Laborator 4

Utilizarea instrumentelor interne de tip ceas și a afişajelor pe 7 segmente

Excepții de sistem și întreruperi

Secțiuni lucrare:

Excepții de sistem și întreruperi

Ce sunt excepțiile ?

Excepțiile sunt evenimente care sunt generate într-un mod asincron, care provin fie din exteriorul sistemului sau din interiorul acestuia. Când acestea sunt generate, procesorul schimbă modul normal de funcționare pentru a se putea ocupa de excepția generată. Cu alte cuvinte, orice entitate care perturbă modul normal de funcționare a procesorului este o ***excepție***.

Ce sunt întreruperile?

O întrerupere este o excepție. O întrerupere se referă la o excepție provenită din exteriorul sistemului de exemplu (RTC, I2C, Pini de intrare ieșire, USART, etc). Prin exteriorul sistemului se face referire la toate sistemele care nu sunt incluse în nucleul procesorului. Procesoarele Cortex M3/M4 permit un număr maxim de 240 de întreruperi.

Tipuri de excepții

Există două tipuri de excepții:

1. Excepții de sistem (interne procesorului)
2. Excepții externele (întreruperi)

Există 15 excepții de sistem numerotate de la 1 la 15 și 240 de întreruperi numerotate de la 16 la 255. De exemplu, a 16-a excepție nu este altceva decât întreruperea #0. În tabelul următor se pot vedea toate excepțiile de sistem (după cum se poate vedea, excepțile încep de la 1):

No.	Exception type	Priority	Description
0	*Stack*	*N/A*	*Initial main stack pointer*
1	Reset	-3 (fixed)	Reset vector
2	NMI	-2 (fixed)	Non mask-able interrupt
3	Hard fault	-1 (fixed)	Hard fault
4	Mem manage fault	Settable	MPU violations
5	Bus fault	Settable	Bus error
6	Usage fault	Settable	Program errors
7-10	Reserved	N/A	Reserved
11	SVC	Settable	Supervisor call
12	Debug Monitor	Settable	Debug requests
13	Reserved	N/A	Reserved
14	PendSV	Settable	Pendable service call
15	SysTick	Settable	System Tick timer
16	ExtInt0	Settable	External interrupt #0
17	ExtInt1	Settable	External interrupt #1
…	…	…	…
256	Interupt240	Settable	Interrupt #240

Tabel 1. Tabelul cu excepțiile implementate pe familia de microcontrolere STM32F1xx

Primele trei excepții de sistem au prioritatea fixă. Aceasta nu se poate modifica. **Un număr mai mic reprezintă o prioritate mai mare**. Toate ***întreruperile*** au prioritatea programabilă. Cu alte cuvinte, programatorul poate să seteze prioritatea în funcție de necesități. Excepția de sistem 1 și excepția de sistem 2, nu pot fi mascate. Cu alte cuvinte, dacă procesorul execută excepția numărul 1, RESET, nu există o altă întrerupere care să deturneze procesorul de la executarea excepției

RESET. Același lucru este valabil și pentru excepția numărul 2. Toate celalate excepții și întreruperi pot fi mascate și este datoria programatorului să seteze ordinea și prioritatea.

SysTick Timer

Toate procesoarele ARM Cortex M3 si M4 au un timer de sistem pe 24 de biți numit sysTick. Numărătorul din componența timer-ului sysTick este un numărator decremental pe 24 de biți. Când în numărător este încărcată o valoare, el începe să decrementeze valoarea respectivă la fiecare semnal de ceas pe care îl primește procesorul. În momentul în care valoarea din numărător ajunge la 0, se declanșază excepția **systick timer**, se reîncarcă o nouă valoare în numărător și acesta începe să decrementeze valoarea la fiecare semnal de ceas. Timerul SysTick poate să fie folosit pentru a măsura timpul, pentru delay sau ca sursă de întrerupere pentru anumite sarcini ce trebuie executate la o anumită perioadă de timp. Excepția apelează funcția numită ***SysTick_Handler***. Configurarea timer-ului se realizează prin apelarea unor funcții specifice procesor. Pentru configurarea sysTick Timer se va apela funcția **SysTick_Config(*ticks*)**, funcție care face parte din pachetul CMSIS. Această funcție primește ca parametru numărul de cicli de ceas după care se va executa excepția şi se va apela funcția SysTick_Handler. De exemplu, SysTick_Config(2000), va astepta 2000 de cicli de ceas pana când va declanșa excepția. În funcție de frecvența procesorului se poate calcula de câte ori pe secundă se va declanșa excepția de SysTick.

