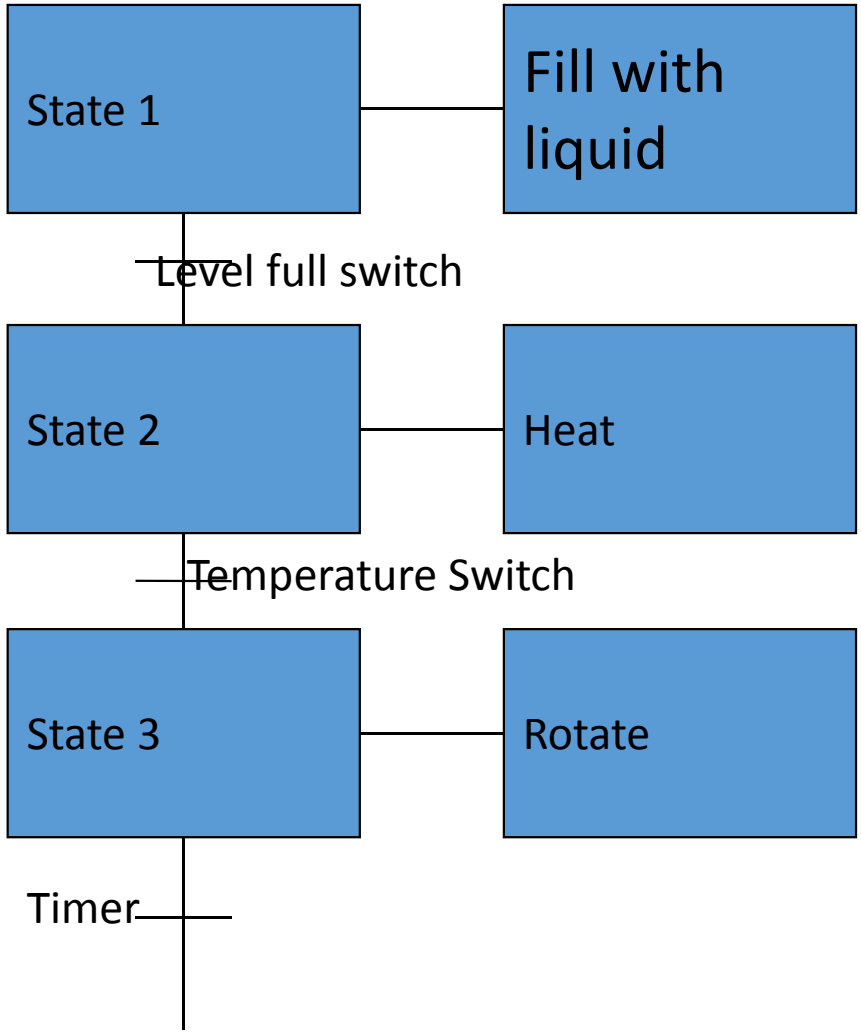
PLC ladder logic program

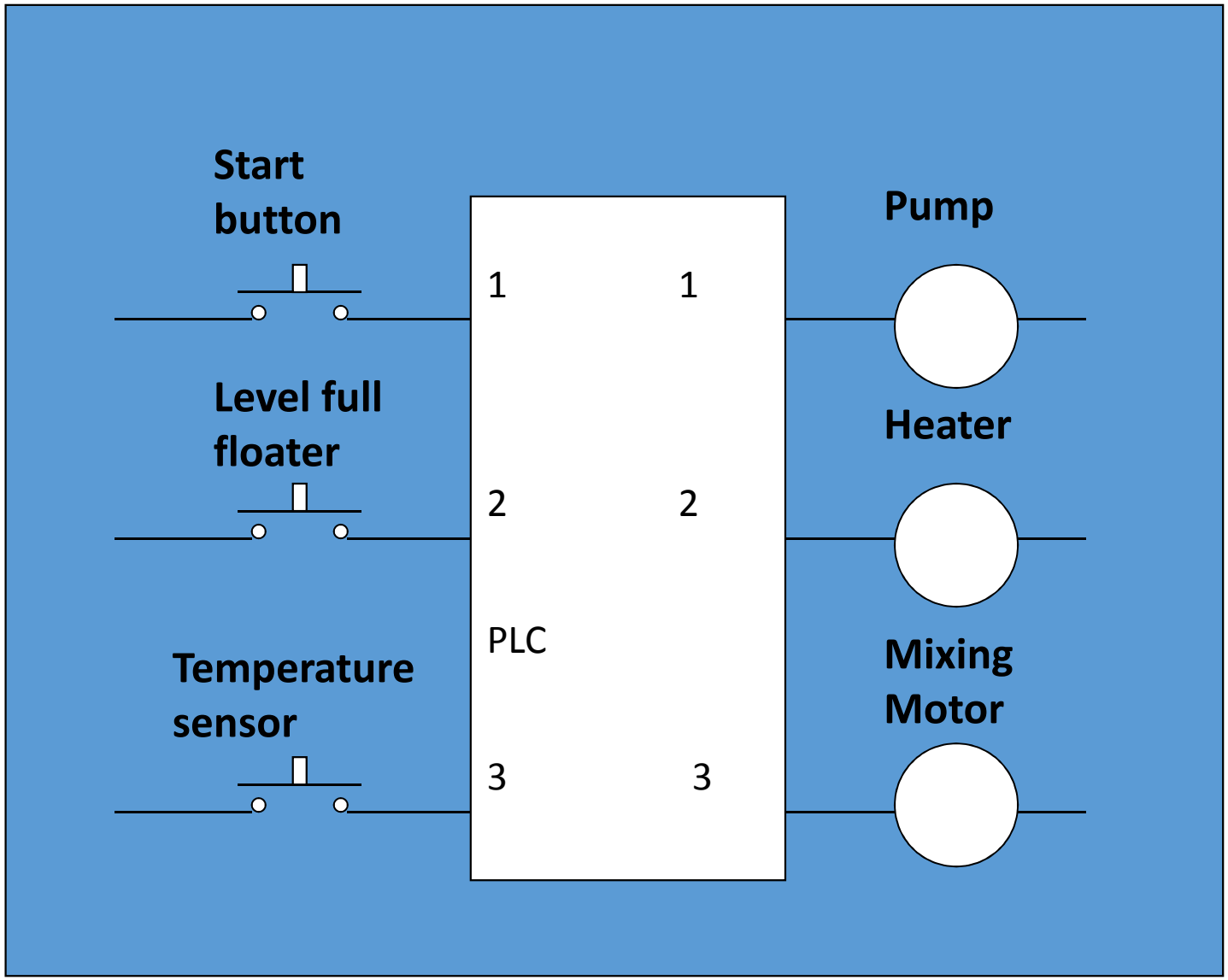


