AUTÒMATS PROGRAMABLES o PLCs: VirPLC

INTRODUCCIÓ 2

EXEMPLES D'APLICACIÓ D'AUTÒMATS PROGRAMABLES 3
AVANTATGES DE L'AUTÒMAT 4
INTRODUCCIÓ ALS PLC's 5
CONCEPTE DE PROGRAMA 5

ARQUITECTURA DE LA UNITAT CENTRAL 6

ESTRUCTURA I DIRECCIONAMENT DE LA MEMÒRIA OMROM 6
DIRECCIONS I CONTINGUTS DE MEMÒRIA 8
ENTRADES I SORTIDES D'UN AUTÒMAT 9
BIT DE MEMÒRIA ASSOCIAT A UN ELEMENT 10

EL PROGRAMA 12

INSTRUCCION AND 16
INSTRUCCIÓ OR : 17
COMBINACIÓ DE FUNCIÓ AND I OR 18
CONDICIÓ D'EXECUCIÓ D'UNA INSTRUCCIÓ 22
INSTRUCCIONS SET I RESET 24
TEMPORITZADORS 25

PRÀCTICA AMB OMROM: CONTROL D'UNA ALARMA 27

PRÀCTIQUES AMB VIRPLC 28

PROJECTE FINAL DE LA UNITAT: 29

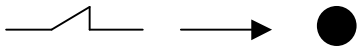
CANVI DE DIRECCIÓ EN MOTORS ASINCRONS MONOFÀSICS DE CONDENSADOR 32

INTRODUCCIÓ

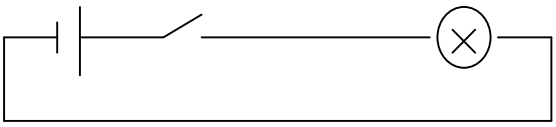
Els autòmats programable o PLCs (Controladors Lògics Programables) s'utilitza pel control de sistemes. Un **sistema** pot ser una màquina d'una indústria, un ascensor, una porta de supermercat,...
Bàsicament, un PLC és un microcomputador al que podem connectar uns elements d'entrada, uns elements de sortida i en el que podem gravar un programa que s'encarrega d'establir la relació que volem entre els elements d'entrada i els de sortida.
Elements d'entrada: interruptors, polsadors, detectors de presència, nivell, temperatura i, en general qualsevol sensor que pot subministrar algun tipus d'informació a l'autòmat.
Elements de sortida: motors, punts de llum, timbres, etc. En general qualsevol dispositiu que pot ser accionat per l'autòmat.

Si tenim un interruptor com element d'entrada i un punt de llum com element de sortida, amb un circuit elèctric convencional podem establir la relació següent entre aquests elements: ***quan tanquem l'interruptor, el punt de llum s'ha d'encendre***.

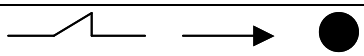
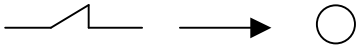
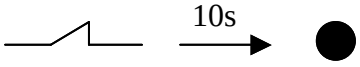
Podem representar aquesta relació de la forma següent :

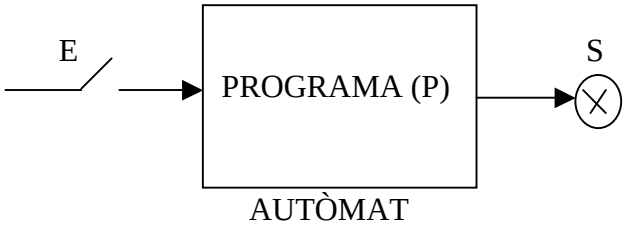


El circuit que aconsegueix aquesta relació és :



Amb un PLC podem aconseguir moltes més relacions entre aquests dos elements en fuinció del programa...

Quan tanquem l'interruptor volem que el punt s'encengui	
Quan tanquem l'interruptor volem que el punt s'apagui	
10 segons després de tancar l'interruptor, volem que el punt s'encengui	



S = f (E , P)

En aquest cas, el corrent que passa per l'interruptor quan el tanquem, no és el mateix que arriba al punt de llum i l'encén. És l'autòmat el que activa o no el punt de llum, en funció del programa que li hem gravat a la memòria.

L'estat de la sortida S depèn, no només de l'estat de l'entrada E, sinó també del programa P que està executant l'autòmat .És el programa el que determina que ha de fer l'autòmat amb el punt de llum quan detecta que hem tancat l'interruptor.

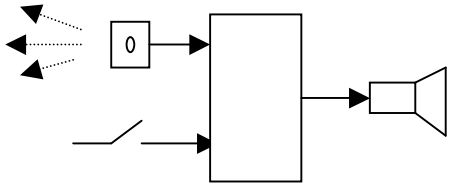
Direm que la sortida S és funció de l'entrada E i del programa P

L'autòmat executa un programa de treball que hem gravat dins la seva memòria. Aquest programa s'encarrega de processar uns senyals generats per uns dispositius d'entrada i obtenir uns resultats que, en funció del modes de funcionament del sistema a controlar, s'apliquen als dispositius de sortida.

Un ordinador també executa un programa de treball que relaciona dispositius d'entrada (teclat, ratolí, escaner) i de sortida (pantalla, impressora, etc). Un PLC té una arquitectura semblant a un ordinador, però està específicament pensat per aplicacions de tipus industrial.

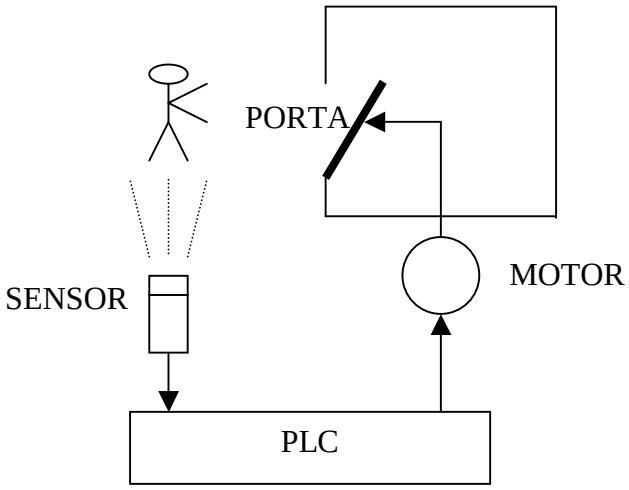
EXEMPLES D'APLICACIÓ D'AUTÒMATS PROGRAMABLES

Control d'un sistema d'alarma



Dispositius d'entrada : detector de presència i interruptor de desconnexió
Dispositiu de sortida : timbre

Control sistema d'obertura i tancament d'una porta de supermercat



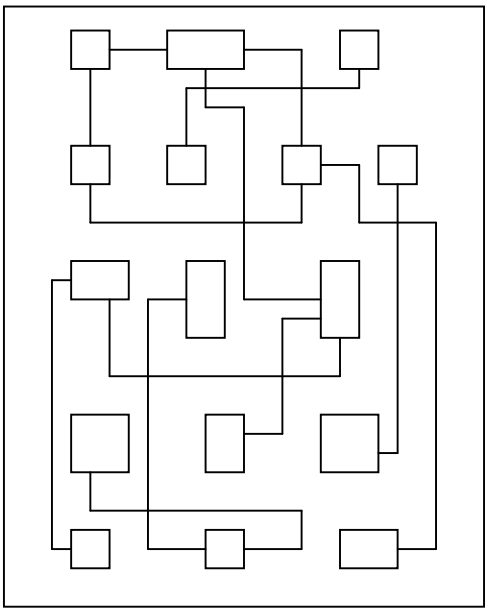
Control d'una cadena de fabricació

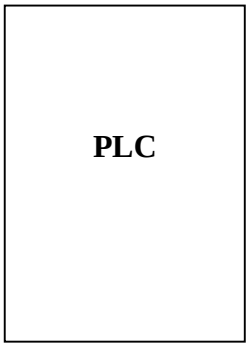
Control de les instal·lacions d'un habitatge
Podem projectar un habitatge amb un autòmat que controli les instal·lacions següents : obertura i tancament automàtic de les persianes de les finestres en funció del nivell de llum de l'exterior, accionament del sistema de calefacció, etc.

AVANTATGES DE L'AUTÒMAT

1.- Flexibilitat (canviant el programa modifiquem fàcilment el treball que desenvolupa)
Imaginem que la porta automatitzada ha funcionat durant un temps i ara volem modificar aquest funcionament. Suposem, per exemple, que volem comptar les persones que hi passen. Si hem fet una automatització tradicional haurem de desfer el cablatge, introduir un comptador i tornar refer el nou cablatge. En canvi, amb un autòmat, només haurem de modificar el seu programa intern amb el dispositiu de programació, no caldrà canviar cap cablatge. Aquesta **flexibilitat** a l'hora de modificar la seqüència d'automatització és un dels principals avantatges que ens proporciona l'autòmat.

2.- Reducció de l'espai necessari
Si una automatització requereix gran nombre d'elements (relés, comptadors, temporitzadors, etc.) i optem per comprar-los individualment i fer el cablatge , el conjunt resultant ocuparà molt espai i tindrà un cablatge complicat. Amb un autòmat estalviarem molt espai i simplificarem molt el cablatge.





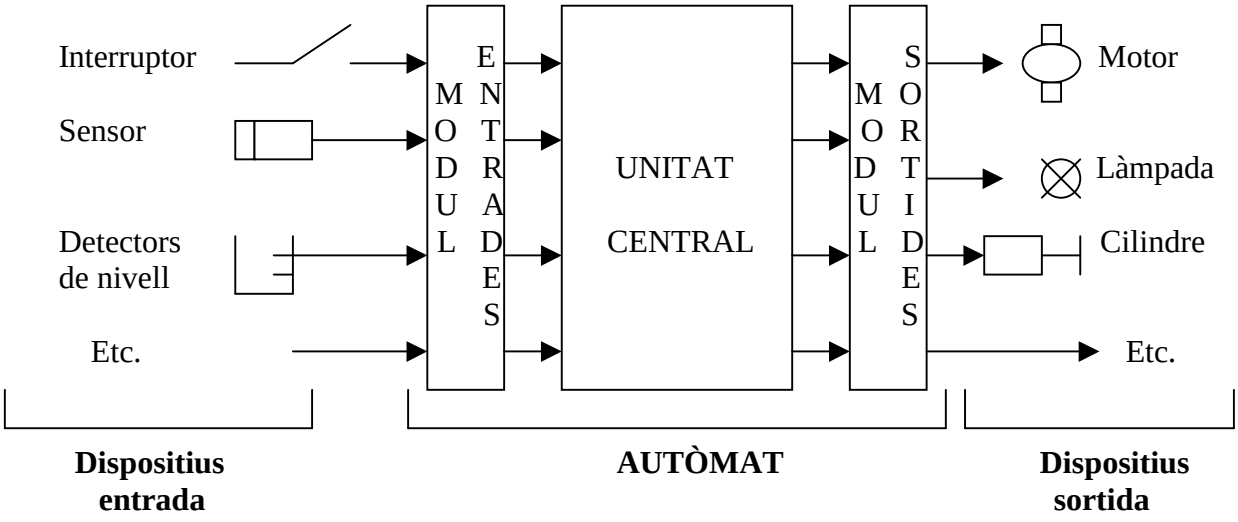
La figura de l’esquerra representa un armari elèctric amb una automatització desenvolupada amb tecnologia tradicional. Els quadres petits simbolitzen els diferents elements (relés, temporitzadors, etc.). La figura de la dreta representa el mateix sistema d'automatització implementat amb un autòmat, i vol visualitzar l'estalvi d'espai i simplificació de cablatge aconseguits amb l'ús del PLC.