NVIC (Nested Vector Interupt Controler)

NVIC nu este altceva decât un bloc hardware care primește informații de la mai multe surse. NVIC este responsabil cu managementul excepțiilor ridicate de diverse surse și trimiterea acestora către procesor în funcție de prioritate. Procesoarele Cortex M au o serie de regiștrii programabili care se ocupă cu management-ul întreruperilor. O mare parte din acești registri se află în interiorul sistemului NVIC. Sunt două moduri prin care se pot administra şi configura regiștrii:

- Acces direct folosind regiștrii NVIC – Accesarea acestora se poate face doar în modul privilegiat.

- Prin folosirea codului pus la dispozitie de CMSIS precum *NVIC_SetPriority(IRQn, uint32_t priority) sau NVIC_EnableIRQ(IRQn_Type IRQn)*

După resetarea microcontrolerului toate întreruperile sunt dezactivate și au prioritatea 0. Înaintea de a folosi orice întrerupere trebuie:

1. Să se seteze nivelul de prioritate pentru întrerupere (opțional). Este obligatoriu acest pas în momentul în care avem mai multe întreruperi care se pot declanșa în același timp și una dintre ele este mai importantă față de cealaltă.
2. Activarea întreruperii în regiștrii NVIC.

Când o întrerupere se activează, ISR-ul corespunzător (Interupt Service Routine) va fi apelat în mod automat.

Prioritatea întreruperilor

În momentul în care apare o întrerupere, aceasta este executată de către procesor în funcție de prioritatea acesteia. O întrerupere cu prioritate mare poate să oprească execuția unei întreruperi cu prioritate mai mică. Această proprietate se numește *nesting of interrupts* sau "grupare de întreruperi". Există o serie de excepții precum (Reset, NMI și HardFault) care au o prioritate fixă. Aceste trei excepții au o prioritate negativă. Cu cât numărul asociat cu prioritatea este mai mic, cu atât prioritatea este mai mare.
Diferite tipuri de microcontrolere au număr diferit de prioritate (***Pre-Empt Priority***). De exemplu există microcontrolere cu 8 nivele de prioritate, cu 16 nivele de prioritate etc. Producătorii de microcontrolere sunt liberi să aleagă câte nivele de prioritate doresc să implementeze. În arhitectura ARM există un registru numit *Priority Level Register* care determină câte nivele de prioritate sunt implementate pe microcontrolerul respectiv. Dacă producătorii de microcontrolere decid să implementeze multe nivele de propritate, acest lucru va duce la creșterea complexității sistemului NVIC și de asemenea va duce la un consum mai mare de energie.

Fiecare excepție/întrerupere are propriul său registru de prioritate (Interupt Priority Level Register). Producătorii de cipuri sunt obligați ca pentru procesoarele ARM Cortex M3/M4 să folosească un minim de 8 nivele de prioritate (trei biți). Pentru procesoarele ARM Cortex M0 doar doi biți sunt implementați adică maxim 4 nivele de prioritate. După cum se poate vedea și în Figura 1, cei mai semnificativi biți sunt implementați. După cum s-a menționat și mai sus, există două moduri prin care se poate seta prioritatea. Un prin mod ar fi prin accesarea regiștrilor NVIC iar cel de al doilea mod este dat de CMSIS. Pentru a păstra portabilitatea codului este recomandat folosirea CMSIS. CMSIS dispune de funcția **NVIC_SetPriority(IRQn,Priority)**. În acest caz este foarte important ca prioritatea sa NU fie shiftată cu numărul de biți neimplmentați, deoarece funcția face acest calcul înainte să salveze datele în registru.