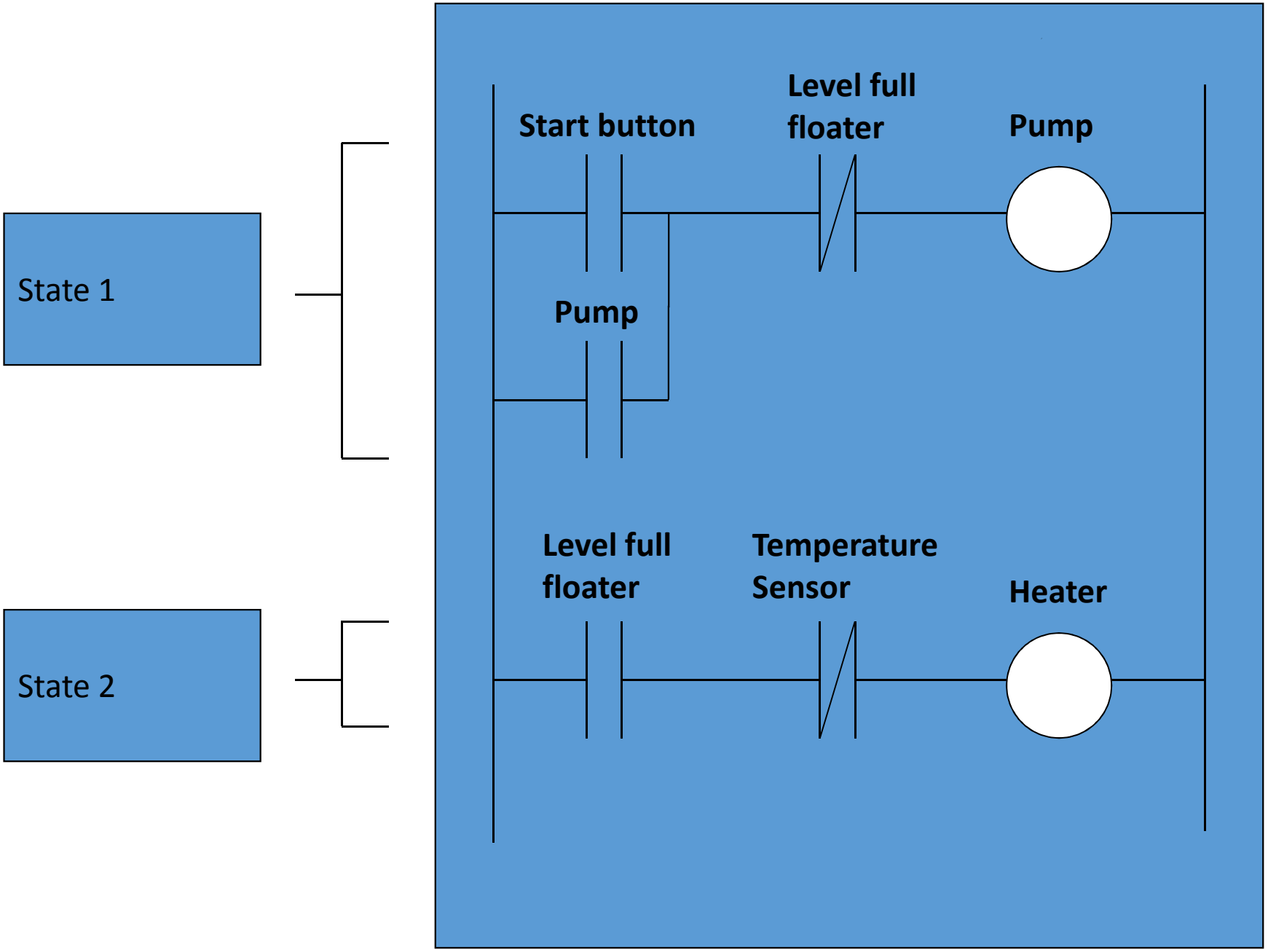
Industrial Mixer

