

Universitatea „Politehnica” din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

**Interfață de comunicare și setare a parametrilor cu scopul testării
smartcardurilor**

Proiect de diplomă

prezentat ca cerință parțială pentru obținerea titlului de
Inginer în domeniul Electronică și Telecomunicații
programul de studii de licență Microelectronică, Optoelectronică și Nanotehnologii

Conducători științifici:

Prof. dr. ing. Corneliu BURILEANU

Ing. Mihai Silviu TOADER

Student:

Ionescu Cristina Alexandra

Universitatea "Politehnica" din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației
Departamentul DCAE

Anexa 1

TEMA PROIECTULUI DE DIPLOMĂ
a studentului **IONESCU C.V. Cristina Alexandra , 441E**

1. Titlul temei: Interfață de comunicare și setare a parametrilor cu scopul testării smartcardurilor

2. Descrierea contribuției originale a studentului (în afara părții de documentare) și specificații de proiectare:

Proiectul de diplomă constă în proiectarea schemei electrice pentru dezvoltarea unei interfețe de testare pentru un smartcard, proiectarea layoutului și programarea unui microcontroler Infineon ce va trimite semnale de control către celelalte dispozitive folosite. El este realizat în colaborare cu firma Infineon Technologies Romania și constituie un proiect ce va ajuta echipa din care fac parte.

Ideea de la baza acestui proiect este de a înlocui cititoarele de carduri folosite în prezent pentru testare cu produse dezvoltate în interiorul echipei pentru a reduce atât dependența de alți distribuitori (pentru achiziționare și calibrare), cât și costurile pe care le implică această dependență. Schema va conține elemente de stabilizare a tensiunilor de alimentare necesare pentru diferitele dispozitive, oscilatoare pentru generarea semnalelor de ceas cu frecvențele dorite, relee pentru întreruperea căilor de semnal când este de dorit acest lucru, și altele. Microcontrolerul trebuie programat astfel încât să se poată realiza comunicarea cu dispozitivul testat, cu un potențiomtru programabil, cu utilizatorul (prin USB) și să poată transmite semnale de control (de exemplu, enable), pentru celelalte dispozitive.

În final, placa pe care o voi proiecta (ca schemă și layout), prin intermediul microcontrolerului programat, va oferi o cale cât mai transparentă de comunicare între utilizator și smartcardurile ce trebuie testate, comunicarea dintre calculator și cititorul proiectat realizându-se prin USB.

3. Resurse folosite la dezvoltarea proiectului:

Altium Designer, IDE specific dispozitivelor Infineon

4. Proiectul se bazează pe cunoștințe dobândite în principal la următoarele 3-4 discipline:

Dispozitive Electronice, Arhitectura Microprocesoarelor, Microcontrolere

5. Proprietatea intelectuală asupra proiectului aparține: companiei

6. Data înregistrării temei: 2018-11-28 17:01:36

Conducător(i) lucrare,

Prof. dr. ing. Corneliu BURILEANU

semnătura:

Ing. TOADER Mihai-Silviu, Infineon Technologies Romania

semnătura:

Student,

semnătura:

Director departament,

Prof. dr. ing. Claudiu DAN

semnătura:

Decan,

Prof. dr. ing. Cristian NEGRESCU

semnătura:

Cod Validare: **6aaf9d8f05**

Copyright © 2019 , *Cristina Alexandra IONESCU* / Infineon Technologies, România

Toate drepturile rezervate

Autorul acordă UPB dreptul de a reproduce și de a distribui public copii pe hârtie sau electronice ale acestei lucrări, în formă integrală sau parțială.

Declarație de onestitate academică

Prin prezenta declar că lucrarea cu titlul “ *Interfață de comunicare și setare a parametrilor cu scopul testării smartcardurilor*”, prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității “Politehnica” din București ca cerință parțială pentru obținerea titlului de *Inginer* în domeniul *Electronică și Telecomunicații*, programul de studii *Microelectronică, Optoelectronică și Nanotehnologii* este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățământ superior din țară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referință la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referință la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor și măsurărilor pe care le prezint ca fiind făcute de mine, precum și metodele prin care au fost obținute, sunt reale și provin din respectivele simulări, experimente și măsurători. Înțeleg că falsificarea datelor și rezultatelor constituie fraudă și se sancționează conform regulamentelor în vigoare.