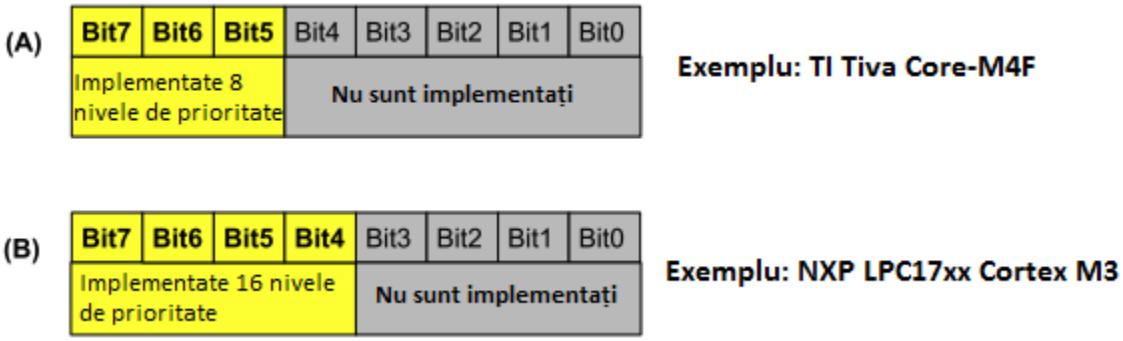


Figura 1. Exemplul de biți implementați în funcție de producător

Gruparea priorităților

Biții implementați pentru a seta nivelele de prioritate, pot fi împărțiti în biți de prioritate şi biți de sub-prioritate. Cu alte cuvinte, pentru un microcontroler cu 3 biţi implementati, se pot seta ca primii doi biți să fie biți de prioritate, iar ultimul bit să fie bit de sub prioritate. Acest mod este necesar în momentul în care dorim ca o întrerupere să nu fie oprită pentru a executa o altă întrerupere.

Prioritate Pre-Empt – Când procesorul execută un heandler al unei întreruperi şi o altă întrerupere apare, vor fi comparate valorile Pre-Empt a celor două întreruperi și va fi executată cea cu prioritatea ce mai mare (cel mai mic număr).

Sub Prioritate – Această valoare va fi folosită doar când două excepții/întreruperi cu aceeași valoare Pre-Empt apar în același timp. În acest caz, excepția/întreruperea cu cea mai mare sub-prioritate va fi procesată prima.

Prioritatea poate să fie setată din registrul **Application Interrupt And Reset Control Register**. Prigroup este format din 3 biți, prin urmare pot exista un număr maxim de 8 priorități Pre-Empt. **Dacă acest registru nu este modificat de programator, el are valoarea zero.** Cand PRIOGRUP are valoarea 0, nu există sub priorități. Diferența între întreruperi se va realiza la nivel de Pre-Empt.

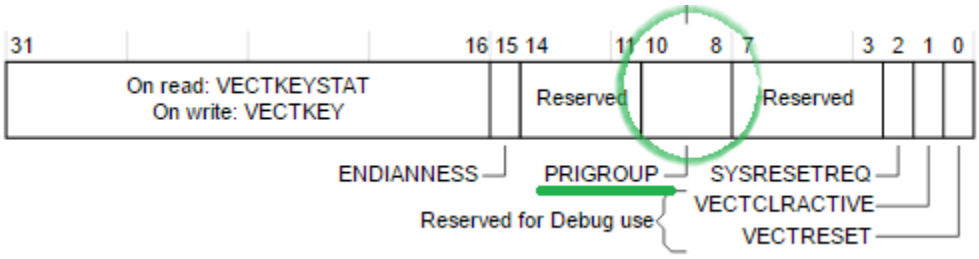


Figura 2. Modificarea biților corespunzători PRIOGRUP duce la setarea nivelelor de prioritate și sub-prioritate

Folosirea pinilor GPIO pentru generare de întreruperi

Fiecare STM32F1xx are o serie de întreruperi externe, aproximativ 20. Acestea sunt împărțite în 2 categorii. Prima categorie o reprezintă pinii P0-P15, pentru fiecare port, cea de a doua categorie reprezintă evenimente precum: întreruperi provenite de la RTC, întreruperi provenite de la Ethernet, întreruperi provenite de la USB etc.

Liniile de întrerupere

Pentru toate porturile GPIO sunt alocate 16 linii de la linia0 la linia15, care reprezintă și numărul pinului. De exemplu pinul PA0 este conectat la linia0 iar PA5 este conectat la linia5. De asemenea, PB0 este conectat la linia0, la fel și PC0 și așa mai departe. Cu alte cuvinte toți pinii 0 (Px0) indiferent de port sunt conectați la linia0. ***Toți pinii cu același număr sunt conectați la linia cu același număr. Toți pinii sunt multiplexați pe linia respectivă. NU SE POT FOLOSI DOI PINI DE PE ACEEAȘI LINIE SIMULTAN. Exemplu: PA0, PB0 și PC0 sunt conectați pe aceeași linie, line0. Prin urmare, se poate folosi doar un singur pin pentru generarea întreruperii. PA0 și PA5 sunt conectați la linii diferite, prin urmare pot să genereze două întreruperi diferite.***

Toate liniile pot să genereze întreruperi atât pe frontul crescător al semnalului cât și pe frontul descrescător.