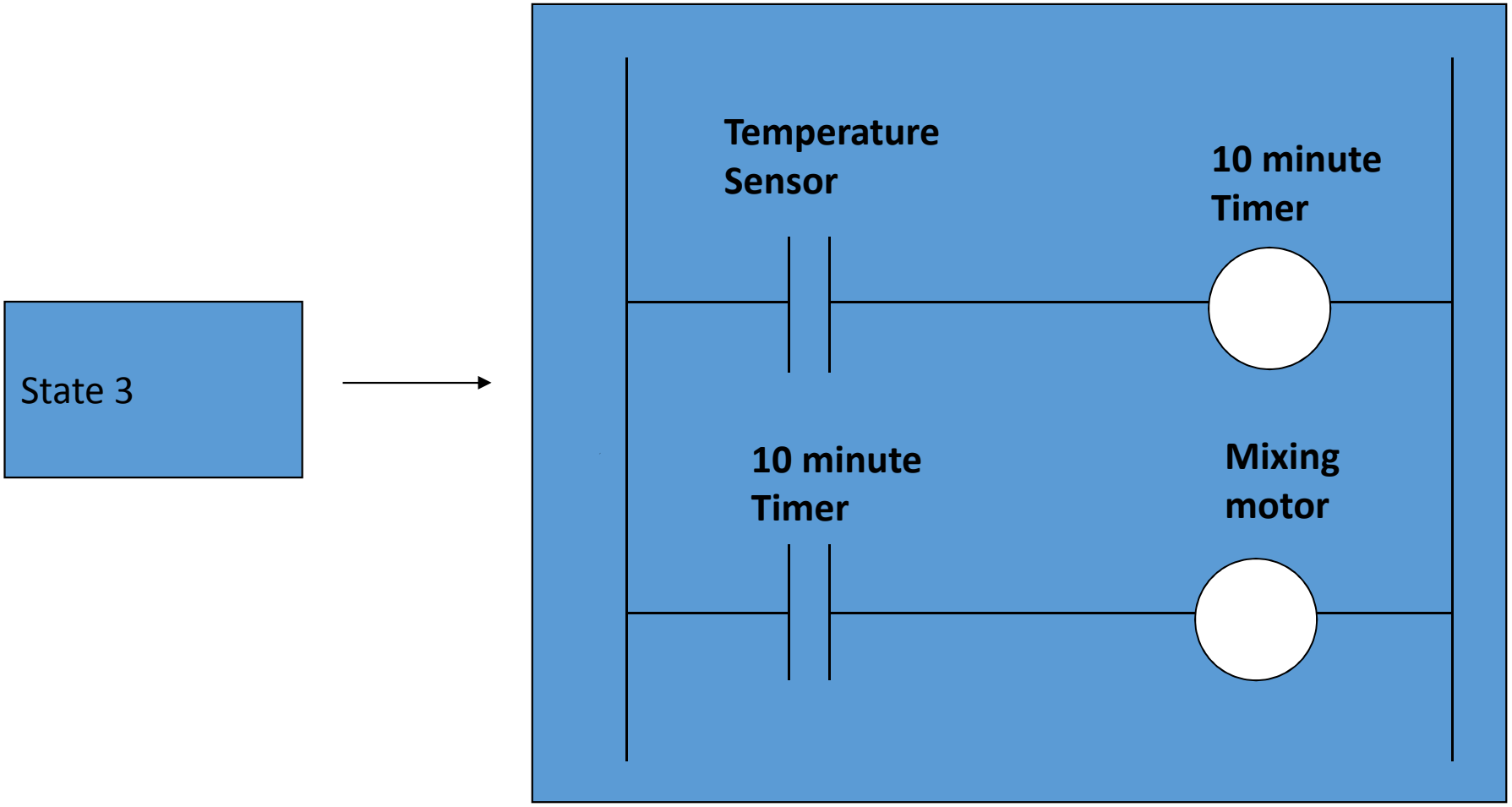
- Filled with liquid
- Heated
- Mix for 10 minutes.