București, 18.06.2019

Absolvent *Cristina Alexandra IONESCU*

(semnătura în original)

Cuprins

Lista figurilor	11
Lista tabelelor.....	13
Lista acronimelor	15
Introducere	17
1. Microcontrolerul de tip smartcard	19
1.1. Scurt istoric.....	19
1.2. Proprietăți electrice.....	19
1.3. Transmiterea datelor conform ISO 7816-3	20
1.4. Tipuri de procesoare	21
1.5. Tipuri de memorii.....	21
1.6. Aplicații ale smartcardurilor.....	21
2. Standardul ISO/IEC 7816-3.....	23
2.1. Activarea și dezactivarea.....	24
2.2. <i>Cold Reset</i> și <i>Warm Reset</i>	25
2.3. Răspunsul la resetare ATR.....	25
3. Obiectivul lucrării	29
3.1. Privire de ansamblu asupra soluției propuse	29
3.2. Privire asupra utilizării unui cititor de carduri disponibil pe piață.....	29
3.3. Privire asupra folosirii unui microcontroler Infineon.....	30
3.4. Privire asupra folosirii unui microcontroler STM32	32
3.5. Potențiale avantaje și dezavantaje ale soluțiilor propuse	33
4. Mediul de dezvoltare Nucleo.....	35
4.1. Microcontrolerul STM32H743.....	35
4.1.1. Memoria	35
4.1.2. Modulul Reset and Clock Control (RCC).....	35
4.1.3. Pinii GPIO.....	36
4.1.4. Gestionarea întreruperilor	36
4.1.5. Modulul USART	38
4.1.6. Modulul SPI	39
4.1.7. Modulul I2C	40
4.1.8. Modulul SWP.....	41

4.2.	Arhitectura plăcii Nucleo	41
4.3.	Interfața ST8034HC	42
5.	Ansamblul fizic propus pentru dezvoltarea proiectului.....	45
6.	Programarea microcontrolerului STM32H743.....	49
6.1.	Conceptul programului.....	49
6.2.	Comunicarea cu cardul și structura comenzilor	50
6.3.	Secvența de activare pentru ATR.....	54
6.4.	Secvența de activare pentru șirul de caractere de identificare a cardului.....	56
6.5.	Schimbarea frecvenței semnalului de ceas	57
6.6.	Schimbarea tensiunii de lucru	58
7.	Rezultate și probleme întâmpinate.....	61
	Concluzii și planuri de dezvoltare viitoare	65
	Bibliografie	67
	Anexe	69
	Anexa A – Schema electrică pentru integrarea interfeței analogice	69
	Anexa B – Alte rezultate obținute	71
	Anexa C – Implementarea unor funcționalități	75
	Anexa D – Afișarea graficelor cu ajutorul Matlab	78

Lista figurilor

Figura 1.1. Propunere de aranjament al modulelor într-un microcontroler smartcard [2].....	21
Figura 2.1 Contactele modulului tipic smartcard [2]	23
Figura 2.2. Secvențele de activare și de dezactivare.....	24
Figura 2.3. Răspunsul la resetare – ATR	26
Figura 2.4. Secvența de activare pentru primirea șirului de caractere de identificare a cardului	27
Figura 3.1. Schema bloc simplificată a soluției propuse.....	31
Figura 3.2. Diagrama conceptuală	32
Figura 4.1. Gestionarea unei întreruperi pentru un model non-nested.....	36
Figura 4.2. Gestionarea întreruperilor în timp cu ajutorul unui model nested [5]	37
Figura 4.3. Modelul interconectării a două module UART	38
Figura 4.4. Modelul interconectării dispozitivelor pentru comunicarea prin SPI.....	40
Figura 4.5. Modelul interconectării dispozitivelor pentru comunicarea prin I2C.....	41
Figura 4.6. Diagrama bloc a plăcii de dezvoltare Nucleo – model simplificat.....	42
Figura 5.1. Diagrama generală a ansamblului construit.....	45
Figura 5.2. Interconectarea interfeței cu pinii corespunzători ai microcontrolerului	45
Figura 5.3. Rezistențele de pull-up și de pull-down	46
Figura 5.4. Condensatoarele de filtrare	46
Figura 5.5. Conectorul pentru smartcard	47
Figura 5.6. LED-uri pentru semnalizare	48
Figura 6.1. Diagrama concepuală a programului.....	49
Figura 6.2. Diagrama conceptuală a implementării comenzilor	50
Figura 6.3. Frecvența semnalului de ceas de lucru implicită.....	54
Figura 6.4. Secvența de activare pentru ATR și răspunsul primit	55
Figura 6.5. ATR-ul unui dispozitiv Infineon, după secvența de resetare generată de microcontrolerul STM	56
Figura 6.6. Șirul de caractere de identificare a cardului	56
Figura 6.7. Frecvența semnalului de ceas, configurată la 500 kHz	57
Figura 6.8. Frecvența semnalului de ceas, configurată la 10 MHz.....	57
Figura 6.9. ATR-ul primit de către PC, vizualizat cu ajutorul Hterm.....	58
Figura 6.10. Tensiunea configurată, pentru început, la 3 V.....	59
Figura 6.11. Configurarea tensiunii de lucru la 1,8 V.....	59
Figura 6.12. Configurarea tensiunii de lucru la 5 V.....	60
Figura 6.13. Configurarea tensiunii de lucru la 0 V - oprirea trecerii semnalelor către DUT	60
Figura 7.1. Întârzierea introdusă de interfața analogică între semnalul de ceas de intrare și cel de ieșire	61
Figura 7.2. Întârzierea dintre semnalul I/O la intrarea în interfață și cel de la ieșire.....	61
Figura 7.3. Întârzierea introdusă pe semnalul I/O la sfârșitul transmisiunii ATR-ului	62
Figura 7.4. Durata răspunsului la resetare.....	62