Interupt Handlers

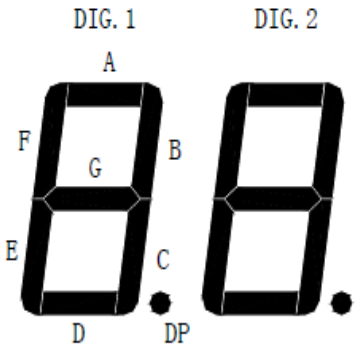
După ce s-a ales pinul care va denclașa întreruperea, trebuie să se îndrepte atenția către handle-uri. STM32F103 are 7 linii de întrerupere pentru pinii de GPIO. Acestea se pot vedea în tabelul de mai jos:

Irq (Interrupt Request)	Handler	Descriere
EXTI0_IRQn	EXTI0_IRQHandler	Handler pentru pinii conectați pe linia 0
EXTI1_IRQn	EXTI1_IRQHandler	Handler pentru pinii conectați pe linia 1
EXTI2_IRQn	EXTI2_IRQHandler	Handler pentru pinii conectați pe linia 2
EXTI3_IRQn	EXTI3_IRQHandler	Handler pentru pinii conectați pe linia 3
EXTI4_IRQn	EXTI4_IRQHandler	Handler pentru pinii conectați pe linia 4
EXTI9_5_IRQn	EXTI9_5_IRQHandler	Handler pentru pinii conectați pe liniile de la 5 la 9
EXTI15_10_IRQn	EXTI15_10_IRQHandler	Handler pentru pinii conectați pe liniile de la 10 la 15

Tabel 2. IRQ indică ce trebuie setat în NVIC pentru a configura întreruperea,iar coloanal Handler indică numele handler-ului ce urmează să fie executat când se declanșază întreruperea

Conectarea unui afișaj cu 2 caractere și 7 segmente

Caracterele pe 7 segmente funcţionează similar cu un număr de leduri egal cu numărul de segmente. În cazul de faţă, deoarece avem câte 7 segmente şi un punct, funcţionarea ar fi echivalentă cu cea a unui circuit de 8 leduri pentru fiecare caracter. Există două posibilităţi de realizare a unui astfel de afișaj: **anod comun sau catod comun**. În cadrul exemplelor următoare vom utiliza un circuit cu ***anod comun***. Chiar dacă la nivel de caracter liniile anod sau catod sunt comune, utilizarea unui astfel de afișaj este dificilă datorită numărului mare de linii de comandă (egal cu numărul total de segmente). Din acest motiv se va utiliza un circuit de tip shift register (TPIC6B595) pentru a scădea numărul liniilor de comandă necesare în lucrul cu acest afișaj. Dacă se dorește utilizarea unui afișaj de tip catod comun se va utiliza un shift register de tipul 74HC595. Nu se va lua în discuție conectarea directă a unui astfel de afișaj cu placa Nucleo datorită numărului mare de linii de comandă necesare (8+8=16).



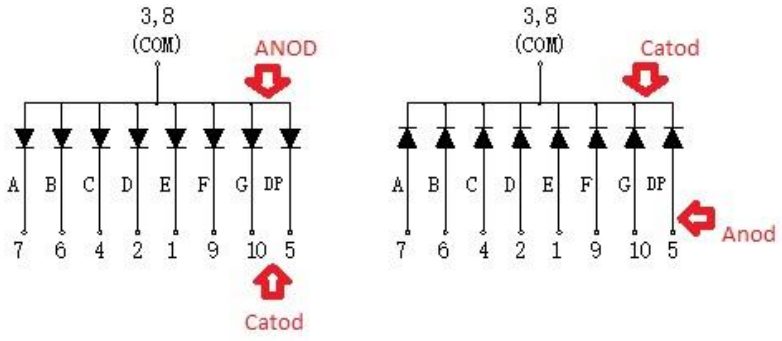


Fig 3. a) Afișaj anod comun b) Afisaj catod comun

Afisajele folosite în laborator sunt anod comun și au 18 pini. Doi din acesti 18 pini sunt pinii de "alimentare" pentru cele două numere, iar restul de 16 pini, sunt catod-urile led-urilor. Pinii sunt legați la led-uri după cum urmează :

