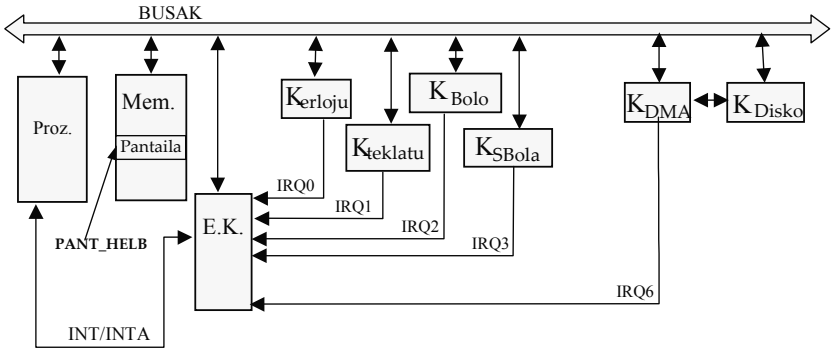
Euskal Herriko Unibertsitatea
Informatika Fakultatea

Konputagailuen Arkitektura eta Teknologia Saila

Konputagailuen Arkitektura I

Sarrera/irteerako azpisistema 2 (ebazpena): Bolatokia

Bolatoki bateko partidak kontrolatzen dituen sistema bat diseinatu nahi dugu. Horretarako, sentsore bat daukagu, bola noiz jaurti den detektatzeko, eta gailu berezi bat, erori diren boloak kontatu eta berriro zutik jartzeko.



Sistemak honako gailuak ditu:

K_BOLO: Erori diren boloak zutitzeko erabiltzen den gailuaren kontroladorea da. Bi erregistro ditu: kontrol-erregistroa (R_KON_KBOLO) eta datu-erregistroa (R_DAT_KBOLO). Kontrol-erregistroan *strobe* sekuentzia bat egitean gailua martxan jartzen da erori diren bolo guztiak altxatzeko. Bere lana bukatzean eten bat sortuko du, eta bere datu erregistroan adieraziko du eroritako bolo kopurua.

K_SBOLA: Bola jaurtia izan den detektatzen duen sentsorearen kontroladorea da. Bola boloetara iristear dagoen unean sentsore honen ondotik pasatzen da, eta kontroladoreak eten bat sortuko du.

K_DMA: DMA kontroladorea da. Memoria eta kanpoko disko-unitate baten artean transferentziak burutzeko erabiltzen da. Honako erregistroak ditu:

- **Kontrol-erregistroa** (R_KON_KDMA): erregistro honetan 1ekoa idaztean transferentzia hasiko da. (Automatikoki jartzen da 0an.)
- **Helbide-erregistroa** (R_HEL_KDMA): transferituko den blokearen hasierako helbidea.
- **Luzera-erregistroa** (R_LUZ_KDMA): transferituko den blokearen luzera bytetan.
- **Egoera-erregistroa** (R_EGO_KDMA): transferentzia amaitzean, erregistro honek adierazten du transferentzia ondo joan den (1) edo erroreren bat gertatu den (0).

K_DISKO: Diskoaren kontroladorea da. Transferentzia bete ahal izateko kontroladore hau hasieratu behar da. Suposatuko dugu *KdiskoProgramatu()* izeneko errutina daukagula hasieratze hori egiteko.

Beste kontroladore guztiak (erlojuarena, teklatuarena eta etenena) ikasgaian landutakoak dira. Sistema honetan **teklatua**rekiko sinkronizazioa **inkesta** bidez egiten da. Pantaila memorian mapeatuta dago, PANT_HELB helbidetik aurrera.

Sistemaren **funtzionamendua** honakoa izan beharko da. Hasieran, bolatokiko jabeak partidari ekiteko baimena eman arte zain egon beharko dugu. Horregatik langa bat dago boloen aurrean. Behin jabeari ordaindu diogunean, honek teklatuko H tekla sakatuko du. Honela, langa altxatuko da eta partida hasi ahalko dugu. Langa altxatzeko *altxatu_langa()* errutina jadanik idatzita dago.

Partida batean 10 jaurtiketa izango ditugu eta bakoitzean botatako boloak kontatuko dira. Jaurtiketa bakoitzaren ondoren, K_BOLO gailua arduratuko da jaurtiketa horretan erori diren bolo guztiak kontatzeaz eta altxatzeaz.

Jaurtiketa bakoitzean, bola pistan zehar doanean, SBOLA sentsoreak detektatuko du bola boloetara iristear dagoela, eta eten bat sortuko du; bola detektatu ondoren, 10 segundo itxaron behar da, eroritako boloak lurrean egonkortu daitezen, eta gero K_BOLO gailua aktibatu beharko dugu bolo guztiak jaso ditzan.

Jaurtiketa bakoitzaren ondoren, ordurarte guztira botatako bolo kopurua pantailaratu beharko da. Horretarako *pantailaratu_botatakoak(Bolo_Kopurua, Jaurtiketa_Zenbakia)* errutina erabiliko dugu, zeinak bi parametro jasotzen dituen: orain arte guztira bota diren bolo kopurua, eta orain arteko jaurtiketa kopurua. Horrela, banan-banan, puntuazio guztiak bata bestearen ondoan pantailaratuz joango dira. Puntuazio bakoitzak 4 byte okupatuko du memorian.

10 jaurtiketak burututakoan partida amaituko da, baina aurretik puntuazioa gordeko da disko batean. Memoriatik diskorako transferentzia honetarako DMA erabiliko dugu. DMA transferentzia ez bada ondo burutzen, berriro saiatuko gara, baina 3 aldiz gehienez. 3 saiakeren ondoren transferentzia burutzea lortu ez badugu *errorea_transferentzian()* errutinari dei egingo diogu, eta partidari bukaera emango.

Partidak gehiegi ez luzatzeko asmoarekin, 5 minutuko denbora tartea dauka jokalariak jaurtiketa bakoitza egiteko. Boloak altxatu direnetik denbora tarte maximo hau pasatzen bada jaurtiketarik egin gabe, partida amaituko da. Azkeneko kasu honetan ez da puntuazioa diskora transferitu beharko.

Edozein kasutan partida amaitzean langa jaitsi beharko da, berriro teklatuan H tekla sakatu arte. Horretarako *jaitsi_langa()* errutina erabil dezakegu.

Hurrengoa eskatzen da: Idatzi lengoaia algoritmikoan beharrezkoak iruditzen zaizkizun zerbitzu-errutina guztiak eta programa nagusia. Komentatu ariketaren ebazpenerako egiten duzun edozein suposaketa. Kontuan hartuko da sistemaren portaera automata batez adieraztea.

