**Mikrorendszerek tervezése  
házi feladat dokumentáció**

Kígyó játék megvalósítása FPGA platformon  
MicroBlaze processzor segítségével

**Készítette:**

Frank János – YHAF2Q

Csőke Lóránt Tibor – EYUNCP

**Konzulens:**

Raikovich Tamás – BME MIT

**Felhasznált eszközök:**

* Logsys Kintex 7 board: Kintex 7 - XC7K70T-FBG676-I FPGA
* Logsys VGA, PS/2 and speaker module
* VGA monitor
* USB kábel
* VGA kábel

Tartalomjegyzék

[1 Bevezetés: 3](#_Toc531718014)

[2 Elvégzendő feladatok 4](#_Toc531718015)

[3 Felépítés 5](#_Toc531718016)

[3.1 Hardver: 5](#_Toc531718017)

[3.1.1 VGA vezérlő modul: 5](#_Toc531718018)

[3.1.2 7-szegmenses kijelző vezérlő modul: 6](#_Toc531718019)

[3.2 Szoftver 8](#_Toc531718020)

[3.2.1 Játékmenet vezérlés 8](#_Toc531718021)

[3.2.2 Adatmodell: 8](#_Toc531718022)

[3.2.3 Függvények: 10](#_Toc531718023)

[3.2.4 Billentyűzet emulátor 12](#_Toc531718024)

[4 Összefoglalás 13](#_Toc531718025)

[5 Ábrajegyzék 14](#_Toc531718026)

[6 Irodalomjegyzék 15](#_Toc531718027)

[7 Irodalomjegyzék 16](#_Toc531718028)

[8 Függelék 17](#_Toc531718029)

[8.1 VGA vezérlő HDL kódja 17](#_Toc531718030)

[8.2 7-szegmenses kijelző vezérlő kódja 20](#_Toc531718031)

[8.3 Snake.c 20](#_Toc531718032)

[8.4 Snake.h 30](#_Toc531718033)

[8.5 Drivers.c 31](#_Toc531718034)

[8.6 Drivers.h 31](#_Toc531718035)

[8.7 main.c 32](#_Toc531718036)

[8.8 Billentyűzet emulátor 32](#_Toc531718037)

# Bevezetés:

A félvezető eszközök megjelenése nagyjából az 1950-as 60-es évekre tehető az informatikában. Először a kezdetleges számítógépekben alkalmazott reléket és elektroncsöveket –melyekből némely akár több tízezer darabot is tartalmazott– cserélték le a már kiforrott tranzisztorokra. Később, a félvezetőgyártásban alkalmazott technológia fejlődésének köszönhetően megjelentek az integrált áramkörök, melyek már több tranzisztorból álltak és képesek voltak egyszerű logikai műveletek végrehatására. Az integráltság Moore törvény szerinti fejlődéssel az 1970-es évek elejére megjelentek az LSI, VLSI IC-k velük együtt a mikroprocesszorok mint pl. az Intel 8080-as . Ezek a berendezések azonban fix logikai és aritmetikai műveletek (összeadás, kivonás, szorzás, osztás, komparálás…) végrehajtására voltak képesek a bemenő adatokon egy előre meghatározott sorrend szerint, melyet a futtatott program határozott meg.

A programozható hardver iránti igény az 1980-as években jelent meg, melyet először EEPROM-ban megvalósított look-up táblákkal alakítottak ki. Később megjelentek, a CPLD-k –komplex programozható logikai eszköz– melyekben már több tízezer kaput lehetett vezérelni. Ezek az eszközök azonban korlátozott szerkezetűek, nem túl flexibilisek voltak . Az FPGA-k megjelenésével egy sokoldalúan alkalmazható logikai eszközhöz jutott a világ. Az első időszakban a feladataik inkább mikroprocesszoros rendszerekben segédfunkciók ellátása volt, mint pl. memóriák illesztése, vagy matematikai műveletek hardveres gyorsítása. Később integráltságuk növekedésével azonban lehetőség nyílt magát a mikroprocesszort is az FPGA-ban létrehozni, melyet manapság „soft-core” processzornak hívnak . Megjelent a működés közbeni parciális újrakonfigurálás lehetősége, valamint az ún. „hard-core” processzorral ellátott SoC –system on chip– rendszerek.

# Elvégzendő feladatok

A házi feladat a soft-core processzoros rendszerekhez kapcsolódik. A cél egy működő összetett hibrid mikroprocesszoros rendszer tervezése és elkészítése volt, mely a népszerű kígyó játékot futtatja 640x480 pixel felbontásban egy VGA monitoron. A probléma megoldásához a MicroBlaze soft-core processzor IP-t használtuk, melyez AXI-stream interfészen egy saját VGA vezérlőt illesztettünk. A rendszert elláttuk a megfelelő kiegészítő blokkokkal, mint pl. a DDR3 memóriavezérlő vagy az AXI-lite interfészen kapcsolódó 7-szegmenses kijelzőt vezérlő modul. A felhasználóval történő kommunikációhoz PS/2-es billentyűzetet szerettünk volna alkalmazni, azonban erre nem volt lehetőségünk, így készítettünk egy C# alkalmazást, mely figyelte a lenyomott billentyűket és továbbküldte azokat soros terminálon keresztül az FPGA-nak, mely feldolgozta azt. Így tulajdonképpen egy billentyűzet emulátort készítettünk a rendszerhez.

**A megoldandó feladatok:**

**Hardver:**

1. MicroBlaze processzor konfigurációja
2. MIG7 – DDR3 memóriavezérlő illesztése, konfigurációja
3. AXI UART modul illesztése, konfigurációja
4. VGA vezérlő modul megírása, illesztése AXI-Stream interfészen keresztül
5. DMA vezérlő konfigurációja, illesztése
6. FIFO buffer létrehozása az DMA és a VGA modul között, konfigurációja
7. AXI – Timer modul illesztése
8. AXI – GPIO modul illesztése
9. 7-szegmenses kijelző vezérlő modul elkészítése, illesztése AXI-Lite buszon keresztül

**Szoftver:**

1. A különböző perifériák inicializálása, felkonfigurálása
2. Képmemória szekciójának létrehozása linker scriptben
3. A játékot megvalósító algoritmus megírása
4. Billentyűzet emulátor készítése C#-ban

# Felépítés

A feladat megoldása során a MicroBlaze processzorhoz számos külső periféria illesztése volt szükséges. Ezeknek a konfigurációját a gyakorlatokon bemutatottaknak megfelelően végeztük, így azok részletes leírásával most nem foglalkozunk, csak az általunk készített modulokat mutatjuk be. A DDR memória illesztéséhez a MIG7 memória vezérlőt használtuk, mely konfigurációjának útmutatója -ben található.

## Hardver:

### VGA vezérlő modul:

A feladat egyik fontos része a saját VGA vezérlő modul elkészítése volt. A megoldáshoz segítségül vettük a Logsys VGA és PS/2 vezérlő modulhoz tartozó dokumentációt, melyben megtalálhatóak voltak a szükséges pixel órajelek valamint a vezérlő jelek létrehozásához szükséges információk, adatok . A blockdesign-ba illeszthető vezérlő modul létrehozásához a Vivado IP generátorát használtuk.