Sequential Function Chart




Start button
Level full floater
Temperature sensor
1
2
3
PLC
1
2
3
Pump
Heater
Mixing Motor


State 1
State 2
Start button
Level full floater
Pump
Pump
Level full floater
Temperature Sensor
Heater


State 3
Temperature Sensor
10 minute Timer
10 minute Timer
Mixing motor

Ladder Logic

- Forcing
- Time checks
- Simulation
- Software testing

Since Ladder logic is essentially a computer program it is subject to bugs and faults. Therefore any program needs to be tested for accuracy and robustness. One method of testing is called forcing. This is where input states are forced to certain states in software. Programming errors can sometimes be found by forcing inputs at various stages in the ladder program.

Time checks can also be built into the ladder logic program. This is where additional ladder rungs might be includes so that when a function starts a timer is started. If the function does not complete when the timer finishes a fault is signaled. The function might be the moving of a piston or filling a drum with liquid.

Many PLCs have a simulate mode where the installed program can be run and inputs and outputs simulated so that they can be checked.

PLC ladder logic software can test against programming syntax errors.

Ladder Logic

- Registers and bits
- Data comparison
- Arithmetic operations
- Functions
- PID control

Ladder logic can do much more than what has been shown so far. It can utilize registers and bits to store and move data. There are data comparison function such as checking for equal to, less than or greater than. There are arithmetic operations such as addition, subtraction and multiplication. There are functions that can transform number to different bases or formats. Some PLCs can provide PID (proportional integral derivative) calculations to control a variable simply by being provided the necessary parameters.

PLCs solve problems

- Flexible
- Cost effective
- Computational abilities
- Trouble shooting
- Reliable

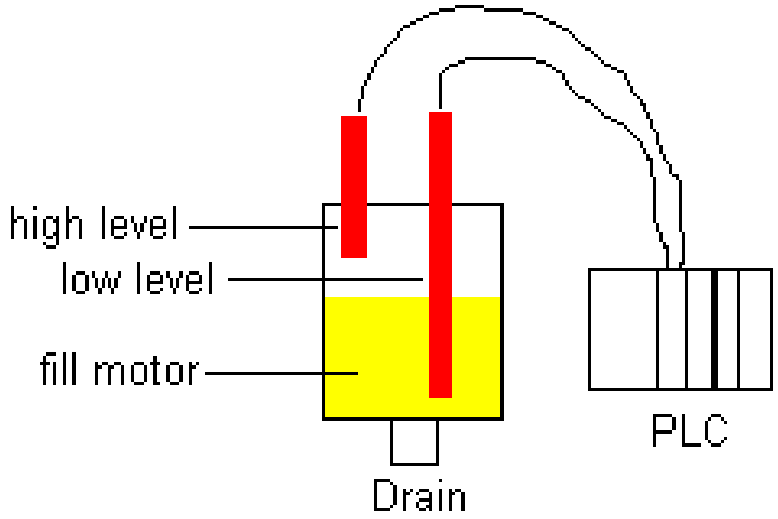
At this point you should realize how PLCs can solve many problems in industry.