Ebazpena

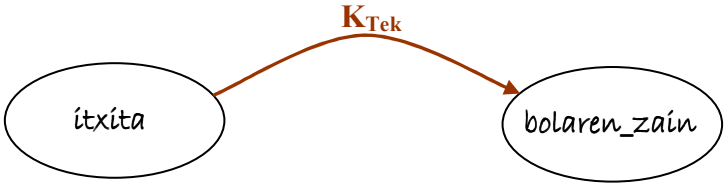
Lehenik, sistemaren funtzionamendua islatzen duen automata bilatzen saiatuko gara. Horretarako, biziki garrantzitsua da sistemaren funtzionamenduaren egoera edo etapa posible guztiak ondo bereiztea, eta baita egoera batetik beste batera pasatzeko egin beharreko urratsak identifikatzea ere. Oro har, egoeren arteko trantsizioak kanpoko gertaeren ondorioak izango dira, eta horiek periferikoen ekintzen bidez adieraziak izango dira, sinkronizazio-modua edozein izanik: inkesta bidezkoa ala etenen bidezkoa.
Ariketa honetan ez digute esaten zein diren sistemaren balizko egoerak, eta horregatik ongi irakurri behar dugu sistemaren funtzionamenduari buruz ematen diguten informazioa, egoerak ondorioztatzeko. Hala, azter ditzagun sistema honetan gerta daitezkeen gertaera posible guztiak eta ikus dezagun zein diren gertaera horien ondorioak, sistemaren egoeraren arabera. Horretarako, arreta handiz berrirakurriko dugu enuntziatua.
Hasteko, esaten digutenaren arabera, bolatokiko jabeari partida ordaintzen ez diogun bitartean, ez dugu baimenik izango jolasteko, eta langa bat egongo da boloen aurrean hori ekiditeko. Hori da, hain zuzen ere, hasierako egoera, *itxita* izendatuko duguna.

1. *itxita* egoera:

- Egoeraren ezaugarriak:
  Aipatu dugun legez, ordaindu gabeko partidak ekiditeko, langa bat dago boloen aurrean. Sistema egoera horretan egongo da jabeak H tekla sakatu arte. Teklatua izango da, beraz, egoera-aldaketa eragingo duen gertaera sortuko duen periferikoa (teklatuarekiko sinkronizazioa inkesta bidez egin behar den arren, horrek ez du eraginik automatan, bertan ez ditugulako bereizten inkestak edo etenak, bakarrik gertaerak, eta garbi dago tekla bat sakatzea gertaera bat dela). Baina H tekla bakarrik hartu beharko dugu kontuan, ez beste edozein tekla, eta bakarrik tekla sakatzean eta ez askatzean –gogoratu teklatuaren kontroladoreak bi gertaera bereizten dituela tekla bakoitzeko: bata, tekla sakatzean (MAKE), eta bestea, askatzean (BREAK)–. Hortaz, egoeren arteko trantsizioa baldintzapekoa izango da: H tekla sakatzen denean, soilik.

- Trantsizio posibleak:

a) Jabeak H tekla sakatzen duenean, partida hasiko da, hau da, jokalariak bola botatzeari ekingo dio, baina horretarako denbora-tarte bat hartu ahal izango du, behar duena –gehienez 5 minutu, enuntziatuan esandakoaren arabera, partida gehiegi ez luzatzearren–. Horregatik, bigarren egoerari *bolaren_zain* deituko diogu.
Automatan, honelaxe adieraziko dugu: bi egoeren arteko geziak trantsizioaren nondik norakoa adierazten du –*itxita* egoeratik *bolaren_zain* egoerara–, eta geziaren gaineko etiketak, berriz, zein den trantsizio hori eragin duen periferikoa edo gertaera.



Teklatua inkestaz sinkronizatu nahi dugunez, haren gertaerak programa nagusian tratatuko dira, ez zerbitzu-errutina batean, etenen bidezko sinkronizazioan egiten den

bezala. Hala, programa nagusiak nolabait detektatu beharko du tekla bat sakatu edo askatu dela. Periferiko arruntekin, inkesta egiteko prozesadoreak kontroladoreko egoera-erregistroa begiratzen du, nahi dugun gertaera detektatu arte. Baina PCetan, teklatuaren kontroladorea pixka bat berezia da, ez baitauka egoera-erregistrorik, Hori dela eta, teklatuaren inkesta egin ahal izateko etenen kontroladoreko IRR erregistroa da aztertu behar dena, tekla bat sakatzean edo askatzean sortzen den etena detektatu ahal izateko, beti sortzen baita etena, nahiz eta inkesta bidez sinkronizatu; horregatik, inkesta bidez sinkronizatu nahi dugunean, beharrezkoa da teklatuaren etenak galaraztea (etenen kontroladoreko IMR maskara erregistroko 1 bita 1ean jarriz). Hala, programa nagusiak IRR erregistroa irakurri beharko du, eta haren 1 bita aztertu: 1 balioa hartzen duenean, tekla sakatu/askatu den seinalea, eta orduan hurrengo eginkizunak burutu beharko ditu programa nagusiak, automatan islatu duguna egingo badu.

- Teklatuaren tratamenduaz arduratzen den programa-zatiaren eginkizunak:

Automatan islatu dugun egoera-trantsizioaz gain, teklatuaren gertaeren tratamenduaz arduratzen den programa-zatiak beste eginkizun batzuk ere baditu (argi dago, inkestan tekla bat sakatu/askatu dela detektatu ondoren exekutatuko da zati hau; inkesta ez dugu hemen idatziko, orokorra delako, ez automataren mendekoa, horregatik, programa nagusia idaztean komentatuko dugu nola egin inkesta):

Tek.1) Teklatuaren kontroladoreko datu-erregistroa (`R_DAT_Ktek`) irakurri beharko du, jabeak zein tekla sakatu duen jakiteko:

```
tekla_kodea = InPort(R_DAT_Ktek);
```

Tek.2) Teklatuaren kasuan, kontroladoreko kontrol-erregistroaren gainean (`R_KON_Ktek`) "strobe" sekuentzia bat egin behar da. Honelaxe:

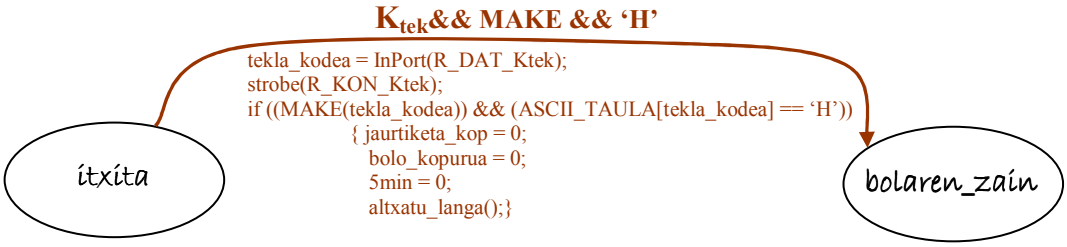
```
strobe(R_KON_Ktek);
```

Tek.3) Orain, tekla sakatu duen (MAKE) ala askatu duen (BREAK) begiratu behar dugu, eta gainera begiratu behar dugu ea H tekla den, eta hori guztia *itxita* egoeran soilik, beste edozein egoeratan ez baitzaio kasurik egin behar teklatuari. Eta baldintza horiek betetzen direnean, egoera-aldaketa eragiteaz gain, boloen aurrean dagoen langa altxatu behar da, eta horretarako esaten digute badagoela jadanik idatzita dagoen errutina berezi bat: `altxatu_langa()` izenekoa. Horrez gain, partidaren hasiera denez, aldagai global batzuk hasieratu behar dira, partida egokiro kontrolatu ahal izateko. Zehazki, hauek: a) partida bakoitzean gehienez 10 jaurtiketa egin daitezke; hala, aldagai bat behar dugu jaurtiketa-kopurua kontatzeko: `jaurtiketa_kop` izena emango diogu eta 0 balioa esleituko diogu hasieran; b) jaurtiketa bakoitzean puntuazioa kalkulatu eta pantailaratu behar da, eta horretarako kontuan hartzen da une horretara arte egindako jaurtiketa guztietan botatako bolo-kopurua; hala, beste aldagai global bat izango dugu, `bolo_kopurua`, 0rekin hasieratuko dena; c) azkenik, badakigu jokalariak gehienez 5 minutuko denbora-tartea duela bola botatzeko, eta ez badu botatzen, orduan partida amaitu behar dela; horregatik, `5min` izeneko aldagai globala izango

dugu denbora-tarte hori kontrolatzeko, eta partida hasten denean, 0 balioarekin hasieratu behar da. Hortaz, honelaxe geratuko da kodea:

```
if ((MAKE(tekla_kodea))&&(ASCII_TAULA[tekla_kodea]== 'H')
&&(egoera_automata == itxita))
                {
                    jaurtiketa_kop = 0;
                    bolo_kopurua = 0;
                    5min = 0;
                    altxatu_langa();
                    egoera_automata = bolaren_zain;
                }
```

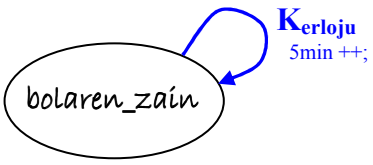
Gauza edo ekintza horiek automatan bertan argitzea komeni da, hori lagungarria izango baita programaren kodea idaztean. Horretarako, trantsizioaren geziaren azpian idatziko ditugu ekintza horiek guztiak, honelaxe:



b) Printzipioz, *itxita* egoeratik ateratzeko ez dago trantsizio posible gehiagorik.

2. *Bolaren_zain* **egoera:**

- Egoeraren ezaugarriak:
  Izenak adierazten duen bezala, egoera honetan sistema jokalariak bola noiz botako zain dago. Jokalariak bola botatzen duenean, SBola sentsoreak detektatuko du, eta bere kontroladoreak etena sortuko du hori adierazteko; horren ondorioz, sistema beste egoera batera pasatuko da. Baina sistema *bolaren_zain* egoeran dagoen bitartean, bola botatzeko 5 minutu ditu gehienez jokalariak: denbora-tarte hori pasatzen bada egoera honetan, beraz, bola bota gabe, hasierako egoerara itzuliko da sistema, partidari amaiera emateko. Hortaz, egoera honetan `5min` aldagaia modu egokian kontrolatu behar da, jakiteko noiz pasatu diren 5 minutu bola bota gabe.
  Beraz, hortik aukera batzuk aurreikusten dira:
  - Alde batetik, sistema *bolaren_zain* egoeran dagoen bitartean, `5min` aldagaia kontrolatu behar da, erlojuaren eten bakoitzarekin eguneratuz, baina horrek ez du eraginik izango sistemaren egoeran, hots, jokalariak bola botatzen ez badu, sistema egoera horretan mantenduko da 5 minutu pasatu arte. Hortaz, automatan honelaxe adieraziko dugu aukera hau:



↳ Erlojuaren zerbitzu-errutinaren eginkizunak:

Erloju.1) `5min` aldagaiaren eguneraketa erlojuaren eten bakoitzean:

```
if (egoera_automata == bolaren_zain)
   5min++;
```

Modu honetan jakingo dugu 5 minutu pasatu direla `5min` aldagaiak 5 × 60 × 18 balioa hartzen duenean, erlojuaren eten-kopurua kontatzen ari baikara.

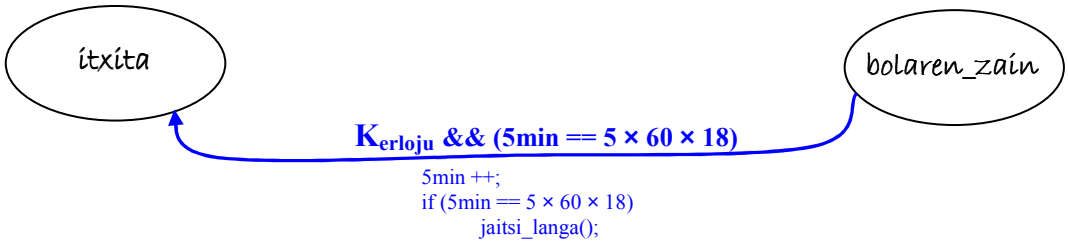
- Hala, sistema *bolaren_zain* egoeran dagoen bitartean, jokalariak bola ez badu botatzen 5 minutuak pasatu baino lehen, sistema *itxita* egoerara itzuliko da, hurrengo partida esleitzeko. Hori egoera-trantsizio bat denez gero, hurrengo atalean azalduko dugu.
- Azkenik, 5 minutu pasatu baino lehen, jokalariak bola botatzen badu, SBola sentsoreak detektatuko du, eta etena sortuko du hori adierazteko; horren ondorioz, sistema beste egoera batera pasatuko da. Garbi dago kasu honetan ere egoera-trantsizio bat gertatuko dela, eta horrexegatik hurrengo atalean azalduko dugu.

- Trantsizio posibleak:

a) Erlojuak eteten duenean, `5min` aldagaia eguneratu ondoren, haren balioa 5 × 60 × 18 baldin bada, orduan sistema *itxita* egoerara itzuliko da, horrek esan nahi baitu jokalariak ez duela bola bota. Horrekin batera, langa jaitsi beharko dugu, `jaitsi_langa()` funtzioaren bitartez. Hori guztia, automatan, gezi baten bidez adieraziko dugu, eta erlojuaren zerbitzu-errutinean, honelaxe:

Erloju.2) Egoera-trantsizioa, beharrezkoa baldin bada:

```
if (5min == 5 × 60 × 18)
   {
        jaitsi_langa();
        egoera_automata = itxita;
   }
```



b) Baina jokalariak bola botatzen badu 5 minutu pasatu baino lehen, orduan SBola sentsoreak detektatuko du eta etena sortuko du; horren ondorioz, sistema beste egoera batera pasatuko da, non 10 segundotan zain egon behar duen bolak botatako boloak egonkortu arte. Garbi dago, beraz, K_SBola kontroladorearen eten-eskaera izango dela gertaera horren adierazlea. Egoera berriari *egonkortzen* izena jarriko diogu, sistema horren zain egongo baita.

