Kārlis Raimonds Eniņš, Gints Romanovskis LLU ITF MAG, 1.kurss. 14.03.2020.

## **Darbs tiešsaistē priekšmetā “Datorizētā projektēšana, modelēšana un ražošana”**

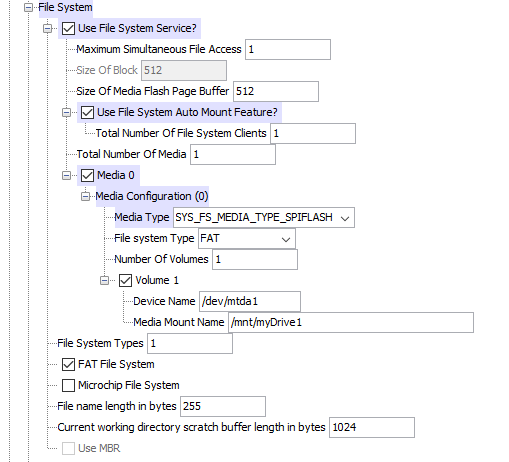
# **CASE rīki iegultajā programmēšanā.**

5.variants: USB flash hosts

Izmantotais mikrokontrolieris: PIC32MX795F512L

## MPLAB Harmony configurator konfigrācija

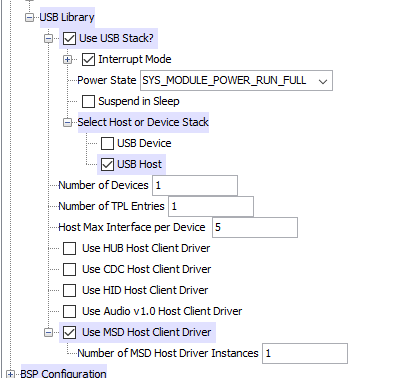
Pie opcijām sadaļā “Harmony Framework Configuration” ->”System Services” -> “File System” jāieslēdz failu sistēmas izmantošana. Jāiestata Failu sistēmas tips uz FAT, kā arī jāpārbauda kāds ir kokrētās ierīces vārds un piemontēšanas vārds.



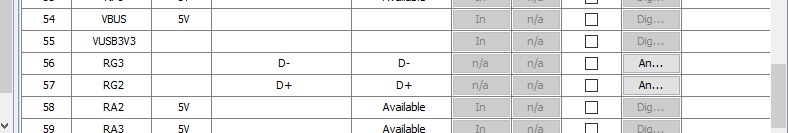
Piemontēšanas vārds vēlāk tiek izmantots kodā, lai norādītu, kur izvietot failu.

appData.fileHandle = SYS\_FS\_FileOpen("/mnt/myDrive1/simpleText.txt",(SYS\_FS\_FILE\_OPEN\_APPEND\_PLUS));

Pie opcijām sadaļā “Harmony Framework Configuration” ->”System Services” -> “USB Library” Jāieslēdz USB Stack, jānomaina tips no Device uz Host Stack un jāiesta MSD (Mass storage device) Host Client driver.



Mikrokontrolierim ar Harmony configurator palīdzību nokonfigurē USB paredzētās datu kājiņas RG2(57) un RG3 (56) kā D+ un D-. Attēlā redzams, ka VBUS(54) un VUSB3V3(55) jau ir pēc noklusējuma nokonfigurētas attiecīgi nepieciešamajām funkcijām.