PLCS are flexible and can be reapplied to control other systems quickly and easily. They are cost effective for controlling complex systems. They posses high computational abilities that allow more sophisticated control through ladder logic. Trouble shooting aids make programming easier and reduce downtime. Reliable components make PLCs likely to operate for years before failure. [1]

You should also realize that right now, if you had a PLC and a enough ladder logic knowledge you could construct a sophisticated machine with little problem as far as operational logic is concerned.

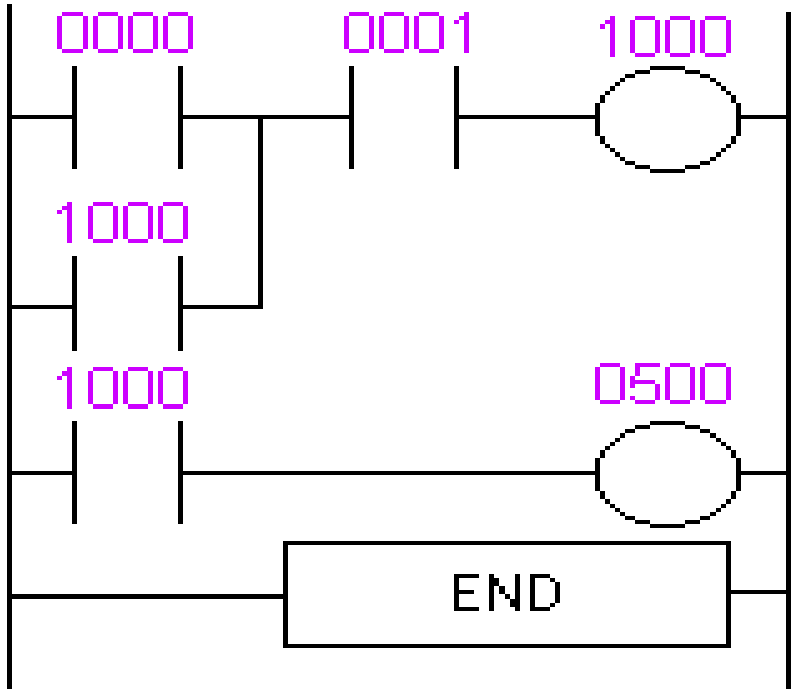
Programming a PLC

Oil is consumed randomly. The tank needs to be refilled by turning on a pump. Two hydrostatic switches are used to detect a high and low level.