Lista tabelelor

Tabelul 1.1. Descrierea clasei dispozitivelor în funcție de trei parametri electrici. Toleranțele pentru parametrii discutați [2]	20
Tabelul 4.1. Configurarea tensiunii de alimentare/nivelului de tensiune pentru semnalele transmise către DUT [6]	43
Tabelul 6.1. Structura comenzilor de bază implementatec	51
Tabelul 6.2. Comanda de schimbare a frecvenței semnalului de ceas	52
Tabelul 6.3. Comanda de schimbare a tensiunii de lucru	53
Tabelul 6.4. Implementarea pentru interpretarea comenzii primite.....	53

Lista acronimelor

Acronim	Denumire	Explicații
I^2C	Inter-Integrated Circuit	Protocol de comunicare
ARM	Advanced RISC Machine	
ATR	Answer To Reset	
C1, C2, ..., C8	Contactele 1, 2, ..., 8	
CLK	Clock	Semnalul sau contactul de ceas
CMOS	Complementary Metal Oxide Semiconductor	Tehnologie pentru construirea circuitelor integrate
CS	Chip Select	Semnal trimis de la un dispozitiv pentru a selecta un alt dispozitiv – spre a-i activa funcționalitățile
DSS	Digital Security Solutions	
DUT	Device Under Test	
ECC	Error Code Correction	
ETH	Ethernet	
FPU	Floating Point Unit	
GPIO	General Purpose Input Output	
GSM	Global System for Mobile Communications	
HAL	Hardware Abstraction Layer	
I/O	Input/Output	Semnalul sau contactul de intrare/ieșire
ISO/IEC 7816-3	International Organization for Standardization/ International Electrotechnical Commission 7816-3	
ISR	Inerrupt Service Routine	
ITP	Interrupt and Peripheral Event Channel Controller	
JTAG	Joint Test Action Group	
LDO	Low Drop-Out Regulator	
LED	Light Emitting Diode	
MISO	Master Input Slave Output	
MOSI	Master Output Slave Input	
NMI	Non-maskable Interrupt	
NMOS	N-type Metal Oxide Semiconductor	
NVIC	Nested Vectored Interrupt Controller	
NVM	Non-Volatile Memory	
PC	Personal Computer	
PCB	Printed Circuit Board	Circuitul Printat

PCB	Printed Circuit Board	
PLL	Phase Locked Loop	Circuit ce poate urmări frecvența unui semnal de intrare, sau genera un semnal cu o frecvență multiplu al frecvenței semnalului de intrare
RAM	Random Access Memory	
RCC	Reset and Clock Control	
RISC	Reduced Instruction Set Computer	
RST	Reset	Semnalul sau contactul de resetare
RTI	Rutina de Tratare a Întreruperii	
RX	Receive	Pinul de recepționare a datelor prin UART
SCLK	Serial Clock	Semnalul de ceas utilizat în protocolul SPI
SECEDED	Single-Error Correcting and Double-Error Detecting	
SIM	Subscriber Identity Module	
SPI	Serial Peripheral Interface	Protocol de comunicare
SRAM	Static Random Access Memory	
SWD	Serial Wire Debug	
SWI	Software Interrupt	Întrerupere ce poate fi generată software, de către utilizator
SWP	Single Wire Protocol	Protocol de comunicare
SWPMI	Single Wire Protocol Master Interface	
TCM	Tightly-Coupled Memory	
TX	Transmit	Pinul de transmitere a datelor prin UART
UART	Universal Asynchronous Receiver Transmitter	
UCP	Unitatea centrală de prelucrare	
USART	Universal Synchronous/Asynchronous Receiver Transmitter	
USB	Universal Serial Bus	
VCC	Contact de alimentare	