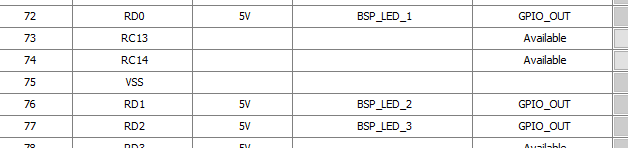
RF3(51) kājiņu kā USB ID



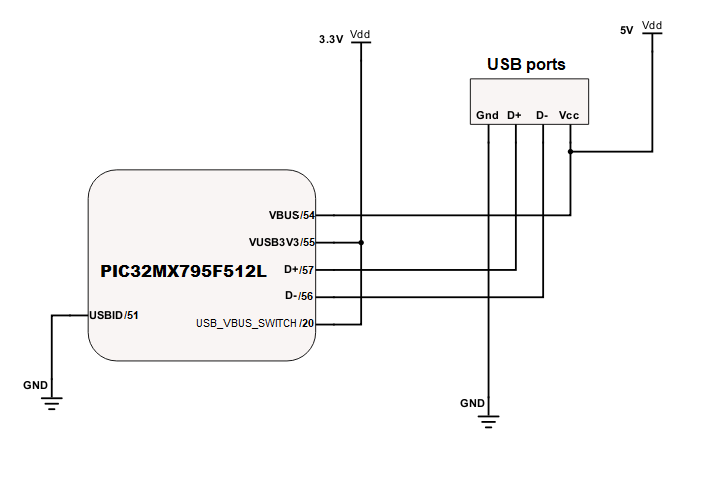
Un RB5(20) kā USB\_VBUS\_SWITCH (VBUS)



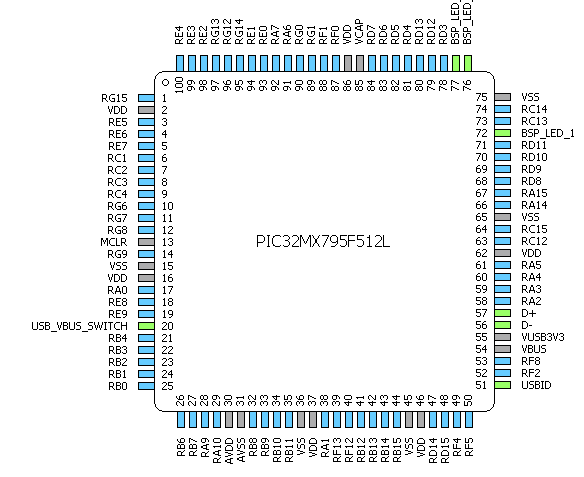
Papildus nokonfigurē visas trīs uz prototipēšanas plates esošās LED, kur attiecīgi BSP\_LED\_1 ir pieslēgts pie RD0(72), BSP\_LED\_2 pie RD1(76) un BSP\_LED\_3 pie RD2(77). Šīs LED tiek izmantotas, piemēram, programmatūras izpildes kļūdas parādīšanai.



USB ports uz prototipēšanas platformas “PIC32 Ethernet Starter Kit” pie mikrokontrolera pieslēgts pēc šādas shēmas.



Un nokonfigurējot pinus, Pin diagram izskatās šādi.



## Pievienotais kods

Papildus, jau klasē izstrādātajai projekta daļai ar 2 mirgojošiem LED, bija jāpievieno kods darbībai ar failsistēmu un ar USB ierīces pieslēgumiem.

Visa programmas darība ir organizēta ar “switch” palīdzība. Katrā “case” tiek organizēta zināmu procedūru izpilde un to izpildes pārbaude. Ja scenārija izpilde ir korekta, notiek pāriešana uz nākamo “case”, ja neizpildās, pāriet uz “Error case”.

Lai saprastu kādā stāvoklī atrodas sistēma, ir sekojoši uz prototipēšanas plates esošo LED kodi:

|  |  |  |
| --- | --- | --- |
| LED numurs | Stāvoklis(On/Off) | Nozīme |
| LED1 | On | Ir pievienota USB ierīce |
| LED2 | On | Gaidīšanas režīms |
| LED3 | On | Kļūda |
| Visi LED | Off |  |

# app.h koda apraksts

1. **void** APP\_Initialize(**void**) {
2. /\* Piefiksē programmas sākotnējo stāvoli . \*/
3. appData.state = APP\_STATE\_BUS\_ENABLE;
4. appData.deviceIsConnected = **false**;
5. }

Sakumā tiek inicializēts app sākotnējais stāvoklis. AppData.state = APP\_STATE\_BUS\_ENABLE tiek uzstādīta pazīme par to ka ir nepieciešams sākt ar bloku, kurā tiek uzstādīts USB BUS. Kā arī tiek uzstādīta pazīme par to ka USB draiveris sākotnēji nav pievienots.

