Sarrera/irteerako ariketen ebazpena errazteko asmoz, baina errealitatetik gehiegi aldendu gabe, C lengoaiaren antzeko pseudo-kodea erabiliko dugu. Aurkezten diren ariketen ebazpena luzeegia izan ez dadin, programatu beharko liratekeen behe-mailako funtzio batzuk jadanik kodetuta daudela suposatuko dugu. Hain zuzen, horrelako funtzioen behe-mailako programazioa laborategiko ariketetan lantzen da.

Honako funtzio hauek emango ditugu programatutzat:

`unsigned char` **`InPort`**`(erregistroa);`

Parametro gisa sarrera/irteerako espazioko erregistro baten helbidea jasotzen du, eta haren edukia itzultzen du. Ariketetan ez ditugu emango sarrera/irteerako erregistroen helbide zehatzak, etiketak edo izenak baizik. Hala, teklatuaren kontroladoreko datu-erregistroa honela adieraziko dugu: R_DAT_Ktek, eta hori izango da funtzioaren parametroa: `InPort (R_DAT_Ktek)`.

`void` **`OutPort`**`(erregistroa, balioa);`

Parametro gisa sarrera/irteerako espazioko erregistro baten helbidea eta balio bat jasotzen ditu, eta balioa idazten du erregistroan. Aurreko kasuan bezala erabiliko dugu erregistroaren etiketa; adibidez, periferiko baten kontrol-erregistroan idazteko: `OutPort (R_KON_Kper, balioa)`.

`void` **`Eoi`**`();`

End Of Interruption.Etenaren zerbitzu-errutinaren exekuzioa amaitu dela adierazteko: etenen kontroladoreak hori jasotzen duenean, ISR erregistroan 1 balioa duten bit guztien artean pisu handieneko bita desaktibatzen du, posible eginez lehentasun txikiagoko etenak zerbitzatzea.

`void` **`Strobe`**`(erregistroa);`

Periferiko baten kontrol-erregistroan "*strobe*" sekuentzia egiten du. Ariketa hauetan ez dugu kontuan hartuko zein den *strobe* egiteko erabili behar den bit zehatza; hori, laborategiko ariketetan lantzen da.

`int` **`MAKE`**`(tekla-kodea);`

Teklatuan tekla bat sakatzean/askatzean datu-erregistroan irakurtzen den tekla-kodea MAKE (tekla sakatu) ala BREAK (tekla askatu) den adierazten duen funtzio logikoa: 1 balioa itzultzen du tekla sakatu denean (MAKE), eta 0 balioa tekla askatzean (Break).

`unsigned char` **`IrakurriIRR`**`();`

PCaren etenen-kontroladoreko IRR erregistroaren edukia itzultzen du.

Beste aldetik, periferiko jakin eta bakar baten etenak galarazteko (maskaratzeko) edo baimentzeko, etenen kontroladoreko IMR maskara-erregistoaren bitartez, honako funtzio hauek kodetuta daudela suposatuko dugu:

```
void KperGalarazi(IMR_sarrera);
void KperBaimendu(IMR_sarrera);
```

Bi funtzio horiek parametro gisa pasatutako "IMR_sarrera" zenbakia duen bitaren balioa aldatzen dute IMR erregistroan, sarrera horrekin lotuta dagoen periferikoaren etenak maskaratzeko (1 balioa idatziz bit horretan) edo baimentzeko (0 balioa idatziz). Esate baterako, teklatuaren etenak galarazi nahi badira, teklatua etenen kontroladoreko IRQ1 sarreran dagoela kontuan hartuta, IMR erregistroaren 1 bita da aldatu behar dena; hortaz, deia honelaxe egin beharko genuke: `KperGalarazi(1)`.

Azkenik, gure aplikazioetan idatzitako berariazko zerbitzu-errutinak exekuta daitezen, eten-bektorearen sarrera egokiak aldatu behar dira programa nagusiaren hasieran, eta aldaketa egin aurretik sarrera horiek zituzten balioak berreskuratu behar dira programaren bukaeran. Horretarako, suposatuko dugu honako bi funtzio hauek ditugula kodetuta:

```
void Aldatu_EB(EB_sarrerak);
void Berreskuratu_EB(EB_sarrerak);
```

Bi funtzio hauek parametro gisa jasotzen dituzte eten-bektorean aldatu edo berreskuratu nahi diren sarrerak. Esate baterako, erlojuari eta teklatuari dagozkien sarrerak aldatzeko, honako funtzio-deia erabiliko dugu: `Aldatu_EB(0x1C,0x09)`, non `0x1C` sarrera erlojuari dagokion eta `0x09` sarrera teklatuari.

Beste aldetik, suposatuko dugu jadanik definituta dagoela bektore bat, ASCII_TAULA[], non teklen posizio-kodea indize gisa hartuta, tekla bakoitzari dagokion ASCII kodea lor daitekeen, baina MAKE kasuan lortutako posizio-kodea bakarrik erabil daiteke indize gisa, ez BREAK kasukoa. Hau da, taula horretan begiratu dezakegu zein den tekla bati dagokion ASCII kodea tekla sakatzen denean, inoiz ez tekla askatzean.