1.Digit 1 catod E	2.Digit 1 catod D	3 – Digit 1 catod C	4 – Digit 1 catod DP	5 – Digit 2 catod E	6 – Digit 2 catod D
7 – Digit 2 catod G	8 – Digit 2 catod C	9 – Digit 2 catod DP	10 – Digit 2 catod B	11 – Digit 2 catod A	12 – Digit 2 catod F
13 – Digit 2 anod comun	14 – Digit 1 anod comun	15 – Digit 1 catod B	16 – Digit 1 catod A	17 – Digit 1 catod G	18 – Digit 1 catod F

Tabel 3. Conectarea pinilor la afisajul cu două numere

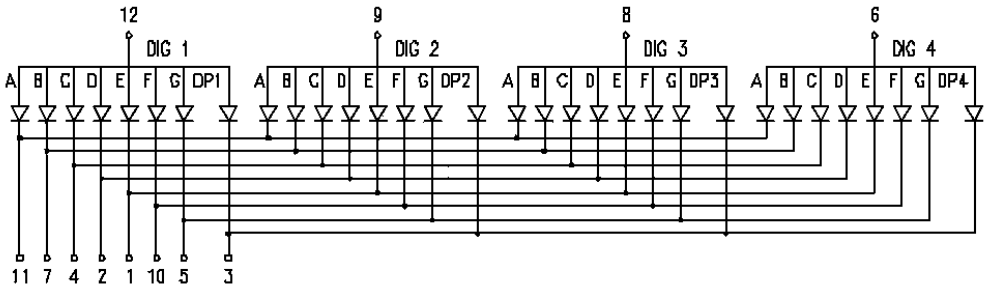
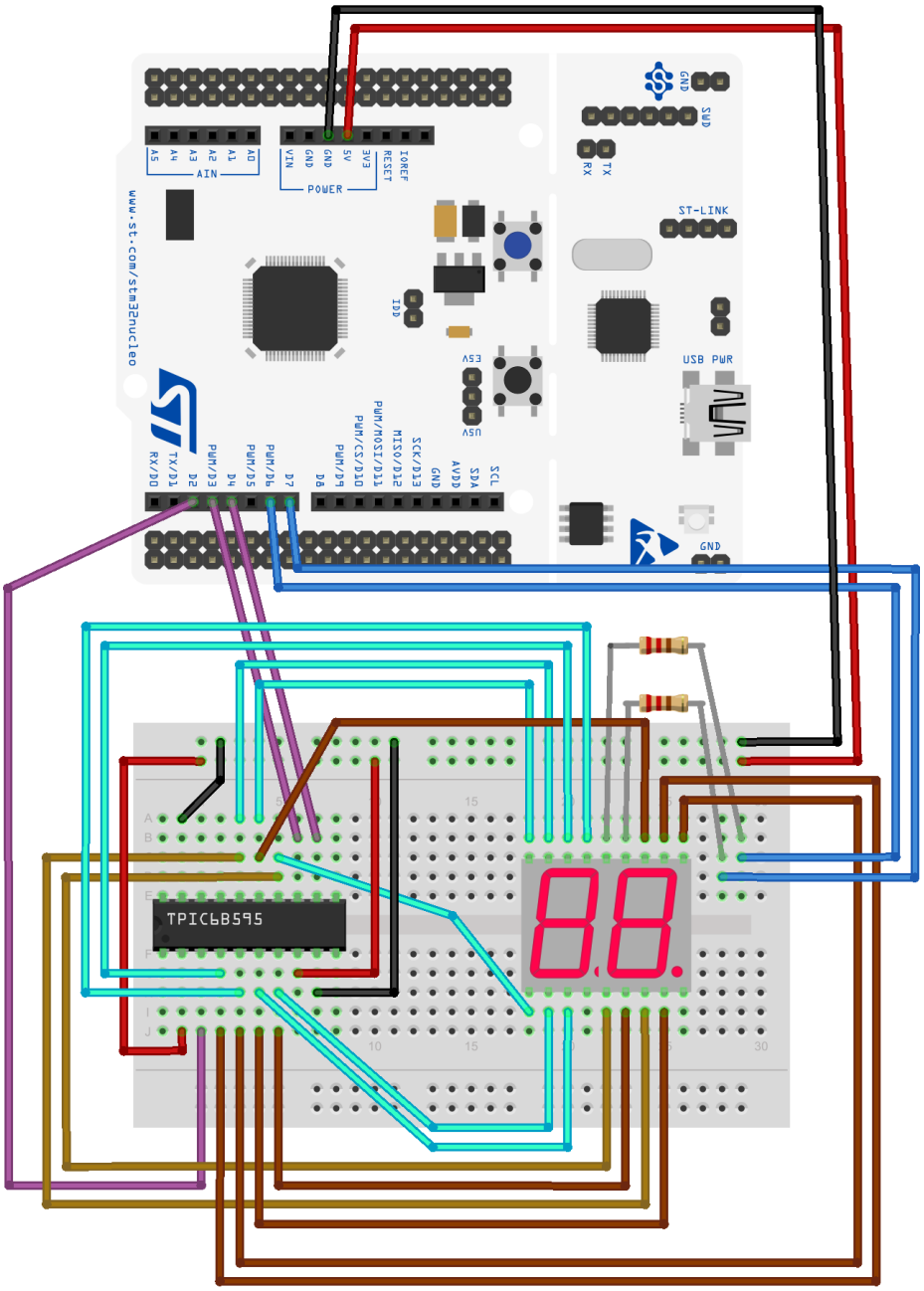
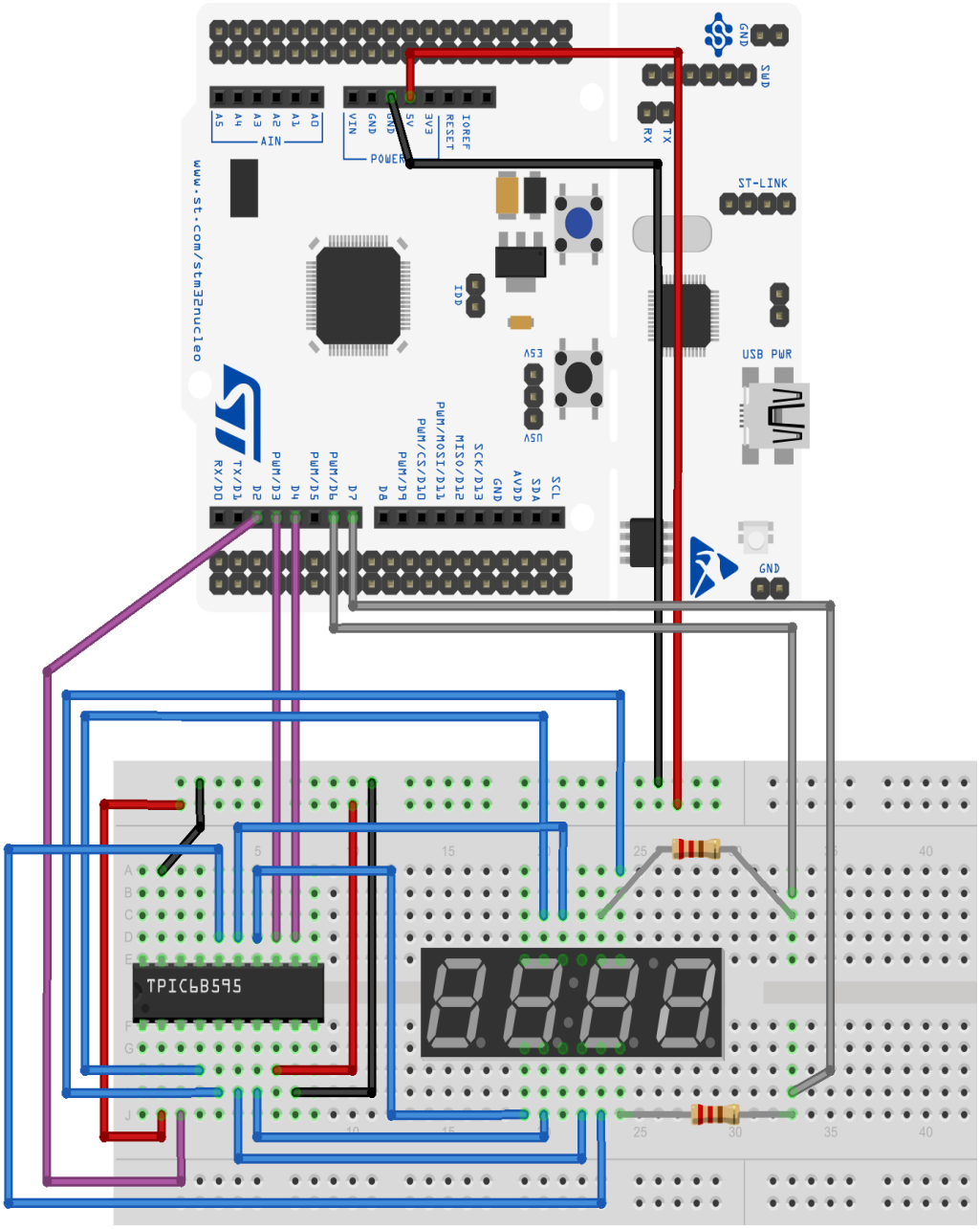