A modul AXI-Stream interfészen keresztül fogadja a pixeladatokat a DMA vezérlőtől, egy FIFO bufferen keresztül. A kép pixelei két darab, a külső DDR3 memóriában foglalt tömbben kerültek eltárolásra, melyből a DMA felváltva továbbította az adatokat. Az így megvalósított dupla pufferelt megjelenítés segítségével nem történik hozzáférési ütközés a memóriában, hisz még a processzor az új képhez szükséges adatokat írja addig a DMA vezérlő a másik képet olvassa.

**A modul neve: AXI4S\_VGA\_v1\_0\_AXI4S\_0**

**A modul bemeneti portjai:**

* S\_AXIS
* S\_AXIS\_ACLK
* S\_AXIS\_ARESETN

**A modul kimeneti portjai:**

* R0, R1, G0, G1, B0, B1
* H\_sync
* V\_sync

**Blokkvázlat:**

|  |
| --- |
|  |
| . ábra: VGA vezérlő modul blokkvázlat |

### 7-szegmenses kijelző vezérlő modul:

A Kintex-7 kártyán található egy 7-szegmenses kijelző, melyet a játékban elért pontok megjelenítésére alkalmaztunk . Ehhez írtunk egy egyszerű Verilog kódot, mely megvalósította a kijelző időmultiplexált vezérlését. A megjelenítendő adatokat AXI-Lite interfészen keresztül juttattuk el a processzortól a modulnak, melyet a Vivado IP Packager-rel illesztettük a blockdesign-ba.

**A modul neve: SevenSegmentDriver\_0**

**A modul bemeneti portjai:**

* S00\_AXI
* s00\_axi\_aclk
* s00\_axi\_aresetn

**A modul kimeneti portjai:**

* SegmentOut[7:0]
* DigitOut[3:0]

**Blokkvázlat:**

|  |
| --- |
|  |
| 3. ábra: 7-szegmenses kijelző vezérlő modul blokkvázlat |

## Szoftver

A szoftver elkészítéséhez először legeneráltuk a Board Support Ppackage-et (BSP), mely tartalmazza a processzor és a hozzá illesztett perifériák használatához szükséges memóriacímeket, függvényeket. A fejlesztéshez a Vivado beépített SDK-ját használtuk. A képmemória létrehozásához kijelöltünk egy szekciót a linker scriptben, megadva azzal, hogy az adott terület a DDR3-memóriába kerüljön. A szükséges hardver elemek inicializálását követően elindítottuk a játék ütemezéséhez használt AXI Timert, valamint az algoritmust.

### Játékmenet vezérlés

A játék egy időzítőre alapul, mely ütemezi a játékmenetet. A különböző szintek közötti előrelépés a timer periódusidejének csökkentésével történik, így gyorsítva a játékot. Minden egyes ütemezéskor a következő frame kiszámolódik, miközben a DMA felváltva továbbítja a képadatokat a VGA vezérlőnek a két pufferből. Új játék indulása esetén a képek törlését darabokban kell elvégezni, mivel a processzor és a DDR memória közötti adatátvitel lassú. Erre azt a megoldást választottuk, hogy míg az egyik kép törlése zajlik, addig a DMA a másik képet továbbítja a vezérlőnek. Így a kép egy rövid időre befagy, majd kezdődik a játék elölről. A kígyó leírását annak töréspontjai eltárolásával oldottuk meg. Az adott pontok között a kirajzoláskor csak egy egyenest kell húzni, így megkapható a kígyó alakzata. Az alakzat mozgatása úgy történt, hogy az utolsó szegmensből (a kígyó farka) elvettünk egy egységet, majd a fejénél kirajzoltattuk. Étel találata esetén az elvétel lépése, azaz a kígyó rövidítése elmarad.

A szükséges lépéseket az időzítő által megszabott periódusonként számoltuk újra, majd írtuk ki az éppen szabad, azaz a DMA által nem olvasott képmemóriába.

### Adatmodell:

A szoftverben alkalmazott tömbök struktúrák leírása.

**picture<#n>:**

A linker scriptben megadot *.extmem* memóriaterületen lefoglalt dupla képmemória, mely a dupla pufferelt megjelenítést szolgálja.

unsigned char picture1**[**480**][**640**]** \_\_attribute\_\_**((**aligned**(**128**),** section**(**".extmem"**)));**

unsigned char picture2**[**480**][**640**]** \_\_attribute\_\_**((**aligned**(**128**),** section**(**".extmem"**)));**

**point\_typedef:**

A képen megjelenítendő pontokat leíró struktúra.

**typedef** struct

**{**

int x**;**

int y**;**

**}** point\_typedef**;**

**direction\_typedef:**

A lenyomott billentyűk és a haladási irány leírására szolgál.

**typedef** enum

**{**

UP **=** 0**,**

DOWN **=** 1**,**

LEFT **=** 2**,**

RIGHT **=** 3**,**

NONE **=** 4

**}**direction\_typedef**;**

**fillData\_typedef:**

Négyzetesen kitöltendő terület leíró struktúrája a megjelenítéshez.

**typedef** struct

**{**

int x0**;**

int y0**;**

int x1**;**

int y1**;**

uint8\_t color**;**

**}**fillData\_typedef**;**

**SEG7:**

7-szegmenese kijelző kódolása.

const uint8\_t SEG7**[**10**]** **=** **{**0x3F**,** 0x06**,** 0x5B**,** 0x4F**,** 0x66**,** 0x6D**,** 0x7D**,** 0x07**,** 0x7F**,** 0x6F**};**

### Függvények:

Az alkalmazott függvények bemutatása, funkciójuk leírása.

**snake.c/.h**

void swap**(**int**\*** x**,** int **\***y**);**

Két pontkoordináta fölcserélése.

void FillRectangle**(**fillData\_typedef**\*** in**,** uint8\_t**\*** picture**)**;

Négyzet kirajzolása a kijelzőre.

int GetRandom**(**int min**,** int max**)**;

Visszaad egy random számot „min” és „max” között.

int PointFitsLine**(**point\_typedef p**,** point\_typedef l0**,** point\_typedef l1**)**;

Ellenőrzi, hogy rajta van-e a „p” pont az „l0” és „l1” pontok közötti szakaszon;

void ShiftPoints**()**;

Eggyel lépteti a kígyót leíró pontokat.

void StepAhead**()**;

A kígyó léptetését számoló függvény. Ez tölti fel az „addHead” és „removeTail” változókat, melyek a képeken történő alakzatok megjelenítését szolgálják.

void RemoveTail**()**;

A kígyó léptetése során lekezeli, ha egy töréspont „elfogy”.

int CheckWall**()**;

Fallal való ütközés a megállapítása.

int CheckBite**();**

Saját magába harapás megállapítása.

void PlaceNewFood**();**

Új élelem elhelyezése.

int FoundFood**();**

Étel találata során történő módosítások végrehajtása.

int StepSnake**();**

A kígyó léptetésének menedzselése.

int ClearScreen**(**uint8\_t **\***pic**,** int first**);**

Képernyő törlése, alaphelyzetbe állítás.

int ClearManage**(**int **\***clearStart**);**

Részekre bontott törlés végrehajtása.

void ScanButtons**();**

Lenyomott billentyű parancsértelmező.

void UpdateGame**();**

Játékmenet léptetés, új koordináták számolása.

void SnakeMain**();**

A játék indítása.