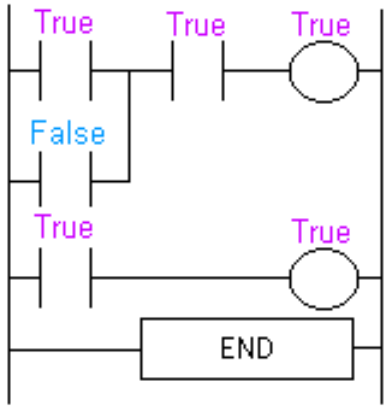


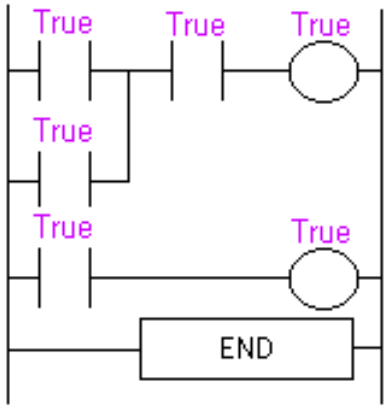
Dispensing oil from a tank

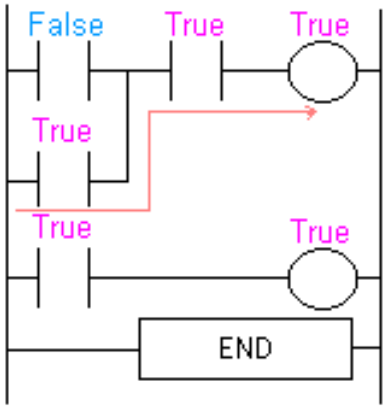
Ladder Logic for Tank

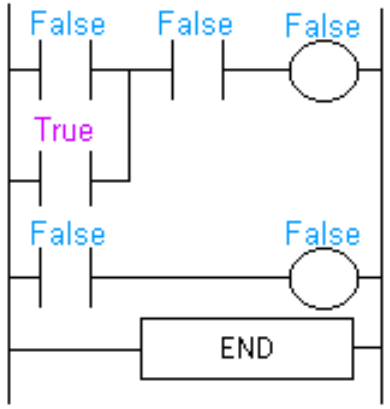


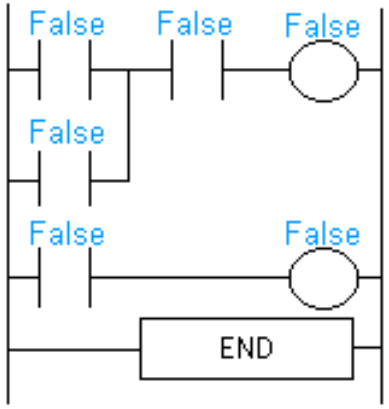
Logic for Ladder Solution

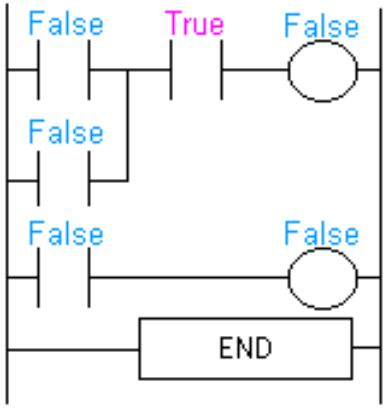




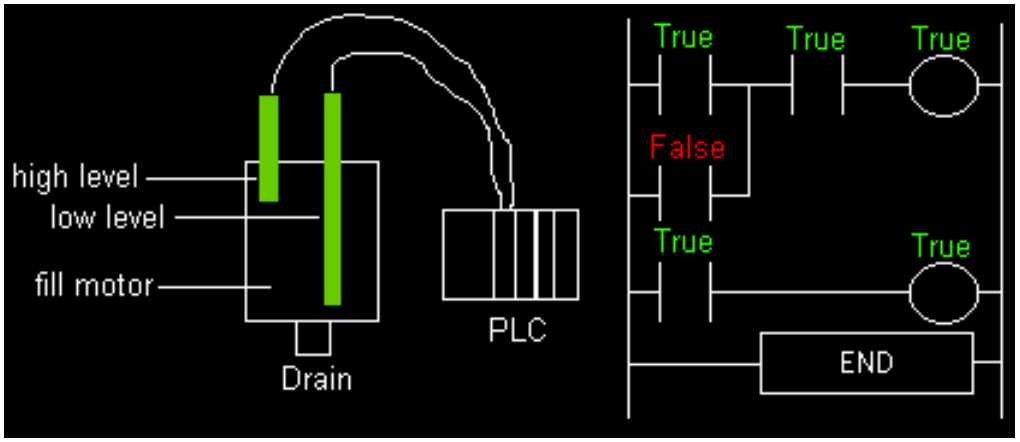




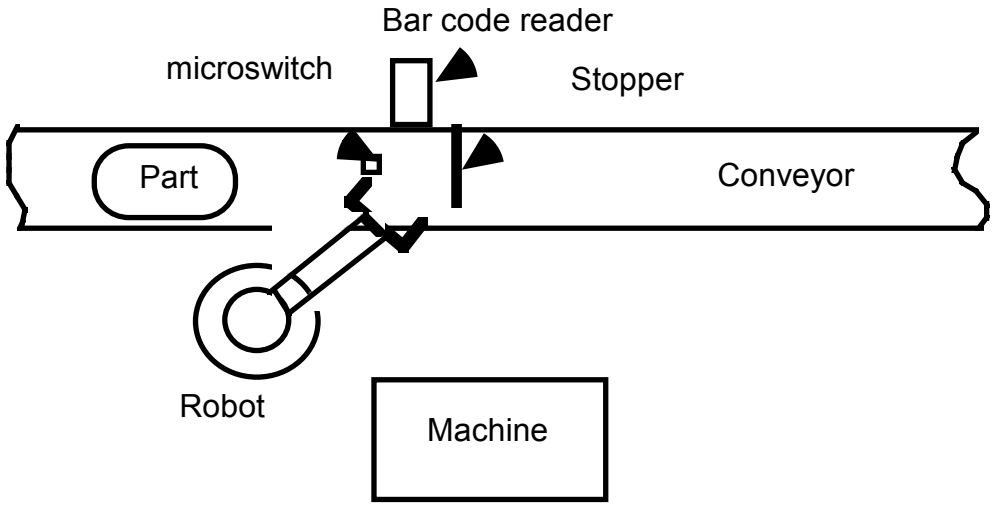




How does it work?