3.- Reducció del cost econòmic (en sistemes un xic complexos)
Si un procés d'automatització requereix un nombre d'elements (relés, temporitzadors, comptadors, comparadors, etc.) mitjà o elevat, resulta més econòmic utilitzar un autòmat

4.- FIABILITAT: la principal de les avantatges des del punt de vista de la indústria.
Menys dispositius i menys electromecanics vol dir també menys possibilitat de fer-se malbé un d’ells.

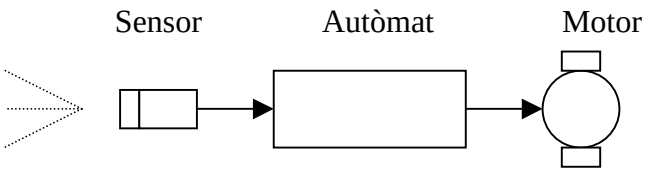
INTRODUCCIÓ ALS PLC's

Així, un autòmat és un dispositiu que executa un programa de treball que hem gravat dins la seva memòria. Aquest programa s'encarrega de processar uns senyals generats per uns dispositius d'entrada i obtenir uns resultats que s'apliquen als dispositius de sortida.



La figura superior representa una primera aproximació a l'arquitectura interna d'un autòmat. El dividim en 3 parts : mòdul d'entrades, unitat central i mòdul de sortides. Els mòduls d'entrades i sortides serveixen per connectar-hi els dispositius d'entrada i sortida. En la unitat central s'executa el programa que determina la seqüència d'automatització desitjada.

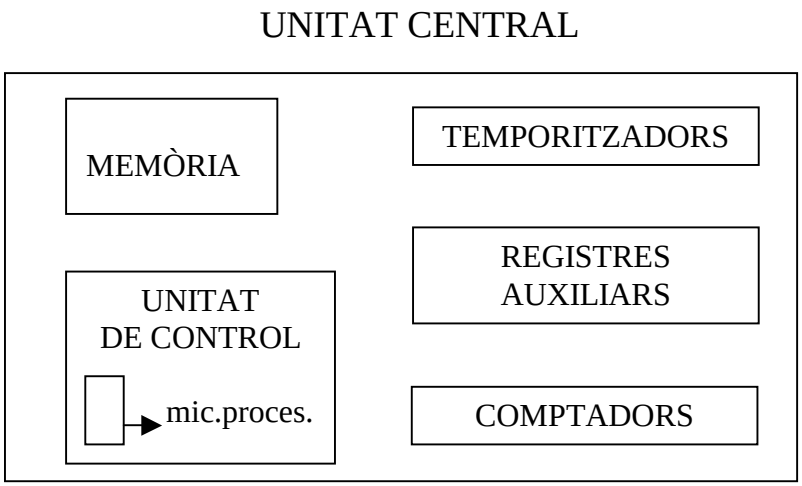
L'exemple inicial de l'automatització d'una porta es representaria de la següent forma :



CONCEPTE DE PROGRAMA

El programa que executa l'autòmat estableix de quina manera processem els senyals d'entrada per obtenir els de sortida. Resideix dins la unitat central en un lloc anomenat **memòria.** Programar l'autòmat serà carregar el programa dins la memòria. Aquest programa determinarà la feina que farà l'autòmat.

ARQUITECTURA DE LA UNITAT CENTRAL

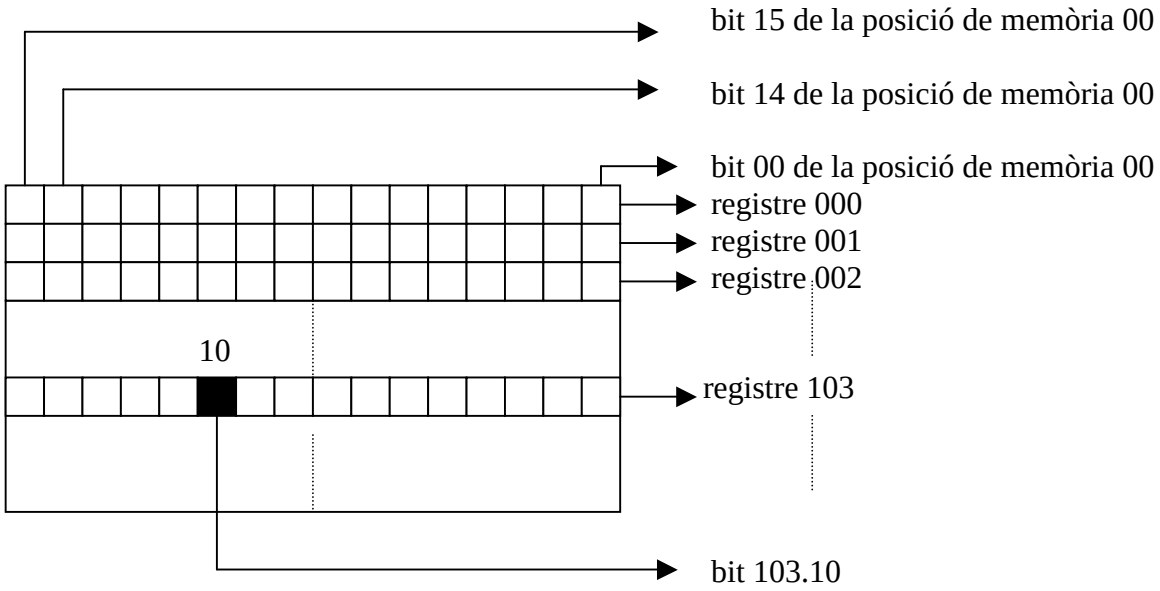


Memòria : element encarregat d'emmagatzemar el programa de treball
Unitat de control : element encarregat d'executar el programa. a més d'altres feines de control. Dins d'ella tenim el xip microprocessador que és el cervell de tot el sistema.
Temporitzadors, relés, comptadors, etc. : elements de treball que utilitzarem en funció del programa que haurà d'executar l'autòmat.
La unitat de control s'anomena també CPU (unitat de procés i control). Algunes vegades amb el terme CPU ens referirem només al xip microprocessador que forma part de la unitat de control.

ESTRUCTURA I DIRECCIONAMENT DE LA MEMÒRIA OMROM

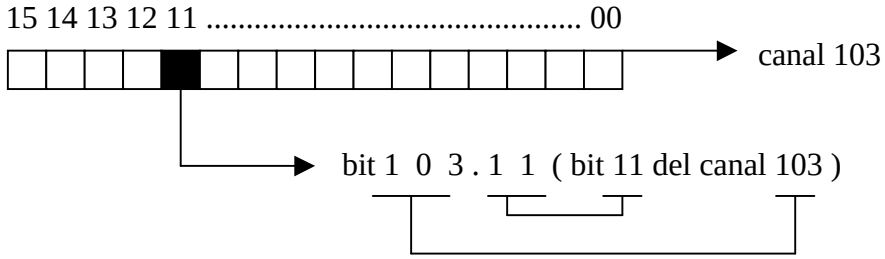
La memòria és el lloc on emmagatzemem el programa i les dades. Està estructurada en uns "calaixos" que s'anomenen **registres o posicions de memòria.** Cada calaix està format per 16 calaixets més petits anomenats **bits.** Quan més llarg sigui el programa, més posicions de memòria ocuparà.
Per poder accedir a un determinat bit, aquest ha de tenir un nom. Aquest nom s'anomena **direcció.** La forma d'associar direccions als bits s'anomena **direccionament de la memòria.** Cada fabricant d'autòmat pot definir un tipus de direccionament propi. A continuació explicarem el direccionament que utilitza la casa comercial Omron, fabricant dels autòmats que utilitzarem en les nostres pràctiques.

A cada registre li assignem una direcció formada per 3 dígits. Dins d'un registre, designarem cada un dels seus 16 bits amb 2 dígits més.
Nosaltres treballarem amb autòmats de la casa comercial Omron. En la terminologia d'aquesta casa les posicions de memòria s'anomenen **canals o registres.**



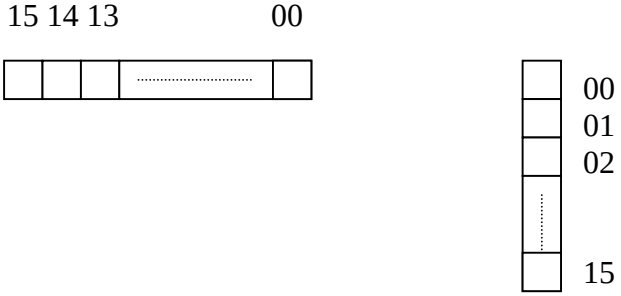
Quan calgui direccionar més de 999 registres, utilitzarem 4 dígits en comptes de 3. Per tant, pot ser que la direcció d'un registre tingui més de 3 dígits

Els 16 bits d'un registre es designem amb els dígits que van del 00 al 15 (no al 16). La direcció complerta d'un bit tindrà 5 dígits : els 3 primers designaran la posició de memòria i, els 2 restants el bit d'aquesta posició de memòria.



Si ens referim a una posició de memòria amb menys de 3 dígits caldrà suposar que els primers dígits són zeros. Així la posició 10 i la 010 són la mateixa .

En el decurs d'aquests apunts podrem dibuixar els canals horitzontals o verticals :

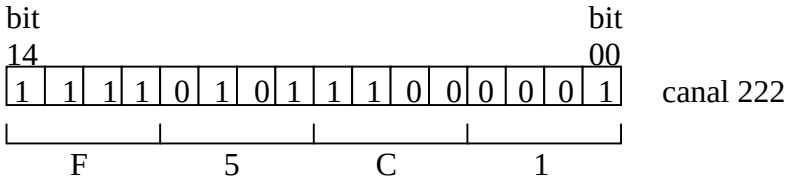


Del total de posicions de memòria o canals, n'hi ha uns quants que podem fer servir en els nostres programes com a registres de treball, són els 32 registres de la zona de memòria anomenada IR que tenen la direcció compresa entre 200 i 231.

No feu servir altres canals diferents dels compresos entre 200 i 231 com a canals de treball dels vostres programes.

La memòria de l'autòmat OMRON - SYSMAC - CPM1A es divideix en 8 zones o àrees : àrea IR, àrea SR, àrea TR, àrea HR, àrea AR, àrea LR, àrea de temporitzadors-comptadors i àrea DM. Cadascuna d'aquestes àrees te una funció especifica. Dins l'àrea IR tenim els bits de treball que podem fer servir lliurament en els nostres programes. Són els continguts en els canals 200-231. En total tindrem 32 canals X 16 bits = 512 bits de treball. Les direccions d'aquests bits estan compreses entre IR 20000 i IR 23115.

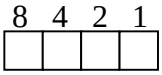
DIRECCIONS I CONTINGUTS DE MEMÒRIA



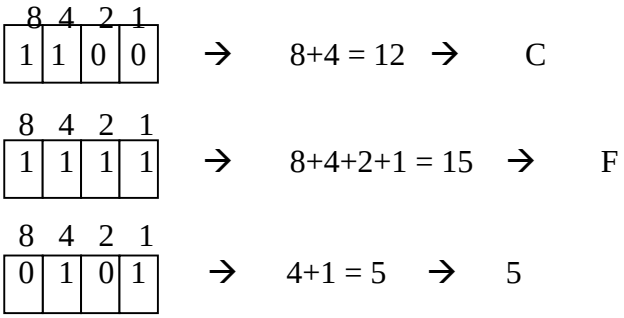
El dibuix de la figura representa el contingut binari de la posició de memòria o canal 222. Aquest contingut binari es representa en codi hexadecimal (F5C1).