**Drivers.c/.h**

void dma\_init**(**unsigned long baseaddr**);**

DMA vezérlő inicializálása.

void dma\_mm2s\_start**(**unsigned long baseaddr**,** void **\***src**,** unsigned long length**);**

DMA transzfer indítása.

uint32\_t dma\_mm2s\_finished**(**unsigned long baseaddr**);**

Befejeződött DMA transzfer jelentése. Pollingolt megoldás.

void timer1\_Init**();**

AXI Timer inicializálása.

uint32\_t GetTick**();**

Aktuális timer érték kiolvasása.

### Billentyűzet emulátor

A PS/2 billentyűzet helyett egy C# alkalmazás segítségével lehetőség nyílik soros terminálon keresztül elküldeni a host számítógépen lenyomott billentyűket a kártyának. A program egy egyszerű soros terminált nyit, majd loggolja lenyomott billentyűket és továbbítja azokat az FPGA-nak.

|  |
| --- |
|  |
| . ábra: Billentyűzet emulátor C# alkalmazás |

# Összefoglalás

A feladat megoldása során rendkívül sok hasznos tapasztalatra tettünk szert egy összetett FPGA-s rendszer tervezésével kapcsolatban. Megismertük a Vivado blockdesign tervezőjét, valamint a szükséges modulok felkonfigurálásának módját. Meg tanultuk hogyan kell saját AXI perifériát készíteni, valamint beilleszteni azt a meglévő projektbe. Rálátást nyertünk, hogyan történik egy hibrid mikroprocesszoros és hardveres rendszer elkészítése, annak tesztelése, szimulációja.

|  |
| --- |
| KigyoJatek.JPG |
| . ábra: Játékmenet képernyőképe |

# Ábrajegyzék

[3.1 ábra: VGA vezérlő modul blokkvázlat 6](#_Toc531718038)

[3.2 ábra: 7-szegmenses kijelző vezérlő modul blokkvázlat 7](#_Toc531718039)

[3.3 ábra: Billentyűzet emulátor C# alkalmazás 12](#_Toc531718040)

[4.1 ábra: Játékmenet képernyőképe 13](#_Toc531718041)

# Irodalomjegyzék

# Irodalomjegyzék

|  |  |
| --- | --- |
| [1] | Wikipedia, „Microprocessor,” 20 11 2018. [Online]. Available: https://en.wikipedia.org/wiki/Microprocessor. [Hozzáférés dátuma: 25 11 2018]. |
| [2] | Wikipedia, „Very Large Scale Integration,” 09 11 2018. [Online]. Available: https://en.wikipedia.org/wiki/Very\_Large\_Scale\_Integration. [Hozzáférés dátuma: 25 11 2018]. |
| [3] | Wikipedia, „Field-programmable gate array,” 25. 11. 2018.. [Online]. Available: https://hu.wikipedia.org/wiki/Field-programmable\_gate\_array. [Hozzáférés dátuma: 25. 11. 2018.]. |
| [4] | BME-MIT, „A Memory Interface Generator (MIG) beállítása a Logsys Kintex-7 FPGA kártyához,” LOGSYS, Budapest, 2018.. |
| [5] | BME-MIT, „LOGSYS VGA, PS/2 ÉS HANGSZÓRÓ MODUL FELHASZNÁLÓI ÚTMUTATÓ,” LOGSYS, Budapest, 2010.06.25.. |
| [6] | BME-MIT, „LOGSYS KINTEX-7 FPGA KÁRTYA FELHASZNÁLÓI ÚTMUTATÓ,” LOGSYS, Budapest, 2018.04.03.. |

# Függelék

## VGA vezérlő HDL kódja

`timescale 1 ns **/** 1 ps

**module** AXI4S\_VGA\_v1\_0\_AXI4S\_In

**(**

// Users to add ports here

**output** R1**,**

**output** R0**,**

**output** G1**,**

**output** G0**,**

**output** B1**,**

**output** B0**,**

**output** H\_sync**,**

**output** V\_sync**,**

// User ports ends

// Do not modify the ports beyond this line

// AXI4Stream sink: Clock

**input** **wire** S\_AXIS\_ACLK**,**

// AXI4Stream sink: Reset

**input** **wire** S\_AXIS\_ARESETN**,**

// Ready to accept data in

**output** **wire** S\_AXIS\_TREADY**,**

// Data in

**input** **wire** **[**31 **:** 0**]** S\_AXIS\_TDATA**,**

// Byte qualifier

**input** **wire** **[**3 **:** 0**]** S\_AXIS\_TSTRB**,**

// Indicates boundary of last packet

**input** **wire** S\_AXIS\_TLAST**,**

// Data is in valid

**input** **wire** S\_AXIS\_TVALID

**);**

// function called clogb2 that returns an integer which has the

// value of the ceiling of the log base 2.

**function** **integer** clogb2 **(input** **integer** bit\_depth**);**

**begin**

**for(**clogb2**=**0**;** bit\_depth**>**0**;** clogb2**=**clogb2**+**1**)**

bit\_depth **=** bit\_depth **>>** 1**;**

**end**

**endfunction**

// 640x480 @ 60 Hz

//Pixel clk 25M MHz

**reg** **[**1**:**0**]** R**;**

**reg** **[**1**:**0**]** G**;**

**reg** **[**1**:**0**]** B**;**

**reg** h\_sync\_reg**;**

**reg** v\_sync\_reg**;**

**reg** **[**9**:**0**]** h\_cntr**;**

**reg** **[**9**:**0**]** v\_cntr**;**

**reg** **[**1**:**0**]** ce**;**

**assign** H\_sync **=** h\_sync\_reg**;**

**assign** V\_sync **=** v\_sync\_reg**;**

**assign** R1 **=** R**[**1**];**

**assign** R0 **=** R**[**0**];**

**assign** G1 **=** G**[**1**];**

**assign** G0 **=** G**[**0**];**

**assign** B1 **=** B**[**1**];**

**assign** B0 **=** B**[**0**];**

**parameter** **[**1**:**0**]** waitForTlast **=** 2'd0**,** waitForData **=** 2'd1**,** inSync **=** 2'd2**;**

**reg** **[**1**:**0**]** streamSync**;**

**reg** aready**;**

**reg[**31**:**0**]** dataIn**;**

**assign** S\_AXIS\_TREADY **=** aready**;**

**always** **@(posedge** S\_AXIS\_ACLK**)**

**begin**

**if(!**S\_AXIS\_ARESETN **||** **(**streamSync **!=** inSync**))**

**begin**

h\_sync\_reg **<=** 0**;**

v\_sync\_reg **<=** 0**;**

**end**

**else** **if(**ce **==** 2'b11**)**

**begin**

h\_sync\_reg **<=** **!((**h\_cntr **>=** **(**640 **+** 16 **-**1**))** **&&** **(**h\_cntr **<** **(**640 **+** 16 **+** 96 **-**1**)));**

v\_sync\_reg **<=** **!((**v\_cntr **>=** **(**480 **+** 10 **-**1**))** **&&** **(**v\_cntr **<** **(**480 **+** 10 **+** 2 **-**1**)));**