Hauek dira trantsizio honetan egin beharreko eginkizunak:

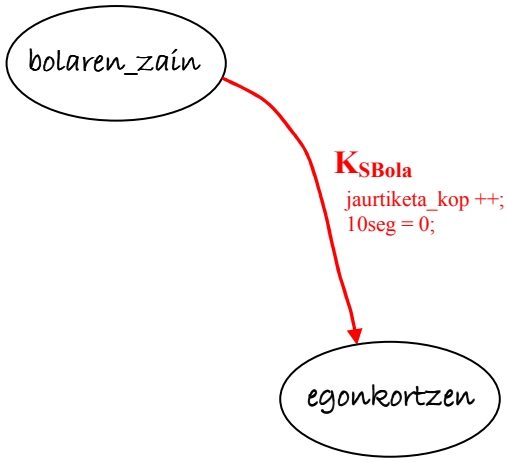
↳ SBola sentsorearen zerbitzu-errutinaren eginkizunak:

SBola.1) Alde batetik, bola detektatzeak esan nahi du jokalariak jaurtiketa bat gehiago egin duela; ondorioz, `jaurtiketa_kop` aldagaia eguneratuko dugu:

```
jaurtiketa_kop++;
```

SBola.2) Beste aldetik, egoera berrian sistemak 10 segundo zain egon behar du, eroritako boloak egonkortu zain. Horregatik, `10seg` izeneko aldagai orokorra izango dugu denbora-tarte hori kontrolatzeko, eta trantsizio honetan 0 balioa esleituko diogu, trantsizioa gertatzen den unetik aurrera kontatzen hasten baitira 10 segundoak, ez lehenago:

```
10seg = 0;
```

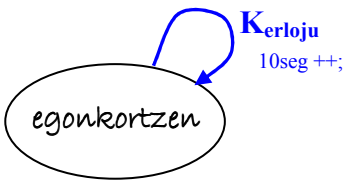


Baina eginkizun horiek bakar-bakarrik exekutatu behar dira baldin eta bola detektatzen bada sistema *bolaren_zain* egoeran dagoen bitartean, gerta baitaiteke bola bat detektatzea beste edozein unetan, eta horri ez zaio kasurik egin behar. Hala, K_SBola periferikoaren zerbitzu-errutinean, aurreko eginkizunen aurretik automataren egoeraz galdetuko dugu, honelaxe:

```
if (egoera_automata == bolaren_zain)
  {
      jaurtiketa_kop++;
      10seg = 0;
      egoera_automata = egonkortzen;
  }
```

3. *Egonkortzen* **egoera:**

- Egoeraren ezaugarriak:
  Egoera honetan sistema boloak egonkortu zain dago, beraz, denbora pasa besterik ez da egiten, erlojuaren zerbitzu-errutinak `10seg` aldagaia eguneratu behar du:



↳ Erlojuaren zerbitzu-errutinaren eginkizunak:

Erloju.3) `10seg` aldagaiaren eguneraketa erlojuaren eten bakoitzean:

```
if (egoera_automata == egonkortzen)
   10seg++;
```

- Trantsizio posibleak:

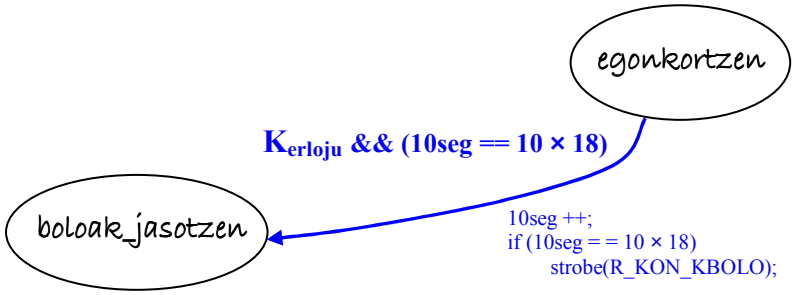
a) 10 segundo pasatu ondoren, suposatzen da boloak egonkortu direla, eta jaso daitezkeela: bolak zenbat bolo bota dituen kontatzeko, eta zutik uzteko hurrengo jaurtiketarako prest. Horretaz, K_Bolo periferikoa arduratzen da. Horregatik, 10 segundoko tartea amaitzean, K_Bolo aktibatu beharko dugu, martxan jar dadin eta egin beharrekoa egin dezan. Horretarako, K_Boloren kontrol-erregistroan `strobe` sekuentzia bat idatzi behar dugu, eta horren ondorioz, martxan jarriko da periferikoa. Hala, sistema beste egoera batera igaroko da, non K_Bolok boloak jasotzen dituen bitartean egongo den; *boloak_jasotzen* izena jarriko diogu beste egoera horri.

Hala, honelaxe geratuko zaigu erlojuaren zerbitzu errutina egoera-trantsizio honi dagokionez:

Erloju.4) Egoera-trantsizioa, beharrezkoa baldin bada:

```
if (10seg == 10 × 18)
   {
      strobe(R_KON_KBOLO);
      egoera_automata = boloak_jasotzen;
   }
```

Eta automatan:



4. *Boloak_jasotzen* **egoera:**

- Egoeraren ezaugarriak:

Egoera honetan, sistema K_Bolok boloak jaso zain dago, eta jaso dituela jakingo du horrek etena eskatzen duenean. K_Bolo izango da, beraz, gertaeraren sortzailea, eta, horren ondorioz, sistema beste egoera batera pasatuko da, baina trantsizioa jaurtiketa kopuruaren araberakoa izango da. Hurrengo atalean ikusiko ditugu bi trantsizio posibleak.

Baina, egoera-trantsizioaz gain, enuntziatuan esaten digute puntuazioa pantailaratu behar dela `pantailaratu_botatakoak(Bolo_kopurua,jaurtiketa_zenbakia)` funtzioari deituz. Horretarako, aldez aurretik jakin behar dugu zenbat bolo erori diren amaitu berri den jaurtiketan: hau da, K_Bolo kontroladorearen datu-erregistroa irakurri behar dugu, horrek azken jaurtiketan botatako boloen kopurua ematen baitigu. Hori gehitu behar diogu `bolo_kopurua` aldagai orokorrari, jakiteko

zenbat bolo bota dituen jokalariak guztira, jaurtiketa guztietan, eta azken aldagai hau da aurreko funtzioari parametro gisa pasatzen zaiona, jaurtiketa zenbakiarekin batera. Emaitza pantailaratzea bi trantsizio-kasu posibleetan egin behar da, partidak jarraitzen duenean eta partida amaitzen denean ere, azken puntuazioa ere pantailaratu behar delako.