Figura 4. Schemă electrică pentru afisajele cu patru numere

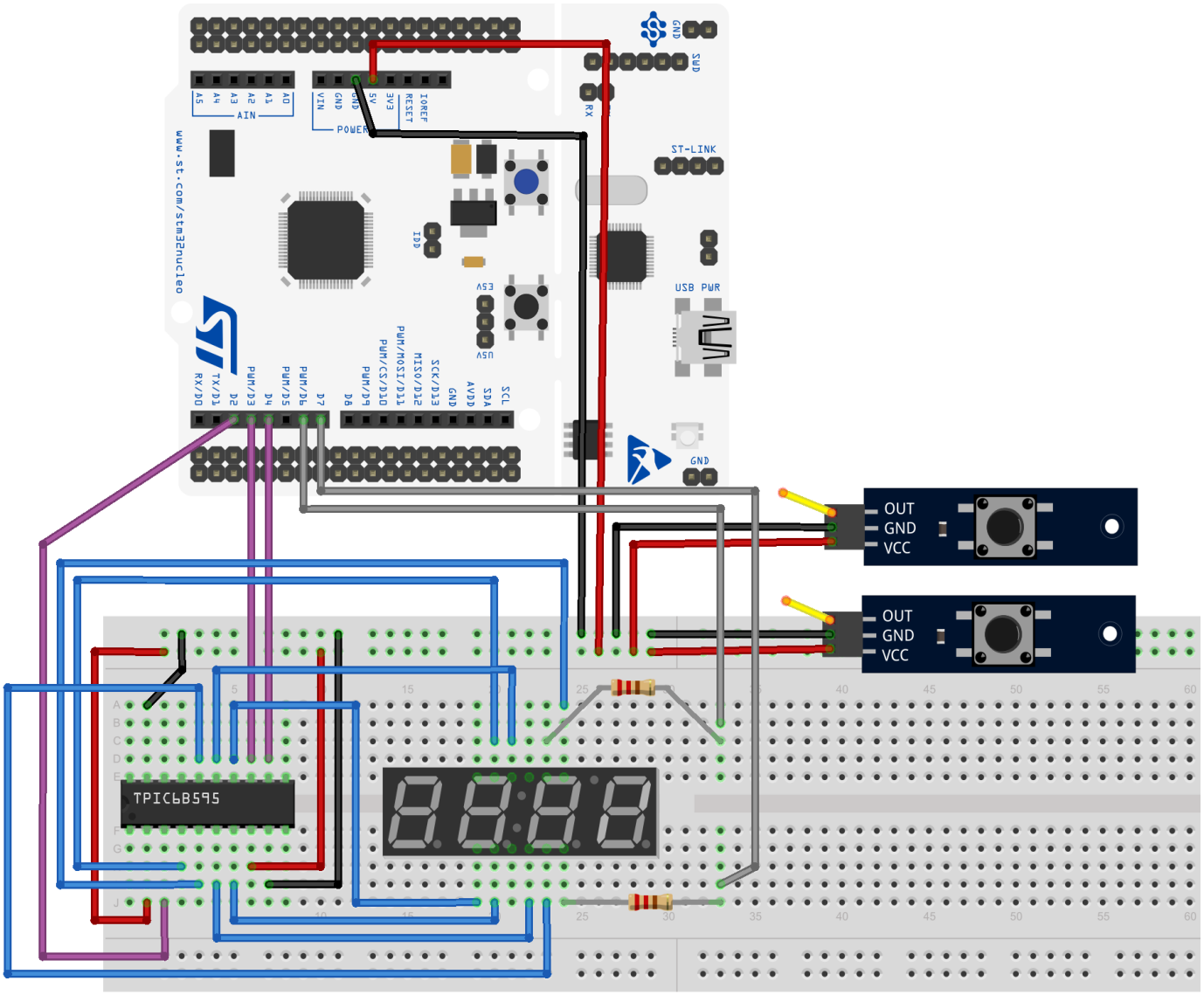
Schema pentru numărător pe două caractere