**end**

**end**

**wire** h\_end **=** h\_cntr **>=** 10'd799**;**

**wire** v\_end **=** v\_cntr **>=** 10'd520**;**

**always** **@(posedge** S\_AXIS\_ACLK**)**

**begin**

**if(!**S\_AXIS\_ARESETN**)**

**begin**

streamSync **<=** waitForTlast**;**

**end**

**else**

**begin**

**if(**streamSync **==** waitForTlast**)**

**begin**

**if(**S\_AXIS\_TLAST**)**

**begin**

streamSync **<=** waitForData**;**

**end**

**end**

**else** **if(**streamSync **==** waitForData**)**

**begin**

**if(**S\_AXIS\_TVALID**)**

**begin**

streamSync **<=** inSync**;**

**end**

**end**

**end**

**end**

**wire** visibleArea **=** **!((**h\_cntr **>=** 639**)** **||** **(**v\_cntr **>=** 479**));**

**always** **@** **(posedge** S\_AXIS\_ACLK**)**

**begin**

**if(**streamSync **!=** inSync**)**

aready **<=** 1**;**

**else**

**if((**h\_cntr**[**1**:**0**]** **==** 2'b00**)** **&&** **(**ce **==** 2'b00**)** **&&** visibleArea**)**

**begin**

aready **<=** 1**;**

dataIn **<=** S\_AXIS\_TDATA**;**

**end**

**else**

aready **<=** 0**;**

**end**

**always** **@(posedge** S\_AXIS\_ACLK**)**

**begin**

**if(!**S\_AXIS\_ARESETN **||** **(**streamSync **!=** inSync**))**

**begin**

h\_cntr **<=** 10'd0**;**

v\_cntr **<=** 10'd0**;**

ce **<=** 2'b0**;**

**end**

**else**

**begin**

ce **<=** ce **+** 1**;**

**if** **(**ce **==** 2'b11**)**

**begin**

**if** **(**h\_end **&&** v\_end**)**

**begin**

h\_cntr **<=** 10'd0**;**

v\_cntr **<=** 10'd0**;**

**end**

**else** **if** **(**h\_end**)**

**begin**

h\_cntr **<=** 10'd0**;**

v\_cntr **<=** v\_cntr **+** 1**;**

**end**

**else**

**begin**

h\_cntr **<=** h\_cntr **+** 1**;**

**end**

**end**

**end**

**end**

**always** **@** **(posedge** S\_AXIS\_ACLK**)**

**begin**

**if(**ce **==** 2'b11**)**

**begin**

**if(!**visibleArea**)**

**begin**

R**[**1**:**0**]** **<=** 0**;**

G**[**1**:**0**]** **<=** 0**;**

B**[**1**:**0**]** **<=** 0**;**

**end**

**else**

**begin**

**case** **(**h\_cntr**[**1**:**0**])**

2'b00**:**

**begin**

R**[**1**:**0**]** **<=** dataIn**[**5**:**4**];**

G**[**1**:**0**]** **<=** dataIn**[**3**:**2**];**

B**[**1**:**0**]** **<=** dataIn**[**1**:**0**];**

**end**

2'b01**:**

**begin**

R**[**1**:**0**]** **<=** dataIn**[**13**:**12**];**

G**[**1**:**0**]** **<=** dataIn**[**11**:**10**];**

B**[**1**:**0**]** **<=** dataIn**[**9**:**8**];**

**end**

2'b10**:**

**begin**

R**[**1**:**0**]** **<=** dataIn**[**21**:**10**];**

G**[**1**:**0**]** **<=** dataIn**[**19**:**18**];**

B**[**1**:**0**]** **<=** dataIn**[**17**:**16**];**

**end**

2'b11**:**

**begin**

R**[**1**:**0**]** **<=** dataIn**[**29**:**28**];**

G**[**1**:**0**]** **<=** dataIn**[**27**:**26**];**

B**[**1**:**0**]** **<=** dataIn**[**25**:**24**];**

**end**

**endcase**

**end**

**end**

**end**

**endmodule**

## 7-szegmenses kijelző vezérlő kódja

???

## Snake.c

#include "Snake.h"

unsigned char picture1**[**480**][**640**]** \_\_attribute\_\_**((**aligned**(**128**),** section**(**".extmem"**)));**

unsigned char picture2**[**480**][**640**]** \_\_attribute\_\_**((**aligned**(**128**),** section**(**".extmem"**)));**

point\_typedef points**[**100**];**

int point\_cntr **=** 0**;**

fillData\_typedef removeTail**;**

fillData\_typedef addHead**;**

fillData\_typedef newFood**;**

int newFoodPlaced**;**

point\_typedef currentFood**;**

uint32\_t last\_tick **=** 0**;**

uint32\_t tick**;**

uint8\_t currentDmaPicture **=** 1**;**

int gameEnded **=** 0**;**

int clearStart **=** 0**;**

int Score**;**

int Level**;**

uint8\_t gameRunning **=** 0**;**

direction\_typedef currentDirection**;**

direction\_typedef lastPressedButton **=** NONE**;**

const uint8\_t SEG7**[**10**]** **=** **{**0x3F**,** 0x06**,** 0x5B**,** 0x4F**,** 0x66**,** 0x6D**,** 0x7D**,** 0x07**,** 0x7F**,** 0x6F**};**

void swap**(**int**\*** x**,** int **\***y**)**

**{**

int tmp **=** **\***x**;**

**\***x **=** **\***y**;**

**\***y **=** tmp**;**

**}**

void FillRectangle**(**fillData\_typedef**\*** in**,** uint8\_t**\*** picture**)**

**{**

**if(**in**->**x0 **>** in**->**x1**)**

swap**(&**in**->**x0**,** **&**in**->**x1**);**

**if(**in**->**y0 **>** in**->**y1**)**

swap**(&**in**->**y0**,** **&**in**->**y1**);**

**for(**int y **=** in**->**y0**;** y **<=** in**->**y1**;** y**++)**

**{**

**for(**int x **=** in**->**x0**;** x **<=** in**->**x1**;** x**++)**

**{**

picture**[**y **\*** 640 **+** x**]** **=** in**->**color**;**

**}**

**}**

**}**

int GetRandom**(**int min**,** int max**)**

**{**

**return** rand**()** **/** **(**RAND\_MAX **/** **(**max**-**min**))** **+** min**;**

**}**

int PointFitsLine**(**point\_typedef p**,** point\_typedef l0**,** point\_typedef l1**)**