El codi hexadecimal utilitza 16 símbols alfanumèrics : 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

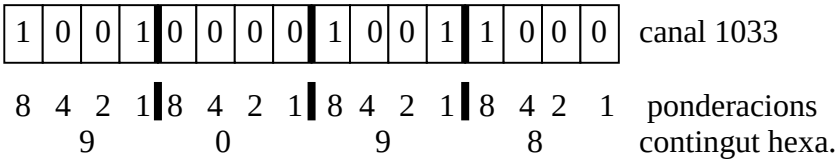
Els bits binaris s'agrupen de 4 en 4 i els hi assignem les següents ponderacions:



Per determinar el símbol hexadecimal que correspon als 4 bits caldrà sumar les ponderacions dels bits que estiguin a "1". Alguns exemples aclariran aquest tema:



Veiem un altre exemple de contingut d'un canal :



En aquest exemple el contingut del canal 1033 és 9098 en codi hexadecimal. Escriurem (1033) = 9098
Cal no confondre la direcció del canal amb el seu contingut . La **direcció** és el nom i el contingut és la **dada** que hi tenim a dins. Si no posem el parèntesi ens referim a la direcció, amb el parèntesi ens referim al contingut o dada

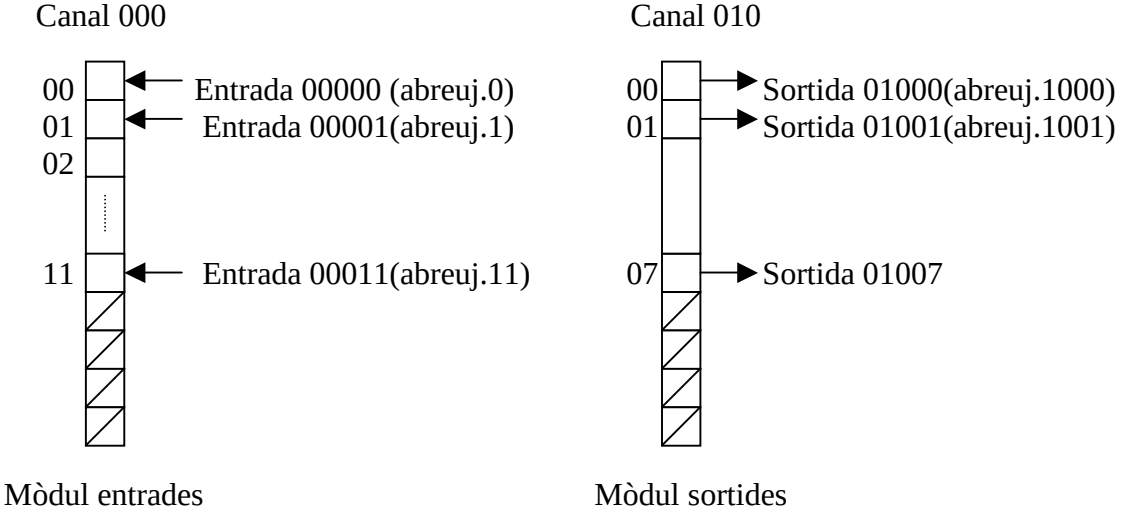
1033 : direcció del canal
(1033) : contingut del canal = 9098 (codi hexadecimal)
(1033) = 1011000010011111 (codi binari)

Una manera de diferenciar la direcció d'un canal de la dada que emmagatzema és l'ús del símbol #. Aquest símbol indica que el número que tenim a continuació és una dada, no una direcció.

1033 : direcció # 9098 : dada

ENTRADES I SORTIDES D'UN AUTÒMAT

Els mòduls d'entrada i sortida s'associen a uns canals determinats de l'autòmat. Així per exemple les 12 entrades de l'autòmat de la casa Omron Sysmac CPM1A estan associades al canal 000 (abreujadament 0) i les seves 8 sortides als canal 010 (abreujadament 10). Això vol dir que aquest model d'autòmat permet la connexió de 12 dispositius d'entrada (interruptors, sensors, etc) i 8 dispositius de sortida (motors, cilindres pneumàtics, llums, timbres, etc.)



Si connectem un dispositiu d'entrada (per exemple un interruptor) al bit 00011 , la direcció d'aquest interruptor passarà a ser 00011.

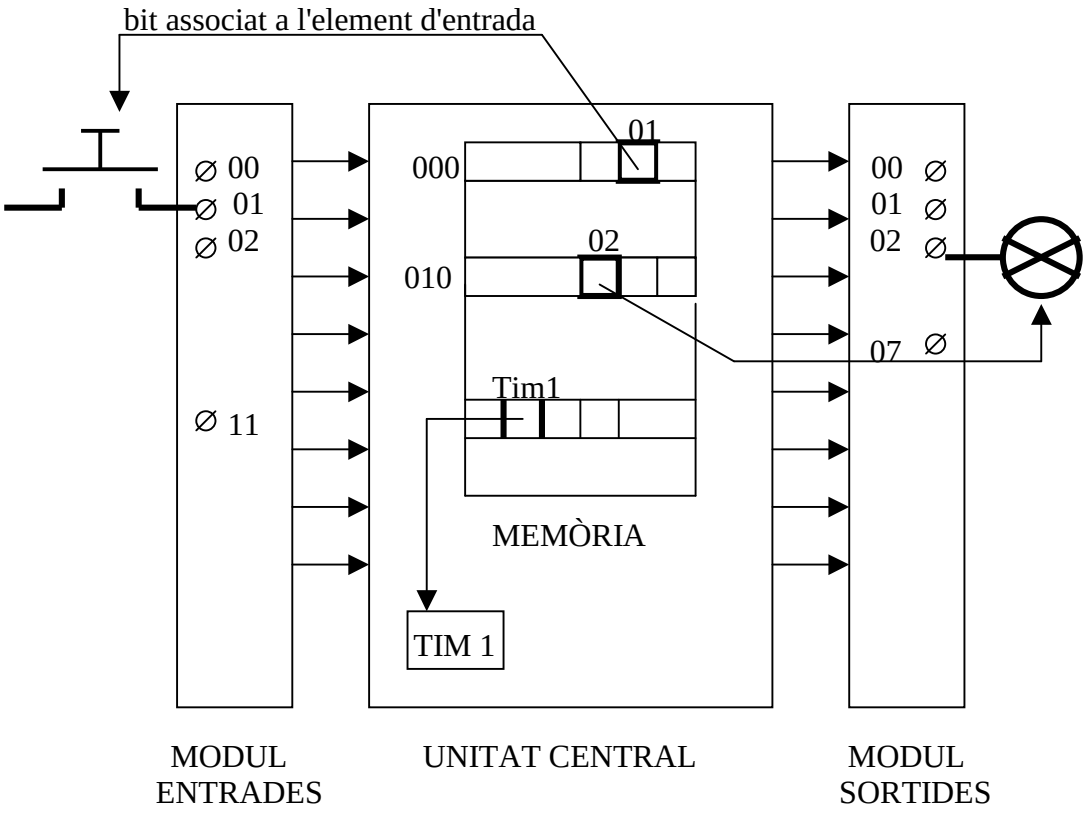
BIT DE MEMÒRIA ASSOCIAT A UN ELEMENT

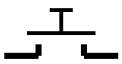
Elements
- externs
  - dispositius d'entrada (interruptors, polsadors, sensors, etc)
  - dispositius de sortida (leds, motors, cilindres, etc)
- interns (temporitzadors, comptadors, comparadors, etc)

Utilitzem un element d'entrada, per exemple un interruptor, per explicar el concepte de bit de memòria associat a un element

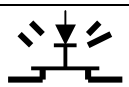
Imaginem que tenim un polsador connectat en l'entrada 000.01 i un led connectat a la sortida 010.03. Volem que l'autòmat executi un programa que encengui el led en el moment de tancar el polsador. L'autòmat necessita saber en tot moment si tenim el polsador obert o tancat. Com s'ho fa ? surt un senyor de dins l'autòmat per comprovar-ho ?, evidentment no.

El que realment passa és que dins la memòria de l'autòmat es reserva un bit que s'associa a l'element d'entrada i s'encarrega de registrar l'estat d'aquest element. Quan l'autòmat vol saber com tenim el polsador només ha de consultar l'estat d'aquest bit. Com reflecteix aquest bit l'estat del polsador ?. Molt fàcil, quan tanquem el polsador entrem un senyal de 24 volts dins el mòdul d'entrades i el bit es col·loca automàticament en "1". Quan el polsador està obert tindrem un "0" en el bit associat.



CONTACTE OBERT 

Adreça		01	
Bit de memòria		**1**	

CONTACTE TANCAT 

Adreça		01	
Bit de memòria		**1**	

El mateix raonament podem fer per un element de sortida. El led que hem connectat a la sortida 010.02 del mòdul de sortides tindrà un bit de memòria (el 02) situat en el canal 010 que registrarà l'estat del led (encès o apagat)

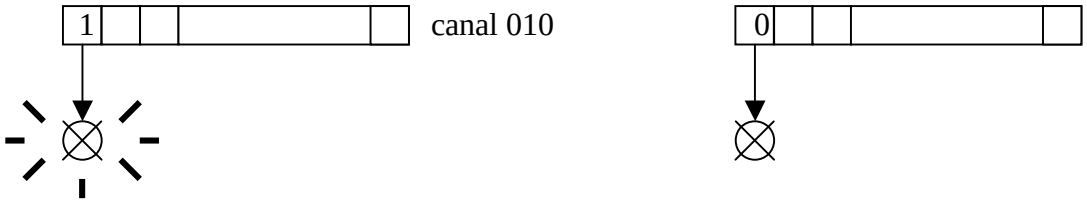
PUNT DE LLUM OFF 

Adreça		000.01	
Bit de memòria		**1**	

PUNT DE LLUM ON 

Adreça		010.02	
Bit de memòria		**1**	

Si l'autòmat ha d'executar una instrucció per encendre el led , primer escriurà un "1" en el bit corresponent del mapa de memòria i després trasllada aquest "1" a l'element associat al bit , en el nostre cas el led.



Els elements interns de l'autòmat (temporitzadors, comptadors, comparadors, etc.) també tenen un bit associat. L'estat "1" o "0" d'aquest bit ens informa de l'estat de l'element que considerem. Per exemple, quan un temporitzador acaba de temporitzar un temps determinat, el seu bit associat canvia de 0 a 1. Aquests bits de memòria que registren l'estat dels elements de l'autòmat s'anomenen **"flags"** o banderes

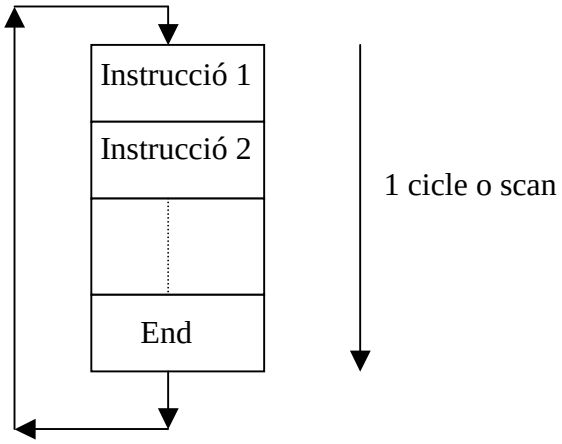
Resumim:
Cal que diferenciem els conceptes d'element d'entrada (per exemple un interruptor) i bit d'entrada associat a aquest element. L'interruptor podrà estar obert o tancat i el seu bit podrà estar a 0 o a 1. Hem estudiat que existeix una relació directe entre l'estat del dispositiu i el del seu bit.

bit sortida = 0 → led apagat	Interruptor **obert** → bit entrada = **0**
bit sortida = 1 → led encès	Interruptor **tancat** → bit entrada = **1**

Amb la mateixes paraula "entrada", "sortida", unes vegades ens referirem a l'element d'entrada o sortida (interruptor/punt de llum) i altres al bit associat a aquest element.