Introducere

Una dintre cele mai importante etape din procesul de producție a unui dispozitiv de orice fel este testarea. Cu cât testarea este efectuată mai eficient și mai riguros, poate salva efortul fizic și financiar depus în repararea sau înlocuirea dispozitivelor care nu funcționează conform specificațiilor. Odată cu evoluția tehnologiei și, implicit, cu creșterea vitezei de producție, devine din ce în ce mai demnă de luat în seamă optimizarea procedurilor de testare pentru a putea livra produse fiabile într-un timp cât mai scurt [1]. Ansamblul de test trebuie să fie creat în așa fel încât să ofere rezultate fidele, să permită implementarea unor scenarii de test diverse, să fie ieftin, fiabil, și performant, oferind un raport convenabil între siguranța (încrederea în funcționarea produselor conform specificațiilor) și costul necesar testării. Rolul inginerilor de testare este, așadar, acela de a automatiza pe cât posibil procesul de testare, de a se asigura de buna funcționare a dispozitivelor atât din punct de vedere electric, cât și din punct de vedere logic, de a se asigura că dispozitivele testate respectă standardele naționale și internaționale și țin seama de specificațiile pentru care au fost proiectate.

Scopul acestei lucrări este acela de a dezvolta o interfață de test pentru dispozitivele Infineon, în special în susținerea proiectelor departamentului DSS. Interfața reprezintă un cititor de carduri și este folosită pentru transmiterea unor comenzi și date către DUT și pentru setarea unor parametri de bază pentru testarea funcționalității DUT-ului, precum tensiune, frecvență. Problema dezvoltării unei astfel de interfețe a apărut din necesitatea înlocuirii cititorului folosit în prezent, MP65, dezvoltat de Micropross. Echipa de sprijin Micropross nu va mai calibra și nu va mai repara cititoarele respective, așadar va fi nevoie de o soluție nouă, ieftină și optimizată ce va fi folosită de către departamentul de verificare la infrastructura de verificare. Proiectul presupune programarea unui microcontroler aflat pe o placă de dezvoltare pentru gestionarea datelor primite de la DUT și a datelor și comenzilor trimise de utilizator, proiectarea unui circuit ce are în centru un dispozitiv de tip „interfață” de control pentru aducerea semnalelor la nivelul de tensiune dorit și implementarea unor funcționalități specifice unui cititor de carduri cu ajutorul acestor dispozitive.

Pentru a atinge acest obiectiv am folosit o placă electrică de dezvoltare Nucleo ce are la bază microcontrolerul STM32H743 și am proiectat o machetă electronică ce conține dispozitivul de tip „interfață” ST8034HC.

În primul capitol este descris microcontrolerul de tip smartcard, sunt prezentate un scurt istoric și câteva aplicații ale smartcardurilor.

În cel de-al doilea capitol sunt prezentate standardele ce trebuie respectate atunci când este vorba despre comunicarea cu un smartcard și câteva funcționalități de bază pe care trebuie să le dețină orice cititor de carduri, cuprinse, și ele, de standardele respective.

Al treilea capitol prezintă obiectivul lucrării, problemele și întrebările de la care a luat naștere conceptul lucrării, soluțiile propuse pentru rezolvarea problemelor.

Capitolul 4 prezintă elementele de bază – de interes pentru această lucrare – referitoare la memoria microcontrolerului folosit, la interfețele și modulele puse la dispoziție, la felul în care acestea funcționează și blocurile principale de pe placa de dezvoltare folosită.

În capitolul 5 vorbim despre implementarea hardware a proiectului, folosirea unei plăci de dezvoltare, proiectarea unei plăci pentru folosirea unei interfețe analogice.

Al șaselea capitol are drept conținut implemetarea software a algoritmului propus pentru aplicația cititorului, dispozitivul programat fiind microcontrolerul STM32H743.

Capitolul 7 conține alte rezultate obținute și probleme întâmpinate pe parcurs.

În final, prezentăm concluziile și planurile de dezvoltare viitoare, urmate de bibliografie și anexe.

1. Microcontrolerul de tip smartcard

Pentru a putea înțelege cum să testăm un astfel de dispozitiv, trebuie să cunoaștem standardele de funcționare ce trebuie respectate și să înțelegem care sunt interfețele de comunicare utilizate cel mai des în domeniu, dar, pentru început, trebuie să aflăm ce este un smartcard și cum a apărut necesitatea folosirii unuia.

Un smartcard este un microcontroler de joasă putere ce are la bază un hardware cu capacități criptografice performante.

1.1. Scurt istoric

Popularitatea smartcardurilor a luat naștere încă din anii '50. Prețul mic al materialului sintetic din care sunt confecționate cardurile și natura materialului au făcut posibilă fabricarea unui plastic durabil, robust și potrivit pentru utilizarea zilnică, putând fi supus atât unei varietăți mari de condiții climatice, cât și unui stres mecanic semnificativ.