**{**

int ret **=** 0**;**

**if((**l0**.**x **==** l1**.**x**)** **&&** **(**l0**.**x **==** p**.**x**))**

**{**

**if((**p**.**y **>=** l0**.**y**)** **&&** **(**p**.**y **<=** l1**.**y**))**

ret **=** **-**1**;**

**if((**p**.**y **>=** l1**.**y**)** **&&** **(**p**.**y **<=** l0**.**y**))**

ret **=** **-**1**;**

**}**

**else** **if((**l0**.**y **==** l1**.**y**)** **&&** **(**l0**.**y **==** p**.**y**))**

**{**

**if((**p**.**x **>=** l0**.**x**)** **&&** **(**p**.**x **<=** l1**.**x**))**

ret **=** **-**1**;**

**if((**p**.**x **>=** l1**.**x**)** **&&** **(**p**.**x **<=** l0**.**x**))**

ret **=** **-**1**;**

**}**

**return** ret**;**

**}**

void ShiftPoints**()**

**{**

**for(**int i **=** point\_cntr**;** i **>** 0**;** i**--)**

**{**

points**[**i**]** **=** points**[**i **-**1**];**

**}**

point\_cntr**++;**

**}**

void StepAhead**()**

**{**

**if((**currentDirection **==** UP**)** **||** **(**currentDirection **==** DOWN**))**

**{**

**if(**lastPressedButton **==** RIGHT**)**

**{**

ShiftPoints**();**

points**[**0**].**x **=** points**[**1**].**x **+** 10**;**

points**[**0**].**y **=** points**[**1**].**y**;**

currentDirection **=** RIGHT**;**

addHead**.**x0 **=** points**[**0**].**x **-** 5**;**

addHead**.**y0 **=** points**[**0**].**y **-** 4**;**

addHead**.**x1 **=** points**[**0**].**x **+** 4**;**

addHead**.**y1 **=** points**[**0**].**y **+** 4**;**

addHead**.**color **=** 0xFF**;**

**}**

**else** **if(**lastPressedButton **==** LEFT**)**

**{**

ShiftPoints**();**

points**[**0**].**x **=** points**[**1**].**x **-** 10**;**

points**[**0**].**y **=** points**[**1**].**y**;**

currentDirection **=** LEFT**;**

addHead**.**x0 **=** points**[**0**].**x **-** 4**;**

addHead**.**y0 **=** points**[**0**].**y **-** 4**;**

addHead**.**x1 **=** points**[**0**].**x **+** 5**;**

addHead**.**y1 **=** points**[**0**].**y **+** 4**;**

addHead**.**color **=** 0xFF**;**

**}**

**else** **if(**currentDirection **==** UP**)**

**{**

points**[**0**].**y **-=** 10**;**

addHead**.**x0 **=** points**[**0**].**x **-** 4**;**

addHead**.**y0 **=** points**[**0**].**y **-** 4**;**

addHead**.**x1 **=** points**[**0**].**x **+** 4**;**

addHead**.**y1 **=** points**[**0**].**y **+** 5**;**

addHead**.**color **=** 0xFF**;**

**}**

**else** **if(**currentDirection **==** DOWN**)**

**{**

points**[**0**].**y **+=** 10**;**

addHead**.**x0 **=** points**[**0**].**x **-** 4**;**

addHead**.**y0 **=** points**[**0**].**y **-** 5**;**

addHead**.**x1 **=** points**[**0**].**x **+** 4**;**

addHead**.**y1 **=** points**[**0**].**y **+** 4**;**

addHead**.**color **=** 0xFF**;**

**}**

**}**

**else** **if((**currentDirection **==** RIGHT**)** **||** **(**currentDirection **==** LEFT**))**

**{**

**if(**lastPressedButton **==** UP**)**

**{**

ShiftPoints**();**

points**[**0**].**x **=** points**[**1**].**x**;**

points**[**0**].**y **=** points**[**1**].**y **-** 10**;**

currentDirection **=** UP**;**

addHead**.**x0 **=** points**[**0**].**x **-** 4**;**

addHead**.**y0 **=** points**[**0**].**y **-** 4**;**

addHead**.**x1 **=** points**[**0**].**x **+** 4**;**

addHead**.**y1 **=** points**[**0**].**y **+** 5**;**

addHead**.**color **=** 0xFF**;**

**}**

**else** **if(**lastPressedButton **==** DOWN**)**

**{**

ShiftPoints**();**

points**[**0**].**x **=** points**[**1**].**x**;**

points**[**0**].**y **=** points**[**1**].**y **+** 10**;**

currentDirection **=** DOWN**;**

addHead**.**x0 **=** points**[**0**].**x **-** 4**;**

addHead**.**y0 **=** points**[**0**].**y **-** 5**;**

addHead**.**x1 **=** points**[**0**].**x **+** 4**;**

addHead**.**y1 **=** points**[**0**].**y **+** 4**;**

addHead**.**color **=** 0xFF**;**

**}**

**else** **if** **(**currentDirection **==** RIGHT**)**

**{**

points**[**0**].**x **+=** 10**;**

addHead**.**x0 **=** points**[**0**].**x **-** 5**;**

addHead**.**y0 **=** points**[**0**].**y **-** 4**;**

addHead**.**x1 **=** points**[**0**].**x **+** 4**;**

addHead**.**y1 **=** points**[**0**].**y **+** 4**;**

addHead**.**color **=** 0xFF**;**

**}**

**else** **if** **(**currentDirection **==** LEFT**)**

**{**

points**[**0**].**x **-=** 10**;**

addHead**.**x0 **=** points**[**0**].**x **-** 4**;**

addHead**.**y0 **=** points**[**0**].**y **-** 4**;**

addHead**.**x1 **=** points**[**0**].**x **+** 5**;**

addHead**.**y1 **=** points**[**0**].**y **+** 4**;**

addHead**.**color **=** 0xFF**;**

**}**

**}**

**}**

void RemoveTail**()**

**{**

removeTail**.**x0 **=** points**[**point\_cntr **-** 1**].**x **-** 5**;**

removeTail**.**y0 **=** points**[**point\_cntr **-** 1**].**y **-** 5**;**

removeTail**.**x1 **=** points**[**point\_cntr **-** 1**].**x **+** 5**;**

removeTail**.**y1 **=** points**[**point\_cntr **-** 1**].**y **+** 5**;**

removeTail**.**color **=** 0x00**;**

**if(**points**[**point\_cntr **-** 1**].**x **==** points**[**point\_cntr **-** 2**].**x**)**

**{**

int d **=** points**[**point\_cntr **-** 1**].**y **-** points**[**point\_cntr **-** 2**].**y**;**

**if(**my\_abs**(**d**)** **==** 10**)**

**{**

point\_cntr**--;**

**}**

**else**

**{**

points**[**point\_cntr **-** 1**].**y **+=** **(**d **>** 10**)** **?** **-**10 **:** 10**;**

**}**

**}**

**else**

**{**

int d **=** points**[**point\_cntr **-** 1**].**x **-** points**[**point\_cntr **-** 2**].**x**;**