EL PROGRAMA

El programa que executa el PLC està format per un seguit d'instruccions. Cada instrucció ocupa una o més posicions de memòria . L'autòmat executa el programa d'una forma cíclica i continuada i quan troba la instrucció END torna executar la primera instrucció.



Una execució del programa (una passada del programa) s'anomena **cicle** o **scan**. A cada cicle l'autòmat avalua (scaneixa, escombra) totes les instruccions i les executa.
El temps d'scan (temps que l'autòmat tarda en fer una passada al programa) depèn del nombre d'instruccions que tingui el programa però, sempre serà un temps molt i molt petit i en conseqüència, és com si una instrucció l'estiguéssim executant contínuament.

Si en una instrucció activem un temporitzador per comptar 10 segons cal tenir en compte que l'autòmat no s'atura en aquesta instrucció fins que han passat els 10 segons, sinó que continua executant la resta d'instruccions de forma continuada. Quan hagin passat els 10 segons l'autòmat executarà la operació lligada amb el temporitzador.

Per introduir de forma més clara el tema de la programació ho farem a cavall d'un exemple senzill (pel que no caldria de cap manera l'ús d'un autòmat).

Exemple 1
Imaginem que volem connectar un interruptor e1(entrada 1) i un punt de llum s1(sortida 1)a l'autòmat. Quan tanquem l'interruptor caldrà que s'encengui el punt de llum.
Primer cal triar a quina de les dotze entrades connectem l'interruptor i a quina de les 8 sortides connectem el punt de llum. Imaginem que triem l'entrada “e1” i la sortida “s1”.

Quan tanquem l’interruptor, arriba Vcc al autòmat i el bit d’entrada que té associat es posa a 1. Però, per encendre el led , caldrà que el programa posi a 1 el bit de sortida que té associat :

La seqüència sencera serà :

Quan tanquem l'interruptor, el bit d'entrada "e1" es posa a 1. El programa detecta aquest 1 i posa a 1 el bit de sortida "s1". Aquest 1 de sortida fa que el led s'encengui.
Quan tenim l'interruptor obert, arriba 0V a l'autòmat i el bit d'entrada associat es posa a zero. Quan el bit de sortida estigui a 1, el seu led associat estarà encès. Quan el bit de sortida estigui a 0, el seu led associat estarà apagat.

El programa que executa la funció desitjada està format per 3 instruccions:

Funció	Instruccions
Entrada = 1 → Sortida = 1 Entrada = 0 → Sortida = 0	LD e1 OUT s1 END

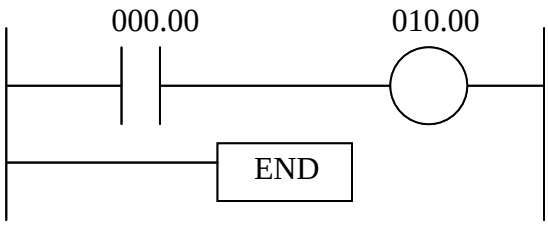
LD e1: LD és abreviació de la paraula anglesa LOAD que vol dir carregar. Quan l'autòmat troba aquesta instrucció "carrega", es a dir **llegeix**, l'estat del bit d'entrada "e1" . Aquest bit pot estar a "1" si l'interruptor esta tancat, o a "0" si està obert.

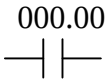
OUT s1: posa els resultat de la lectura a la sortida "s1". Si es un "1" el led s'encendrà, i si ja estava encès, ningú ho notarà. Si estava a "0" el led s'apagarà i si ja estava apagat, ningú ho notarà.

END: Fi del programa. Quan l'autòmat troba aquesta instrucció, salta a executar la primera instrucció del programa. En el VirPLC aquesta instrucció no fa falta.

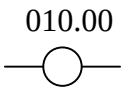
En resum, amb aquestes 3 instruccions l'autòmat vigilarà l'estat de l'entrada considerada i quan canviï a "1" (canvi que es produirà quan tanquem l'interruptor) activarà la sortida 010.00 i s'encendrà el punt de llum que hi hem connectat.

Les instruccions d'un programa es poden representar gràficament en uns diagrames anomenats **diagrames de contactes**, el que en anglès anomenen **"ladder programs"**. Aquests diagrames són molt importants a l'hora de programar l'autòmat. En el nostre cas, tindria el següent aspecte

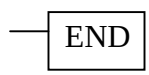


El símbol 

és equivalent a la instrucció LD 000.00

El símbol 

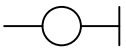
és equivalent a la instrucció OUT 01000. La circumferència representa sempre una sortida

El símbol 

és equivalent a la darrera instrucció END

Abans d'aprofundir el tema de la programació i estudiar les instruccions , veiem com ho faríem per aconseguir que l'autòmat de la cas OMROM executes el programa que ens ha servit d'exemple. Com a dispositiu de programació podem utilitzar la consola de programació o un ordinador. Amb l'ordinador haurem de considerar les següents fases :

1- Obrim el programa SYSwin: Si l'ordinador te la drecera d'entrada d'aquest programa situada en el escriptori, només caldrà que cliquem la icona corresponent. En cas contrari caldrà anar al menú Inici / programes / SYSwin / SYSwin

2- Introduïm el diagrama de relès: Cliquem (premem el botó esquerre del ratolí i el deixem anar) la icona  situada a la part esquerra de la pantalla i l'arrosseguem fins el quadre negre de la pantalla. Quan tenim el símbol dins el lloc desitjat tornem a clicar el ratolí i veurem que s'obre un menú que ens permet escriure la direcció de l'entrada, 00000 en el nostre cas. Només cal teclejar un 0 i prémer acceptar.
Repetim la mateixa seqüència de moviments amb el símbol 
Situem el cursor sobre el rectangle gris "final de bloc" i cliquem el ratolí dues vegades per obrir una nova línia en el diagrama de contactes
Cliquem la icona FUN i l'arrosseguem fins el quadre negre, tornem a clicar i teclegem 01 dins el quadre blanc (01 és el codi de la instrucció END). Cliquem acceptar i veurem com queda situada la instrucció de final de programa.

3- Obrim la comunicació entre ordinador i autòmat: Entrem dins el menú online i escollim la opció connectar (online). En aquest moment l'ordinador comprovarà si es pot comunicar correctament amb l'autòmat i ens adreçarà el missatge corresponent.

4- Col·loquem l'autòmat en posició programació (stop / prog.): Anem al menú online/mode i de les 3 possibilitats que ens ofereix el programa triem stop/prog. Finalment cal confirmar que volem canviar l'estat de l'autòmat.

5- Transferim el programa des de l'ordinador (PC) a l'autòmat: Anem al menú online / transferència PC → PLC . El programa ens avisarà si la transferència s'ha efectuat correctament.

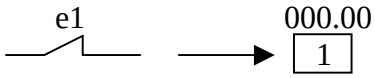
6- Col·loquem l'autòmat en posició execució del programa (run): Anem al menú online / mode, i cliquem opció RUN. Caldrà confirmar que volem canviar l'estat del PLC.

A partir d'aquest moment l'autòmat està executant el programa. Contínuament està comprovant l'estat de l'entrada 00000 i quan la troba a "1" activa la sortida 01000. Si tanquem l'interruptor connectat a l'entrada 00000, veurem com s'encén el led corresponent a la sortida 01000 .
Si volem visualitzar en la pantalla de l'ordinador com es va executant el programa caldrà anar al menú online i accedir al submenú monitorizar.

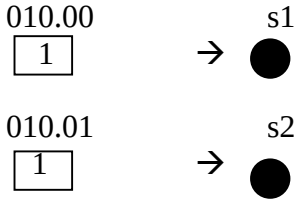
Exemple 2

Tenim un autòmat amb un interruptor e1 connectat a l'entrada 000.00i dos leds s1. i s2 connectats a les sortides 010.00 i 010.01. Quan tanquem l'interruptor, volem que s'encenguin el dos leds simultaniament.

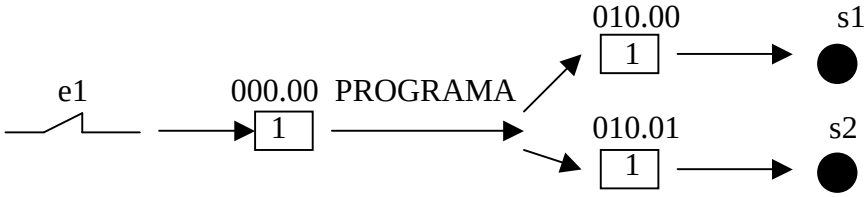
Quan tanquem l'interruptor e1, arriba corrent al autòmat i el bit d'entrada 000.00 que té associat es posa a 1 :



Per encendre els leds s1 i s2, caldrà que el programa posi a 1 els bits de sortida que tenen associats:

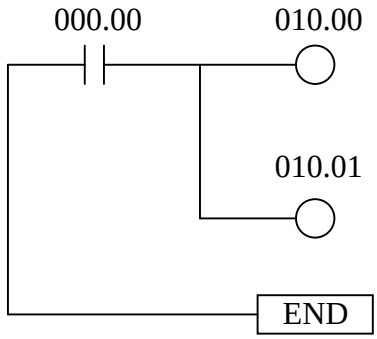


La seqüència sencera serà :

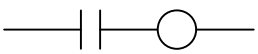


El programa que soluciona el problema plantejat és :

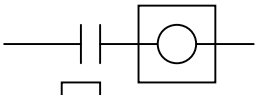
```
LD 000.00
OUT 010.00
OUT 010.01
END
```



Per inserir la línia vertical en el diagrama de contactes amb el software de programació de l'autòmat OMRON cal :

a) dibuixar la línia 

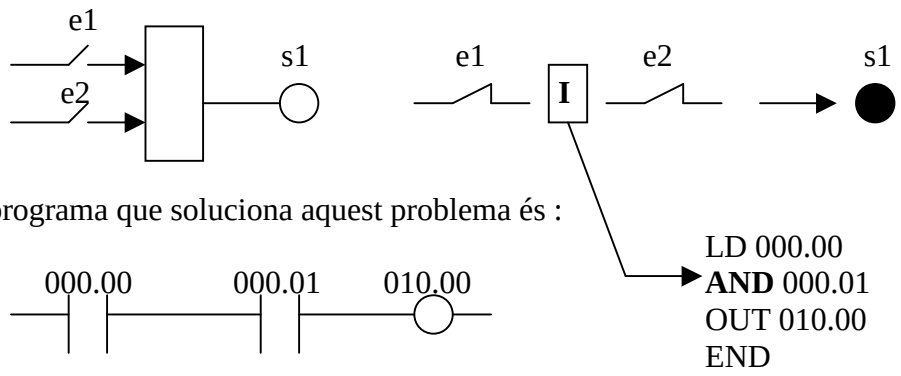
b) situar el cursor sobre la instrucció OUT de forma que quedi emmarcada pel rectangle negre.



c) seleccionar la icona 

INSTRUCCION AND

Suposem que tenim dos interruptors e1 i e2 connectats a les entrades 000.00 i 000.01 i un led s1 connectat a la sortida 010.00. Volem que el led s'encengui **només quan els dos interruptors estiguin tancats**.