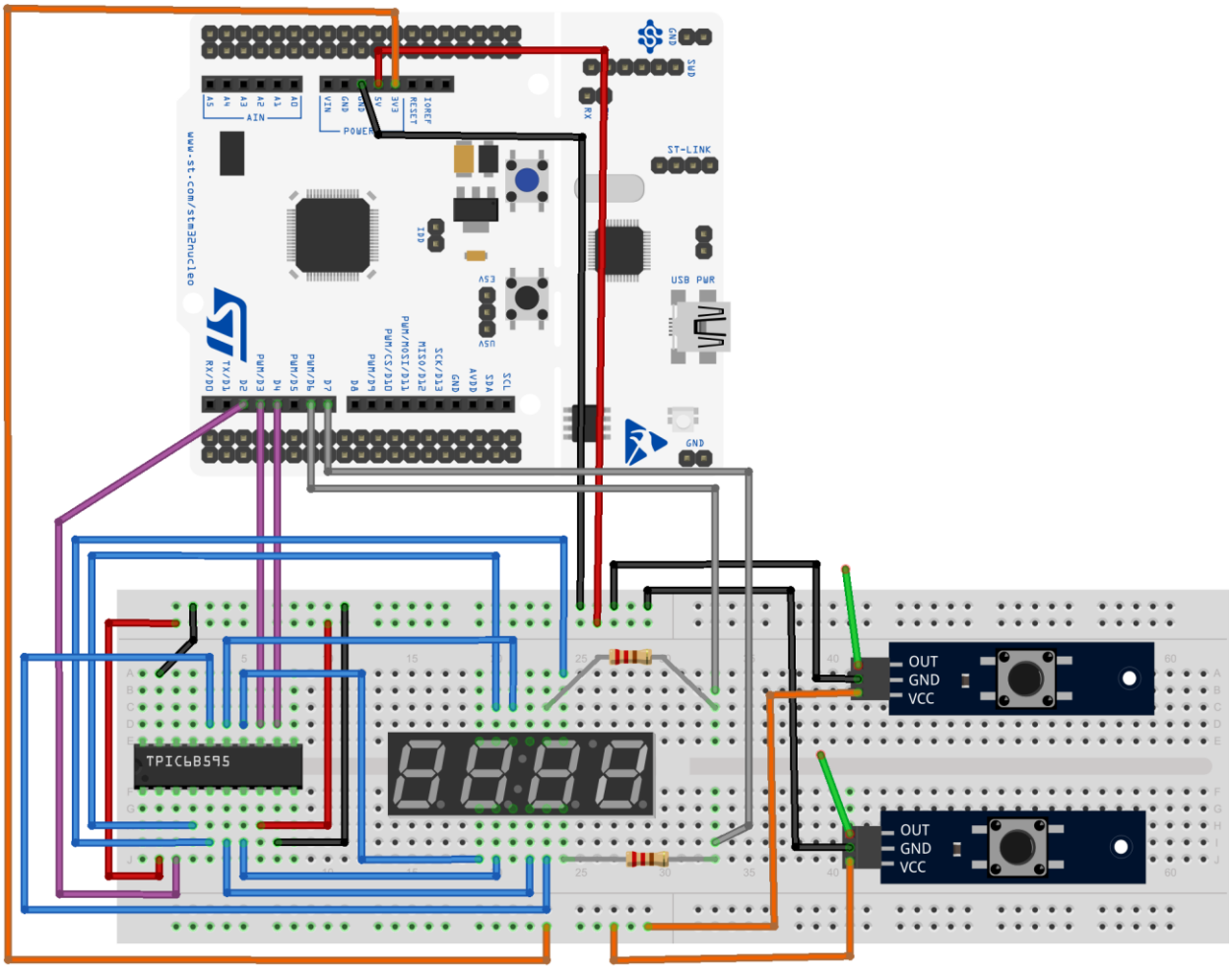
Schema pentru numărător cu patru caractere



Schema pentru numărător cu patru caractere și butoane externe pentru pini care sunt toleranti la 5V



Schema pentru numărător cu patru caractere și butoane externe pentru pini care NU sunt toleranti la 5V



Pinout Nucleo F103RB