**if(**my\_abs**(**d**)** **==** 10**)**

**{**

point\_cntr**--;**

**}**

**else**

**{**

points**[**point\_cntr **-** 1**].**x **+=** **(**d **>** 10**)** **?** **-**10 **:** 10**;**

**}**

**}**

**}**

int CheckWall**()**

**{**

**if(**points**[**0**].**x **<** 0 **||** points**[**0**].**y **<** 0 **||** points**[**0**].**x **>** 638 **||** points**[**0**].**y **>** 478**)**

**return** **-**1**;**

**return** 0**;**

**}**

int CheckBite**()**

**{**

int ret **=** 0**;**

**for(**int i **=** 1**;** i **<** point\_cntr **-** 1**;** i**++)**

**{**

**if(**PointFitsLine**(**points**[**0**],** points**[**i**],** points**[**i **+** 1**])** **!=** 0**)**

**{**

ret **=** **-**1**;**

**break;**

**}**

**}**

**return** ret**;**

**}**

void PlaceNewFood**()**

**{**

int ok **=** 0**;**

**while(!**ok**)**

**{**

currentFood**.**x **=** GetRandom**(**0**,** 20**)** **\*** 10 **+** 9**;**

currentFood**.**y **=** GetRandom**(**0**,** 10**)** **\*** 10 **+** 9**;**

//currentFood.x = GetRandom(0, 63) \* 10 + 9;

//currentFood.y = GetRandom(0, 47) \* 10 + 9;

ok **=** 1**;**

**for(**int i **=** 0**;** i **<** point\_cntr **-** 1**;** i**++)**

**{**

**if(**PointFitsLine**(**currentFood**,** points**[**i**],** points**[**i **+** 1**])** **!=** 0**)**

ok **=** 0**;**

**}**

**}**

newFood**.**x0 **=** currentFood**.**x **-**4**;**

newFood**.**y0 **=** currentFood**.**y **-**4**;**

newFood**.**x1 **=** currentFood**.**x **+**4**;**

newFood**.**y1 **=** currentFood**.**y **+**4**;**

newFood**.**color **=** 0x30**;**

newFoodPlaced **=** 1**;**

**}**

int FoundFood**()**

**{**

**return** **(**points**[**0**].**x **==** currentFood**.**x**)** **&&** **(**points**[**0**].**y **==** currentFood**.**y**);**

**}**

int StepSnake**()**

**{**

int ret **=** 0**;**

StepAhead**();**

**if(**FoundFood**())**

**{**

PlaceNewFood**();**

int tmp **=** Score **/** 10 **+** 1**;**

Level **=** tmp **<** 8 **?** tmp **:** 8**;**

**if(**Score **<** 9998**)**

Score**++;**

**}**

**else**

**{**

RemoveTail**();**

**}**

**if(**CheckWall**()** **||** CheckBite**())**

**{**

ret **=** **-**1**;**

**}**

**return** ret**;**

**}**

int ClearScreen**(**uint8\_t **\***pic**,** int first**)**

**{**

static int state**;**

static int line**;**

**if(**first**)**

**{**

state **=** 0**;**

line **=** 0**;**

**}**

**if(**state **==** 0**)**

**{**

int lineStart **=** line**;**

int j**;**

**for(**j **=** lineStart**;** **(**j **<** 480**)** **&&** **(**j **<** lineStart **+** 2**);** j**++)**

**{**

**for(**int i **=** 0**;** i **<** 640**;** i**++)**

**{**

**\*(**pic **+** j **\*** 640 **+** i**)** **=** **(**unsigned char**)(**0x00**);**

**}**

**}**

line **=** j**;**

**if(**j **==** 480**)**

state **=** 1**;**

**}**

**else** **if(**state **==** 1**)**

**{**

**for(**int i **=** 0**;** i **<** 639**;** i**++)**

**{**

**\*(**pic **+** 0 **\*** 640 **+** i**)** **=** **(**unsigned char**)(**0x03**);**

**\*(**pic **+** 1 **\*** 640 **+** i**)** **=** **(**unsigned char**)(**0x03**);**

**\*(**pic **+** 2 **\*** 640 **+** i**)** **=** **(**unsigned char**)(**0x03**);**

**\*(**pic **+** 3 **\*** 640 **+** i**)** **=** **(**unsigned char**)(**0x03**);**

**}**

state **=** 2**;**

**}**

**else** **if(**state **==** 2**)**

**{**

**for(**int i **=** 0**;** i **<** 639**;** i**++)**

**{**

**\*(**pic **+** 475 **\*** 640 **+** i**)** **=** **(**unsigned char**)(**0x03**);**

**\*(**pic **+** 476 **\*** 640 **+** i**)** **=** **(**unsigned char**)(**0x03**);**

**\*(**pic **+** 477 **\*** 640 **+** i**)** **=** **(**unsigned char**)(**0x03**);**

**\*(**pic **+** 478 **\*** 640 **+** i**)** **=** **(**unsigned char**)(**0x03**);**

**}**

state **=** 3**;**

**}**

**else** **if(**state **==** 3**)**

**{**

**for(**int i **=** 0**;** i **<** 479**;** i**++)**

**{**

**\*(**pic **+** i **\*** 640 **+** 0**)** **=** **(**unsigned char**)(**0x03**);**

**\*(**pic **+** i **\*** 640 **+** 1**)** **=** **(**unsigned char**)(**0x03**);**

**\*(**pic **+** i **\*** 640 **+** 2**)** **=** **(**unsigned char**)(**0x03**);**

**\*(**pic **+** i **\*** 640 **+** 3**)** **=** **(**unsigned char**)(**0x03**);**

**}**

state **=** 4**;**

**}**

**else** **if(**state **==** 4**)**

**{**

**for(**int i **=** 0**;** i **<** 479**;** i**++)**

**{**

**\*(**pic **+** i **\*** 640 **+** 635**)** **=** **(**unsigned char**)(**0x03**);**

**\*(**pic **+** i **\*** 640 **+** 636**)** **=** **(**unsigned char**)(**0x03**);**

**\*(**pic **+** i **\*** 640 **+** 637**)** **=** **(**unsigned char**)(**0x03**);**

**\*(**pic **+** i **\*** 640 **+** 638**)** **=** **(**unsigned char**)(**0x03**);**

**}**

state **=** 5**;**

**}**

**else** **if(**state **==** 5**)**

**{**

fillData\_typedef init**;**

addHead**.**x0 **=** init**.**x0 **=** 315**;**

addHead**.**y0 **=** init**.**y0 **=** 235**;**

addHead**.**x1 **=** init**.**x1 **=** 323**;**

addHead**.**y1 **=** init**.**y1 **=** 195**;**

addHead**.**color **=** init**.**color **=** 0xFF**;**

FillRectangle**(&**init**,** pic**);**

removeTail **=** addHead**;**

points**[**0**].**x **=** 319**;**

points**[**0**].**y **=** 199**;**

points**[**1**].**x **=** 319**;**

points**[**1**].**y **=** 239**;**

point\_cntr **=** 2**;**

currentDirection **=** UP**;**

FillRectangle**(&**newFood**,** pic**);**

newFoodPlaced **=** 0**;**

Score **=** 0**;**

Level **=** 1**;**

state **=** 6**;**

**}**

**return** **!(**state **==** 6**);**

**}**

int ClearManage**(**int **\***clearStart**)**

**{**

int ret **=** 0**;**

**if(\***clearStart **==** 1**)**

**{**

dma\_mm2s\_start**(**XPAR\_AXIDMA\_0\_BASEADDR**,** picture2**,** 640**\***480**);**