Inițial, cardurile au fost acceptate drept dispozitive de plată numai în locații exclusiviste, urmând apoi să fie recunoscute avantajele pe care le aduc la un nivel mult mai mare. Odată cu apariția cardurilor, riscul de pierdere a banilor din cauza vreunei fraude sau a unor alte hazarduri imprevizibile a scăzut. Totodată, folosirea unui card elimină necesitatea schimbului valutar la trecerea graniței către o țară cu o monedă națională diferită. În prezent, sute de milioane de carduri sunt fabricate și utilizate anual.

Pentru început, funcționalitățile pe care le-a deținut un card au fost simple: a fost nevoie de un dispozitiv care să conțină informații legate de deținătorul cardului, o semnătură pentru identificarea acestuia, numărul cardului, date stocate cu un oarecare nivel de securitate. Cardurile au devenit din ce în ce mai puțin exclusiviste, fapt ce a adus în discuție noi tehnologii de citire a lor și de securizare a datelor stocate cu ajutorul lor. Funcționalitățile de bază și protecția împotriva furtului și a folosirii frauduloase a datelor stocate pe carduri au fost îmbunătățite. Primul element de perfecționare a fost banda magnetică de pe spatele cardului, care a permis ca datele digitale să poată fi stocate pe card într-o formă ce poate fi citită de o mașină, adiacent informației vizuale. Tranzacțiile pe bază de hârtie au fost înlocuite de procesarea electronică a datelor, pentru care a fost necesară o nouă metodă de autentificare: PIN, Personal Identification Number, sau Număr Personal de Identificare. Acesta este un număr secret, comparat cu un număr de referință. Cu toate acestea tehnologia benzii magnetice are unele slăbiciuni, cea mai importantă fiind aceea că datele stocate pe card pot fi citite, șterse și rescrise de oricine are echipamentul necesar pentru a comunica cu cardul. Așadar, nu este potrivit pentru stocarea datelor confidențiale [2]. Trebuie folosite tehnici auxiliare pentru a asigura confidențialitatea datelor și pentru a preveni manipularea acestora.

1.2. Proprietăți electrice

Proprietățile electrice ale smartcardurilor depind în întregime de microcontrolerul integrat, fiind singura componentă electronică din card. În funcție de aplicațiile în care au fost folosite, cardurilor li s-au impus anumiți parametri de funcționare prin intermediul unor standarde internaționale. Astfel, în tabelul 1.1 sunt precizați cei mai relevanți trei parametri electrici pentru această lucrare,

reglementați de standardele ISO/IEC 7816-3. Acești parametri de funcționare stau la baza tuturor testelor aplicate smartcardurilor pentru a asigura buna funcționare.

Tabelul 1.1. Descrierea clasei dispozitivelor în funcție de trei parametri electrici. Toleranțele pentru parametrii discutați [2]

Clasă	Tensiune	Frecvența de lucru	Curent maxim
Clasa A	5 V $\pm$ 10 % => 4.5 ÷ 5.5 V	1 – 5 MHz	60 mA la 5MHz
Clasa B	3 V $\pm$ 10 % => 2.7 ÷ 3.3 V	1 – 5 MHz	50 mA la 4MHz
Clasa C	1.8 V $\pm$ 10 % => 1.62 ÷ 1.98 V	1 – 5 MHz	30 mA la 4MHz
Clasele A, B și C fără semnalul de ceas	-	-	0.5 mA (semnalul de ceas oprit)

Conform standardelor, interfața dintre cititor și smartcard este de tip *master – slave*, iar masterul, cititorul, este responsabil pentru generarea semnalului de ceas. Acest semnal de ceas va oferi, de asemenea, o referință pentru rata de transmitere a datelor. Conform standardelor, semnalul de ceas trebuie să aibă un factor de umplere de 50 %, dar toleranța permite ca semnalul de ceas să aibă un factor de umplere în intervalul 40 ÷ 60 %.

1.3. Transmiterea datelor conform ISO 7816-3

Posibilitatea comunicării bidirecționale este indispensabilă pentru toate interacțiunile dintre un smartcard și un terminal. În orice caz, numai o singură linie electrică este disponibilă pentru comunicarea cu smartcardul. Datele digitale sunt interschimbate între card și terminal prin această conexiune electrică. Astfel, cardul și terminalul trebuie să transmită date pe rând; această procedură poartă denumirea de *half-duplex*.