1. USB\_HOST\_EVENT\_RESPONSE APP\_USBHostEventHandler(USB\_HOST\_EVENT event, **void** \* eventData, **uintptr\_t** context) {
2. **switch** (event) {
3. **case** USB\_HOST\_EVENT\_DEVICE\_UNSUPPORTED:
4. **break**;
5. **default**:
6. **break**;
7. }
9. **return** (USB\_HOST\_EVENT\_RESPONSE\_NONE);
10. }

USB\_HOST\_EVENT\_RESPONSE APP\_USBHostEventHandler funkcija ķer ierīces notikumus.

USB\_HOST\_EVENT\_DEVICE\_UNSUPPORTED šis notikums notiek, ja resursdatora slānis nevarēja pievienot draiverus pievienotā ierīce vai kad ir parādījusies kļūda. Nav datu par notikumiem, kas saistīti ar šo notikumu.

1. **void** APP\_SYSFSEventHandler(SYS\_FS\_EVENT event, **void** \* eventData, **uintptr\_t** context) {
2. **switch** (event) {
3. **case** SYS\_FS\_EVENT\_MOUNT:
4. appData.deviceIsConnected = **true**;
5. **break**;
7. **case** SYS\_FS\_EVENT\_UNMOUNT:
8. appData.deviceIsConnected = **false**;
10. **break**;
12. **default**:
13. **break**;
14. }
15. }

APP\_SYSFSEventHandler Funkcija klausās faila sistēmas notikumus. Ja notikums ir SYS\_FS\_EVENT\_MOUNT, tad faila sistēma ir fiksējusi to, ka ierīce ir pieslēgta. Savukārt pretēji SYS\_FS\_EVENT\_MOUNT, SYS\_FS\_EVENT\_UNMOUNT norāda uz to ka ierīce nav pievienota.