ClearScreen**((**uint8\_t**\*)**picture1**,** 1**);**

**\***clearStart **=** 2**;**

**}**

**else** **if(\***clearStart **==** 2**)**

**{**

dma\_mm2s\_start**(**XPAR\_AXIDMA\_0\_BASEADDR**,** picture2**,** 640**\***480**);**

**if(!**ClearScreen**((**uint8\_t**\*)**picture1**,** 0**))**

**{**

**\***clearStart **=** 3**;**

**}**

**}**

**else** **if(\***clearStart **==** 3**)**

**{**

dma\_mm2s\_start**(**XPAR\_AXIDMA\_0\_BASEADDR**,** picture1**,** 640**\***480**);**

ClearScreen**((**uint8\_t**\*)**picture2**,** 1**);**

**\***clearStart **=** 4**;**

**}**

**else** **if(\***clearStart **==** 4**)**

**{**

dma\_mm2s\_start**(**XPAR\_AXIDMA\_0\_BASEADDR**,** picture1**,** 640**\***480**);**

**if(!**ClearScreen**((**uint8\_t**\*)**picture2**,** 0**))**

**{**

ret **=** 1**;**

**}**

**}**

microblaze\_flush\_dcache**();**

**return** ret**;**

**}**

void ScanButtons**()**

**{**

uint8\_t c**;**

**if(**MEM32**(**XPAR\_UARTLITE\_0\_BASEADDR **+** 8**)** **&** 0x1**)**

**{**

c **=** MEM32**(**XPAR\_UARTLITE\_0\_BASEADDR**);**

**switch(**c**)**

**{**

**case** 'U'**:**

lastPressedButton **=** UP**;**

**break;**

**case** 'D'**:**

lastPressedButton **=** DOWN**;**

**break;**

**case** 'L'**:**

lastPressedButton **=** LEFT**;**

**break;**

**case** 'R'**:**

lastPressedButton **=** RIGHT**;**

**break;**

**}**

**}**

uint32\_t in **=** MEM32**(**XPAR\_AXI\_GPIO\_0\_BASEADDR **+** 8**);**

**if(**in **&** 0x01 **<<** 0**)**

lastPressedButton **=** UP**;**

**if(**in **&** 0x01 **<<** 1**)**

lastPressedButton **=** DOWN**;**

**if(**in **&** 0x01 **<<** 2**)**

lastPressedButton **=** RIGHT**;**

**if(**in **&** 0x01 **<<** 3**)**

lastPressedButton **=** LEFT**;**

gameRunning **=** **(**in **&** 0x01 **<<** 4**);**

**}**

void UpdateGame**()**

**{**

uint8\_t**\*** pic**;**

**if(**currentDmaPicture **==** 1**)**

**{**

dma\_mm2s\_start**(**XPAR\_AXIDMA\_0\_BASEADDR**,** picture2**,** 640**\***480**);**

currentDmaPicture **=** 2**;**

pic **=** picture1**;**

**}**

**else**

**{**

dma\_mm2s\_start**(**XPAR\_AXIDMA\_0\_BASEADDR**,** picture1**,** 640**\***480**);**

currentDmaPicture **=** 1**;**

pic **=** picture2**;**

**}**

FillRectangle**(&**removeTail**,** pic**);**

FillRectangle**(&**addHead**,** pic**);**

FillRectangle**(&**newFood**,** pic**);**

tick **=** GetTick**();**

uint32\_t tmp **=** tick **-** last\_tick**;**

int speed **=** **(**100000 **\*** **(**SPEED **-** Level **\*** 40**));**

**if(**tmp **>** speed**)**

**{**

last\_tick **+=** speed**;**

**if(**gameRunning**)**

**{**

gameEnded **=** StepSnake**();**

uint32\_t segData **=** 0**;**

int s **=** Score**;**

**for(**int i **=** 0**;** i **<** 4**;** i**++)**

**{**

segData **|=** **(**SEG7**[**s **%** 10**]** **<<** **(**8 **\*** i**));**

s **/=** 10**;**

**}**

MEM32**(**XPAR\_GPIO\_0\_BASEADDR**)** **=** 0x01 **<<** **(**Level **-** 1**);**

MEM32**(**XPAR\_SEVENSEGMENTDRIVER\_0\_S00\_AXI\_BASEADDR**)** **=** segData**;**

**if(!**gameEnded**)**

**{**

FillRectangle**(&**removeTail**,** pic**);**

FillRectangle**(&**addHead**,** pic**);**

**if(**newFoodPlaced**)**

FillRectangle**(&**newFood**,** pic**);**

microblaze\_flush\_dcache**();**

**}**

**else**

**{**

clearStart **=** 1**;**

**}**

**}**

lastPressedButton **=** NONE**;**

**}**

**}**

void SnakeMain**()**

**{**

PlaceNewFood**();**

ClearScreen**((**uint8\_t**\*)**picture1**,** 1**);**

**while(**ClearScreen**((**uint8\_t**\*)**picture1**,** 0**))**

**{}**

ClearScreen**((**uint8\_t**\*)**picture2**,** 1**);**

**while(**ClearScreen**((**uint8\_t**\*)**picture2**,** 0**))**

**{}**

dma\_mm2s\_start**(**XPAR\_AXIDMA\_0\_BASEADDR**,** picture1**,** 640**\***480**);**

**while(**1**)**

**{**

**if(**dma\_mm2s\_finished**(**XPAR\_AXIDMA\_0\_BASEADDR**))**

**{**

**if(**gameEnded**)**

**{**

**if(**ClearManage**(&**clearStart**))**

**{**

gameEnded **=** 0**;**

currentDmaPicture **=** 1**;**

last\_tick **=** GetTick**();**

**}**

**}**

**else**

**{**

UpdateGame**();**

**}**

**}**

ScanButtons**();**

**}**

**}**

## Snake.h

#ifndef SRC\_SNAKE\_H\_

#define SRC\_SNAKE\_H\_

#include <inttypes.h>

#include "Drivers.h"

#include <stdlib.h>

**typedef** struct

**{**

int x**;**

int y**;**

**}** point\_typedef**;**

**typedef** enum

**{**

UP **=** 0**,**

DOWN **=** 1**,**

LEFT **=** 2**,**

RIGHT **=** 3**,**

NONE **=** 4

**}**direction\_typedef**;**

**typedef** struct

**{**

int x0**;**

int y0**;**

int x1**;**

int y1**;**

uint8\_t color**;**

**}**fillData\_typedef**;**

#define SPEED 325 // ms/10pixel

#endif /\* SRC\_SNAKE\_H\_ \*/

## Drivers.c

#include "Drivers.h"

void dma\_init**(**unsigned long baseaddr**)**

**{**

//Az MM2S csatorna enged鬹ez鳥: a vez鲬g. RS bitj鮥k 1-be ᬬsa.