O procedură *full-duplex*, în care ambele instanțe pot transmite și recepționa date în același timp, nu este încă implementată în cazul cardurilor, deși este, tehnic, realizabilă. Comunicarea cu smartcardul este întotdeauna inițiată de către terminal [2]. Cardul răspunde la comenzi primite, așadar cardul nu va trimite niciodată date fără a primi, înainte, un stimul exterior. Acest lucru stabilește o relație pur *master-slave*, unde terminalul este *master*, iar cardul este *slave*.

După ce un card este inserat în terminal, contactele lui sunt, pentru început, mecanic conectate cu cele ale terminalului. Cele cinci contacte active sunt, mai apoi, activate din punct de vedere electric, cu ajutorul unei secvențe de activare. După aceea, cardul realizează automat o resetare la pornire – *power-on reset* – și trimite răspunsul la resetare – ATR – către terminal. Terminalul evaluează ATR-ul primit, iar apoi trimite prima comandă către card. Cardul procesează comanda și generează un răspuns, pe care îl va transmite terminalului [2]. Acest schimb de tip comandă-răspuns are loc până când cardul este dezactivat.

1.4. Tipuri de procesoare

Procesoarele utilizate în construcția smartcardurilor nu au arhitecturi deosebite, ci sunt mai degrabă dispozitive testate, dovedite a fi funcționale, folosite în alte domenii pentru o perioadă lungă de timp. În această industrie, nu este ceva obișnuit să se dezvolte ceva nou pentru aplicații specifice deoarece costurile ar fi foarte mari. Motivul principal rămâne totuși siguranța: procesoarele mai vechi, chiar dacă tehnologia avansează zi de zi, sunt deja recunoscute pentru fiabilitatea lor, deci este mai sigur că ele pot fi folosite în domeniul securității. Cardurile trebuie să fie de mare încredere, mai ales în domenii precum industria aerospațială, care prezintă un mare interes pentru securitate.

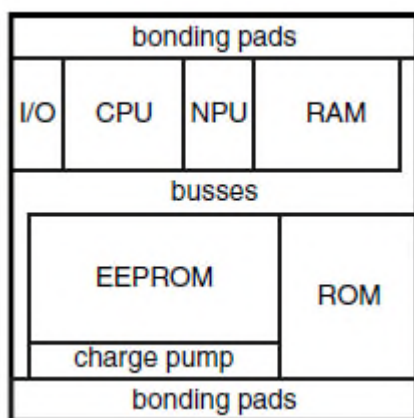


Figura 1.1. Propunere de aranjament al modulelor într-un microcontroler smartcard [2]

Microcontrolerele smartcard au avut o memorie adresabilă între 6 KB și 30kB, pentru început, iar mai apoi cu performanțe tot mai ridicate, procesoarele utilizate în general sunt de tip CISC, cu un set mare de instrucțiuni, care necesită un număr mai mare de cicli de ceas pentru executare.

În figura 1.1 este prezentat un model de aranjament al modulelor și interfețelor uzual în construcția unui microcontroler smartcard. Putem observa poziționarea modului de intrare-ieșire, a CPU (*Central Processing Unit*) – unitatea centrală de prelucrare, a unui microprocesor NPU (*Neural Processing Unit*) folosit cu precădere în aplicații de telefoane mobile, pentru accelerarea algoritmilor folosiți prin *machine learning*, a memoriilor RAM, EEPROM, ROM, a unui circuit de tip *charge pump* pentru controlul tensiunii.

1.5. Tipuri de memorii

Pe lângă procesor, cele mai importante componente ale unui microcontroler sunt diversele tipuri de memorie, care servesc la stocarea codului programului și a datelor. Fiindcă microcontrolerele smartcard trebuie să fie mașini de calcul complete, ele dețin toate cele trei memorii: RAM, memorie ROM și memorie non-volatilă. Frația din fiecare memorie depinde de aplicațiile în care va fi folosit dispozitivul. Efortul mai mare este de a păstra memoriile RAM și EEPROM cât mai mici posibil, deoarece ele au nevoie de cel mai mult spațiu pentru un bit de informație [2].

1.6. Aplicații ale smartcardurilor

Primele smartcarduri folosite într-un număr foarte mare au fost cardurile de memorie pentru aplicații telefonice. Logica integrată special pentru securitate face posibilă protecția datelor stocate

pe card. Au urmat microprocesoarele de tip smartcard, folosite pentru prima dată în băncile din Franța. Producția în masă a cardurilor a dus la o reducere semnificativă a costului lor, deci la introducerea unor noi aplicații în fiecare an – au fost utilizate în telefoanele mobile, în sisteme ce au nevoie de un control al accesului în zone cu acces limitat (autentificare), semnături electronice, stocare securizată a datelor, cardurile devenind și multifuncționale, incorporând mai multe aplicații într-un singur dispozitiv.