↳ Bolo periferikoaren zerbitzu-errutinaren eginkizunak:

Bolo.1) Datu-erregistroa irakurri:

```
botatakoak = InPort(R_DAT_KBolo);
```

Bolo.2) `bolo_kopurua` aldagaia eguneratu:

```
bolo_kopurua = bolo_kopurua + botatakoak;
```

Bolo.3) Pantailaratu puntuazioa:

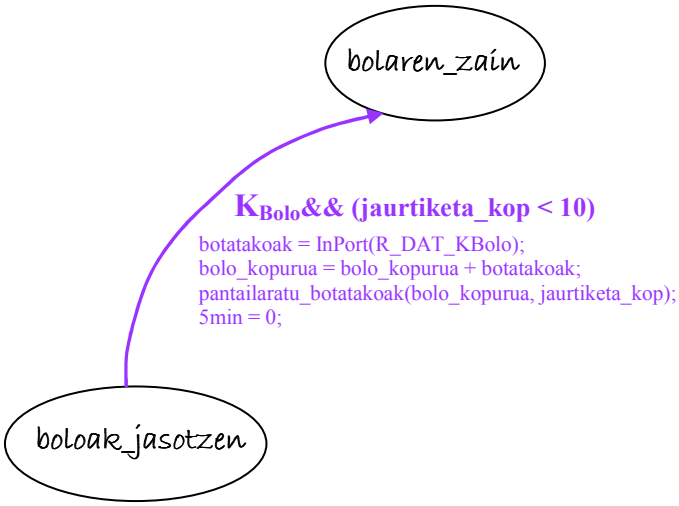
```
pantailaratu_botatakoak(bolo_kopurua,jaurtiketa_kop);
```

Hemen ez dugu egoeraren baldintza jarri lehenengo eginkizun gisa, suposatzen delako K_Bolok bakarrik eskatuko duela etena aldez aurretik aktibatua izan bada, eta hori gertatzen da *boloak_jasotzen* egoerara iristen denean sistema, ez beste egoeretan. Dena den, funtzionamendu txarra gerta daitekeela susmatzen bada, orduan baldintza jar daiteke aurreko sententzien aurretik:

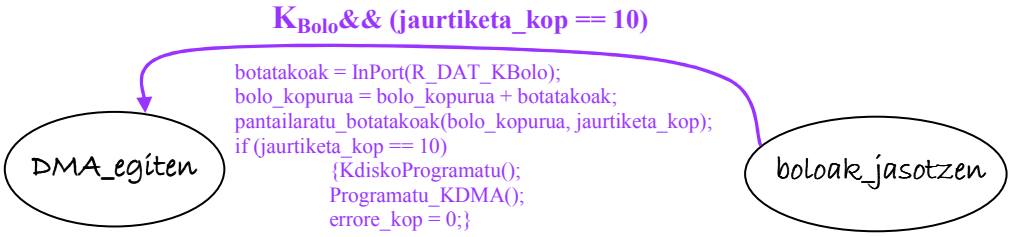
Bolo.0) `if (egoera_automata == boloak_jasotzen)`

- Trantsizio posibleak:

a) Dakigunez, partidan 10 jaurtiketa egin daitezke. Horregatik, K_Bolok eteten duenean abisatzeko boloak jaso dituela, `jaurtiketa_kop` aldagaia aztertu behar da: 10 baino txikiagoa baldin bada, horrek esan nahi du partidak jarraitzen duela, eta horregatik sistema *bolaren_zain* egoerara joango da berriro, hurrengo jaurtiketaren zain.
Kasu horretan, `5min` aldagaia berriro hasieratu beharko dugu 0 balioarekin, sistema berriro bolaren zain geratuko delako.



b) `jaurtiketa_kop` aldagaia 10 baino txikiagoa ez bada, partida amaitu da, jadanik 10 jaurtiketa egin dituelako jokalariak. Horregatik, puntuazioa pantailaratzeaz gain, DMA kontroladorea programatu behar da, puntuazioari buruz memoria nagusian gorde den informazio guztia diskora pasatzeko. Horretarako, diskoaren kontroladorea ere programatu behar da, `KdiskoProgramatu()` funtzioari deituz. Horregatik, kasu honetan, sistema beste egoera batera joango da, DMA transferentzia bukatu arte zain egoteko. `DMA_egiten` izena jarriko diogu egoera horri. Enuntziatuan esaten digutenaren arabera, DMA transferentzian errorea gerta daiteke, eta kasu horretan transferentzia berriro egin behar da, baina hiru aldiz, bakarrik. Horregatik, aldagai orokor bat behar dugu jakiteko zenbat aldiz gertatu den errorea DMA transferentzian, `errore_kop` deituko diogu, eta azken trantsizio honetan hasieratu behar dugu 0 balioarekin, argi baitago transferentzia egin baino lehen ez dela errorerik gertatu.
Honelaxe adieraziko dugu egoera-trantsizio hau automatan:



Honelaxe adieraziko ditugu bi egoera-trantsizio hauek eta dagozkien eginkizunak K_Boloren zerbitzu-errutinan:

↳ Bolo periferikoaren zerbitzu-errutinaren eginkizunak:

Bolo.4) Egoera-trantsizioak baldintzaren arabera:

```
if (jaurtiketa_kop < 10)
{
   5min = 0;
   egoera_automata = bolaren_zain;
}
else
{
   KdiskoProgramatu();
   Programatu_KDMA();
   errore_kop = 0;
   egoera_automata = DMA_egiten;
}
```

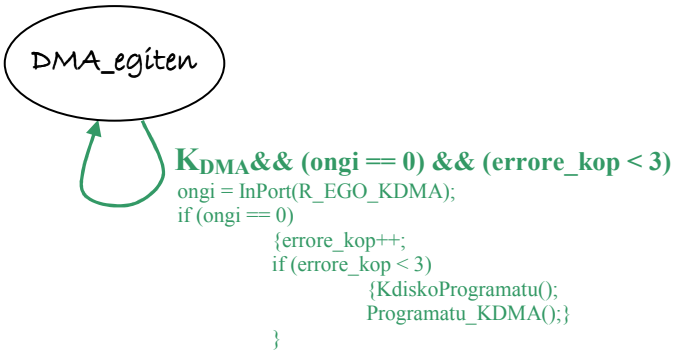
Hor erabili dugun `Programatu_KDMA()` funtzioa hauxe da:

```
OutPort(R_HEL_KDMA,PANT_HELB);
OutPort(R_LUZ_KDMA,4 × 10);
OutPort(R_KON_KDMA,1);
```

Helbidea PANT_HELB da pantailan dagoelako diskora transferitu behar dugun informazioa; transferitu behar den blokearen luzera 4 × 10 da, 10 jaurtiketa direlako, eta jaurtiketa bakoitzari dagokion puntuazioak 4 byte okupatzen dituelako memorian. Azkenik, kontrol-erregistroan 1 idaztean, transferentzia automatikoki hasiko da.