//Megszaksokat nem hasznᬵnk.

MEM32**(**baseaddr **+** 0x00**)** **=** **(**1 **<<** 0**);**

//AZ S2MM csatorna enged鬹ez鳥: a vez鲬g. RS bitj鮥k 1-be ᬬsa.

//Megszaksokat nem hasznᬵnk.

//MEM32(baseaddr + 0x30) = (1 << 0);

**}**

void dma\_mm2s\_start**(**unsigned long baseaddr**,** void **\***src**,** unsigned long length**)**

**{**

//A forr᳣beᬬsa. A felsbit mindig 0.

MEM32**(**baseaddr **+** 0x18**)** **=** **(**unsigned long**)**src**;**

MEM32**(**baseaddr **+** 0x1c**)** **=** 0**;**

//Az adatm鲥t beᬬsa, ennek hat᳡ra indul az MM2S DMA ᴶitel.

MEM32**(**baseaddr **+** 0x28**)** **=** length**;**

**}**

uint32\_t dma\_mm2s\_finished**(**unsigned long baseaddr**)**

**{**

unsigned long status**;**

status **=** MEM32**(**baseaddr **+** 0x04**);**

**return** status **&** **(**1 **<<** 1**);**

**}**

void timer1\_Init**()**

**{**

MEM32**(**XPAR\_AXI\_TIMER\_0\_BASEADDR **+** 4**)** **=** 0**;**

MEM32**(**XPAR\_AXI\_TIMER\_0\_BASEADDR**)** **=** 0x90**;**

**return;**

**}**

uint32\_t GetTick**()**

**{**

**return** MEM32**(**XPAR\_AXI\_TIMER\_0\_BASEADDR **+** 8**);**

**}**

## Drivers.h

#ifndef SRC\_DRIVERS\_H\_

#define SRC\_DRIVERS\_H\_

#include <inttypes.h>

#include <xparameters.h>

#define MEM32(addr) (\*(volatile unsigned long \*)(addr))

#define my\_abs(x) (x < 0 ? -x : x)

void dma\_init**(**unsigned long baseaddr**);**

void dma\_mm2s\_start**(**unsigned long baseaddr**,** void **\***src**,** unsigned long length**);**

uint32\_t dma\_mm2s\_finished**(**unsigned long baseaddr**);**

void timer1\_Init**();**

uint32\_t GetTick**();**

#endif /\* SRC\_DRIVERS\_H\_ \*/

## main.c

#include <xparameters.h>

#include <mb\_interface.h>

#include <stdio.h>

#include <stdlib.h>

#include "Snake.h"

int main**()**

**{**

srand**(**0**);**

//Az AXI DMA vez鲬icializᬡsa.

dma\_init**(**XPAR\_AXIDMA\_0\_BASEADDR**);**

//Timer init 10ns

timer1\_Init**();**

SnakeMain**();**

**return** 0**;**

**}**

## Billentyűzet emulátor

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**using** System**.**Windows**.**Forms**;**

**using** System**.**IO**.**Ports**;**

**namespace** Keys2UART

**{**

**public** partial class Form1 **:** Form

**{**

**public** Form1**()**

**{**

InitializeComponent**();**

**}**

**private**

SerialPort sPort **=** **new** SerialPort**();**

**private** void Form1\_FormClosing**(object** sender**,** FormClosingEventArgs e**)**

**{**

**}**

**private** void Form1\_Load**(object** sender**,** EventArgs e**)**

**{**

**this.**KeyPreview **=** **true;**

CBPorts**.**Items**.**AddRange**(**SerialPort**.**GetPortNames**());**

CBBaud**.**Items**.**AddRange**(new** string**[]** **{** "115200" **});**

CBParity**.**Items**.**AddRange**(new** string**[]{**"Even"**,** "Odd"**});**

CBPorts**.**SelectedIndex **=** 0**;**

CBBaud**.**SelectedIndex **=** 0**;**

CBParity**.**SelectedIndex **=** 0**;**

**}**

**private** void BRefresh\_Click**(object** sender**,** EventArgs e**)**

**{**

CBPorts**.**Items**.**Clear**();**

CBPorts**.**Items**.**AddRange**(**SerialPort**.**GetPortNames**());**

**}**

**private** void BConnect\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(!**sPort**.**IsOpen**)**

**{**

sPort**.**PortName **=** CBPorts**.**SelectedItem**.**ToString**();**

sPort**.**BaudRate **=** Convert**.**ToInt32**(**CBBaud**.**SelectedItem**);**

sPort**.**Parity **=** Parity**.**None**;**

**try**

**{**

sPort**.**Open**();**

BConnect**.**BackColor **=** Color**.**Green**;**

CBPorts**.**Enabled **=** **false;**

CBBaud**.**Enabled **=** **false;**

CBParity**.**Enabled **=** **false;**

TBLog**.**Focus**();**

**}**

**catch** **(**Exception ex**)**

**{**

MessageBox**.**Show**(**ex**.**Message**);**

**}**

**}**

**}**

**private** void BDisconnect\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**sPort**.**IsOpen**)**

**{**

**try**

**{**

sPort**.**Close**();**

TBLog**.**Clear**();**

BConnect**.**BackColor **=** SystemColors**.**Control**;**

**}**

**catch** **(**Exception ex**)**

**{**

MessageBox**.**Show**(**ex**.**Message**);**

**}**

**}**

**}**

**private** void Form1\_KeyDown**(object** sender**,** KeyEventArgs e**)**

**{**

**try**

**{**

**switch** **(**e**.**KeyCode**)**

**{**

**case** Keys**.**Up**:**

System**.**Diagnostics**.**Debug**.**Write**(**"Up\n"**);**

//TBLog.Text.Insert(0, "Up");

TBLog**.**Text **+=** "Up\n"**;**

//sPort.Write("U");

**break;**

**case** Keys**.**Down**:**

System**.**Diagnostics**.**Debug**.**Write**(**"Down\n"**);**

TBLog**.**Text **+=** "Down\n"**;**

//sPort.Write("D");

**break;**

**case** Keys**.**Right**:**

System**.**Diagnostics**.**Debug**.**Write**(**"Right\n"**);**

TBLog**.**Text **+=** "Right\n"**;**

//sPort.Write("R");

**break;**

**case** Keys**.**Left**:**

System**.**Diagnostics**.**Debug**.**Write**(**"Left\n"**);**

TBLog**.**Text **+=** "Left\n"**;**

//sPort.Write("L");

**break;**

**case** Keys**.**S**:**

//sPort.Write("S");

**break;**

**}**

**}**

**catch** **(**Exception ex**)**

**{**

MessageBox**.**Show**(**ex**.**Message**);**

**}**

**}**

**private** void BClear\_Click**(object** sender**,** EventArgs e**)**

**{**

TBLog**.**Clear**();**

TBLog**.**Focus**();**

**}**

**}**

**}**