Smartcardurile sunt utilizate, în zilele noastre, pentru acces în locații restricționate, situație în care datele de pe smartcard trebuie să fie criptate, sistemul de detecție al unui atac trebuie să fie complex, în aplicații de plată, pentru carduri de credit securizate. Cardurile sunt folosite și în comunicații – SIM (Subscriber Identity Module) – și conțin aplicații și date introduse pe card de către operatorii mobili. Smartcardurile pot fi folosite și drept generatoare de chei de securitate pentru aplicații software [2].

O importanță foarte mare în a dicta cum trebuie să se comporte un dispozitiv, ce dimensiuni, caracteristici electrice, fizice, ce funcționalități ar trebui să aibă un smartcard o constituie standardele internaționale.

2. Standardul ISO/IEC 7816-3

ISO/IEC 7816 reprezintă o serie de standarde internaționale ce menționează cardurile bazate pe circuite integrate și folosirea unor astfel de carduri în schimburi de date. Aceste carduri sunt carduri de identificare pentru schimbul de informație negociat între lumea exterioară și circuitul integrat al cardului. Ca rezultat al schimbului de date, cardul livrează informația (rezultatele de calcul, datele stocate), sau le modifică conținutul [4]. Suntem interesați în mod deosebit această serie de standarde deoarece, pentru testarea corectă și completă a cardurilor noastre, trebuie să respectăm specificațiile prevăzute aici.

Există cinci fracțiuni ale acestui standard care sunt de interes pentru lucrarea de față, iar trei dintre ele conțin și specificații în legătură cu interfețele electrice:

- ISO/IEC 7816-1 conține specificații în legătură cu caracteristicile fizice ale cardurilor cu contacte;
- ISO/IEC 7816-2 se ocupă de dimensiunile și localizarea contactelor;
- ISO/IEC 7816-3 conține specificații ce au impact asupra interfețelor electrice și a protocoalelor de transmisiune pentru carduri asincrone;
- ISO/IEC 7816-10 conține specificații despre interfețele electrice și rutina de răspuns la resetare ale cardurilor sincrone;
- ISO/IEC 7816-12 conține informații legate de interfețele electrice și procedurile de operare pentru carduri USB [4].

Cel mai mult, însă, ne vom folosi de partea a treia a acestui standard, în elaborarea lucrării. Ea oferă specificații în legătură cu puterea și structura semnalelor, cu schimbul de informație dintre card, circuitul integrat, și o un dispozitiv de interfață, terminalul. Totodată, această parte a standardului acoperă nivelele de tensiune, valorile specifice ale curentului, convențiile de paritate, mecanisme de transmisiune a datelor și de comunicare cu cardul. Nu conține constrângeri legate de conținutul instrucțiunilor, caracteristici de securitate, identificare a utilizatorilor [4].

C1		C5
C2		C6
C3		C7
C4		C8

VCC		GND
RST		VPP
CLK		I/O
AUX1		AUX2

Figura 2.1 Contactele modului tipic smartcard [2]

Contactele descrise în conținutul acestui standard sunt:

- C1: contact folosit pentru alimentarea cardului (VCC). În standard sunt definite trei clase de condiții de operare, bazate pe tensiunea nominală de alimentare furnizată cardului prin acest contact: 5V (clasa A), 3V (clasa B), 1,8V (clasa C);
- C2: acest contact este folosit pentru a trimite cardului semnalul de resetare (RST);
- C3: contact utilizat pentru asigurarea semnalului de ceas (CLK);
- C4: contact suplimentar – AUX1;
- C5: masa (GND, tensiune de referință);
- C6: contact folosit pentru tensiune programabilă – VPP – nu a mai fost utilizat; în prezent, este utilizat adesea pentru comunicație prin SWP;
- C7: acest contact (denumit I/O) este folosit ca intrare (în modul de recepție) sau ieșire (în modul de transmisie);
- C8: contact suplimentar – AUX2;

În figura 2.1 este prezentată structura tipică a contactelor unui smartcard și felul în care sunt dispuse.