El programa que soluciona aquest problema és :

Direm que els bits 000.00 i 000.01 estan connectats en **sèrie.** El programa realitza aquesta connexió sèrie a través de la instrucció AND.

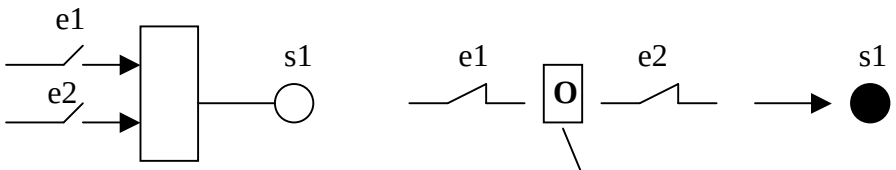
La taula de veritat que defineix la funció AND és la següent :

FUNCIÓ AND		
bit entrada 000.00	**bit entrada 000.01**	**bit sortida 010.00**
0	0	0
1	0	0
0	1	0
1	1	1

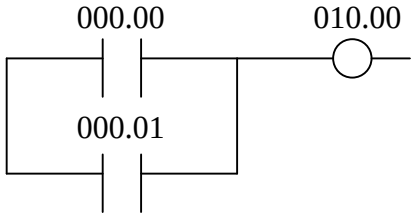
Observa que només obtenim un 1 a la sortida quan tenim totes les entrades a 1
La funció **AND** s'associa a la **multiplicació** aritmètica : bit 010.00 = bit 000.00 · bit 000.01

INSTRUCCIÓ OR :

Suposem que, en el mateix exemple anterior, ara ens interessa que el led s'encengui quan un interruptor **O** l'altre **O** tots dos estiguin tancats



El programa que genera aquesta funció és :




```
LD 000.00
OR 000.01
OUT 010.00
END
```

La taula de veritat és :

FUNCIÓ OR		
bit entrada 000.00	**bit entrada 000.01**	**bit sortida 010.00**
0	0	0
1	0	1
0	1	1
1	1	1

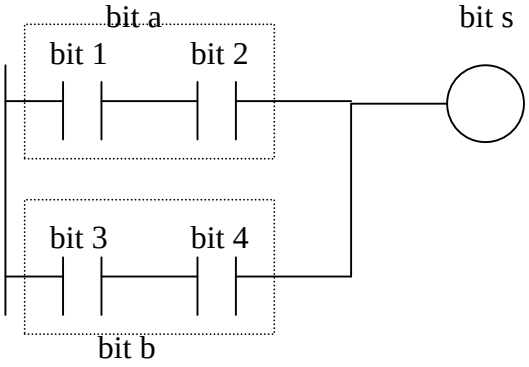
Observem que la sortida només és 0 quan les dues entrades estan a 0.

la funció **OR** s'associa a la **suma** aritmètica : bit 010.00 = bit 000.00 + bit 000.01

0+0=0 1+0=1 0+1=1 1+1=1

COMBINACIÓ DE FUNCIÓ AND I OR

1) Combinació paral·lel de bits en sèrie



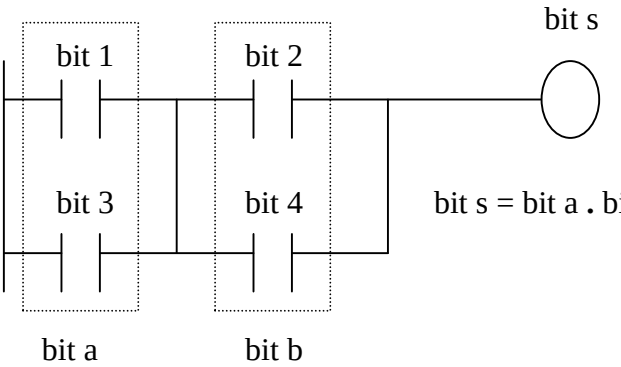
bit s = bit a + bit b = (bit 1 . bit 2)+(bit 3 . bit 4)

Veiem amb un exemple com calcular l'estat del bit de sortida : suposem que, en un moment determinat, els bits d'entrada es troben en la situació següent :

bit 1 =1 bit 2=0 bit 3=1 bit 4=1

Tindrem : bit s = (1 . 0) + (1 . 1) = 0 + 1 = 1

2) Combinació sèrie de bits en paral·lel



bit s = bit a . bit b = (bit 1 + bit 3) . (bit 2 + bit 4)

Veiem un exemple de càlcul : suposem que els bits d'entrada es troben en la situació següent : bit 1=0 bit 2=1 bit 3=1 bit 4=0

bit s = (0+1) . (1+0) = 1 . 1 = 1

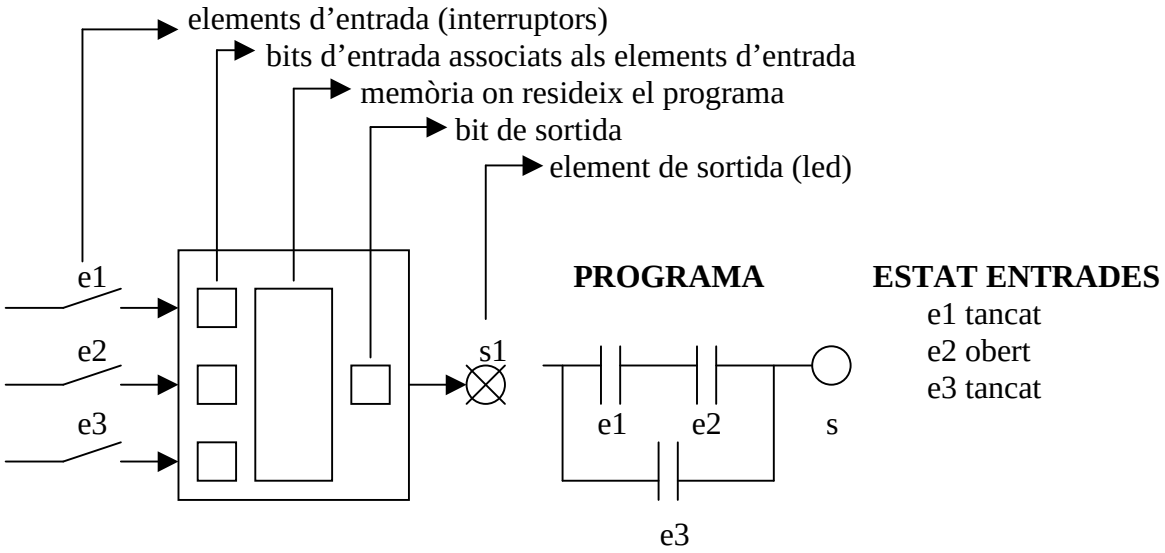
Norma per avaluar ràpidament el resultat d'un diagrama de contactes :

⊣ ⊢ deixa passar el senyal quan el bit està a 1 i la bloqueja quan està a 0

Veiem com podem aplicar aquesta norma a uns exemples

Exemple 3 :

DISPOSICIÓ AUTÒMAT

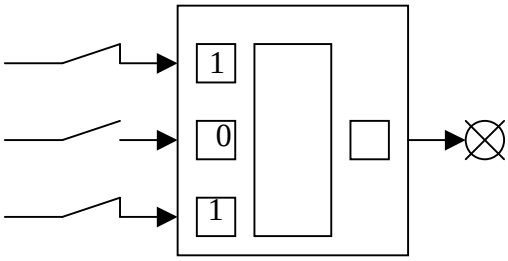


Primer determinem l'estat dels bits d'entrada en funció de l'estat dels interruptors . Recordeu que quan tenim l'interruptor obert no pot arribar el corrent a l'autòmat i el bit corresponent està a **0.** Amb l'interruptor tancat, arriba el corrent a l'autòmat i el bit d'entrada es posa a **1.**

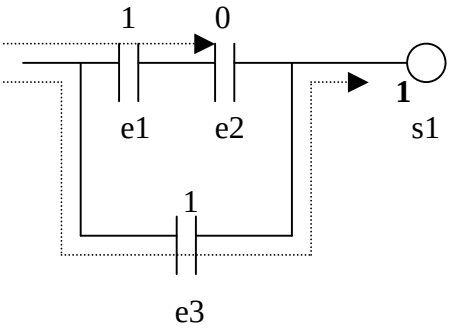
Interruptor obert	→	bit = 0
Interruptor tancat	→	bit = 1

En el nostre exemple tindrem :

interruptor e1 tancat	→	bit e1 = 1
interruptor e2 obert	→	bit e2 = 0
interruptor e3 tancat	→	bit e3 = 1

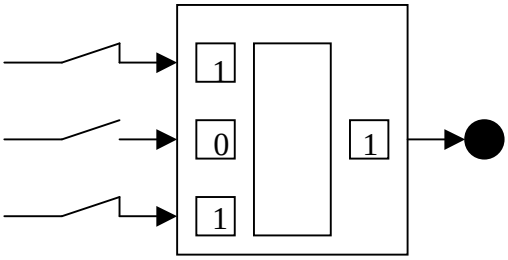


Seguidament analitzem el programa per determinar l'estat del bit de sortida :

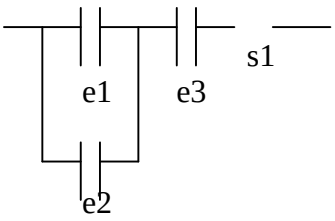


Observem que podem anar de l'entrada a la sortida a través de e3 → bit de sortida = 1 → led encès

La conclusió final és que el led estarà encès :

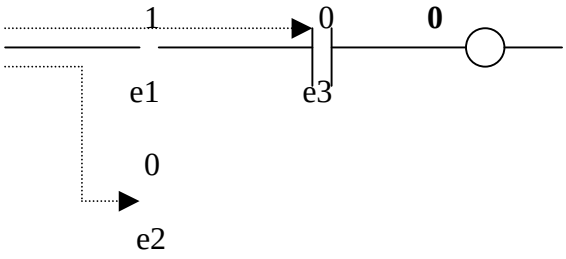


Exemple 4 :

DISPOSICIÓ AUTÒMAT	PROGRAMA	ESTAT ENTRADES
la mateixa de l'exemple 1 (3 entrades i 1 sortida)		e1 tancat e2 obert e3 obert

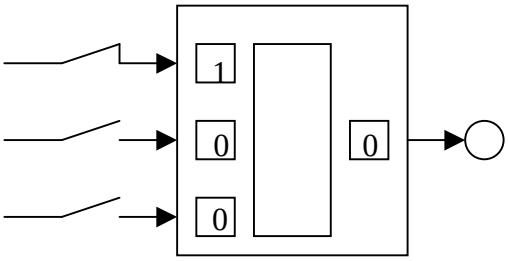
bits d'entrada : bit e1 = 1 (interruptor tancat)
bit e2 = 0 (interruptor obert)
bit e3 = 0 (interruptor obert)