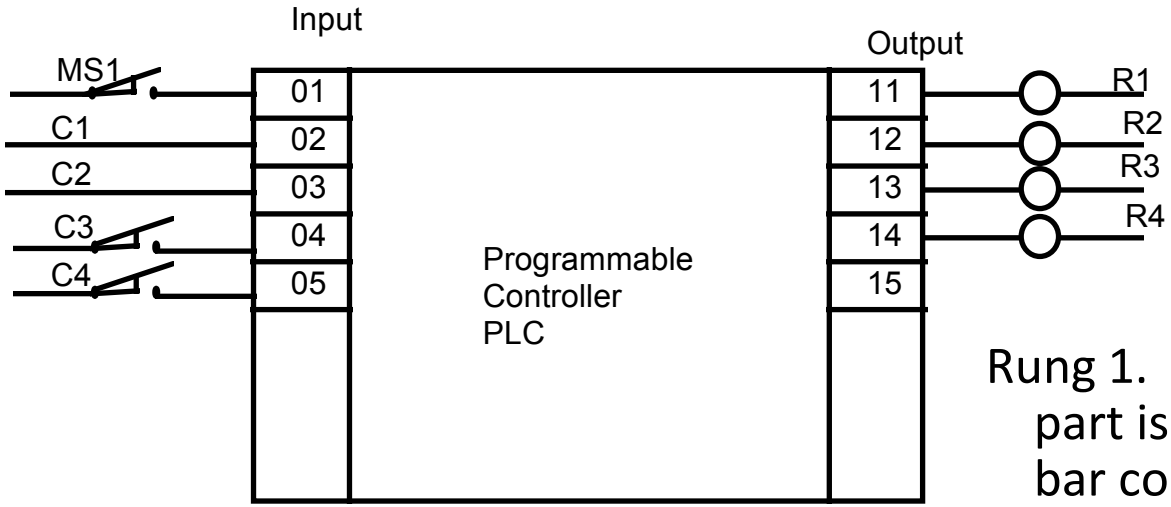


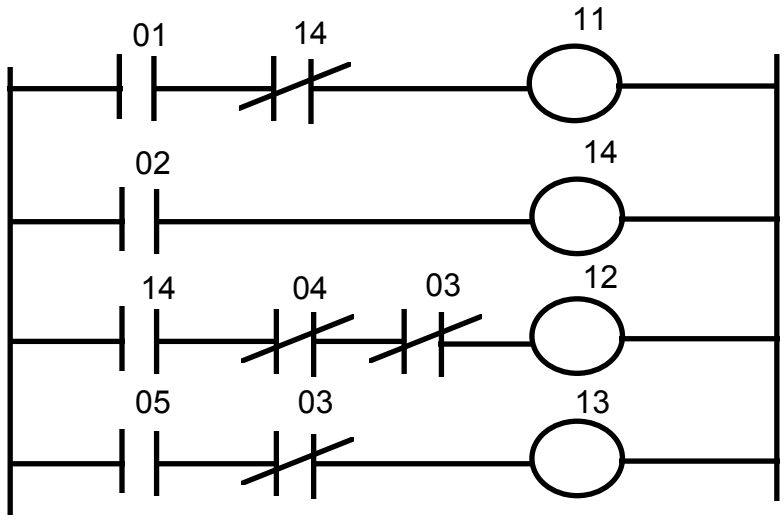
PROGRAMMING EXAMPLE 1



id	description	state	explanation
MSI	microswitch	1	part arrive
R1	output to bar code reader	1	scan the part
C1	input from bar code reader	1	right part
R2	output robot	1	loading cycle
R3	output robot	1	unloading cycle
C2 input from robot		1	robot busy
R4	output to stopper	1	stopper up
C3 input from machine		1	machine busy
C4	input from machine	1	task complete

SOLUTION





Rung 1. If part arrives and no part is stopped, trigger the bar code reader.

Rung 2. If it is a right part, activate the stopper.

Rung 3. If the stopper is up, the machine is not busy and the robot is not busy, load the part onto the machine.