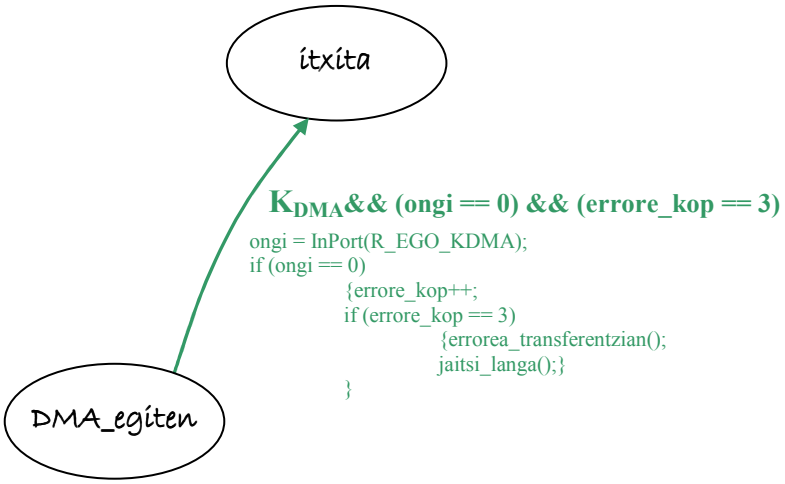
5. DMA_egiten **egoera:**

- Egoeraren ezaugarriak:
  Egoera honetan, DMA transferentzia burutu zain dago sistema. Eta transferentzia amaitu dela jakingo du DMA kontroladoreak etena eskatzen duenean. Hiru kasu gerta daitezke transferentzia amaitzean: a) Errorik ez gertatzea transferentzian; orduan, partida guztiz amaitutzat emango du sistemak eta *itxita* egoerara joango da, langa jaitsita, beste partida bat hasteko prest (noski, jokalariak jabeari ordaindu ondoren). b) Errorea gertatzea transferentzian, baina hiru errore baino gutxiago izatea; kasu honetan, egoera berean mantentzen da sistema, baina berriro programatu behar dira KDisko eta KDMA transferentziari berriro ekiteko. c) Errorea gertatzea hirugarren aldiz; kasu honetan, partida amaitutzat emango du sistemak, eta *itxita* egoerara itzuliko da, baina `errorea_transferentzian()` funtzioaren bidez horren berri emango dio jabeari. Edozein kasutan, transferentzian errorea gertatu den ala ez jakiteko, DMA kontroladorearen egoera-erregistroa irakurri behar da; honen balioa 1 baldin bada, transferentzia ongi joan da; 0 baldin bada, berriz, errorea gertatu da.
  Hurrengo atalean ikusiko ditugu bi trantsizio posibleak. Orain, egoera-trantsiziorik eragiten ez duen gertaera nola islatu automatan ikusiko dugu. Kodea, hurrengo atalaren bukaeran ikusiko dugu, beste bi trantsizioekin batera.

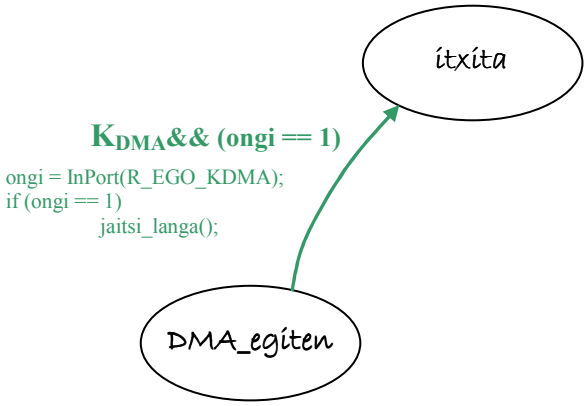


- Trantsizio posibleak:

a) Transferentzian errorea gertatzen bada hirugarren aldiz:



b) Transferentzian errorik ez bada gertatzen.

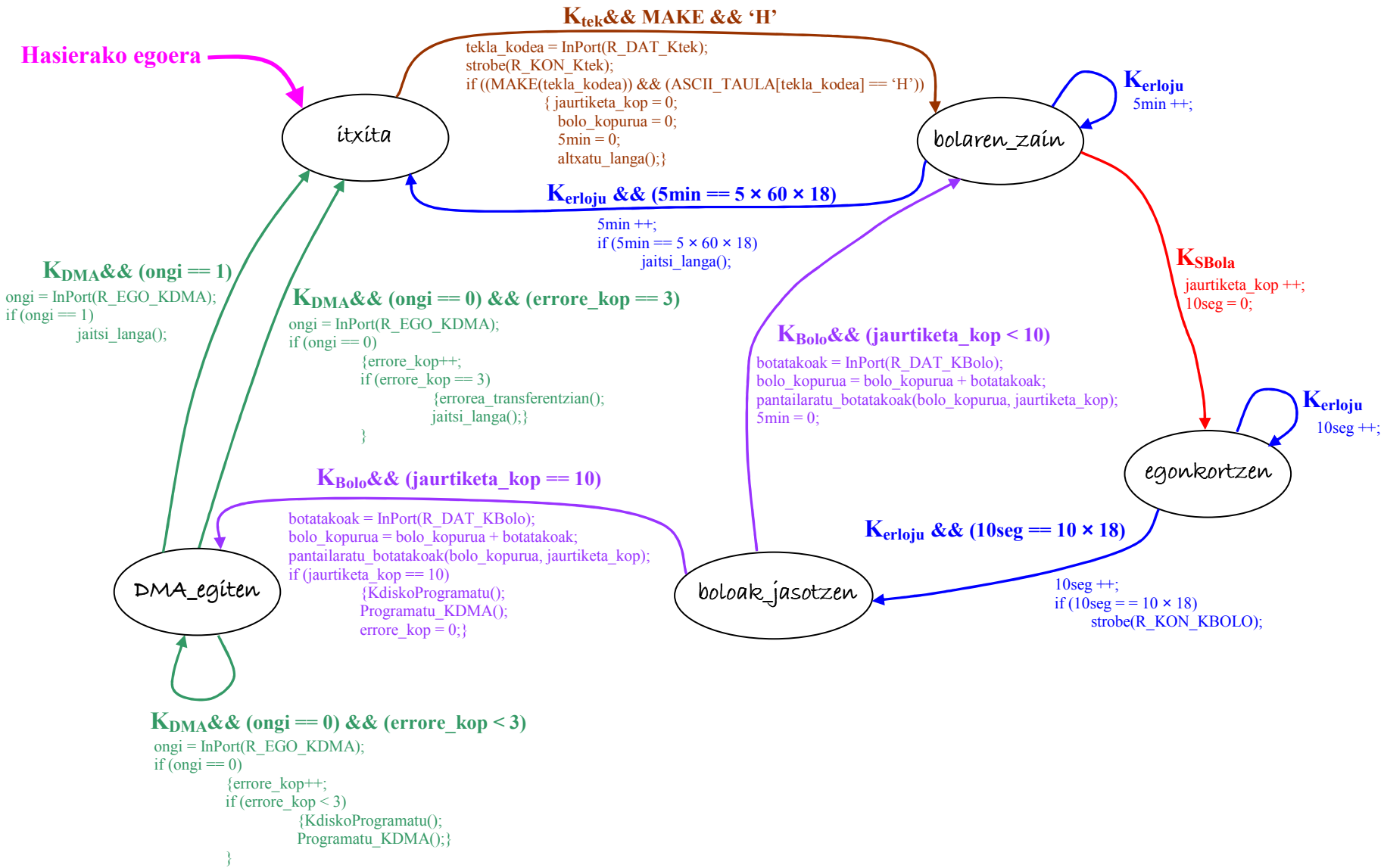


Kodea hiru kasuetarako:

↳ DMA kontroladorearen zerbitzu-errutinaren eginkizunak:

```
ongi = InPort(R_EGO_KDMA);
if (egoera_automata == DMA_egiten)
    {
      if (ongi == 0)
      {
            errore_kop ++;
            if (errore_kop < 3)
            {
                 KdiskoProgramatu();
                 Programatu_KDMA();
            }
            else
            {
                 errorea_transferentzian();
                 jaitsi_langa()
                 egoera_automata = itxita;
            }
      }
      else
      {
        jaitsi_langa();
        egoera_automata = itxita;
      }
    }
```

Kasu posible guztiak denak batera bilduz, honelaxe geratuko da automata:



Eta, orain, kodea idaztea besterik ez da falta. Honako hauek idatzi behar ditugu: programa nagusia, non begizta etengabea exekutatuko baita eta teklatuaren inkesta egingo baita; horrez gain, periferikoen etenen zerbitzu-errutinak ere idatzi behar ditugu: K_{SBola}, K_{Bolo}, K_{erloju} eta K_{DMA}.

Programa nagusia.

Programa nagusian, hainbat ekintza egin behar dira, hala nola: aldagaiak hasieratu (behar izanez gero), eten-bektorearen hainbat osagai aldatu (programaren hasieran) eta berreskuratu (bukaeran) –gure zerbitzu-errutinak exekutatuko direla ziurtatzeko, eta ez balizko sistema eragilearenak–, eta begizta bat ezarri gure programa behin eta berriro exekuta dadin, amaitu behar ez den bitartean. Begizta amaigabe horren barruan, teklatuaren inkesta egin behar da, detektatzeko noiz sakatzen den tekla bat. Teklatua inkestaz sinkronizatu ahal izateko, teklatuaren etenak galarazi behar dira programaren hasieran (bestela, tekla bat sakatzean edo askatzean etena sortuko luke teklatuaren kontroladoreak, eta hori saihestu behar da inkesta bidez sinkronizatu nahi dugunean). Programa amaitzen denean, berriz, teklatuaren etenak baimendu behar dira.

- Hasieratu beharreko aldagaiak:
  Hasteko, suposatuko dugu sistema martxan jartzen denean sistema *itxita* egoeran egongo dela. Garbi izan behar dugu, beraz, sistemaren funtzionamendu egokia ziurtatzeko beharrezkoa dela uneoro jakitea zein egoeratan dagoen sistema, eta horretarako aldagai global bat behar da, `egoera_automata` izendatuko duguna, zeinaren hasierako balioa *itxita* izango baita.
  Hala, honako hauek dira beharrezko hasieraketak:

  ✓ `egoera_automata = itxita;`

  Azpimarratu behar dugu ez dela beharrezkoa beste aldagaiak hasieratzea, erabili behar diren unean bertan balio egokiarekin hasieratzen direlako. Esate baterako, `10seg` aldagaia *bolaren_zain* egoeratik *egonkortzen* egoerara pasatzean hasieratuko da 0 balioarekin, une horretan jarri behar delako martxan "kronometroa".

- Aldatu eta berreskuratu beharreko eten-bektorearen osagaiak:
  Sistema hau kontrolatzeko PC arrunt bat erabiltzen badugu, orduan guztiz beharrezkoa da eten-bektorearen hainbat osagai aldatzea, gure periferikoen etenei dagozkienak hain zuzen, kontrola guk nahi dugun moduan egin dezan PCak, hau da, gure periferikoek etena eskatzen dutenean, guk idatzitako zerbitzu-errutinak exekuta daitezen, eta ez sistema eragileak aurre programatuak dituenak .
  Hala, kasu honetan honako hauek aldatu behar ditugu: erlojuarena (eten-bektorearen 0x1C sarreran dagoena), K_Bolo gailuarena (0x0A sarrerakoa), K_SBola sentsorearena (0x0B sarreran dagoena) eta DMA kontroladorearena (0x0E sarreran dagoena). Horretarako, suposatuko dugu jadanik idatzita dagoen funtzioa badugula, eta nahikoa dela aldatu beharreko sarrera parametro gisa pasatzea. Honelaxe:

  ✓ `Aldatu_EB(0x1C, 0x0A, 0x0B, 0x0E);`

Era berean, gure programa amaitzen denean, orduan eten-bektorearen osagai horiek berreskuratu beharko ditugu, sistema eragileak PCaren kontrola har dezan berriz ere. Honelaxe:

✓ `Berreskuratu_EB(0x1C, 0x0A, 0x0B, 0x0E);`

- Teklatuaren etenak galaraztea eta baimentzea:
  Ziurtatu behar dugu teklatuaren etenak ez direla iristen prozesadorera, orduan etenen bidez sinkronizatuko litzatekeelako teklatuarekin, eta guk inkesta bidez sinkronizatu nahi dugu. Horretarako, teklatuaren etenak galarazi behar dira bakarrik, ez periferiko guztienak; horregatik, etenen kontroladoreko maskara erregistroan 1 balioa idatzi beharko dugu 1 bitean. Modu horretan, tekla bat sakatzean edo askatzean etena gertatuko den arren, eten eskaera ez da iritsiko prozesadorera, eta honek beste modu batean detektatu beharko du gertaera hori. Teklatuaren etenak galarazteko, suposatuko dugu funtzio hau daukagula jadanik idatzita, zeinak parametro gisa IMR erregistroan aldatu beharreko bitaren zenbakia jasotzen baitu:

  ✓ `KperGalarazi(1);`

  Noski, programa amaitzen denean, berriro baimendu behar dira teklatuaren etenak. Horretarako, etenen kontroladoreko IMR erregistroko 1 bita 0an jarri behar da, funtzio honen bitartez:

  ✓ `KperBaimendu(1);`

- Begiztaren kontrola:
  Ziurtatu behar dugu gure programa etengabe exekutatzen dela, guk amaitzeko esaten ez diogun bitartean. Horretarako, begizta bat jarriko dugu, zeina behin eta berriro errepikatuko baita, amaitu artean. Kasu honetan, enuntziatuari erreparatuz, ez dago aurreikusita inolako aukerarik gure programaren exekuzioari amaiera emateko. Hori dela eta honako hau erabil dezakegu:

  ✓ `while (true)`

- Teklatuaren inkesta:
  Esan bezala, teklatuaren etenak galarazita baldin badaude, prozesadoreak beste modu batean detektatu beharko ditu teklatuaren gertaerak. Oro har, periferiko arruntekin nahikoa da egoera-erregistroa atzitzea jakiteko ea gertaeraren bat gertatu den. Baina PCko teklatua berezia da: haren kontroladoreak ez baitauka egoera-erregistrorik. Hori dela eta, tekla bat sakatu ala askatu den jakiteko modu bakarra etenen kontroladoreko IRR erregistroa aztertzea da, bertan islatzen direlako periferikoen eten-eskaerak. Tekla bat sakatzean, beraz, teklatuak etena eskatuko du eta IRR erregistroaren 1 bita 1ean jarriko da; baina teklatuaren etenak galarazita daudenez, hor geldituko da informazio hori, aurrera joan gabe, eta prozesadoreak irakurri beharko du erregistro hori ikusteko zein den bit horren balioa: 0 baldin bada (teklarik sakatu/askatu ez den seinale), behin eta berriro irakurriko du, inkesta jarraitua egiteko; horretarako, baldintzapeko begizta bat jarriko dugu. Balio hori 1 baldin bada, berriz, tekla bat sakatu/askatu den seinale;

orduan, aurreko begiztatik aterako da programa, eta teklatuari dagozkion eginkizunak egingo ditu programak.

Hala, hona hemen inkesta:

```
IRR = IrakurriIRR();
while (IRR1 == 0)
        IRR = IrakurriIRR();
```

Laburbilduz, honelaxe geratuko da programa nagusiaren kodea:

```
void main()
{
        //aldagaien hasieraketa
 egoera_automata = itxita;
        //eten-bektoreko osagaien aldaketa
 Aldatu_EB(0x1C, 0x0A, 0x0B, 0x0E);
        //galarazi teklatuaren etenak, inkestaz sinkronizatzeko
 KperGalarazi(1); //Eten-kontroladoreko IMR1 = 1
        //begizta nagusia
 while (true)
  {
        //teklatuaren inkesta
   IRR = IrakurriIRR();
   while (IRR1 == 0)
        IRR = IrakurriIRR();
        //tekla bat sakatu/askatu da (horregatik atera da aurreko begiztatik)
   tekla_kodea = InPort(R_DAT_KTEK);
   Strobe(R_KON_KTEK);
   if ((egoera_automata == itxita) && MAKE(tekla_kodea) &&
       ASCII_TAULA[tekla_kodea] == 'H')
       {
        jaurtiketa_kop = 0;
        bolo_kopurua = 0;
            //kronometroa martxan jarri 5 minutu kontatzeko
        5min = 0;
        altxatu_langa();
            //egoera-trantsizio egokia bideratu
        egoera_automata = bolaren_zain;
       }
  }
        //baimendu teklatuaren etenak
 KperBaimendu(1);
        //eten-bektoreko osagaien berreskurapena programaren amaieran
 Berreskuratu_EB(0x1C, 0x0A, 0x0B, 0x0E);
}
```

- K_{SBola} kontroladorearen zerbitzu-errutina.

Jadanik ikusi dugu zein diren zerbitzu-errutina honetan exekutatu beharreko eginkizunak. Orain, horiek ordenatzea besterik ez zaigu falta. Hona hemen:

```
void interrupt KSBola_ZE()
{
  if (egoera_automata == bolaren_zain)
  {
        //bola jaurtikia izan da egoera egokian
    jaurtiketa_kop++;
        //kronometroa martxan jarri 10 segundo kontatzeko
    10seg = 0;
        //egoera-trantsizio egokia bideratu
    egoera_automata = egonkortzen;
  }
  Eoi();
  IRET;
}
```

- K_{Bolo} gailuaren zerbitzu-errutina.

```
void interrupt KBolo_ZE()
{
      //une honetan botatako bolo kopurua irakurri
  botatakoak = InPort(R_DAT_KBolo);
      //guztira, jaurtiketa guztietan botatako bolo kopurua eguneratu
  bolo_kopurua = bolo_kopurua + botatakoak;
      //pantailaratu guztira botatako bolo kopurua
  pantailaratu_botatakoak(bolo_kopurua,jaurtiketa_kop);
      //egoera-trantsizio egokiak bideratu
  if (jaurtiketa_kop < 10)
    {
      5min = 0;
      egoera_automata = bolaren_zain;
    }
     else
     {
        KdiskoProgramatu();
        Programatu_KDMA();
        errore_kop = 0;
        egoera_automata = DMA_egiten;
     }
  Eoi();
  IRET;
}
```

Aurreko zerbitzu-errutinan `Programatu_KDMA()` funtzioa erabili dugu. Hona hemen haren kodea:

```
void Programatu_KDMA()
{
        OutPort(R_HEL_KDMA,PANT_HELB);
        OutPort(R_LUZ_KDMA,4*10);
        OutPort(R_KON_KDMA,1);
}
```

- K_{erloju} erlojuaren zerbitzu-errutina.

```
void interrupt K_erloju_ZE()
{
if (egoera_automata == bolaren_zain)
    {
            //5min aldagaia eguneratu erlojuaren eten bakoitzean
        5min++;
            //egoera-trantsizio egokiak bideratu
        if (5min == 5*60*18)
        {
            jaitsi_langa();
            egoera_automata = itxita;
        }
    }
if (egoera_automata == egonkortzen)
    {
            //10seg aldagaia eguneratu erlojuaren eten bakoitzean
        10seg++;
            //egoera-trantsizio egokiak bideratu
        if (10seg == 10*18)
        {
            //"strobe" sekuentzia KBolo gailuaren gainean, aktibatzeko
           strobe(R_KON_KBolo);
           egoera_automata = boloak_jasotzen;
        }
    }

IRET;
}
```

K_{DMA} kontroladorearen zerbitzu-errutina.

Oso garbi izan behar dugu zerbitzu-errutina hau DMA kontroladoreak transferentzia amaitzen duenean soilik exekutatuko dela, ez lehenago. Hortaz, une horretan kontroladorearen egoera-erregistroa irakurri behar da ikusteko ea transferentzia ongi amaitu den, ala erroreren bat gertatu den. Azken kasu honetan, berriro ekin behar zaio transferentzia egiteari, baina hiru aldiz gehienez. Hiru errore baino gehiago gertatzen badira, sistema ez da saiatuko egiten beste transferentzia bat, eta automata hasierako egoerara itzuliko da, beste partida bat hasteko prest geratzeko. Modu berean, transferentzia ongi amaitzen bada (errorerik gabe, alegia), sistema hasierako egoerara itzuliko da, partida amaitutzat emanez, eta hurrengo partidarako prest egoteko.

```
void interrupt K_DMA_ZE()
{
 ongi = InPort(R_EGO_KDMA);
 if (egoera_automata == DMA_egiten)
    {
      if (ongi == 0)
      {
            errore_kop ++;
            if (errore_kop < 3)
            {
                 KdiskoProgramatu();
                 Programatu_KDMA();
            }
            else
            {
                 errorea_transferentzian();
                 jaitsi_langa();
                 egoera_automata = itxita;
            }
      }
      else
      {
        jaitsi_langa();
        egoera_automata = itxita;
      }
 Eoi();
 IRET;
}
```