Anàlisi del programa :



Observem que no podem arribar a la sortida ⟶ bit de sortida = 0 ⟶ led apagat

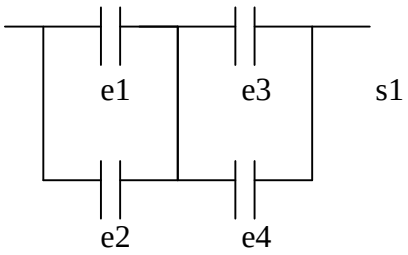
Conclusió final :



Exemple 5 :

AUTÒMAT PROGRAMA ESTAT ENTRADES

4 interruptors d'entrada
1 led de sortida

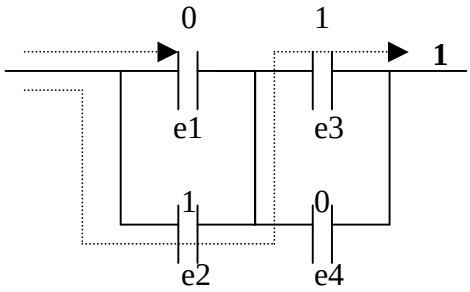


e1 obert
e2 tancat
e3 tancat
e4 obert

Bits d'entrada :

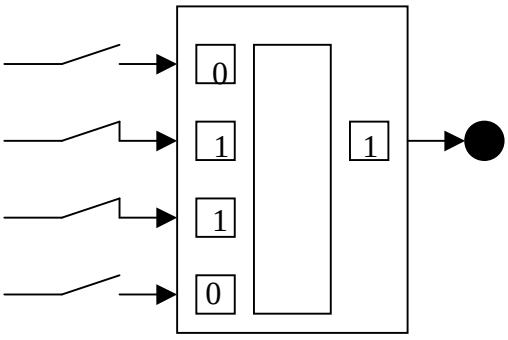
bit e1 = 0 (interruptor obert)
bit e2 = 1 (interruptor tancat)
bit e3 = 1 (interruptor tancat)
bit e4 = 0 (interruptor obert)

Anàlisi del programa :



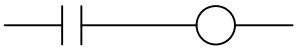
Observem que podem anar de l'entrada a la sortida a través de e1 i e3 → bit de sortida = 1 → led encès

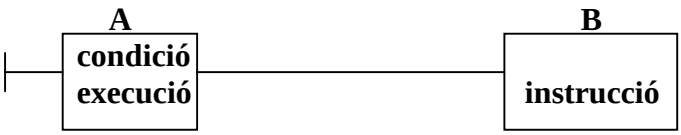
Conclusió final :



CONDICIÓ D'EXECUCIÓ D'UNA INSTRUCCIÓ

A) Format de les instruccions

La línia del diagrama de contactes  correspon al format general següent:



El bloc B representa una instrucció qualsevol. El bloc A representa la condició d'execució d'aquesta instrucció. Aquesta condició d'execució regula la forma en que s'executarà la instrucció .

Una condició d'execució pot estar en estat "ON" o en estat "OFF". Podem distingir dos tipus d'instruccions : per la manera en que s'executen les instruccions B.

a) **Instruccions que s'executen quan la seva condició d'execució (CE) és ON** i no s'executen quan es troba en OFF.

CE = ON	→	s'executa
CE = OFF	→	no s'executa

Més endavant estudiarem una instrucció anomenada SET que pertany a aquest grup. Quan la CE de SET és ON, aquesta instrucció s'executa i activa un bit. Quan la seva CE és OFF aquesta instrucció SET no s'executa i, si havíem activat el bit, aquest bit no es desactiva.

Ex:	CE = **ON**	→	SET s'executa i **activa** un bit
	CE = **OFF**	→	SET no s'executa i **no activa ni desactiva** un bit

b) **Instruccions que s'executen sempre** :

CE = ON	→	s'executa d'una forma
CE = OFF	→	s'executa de forma diferent

La instrucció OUT es limita a traslladar l'estat de la condició d'execució cap al bit de sortida. Quan CE = ON trasllada un 1 al bit de sortida (activa la sortida). Quan CE = OFF, trasllada un 0 al bit de sortida (desactiva la sortida).

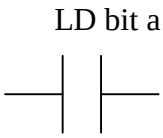
La instrucció OUT NOT trasllada l'estat **invertit** de la condició d'execució cap a la sortida. Quan CE=ON trasllada un 0 i quan és OFF trasllada un 1

Ex	CE = **ON**	→	OUT s'executa i **activa** un bit
	CE = **OFF**	→	OUT s'executa i **desactiva** un bit
	CE = **ON**	→	OUT NOT s'executa i **desactiva** un bit
	CE = **OFF**	→	OUT NOT s'executa i **activa** un bit

(Amb les instruccions OUT i OUT NOT podem activar i desactivar un bit. Amb la instrucció SET només podem activar una sortida.)

B) Format de la condició d'execució (CE)

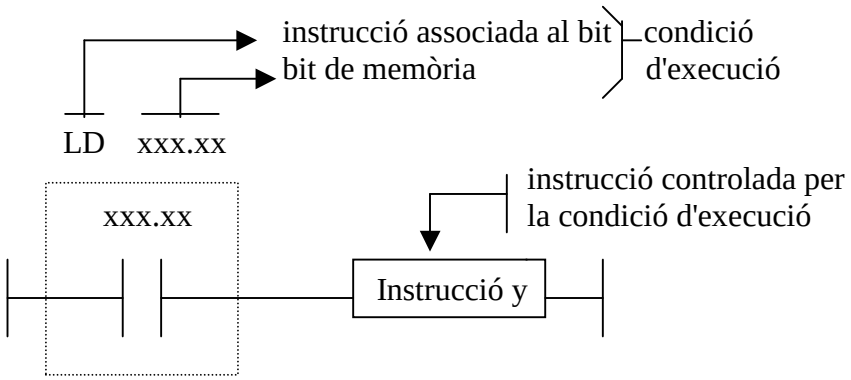
El format de la condició d'execució és el següent :



El bit a pot ser un bit de memòria associat a un element d'entrada (per exemple un interruptor), a un element de sortida (per exemple un led) o a un element intern de l'autòmat (per exemple un temporitzador)

LD bit a : quan el bit a està a 1, la instrucció LD posa la CE en estat ON. Quan el bit a està a 0, la instrucció LD posa la CE en estat OFF.

LD	CE
bit = 1	= ON
bit = 0	= OFF



Gaire bé totes les instruccions exigeixen estar controlades per una condició d'execució.

LD xxx.xx : Comprova l'estat del bit de memòria xxx.xx i si el trobes en "1" col·loca la condició d'execució en "ON"

INSTRUCCIONS SET I RESET

La instrucció SET, quan la seva condició d'execució és ON, activa el bit sobre el que actua, el posa en estat 1. Quan la condició d'execució torna a OFF, la instrucció no s'executa i **el bit continua en 1.** La condició d'execució OFF no afecta al bit. Aquesta és la diferència amb la instrucció OUT.

La instrucció RESET (RSET), quan la seva condició d'execució és ON, desactiva el bit sobre el que actua, el posa a "0" . Quan la seva condició d'execució és OFF, la instrucció no s'executa i no afecta al bit sobre el que actua.

NOTA : En cas de que les dues instruccions SET i RSET tinguin simultàniament condició ON domina la que hem programat en últim lloc.

EXEMPLES :

Programa 1 :

```
LD 00000
SET 01000
LD 00001
RSET 01000
END
```

Programa 2 :

```
LD 00000
RSET 01000
LD 00001
SET 01000
END
```

En aquests 2 programes intentem utilitzar dues entrades per controlar el SET i el RSET de la sortida 01000.

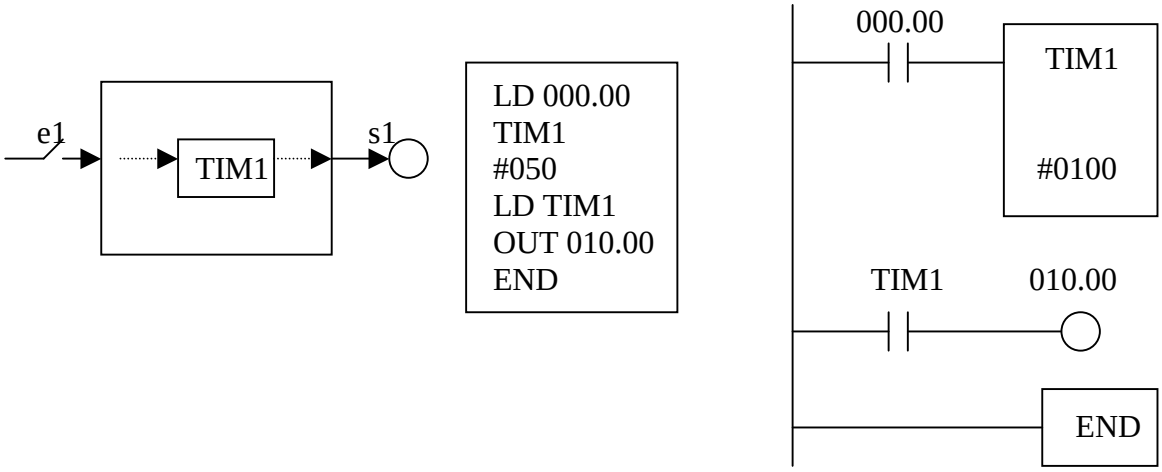
Si en el programa 2 mantenim l'entrada RSET a "1" i enviem un "1" a l'entrada SET, veurem que la sortida s'activa i es manté activada mentre tinguem l'entrada SET a "1".

TEMPORITZADORS

Imaginem que volem aconseguir l'actuació següent : tanquem l'interruptor e1 (associat al bit d'entrada 000.00) i, després de 10 segons, volem que s'activi la sortida s1 (associada al bit de sortida 010.00). Ens caldrà un temporitzador o timer que temporalitzi el temps desitjat. L'interruptor d'entrada arrencarà el timer i el timer, quan hagi acabat de temporalitzar, provocarà l'activació de la sortida.

entrada → timer ..... 10 segons → sortida

El programa que executa aquesta seqüència és :



El funcionament del programa és el següent :
- Quan tanquem l'interruptor el timer arrenca i es comença a decrementar la constant de temps que hem programat, en el nostre cas #0100.
- Els decrements es succeeixen amb el ritme següent : una unitat cada 0,1 segons. Cada dècima de segon la constant es decrementa una unitat. Quan arriba a zero haurà passat un temps de :

100 decrements x 0,1 segons/decrement = 10 segons

Aquest és el motiu que ens obliga a programar el valor 100 per temporalitzar 10 segons.

- L'arribada a zero del timer queda registrada per l'activació (pas de 0 a 1) d'un bit intern o flag

- Quan el timer arriba a zero s'activa la sortida

Si, mentre el timer està temporalitzant, obrim l'interruptor que el governa, s'aturarà el compte i la constant de temps tornarà al seu valor inicial.

Si, un cop finalitzat el compte i activada la sortida, obrim l'interruptor, es desactivarà la sortida i la constant de temps tornarà al seu valor inicial

Quan tanquem l'interruptor direm que fem el **set** del comptador. Quan obrim l'interruptor direm que fem el **reset** del
Per programar el diagrama de contactes caldrà seguir les passes següents :

- clicar i arrossegar el símbol 00000 ⊣ ⊢

- clicar i arrossegar el símbol TIM . S'obrirà un menú en pantalla que ens permetrà definir el número de timer que volem utilitzar i la seva constant de temps.