**App Tasks** metode atbild par galvenajām darbībām kontrolieri. Komentāri ir pielikti kodā.

1. **void** APP\_Tasks(**void**) {
2. **switch** (appData.state) {
3. **case** APP\_STATE\_BUS\_ENABLE:
5. /\* Iestatata notikumu apstradataju un iesledz bus \*/
6. SYS\_FS\_EventHandlerSet(APP\_SYSFSEventHandler, (**uintptr\_t**) NULL);
7. USB\_HOST\_EventHandlerSet(APP\_USBHostEventHandler, 0);
8. USB\_HOST\_BusEnable(0);
9. appData.state = APP\_STATE\_WAIT\_FOR\_BUS\_ENABLE\_COMPLETE;
10. **break**;
12. **case** APP\_STATE\_WAIT\_FOR\_BUS\_ENABLE\_COMPLETE:
13. **if** (USB\_HOST\_BusIsEnabled(0)) {
14. appData.state = APP\_STATE\_WAIT\_FOR\_DEVICE\_ATTACH;
15. }
16. **break**;
18. **case** APP\_STATE\_WAIT\_FOR\_DEVICE\_ATTACH:
19. /\* gaida ierices pievienosanu. Stavokla  tiks parvietots
20. \* uz nakamo stavokli, kad pievienosanas notikums
21. \* ir sanemts. \*/
23. **if** (appData.deviceIsConnected) {
24. BSP\_LED\_1On();
25. BSP\_LED\_2Off();
26. appData.state = APP\_STATE\_DEVICE\_CONNECTED;
27. }
29. **break**;
31. **case** APP\_STATE\_DEVICE\_CONNECTED:
33. /\*Ierice tika pieslegta . Mes varam meginat uzstadit disku \*/
34. appData.state = APP\_STATE\_OPEN\_FILE;
35. **break**;
37. **case** APP\_STATE\_OPEN\_FILE:
39. /\* meginiet atvert failu pievienosanai \*/
41. appData.fileHandle = SYS\_FS\_FileOpen("/mnt/myDrive1/file.txt", (SYS\_FS\_FILE\_OPEN\_APPEND\_PLUS));
42. **if** (appData.fileHandle == SYS\_FS\_HANDLE\_INVALID) {
43. /\* Failu nevareja atvert tiek uzstadita kluda\*/
44. appData.state = APP\_STATE\_ERROR;
46. } **else** {
47. /\* Fails tika veiksmigi atvets . Tiek pariets uz  faila rakstisanu \*/
48. appData.state = APP\_STATE\_WRITE\_TO\_FILE;
50. }
51. **break**;
53. **case** APP\_STATE\_WRITE\_TO\_FILE:
55. /\* Meginiet rakstit fail? \*/
56. **if** (SYS\_FS\_FileWrite(appData.fileHandle, (**const** **void** \*) writeData, 12) == -1) {
57. /\* Rakstisana nebija veiksmaga. Aizvert failu
58. \* un uzstada kludu.\*/
59. SYS\_FS\_FileClose(appData.fileHandle);
60. appData.state = APP\_STATE\_ERROR;
62. } **else** {
63. /\* Rakstisana bija veiksmiga un pabeigta.Fails tiek aizverts \*/
64. appData.state = APP\_STATE\_CLOSE\_FILE;
65. }
67. **break**;
69. **case** APP\_STATE\_CLOSE\_FILE:
71. /\* Aizver failu\*/
72. SYS\_FS\_FileClose(appData.fileHandle);
74. /\* Parbaude vai bija veiksmiga. Aiziet dikstave \*/
75. appData.state = APP\_STATE\_IDLE;
76. **break**;
78. **case** APP\_STATE\_IDLE:
80. /\* aplikacija paradas seit, kad demonstracija ir pabeigta
81. \* veiksmigi. Tiek paradits INDIKATORS. Gaidit ierices atvienosanu
82. \* un, ja tas ir atvienots, pagaidiet, kamer tas tiks  pievienots. \*/
84. BSP\_LED\_3Off();
85. BSP\_LED\_2On();
86. **if** (appData.deviceIsConnected == **false**) {
87. appData.state = APP\_STATE\_WAIT\_FOR\_DEVICE\_ATTACH;
88. BSP\_LED\_2Off();
89. }
90. **break**;
92. **case** APP\_STATE\_ERROR:
94. /\* Datbibas tiek novirzitas uz so bloku ja APP\_STATE
95. \* sadala izkrita ar kludu.\*/
97. BSP\_LED\_1On();
98. **if** (SYS\_FS\_Unmount("/mnt/myDrive") != 0) {
99. /\*Disku nevareja atvienot. \*/
101. appData.state = APP\_STATE\_ERROR;
102. } **else** {
103. /\* Diska atvienosana ir sekmiga \*/
104. appData.state = APP\_STATE\_WAIT\_FOR\_DEVICE\_ATTACH;
106. }
107. **break**;
109. **default**:
110. **break**;
112. }
113. }

Darbību rezultātus var novētot pētot LED darbību.

* Ja ierīce ir pieslēgta un ar to ir savienojums tad tiek ieslēgta 1. LED diode un izslēgta 2. LED diode.
* Pēc veiksmīgas ierakstīšanas programma aiziet dīkstāve un par to signalizē LED2 un LED3. LED3 tiek izslēgta, savukārt LED2 tiek ieslēgta.
* Ja pēc veiksmīgas ierakstīšanas tiek izslēgta LED2, tad tas lēcienā par to ka kontrolieris ir atvienojies no usb draivera.
* Ja APP\_STATE\_ERROR tad tiek ieslēgta LED1