Vaja 4
Zunanje prekinitve in “data abort” prekinitev

1.

Napiši program, ki bo štel število impulzov dveh pravokotnih zunanjih signalov. Trenutno stanje števca naj se prikazuje na LCD prikazovalniku. Za signale uporabite kar tipke, ki so že povezane na pine, kjer se lahko prožijo zunaje prekinitve.

1. Zunanji signal = tipka T2 => stevec +1

2. Zunanji signal = tipka T3 => stevec - 1

2.

Napiši program, ki bo ugotovil velikost pomnilnikov FLASH in RAM. Uporabite prekinitev “data abort”, ki se sproži ko dostopamo do podatkov na neveljavnih pomnilniških naslovih

Tipke naj prožijo zunanje prekinitve (glej extint.c, extint.h)

Pini vzporednih vrat port0, kjer lahko prožimo zunanje prekinitve:
(izbira funkcij pinov v PINSEL0,PINSEL1,PINSEL2):

Pin		tip prekinitve	maska za PINSEL reg.	
P0.1	=>	EINT0	P0_1_EINT0	
P0.3	=>	EINT1	P0_3_EINT1	
P0.7	=>	EINT2	P0_7_EINT2	
P0.9	=>	EINT3	P0_9_EINT3	PINSEL0
P0.14	=>	EINT1	P0_14_EINT1	
P0.15	=>	EINT2	P0_15_EINT2	
P0.16	=>	EINT0	P0_16_EINT0	
P0.20	=>	EINT3	P0_20_EINT3	PINSEL1
P0.30	=>	EINT3	P0_30_EINT3	

Pazite na pin p0.9 (stari ŠARM-i p0.1) => prekinitve lahko proži le prilagodilnik nivojev MAX232

Tipke, LED, LCD => pazi na te izhodne pine (ŠARM shema)

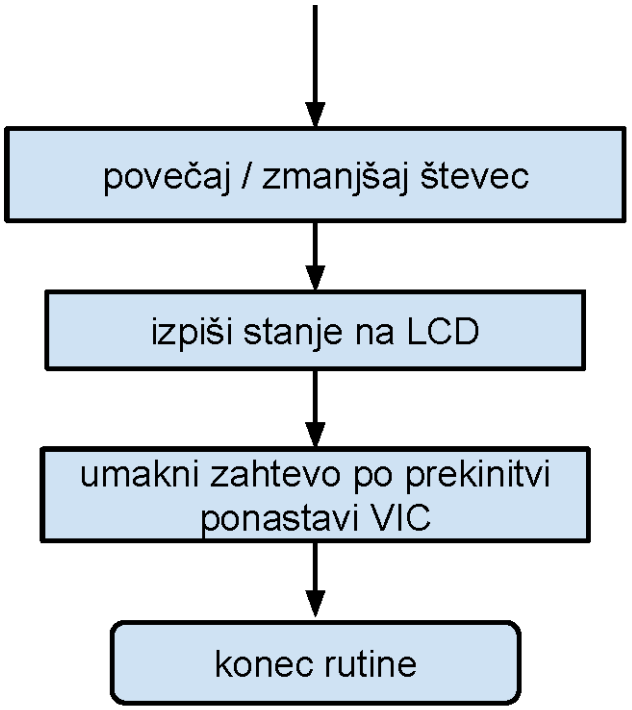
Inicializacija pinov za zunanje prekinitve

void extint_init(int pins, int wake, int sensitivity, int trig);

pins => maska za izbiro funkcije pinov 0-15 (kombinacija zastavic za pine)
(P0_1_EINT0 | P0_3_EINT1…)
wake => maska za omogočanje dviga jedra iz ustavljenega stanja (wake up) z zunanjo prekinitvijo
(EXTWAKE1 | EXTWAKE2 …)
sensitivity => maska za proženje s fronto (privzeto je proženje s stanjem)
(EXTMODE1 | EXTMODE2…)
trig => maska za proženje z visokim stanjem / naraščajočo fronto (privzeto je proženje z nizkim stanjem / padajočo fronto)
(EXTPOLAR1 | EXTPOLAR2…)

T2 in T3 naj prožita vsaka svojo prekinitev, ki nato kličeta funkciji tipkaA(), in tipkaB()
=> nastaviti je potrebno VIC, da bo pravilno prožil zunanje prekinitve

Prekinitvene rutine (ISR) za zunanje prekinitve



T2 in T3 prožijo zunanje prekinitve, ki nato kličejo vsaka svojo prekinitveno rutino:
void tipkaA(){…} : poveča števec
void tipkaB(){…} : zmanjša števec

Ko obdelamo prekinitev, je potrebno umakniti zahtevo po njej => pripravimo VIC na novo prekinitev:

```
EXTINT=EINTi; /* i=0,1,2,3 */
VICVectAddr=0;
```

Prekinitev *data abort*

zanjo ne skrbi VIC, ampak se avtomatsko proži, ko beremo ali pišemo na neveljavno lokacijo v pomnilniku

interrupts.c:
Ob *data abort prekinitvi se kliče funk. V tej funkciji lahko postavite globalno spremenljivko, ki bo signal funkciji main(), da smo prebrali do konca RAM-a (FLASH-a).*

```
void data_abt(void){
…
}
```

Izmerite velikost pomnilnika FLASH in RAM:

RAM	**zač. naslov**	**0x40000000**
FLASH	**zač. naslov**	**0x00000000**

Program naj bere iz pomnilnika toliko časa, da se sproži prekinitev
$\Rightarrow$ Offset pove koliko byte-ov smo uspešno prebrali => torej koliko byte-ov je na voljo v pomnilnikih FLASH in RAM