- clicar dues vegades sobre el quadre gris "final de bloc" per obrir una nova línia

- clicar i arrossegar el símbol Tim 1 ⊣ ⊢

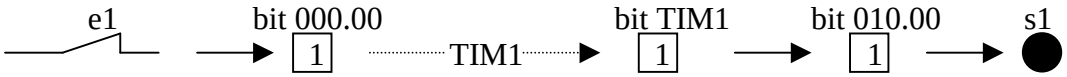
- clicar i arrossegar el símbol 010.00 —◯—|

- clicar dues vegades sobre " final de bloc " per obrir nova línia

- clicar i arrossegar FUN 01 per introduir la instrucció END que senyal la fi del programa.

Amb l'ajut de la monitorització executeu el programa i comproveu tots els aspectes que hem explicat.

El procés intern complert que segueix l'execució d'aquest programa és el següent :

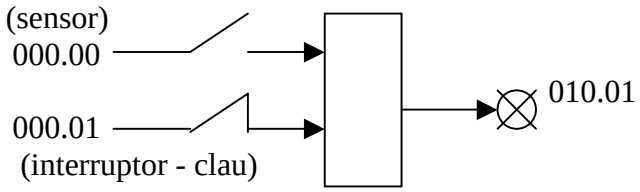


Tanquem interruptor e1 ⇒ el seu bit associat 000.00 es posa a 1 ⇒ arranca el timer...
...quan ha passat el temps es posa a 1 el bit associat al timer ⇒ s'encén el led s1

PRÀCTICA AMB OMROM: CONTROL D'UNA ALARMA

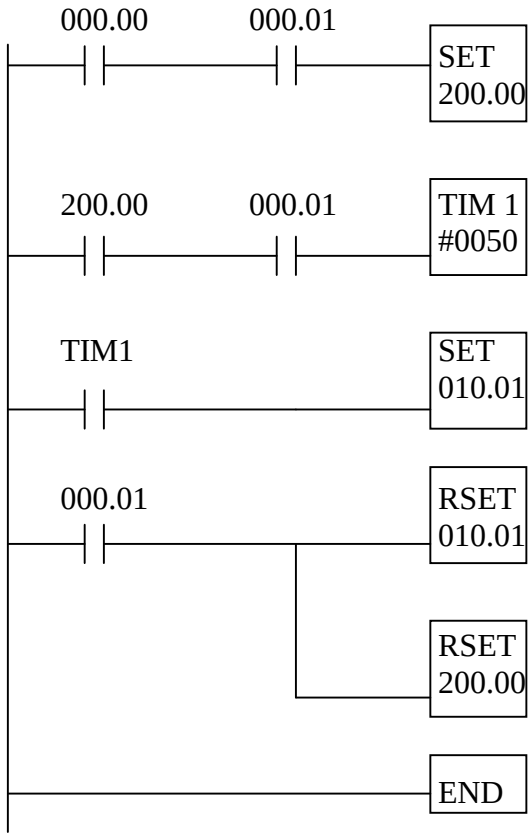
Dissenyar un programa de control del dispositiu d'alarma d'un habitatge amb les següents especificacions tècniques :
- detecció de presència per sensor òptic
- anul·lació d'alarma per interruptor ocult.
- temps de desconnexió programable
- alarma connectada a brunzidor

Utilitzarem un interruptor associat a l'entrada 000.00 per simular el detector i un led associat a la sortida 010.01 per simular el brunzidor. Quan l'interruptor 000.00 estigui tancat idoni un 1 estarem simulant que el sensor està detectant una persona. L'interruptor associat a l'entrada 000.01 simularà l'interruptor de desconnexió ocult. Aquest interruptor representarà un contacte normalment tancat. Quan l'obrim caldrà que es desconnecti l'alarma..



Entrades :	Sortides:	Elements auxiliars interns
Detector òptic: 000.00 Interruptor clau: 000.01	brunzidor : 010.01	TIM 1 (timer1): 200.00

PROGRAMA EN LADDER



PRÀCTIQUES AMB VirPLC

LD, LDNOT, OUT, OUTNOT

- Interruptor tancat => Bombeta encesa i brunzidor apagat. Interruptor obert => Al revés
- En tancar un polsador, encendre una bombeta (2 sistemes)
- En tancar un polsador, apagar una bombeta (2 sistemes)
- En tancar tots 3 interruptors, encendre una bombeta (AND)
- En prémer un dels tres polsadors, encendre una bombeta (OR)
- En tancar 2 i només 2 dels tres polsadors, encendre una bombeta
- Representació d'una funció lògica complexa $F = [[(a \cdot b + (a+b) \cdot e) \cdot c] + f] \cdot d$
- Control de les rodes d'un camió: si una de les 4 rodes rebentades $\Rightarrow$ led indicador; i si les 2 rodes de l'esquerra o les 2 de la dreta rebentades sona sirena.
- Alarma amb: interruptor-clau; 2 sensors passius infraroigs NT (P0, P1), 1 polsador sota el taulell per cas d'atracament que fa sonar sempre l'alarma
- Control d'una grua "segura" mitjançant 2 comandaments de 3 polsadors cada un: 2 x pujar => puja; 2 x Baixar => baixa; 1 x Stop => Stop.

SET, RSET

- Concurs TV: 3 concursants amb 3 polsadors. S'encén la bombeta de qui prem primer. El presentador l'apaga per la propera pregunta.
- Porta tancant i obrint contínuament amb 2 finals de cursa NO
- ... Idem però amb un polsador de marxa i un d'aturada
- ... Idem amb un sol polsador (que engega i para) utilitzant un FLAG
- ... Idem però que si quan el parem, girava a la dreta, que en tornar a engegar continuï girant a la dreta (un altre flag)
- Hostal amb 4 habitacions amb un polsador d'avís a cada una, i dos indicadora a consergeria: 1 bombeta i un brunzidor
- Codi clau: 3 polsadors que s'han de polsar seqüencialment

SET RESET i TIM

- La bombeta s'il·lumina als 20 segons de tancar l'interruptor INT
- Control d'un semàfor de vianants.
- Control dels 2 intermitents d'un cotxe
- Ruleta giratòria
- Alarma amb SET i RSET i TimeOut
- Control d'una porta supermercat amb VirPLC
- Porta de supermercat que reposi el temps de descompte i que torni a obrir.
- Control d'un ascensor de tres plantes
- Control d'una barrera de pàrquing
- Control d'una porta de garatge.

CMPT

- Al prémer 8 cops el polsador CK s'activa la bombeta. Al polsar Reset el comptador s'inicialitza altre cop a 8 i s'apaga la bombeta.
- En prémer P es realitza un moviment esquerra-dreta-esquerra-dreta i es para.

PROJECTE FINAL DE LA UNITAT:

Objectiu: es pretén el disseny i simulació d'un sistema de control habitual amb el que sovint ens hem trobat davant dels nassos i segurament no li hem donat cap importància.
Es proposen els següents 4 exemples però es pot optar per un altre de diferent...

1.- PORTA DE GARATGE:

Sortides a controlar	**Codi**	**Simulat per...**	**Sortida real**
Motor gira tancant	MT	LED inferior dreta	010.
Motor gira obrint	MO	LED superior dreta	010.
Entrades a controlar	**Codi**	**Simulat per...**	**Entrada real**
Final de carrera superior	FCS	Interruptor superior dreta	000.
Final de carrera inferior	FCI	Interruptor inferior dreta	000.
Polsador de pujar (obrir)	PO	Polsador negre esquerra	000.
Polsador d'aturada	PA	Polsador vermell	000.
Polsador de baixar (tancar)	PT	Polsador negre dreta	000.

2.- ASCENSOR 2 PLANTES (Planta baixa - Planta primera)

Sortides a controlar	**Codi**	**Simulat per...**	**Sortida real**
Motor gira baixant	MB	LED inferior dreta	010.
Motor gira pujant	MP	LED superior dreta	010.
Entrades a controlar	**Codi**	**Simulat per...**	**Entrada real**
Final de carrera Pl. 1a.	FCP1	Interruptor superior dreta	000.
Final de carrera Pl. Baixa	FCP0	Interruptor inferior dreta	000.
Polsador Pl. 1a (cridar-lo)	P1	Polsador negre esquerra	000.
Polsador Pl. Baixa (cridar-lo)	P0	Polsador negre dreta	000.

3.- BARRERA DE PARQUING

Sortides a controlar	**Codi**	**Simulat per...**	**Sortida real**
Motor gira tancant	MT	LED inferior dreta	010.
Motor gira obrint	MO	LED superior dreta	010.
Entrades a controlar	**Codi**	**Simulat per...**	**Entrada real**
Final de carrera superior	FCS	Interruptor superior dreta	000.
Final de carrera inferior	FCI	Interruptor inferior dreta	000.
Polsador sol·licitud ticket	PT	Polsador negre esquerra	000.
Detecció de cotxe sota barrera	DC	Polsador vermell	000.

4.- PORTA SUPERMERCAT

Sortides a controlar	**Codi**	**Simulat per...**	**Sortida real**
Motor gira tancant	MT	LED inferior dreta	010.
Motor gira obrint	MO	LED superior dreta	010.
Entrades a controlar	**Codi**	**Simulat per...**	**Entrada real**
Final de carrera porta oberta	FCO	Interruptor superior dreta	000.
Final de carrera porta tancada	FCT	Interrupto superior esquerra	000.
Detector de persones	DPI	Polsador vermell	000.
Detector d'encallades tancant	DET	Polsador negre esquerra	000.

Parts del treball: (Només cal presentar un treball per cada grup. Els folis han de ser DIN A4)

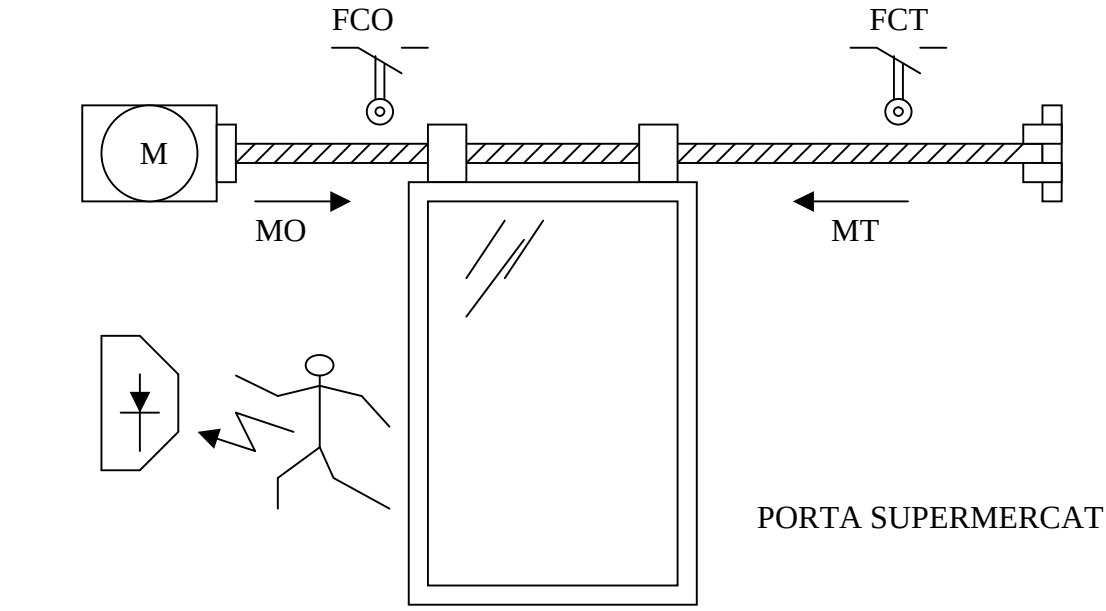
1- (Foli 1) Trieu la opció que vulgueu fer i descriviu el seu funcionament de forma resumida i exacta.
2- (Foli 2) Dibuixeu l'estructura física del sistema. (motor, politges, engranatges, finals de carrera, contrapesos, polsadors,...)
3- (Foli 3) Dibuixeu el sistema de control d'un motor asíncron monofàsic de condensador perquè, mitjançant 2 contactors, pugui girar en 2 direccions.
4- (Foli 4) Elabora, en brut, el programa en forma de diagrama de contactes.
5- Entreu el programa al PLC i comproveu el seu funcionament (adequant-lo si per casualitat no funciona).
6- (Foli 5) Elabora, en net, el programa en forma de diagrama de contactes.
7- (Foli 6) Amb l'ajut del dossier del crèdit, Dibuixeu la connexió elèctrica del autòmat amb les entrades i les sortides,
8- (Foli 7) Calcula la potència que necessitarà tenir el motor.
9- (Foli 8) Creieu que el sistema és millorable?. Com?
10- (Foli 8) Creieu que és aplicable a la realitat?. Què li mancaria?