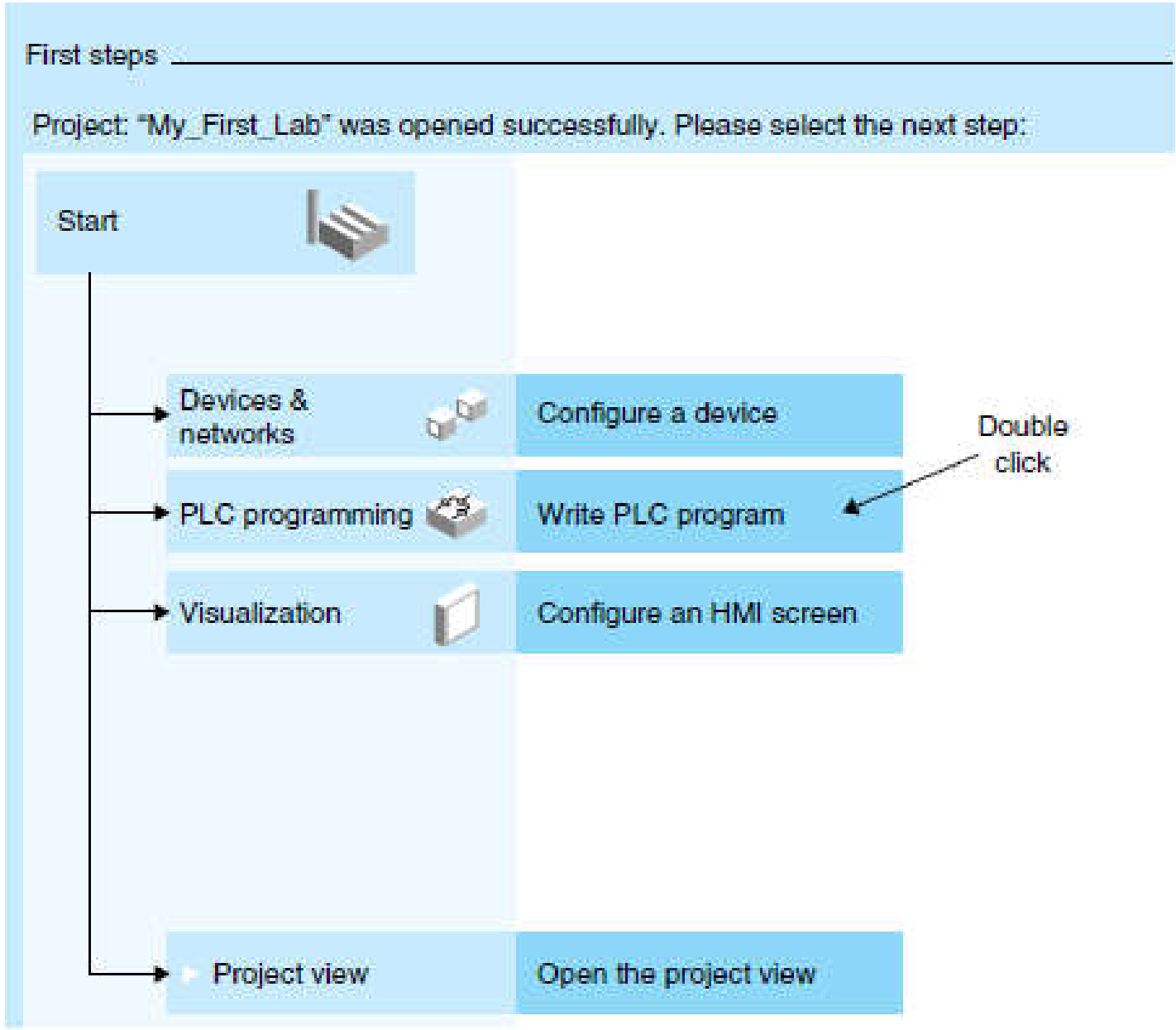
Rung 4. If the task is completed and the robot is not busy, unload the machine.

Софтуер за програмиране на контролери

Unity Pro - SoCollaborative software

Unity Pro е общият софтуер за програмиране, трасиране и експлоатация за гамите контролери Modicon Premium, Atrium и Quantum. Със своите пет езика по IEC61131-3 всички инструменти за трасиране и диагностика, Unity Pro е предназначен да увеличи вашата производителност при разработката и да улесни обслужването. С Unity Pro вашата инвестиция в софтуер се рационализира, разходите за обучение намаляват и вие ползвате предимствата на един несравним потенциал за разработка.

Повече информация - в продуктовия каталог.


First steps
Project: "My_First_Lab" was opened successfully. Please select the next step:
Start
Devices & networks
Configure a device
Double click
PLC programming
Write PLC program
Visualization
Configure an HMI screen
Project view
Open the project view