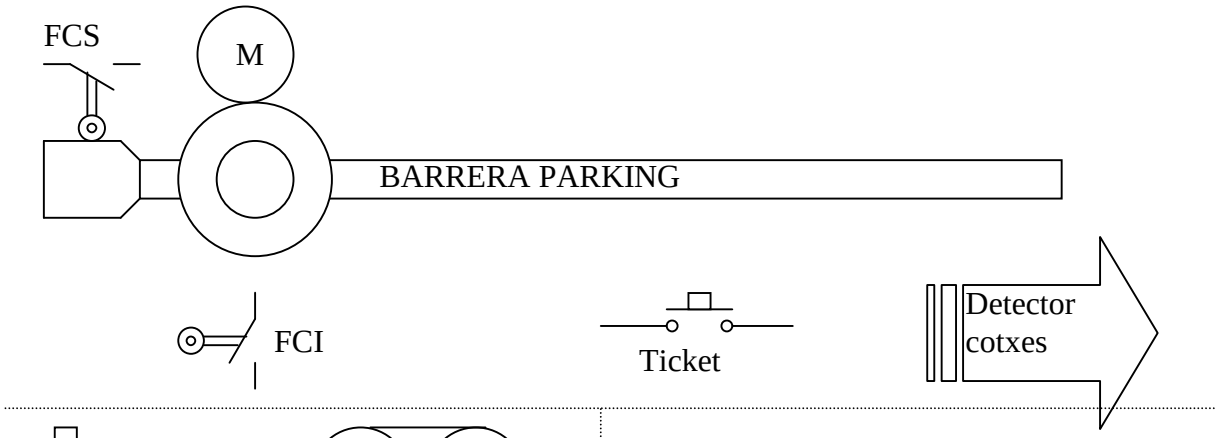
En el propi treball, de forma voluntària i sense por, adjunteu aquest foli al final contestant a les següents preguntes:

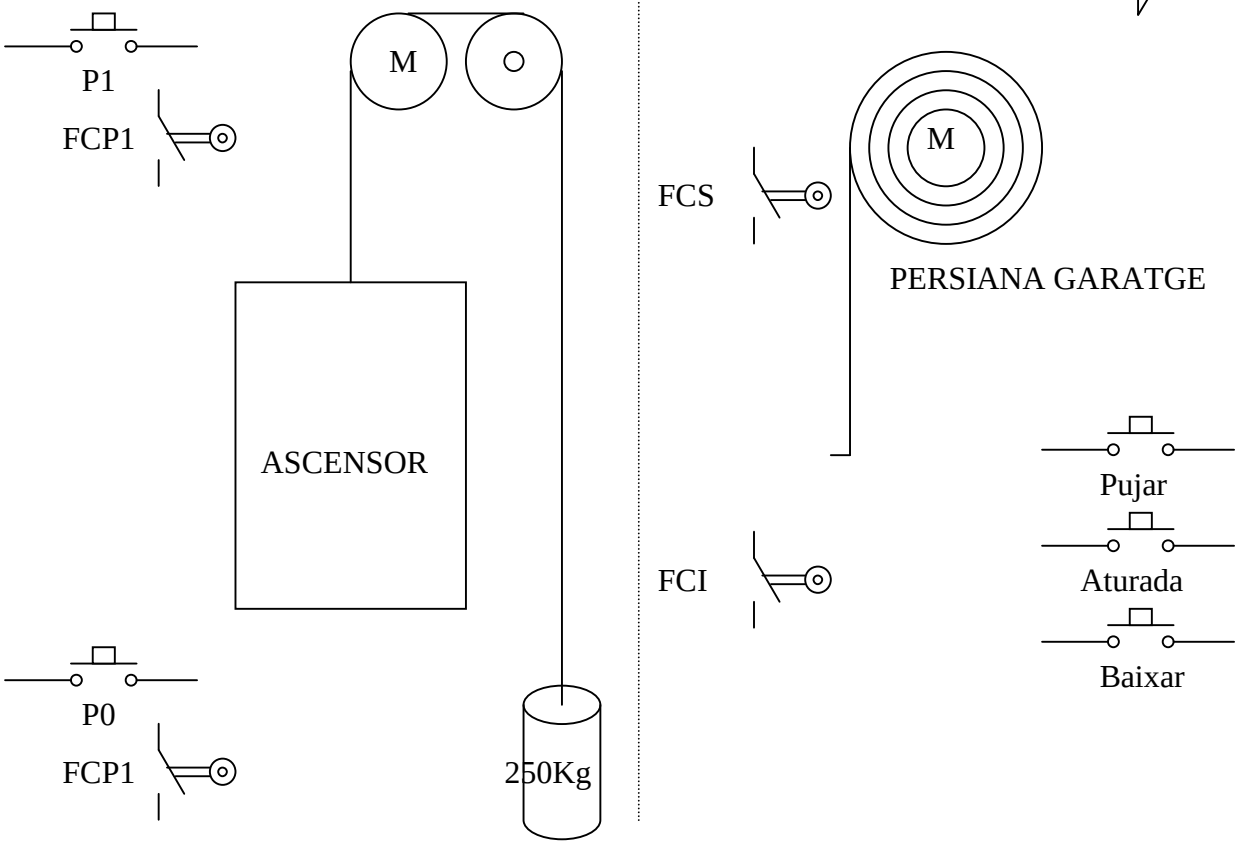
a) Aquest crèdit t'ha agradat? Perquè?

b) Al teu criteri, com es pot millorar?

c) Creus que el que has après t'ha servit per quelcom?

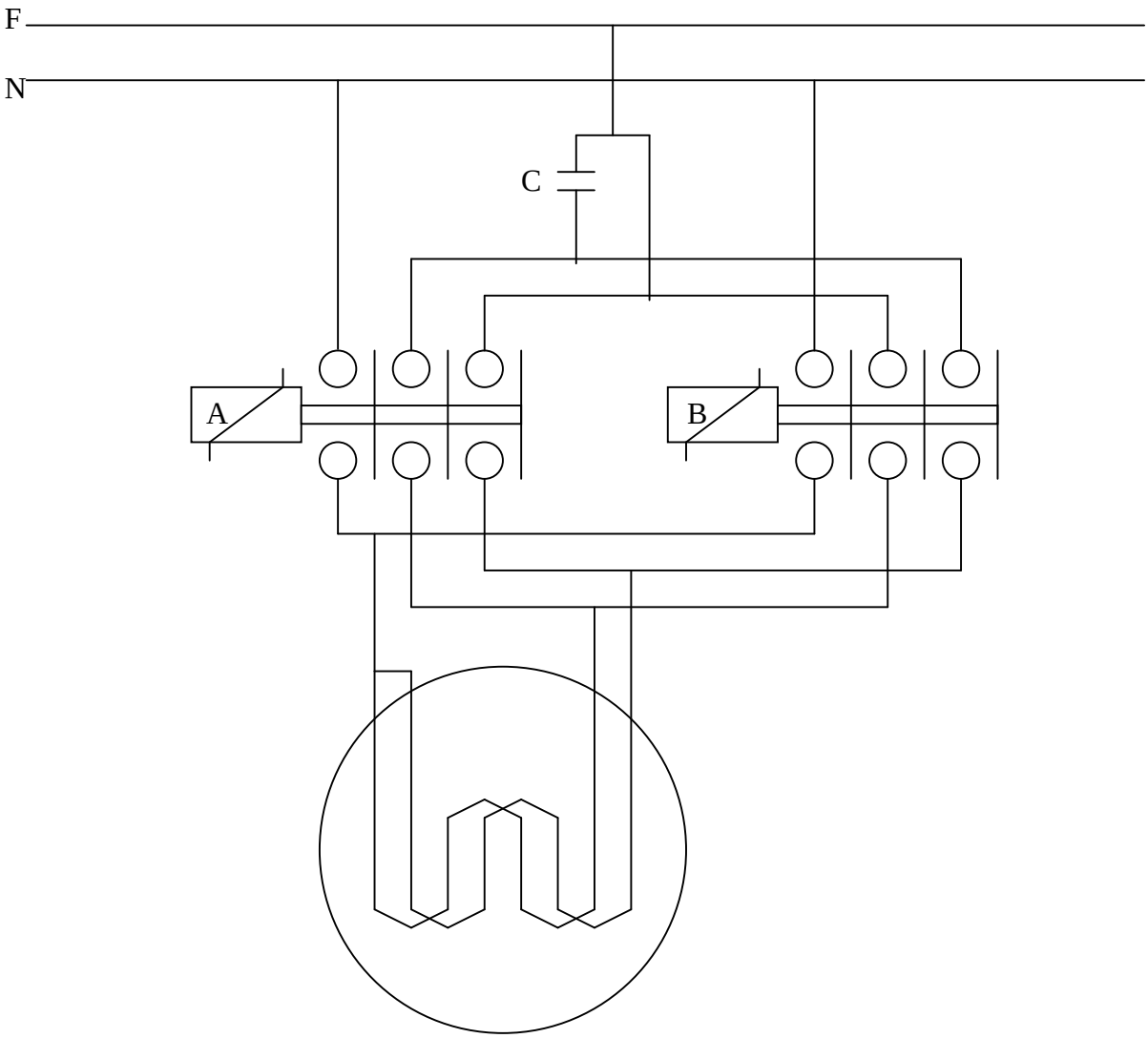

FCO
FCT
M
MO
MT
PORTA SUPERMERCAT


FCS
M
BARRERA PARKING
FCI
Ticket
Detector
cotxes


P1
FCP1
M
ASCENSOR
P0
FCP1
250Kg
FCS
M
PERSIANA GARATGE
FCI
Pujar
Aturada
Baixar

AMPLIACIÖ:

CANVI DE DIRECCIÓ EN MOTORS ASINCRONS MONOFÀSICS DE CONDENSADOR



A	B	Funcionament
0	0	Motor aturat
0	1	Motor activat girant en un sentit
1	0	Motor activat girant en l'altre sentit
1	1	curtcircuit