2.1. Activarea și dezactivarea

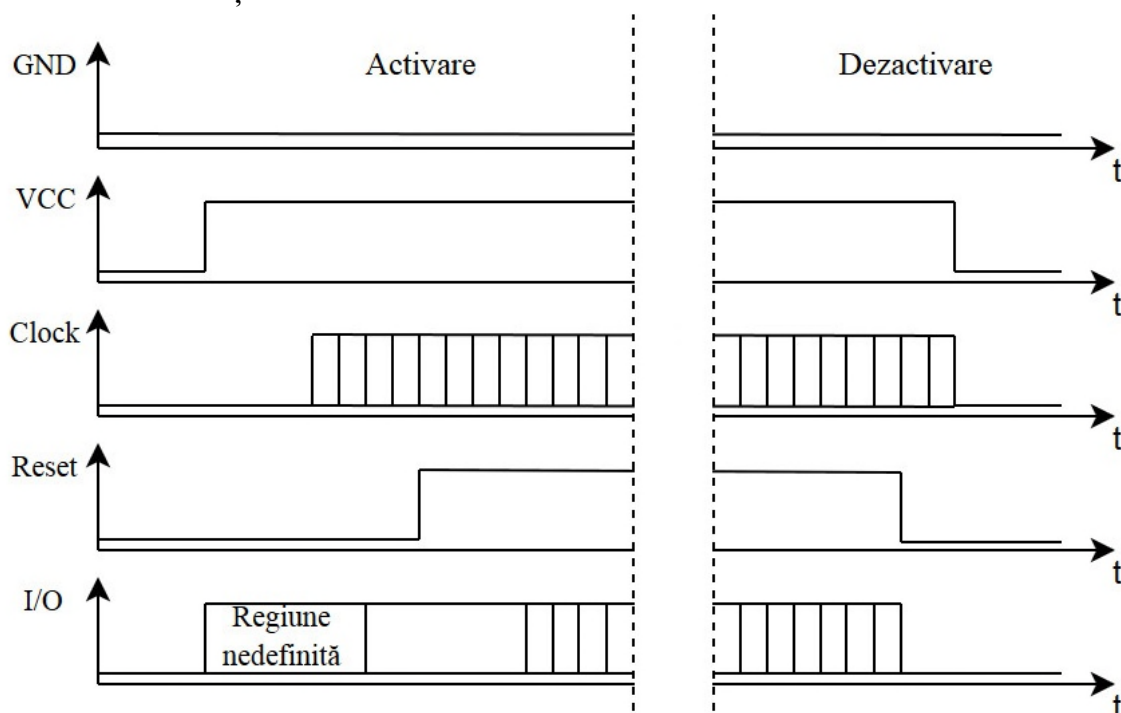


Figura 2.2. Secvențele de activare și de dezactivare

Figura 2.2 prezintă secvențele de activare și dezactivare, comportamentul semnalelor în timp. Pentru a iniția interacțiunea cu un card conectat mecanic, dispozitivul de interfață trebuie să activeze circuitele electrice în următoarea ordine:

- RST trebuie pus în starea în „0” logic;
- La VCC trebuie aplicată alimentarea;
- I/O trebuie pus în modul de recepționare;
- La CLK trebuie aplicat un semnal de ceas.

Atunci când schimbul de informație a avut loc sau a eșuat, dispozitivul de interfață trebuie să dezactiveze circuitele electrice în următoarea ordine:

- RST trebuie pus în „0” logic;
- CLK trebuie pus în „0” logic;
- I/O, de asemenea, în „0” logic;
- VCC – alimentarea trebuie dezactivată.

2.2. Cold Reset și Warm Reset

Resetarea „rece”, sau *Cold Reset* presupune înlăturarea completă a alimentării și restartarea smartcardului.

Prin noțiunea de *warm reset* (resetare „caldă”) înțelegem același lucru ca prin *cold reset*, numai că, de această dată, cardul va primi un semnal de resetare fără ca alimentarea să fie întreruptă.

2.3. Răspunsul la resetare ATR

Răspunsul la resetare (ATR) este un mesaj primit de la un smartcard conform standardelor ISO/IEC 7816, ce urmează unei resetări electrice a cardului, efectuată cu ajutorul cititorului. ATR-ul poate conține informații legate de parametrii de comunicație propuși de card, legate de natura sau starea cardului, sau poate fi specific aplicației încărcate pe card.

ATR-ul oferă o primă informație legată de funcționalitatea cardului și este primul răspuns ce indică faptul că microcontrolerul este operativ. Conținutul acestui șir de caractere este primul lucru verificat [2].

Atunci când cardul este inserat în cititor, nu este aplicată tensiune la niciunul dintre contacte. După ce cardul este inserat corespunzător, cu alinierea corectă a contactelor, alimentarea este aplicată.

Pentru început, contactele sunt aduse într-o stare de inactivitate, pentru ca mai apoi un semnal de resetare să fie transmis cardului prin intermediul liniei RST. Semnalul VCC este stabilit la un potențial fix, contactul I/O este setat în modul de recepție dinspre cititor și un semnal de ceas stabil este aplicat la CLK. Linia de resetare va rămâne activă („low” – „0” logic) pentru cel puțin 40 000 cicluri de ceas înainte ca o secvență validă de resetare să poată fi pornită de către cititor, ridicând linia de RST în „high”, „1” logic [7].

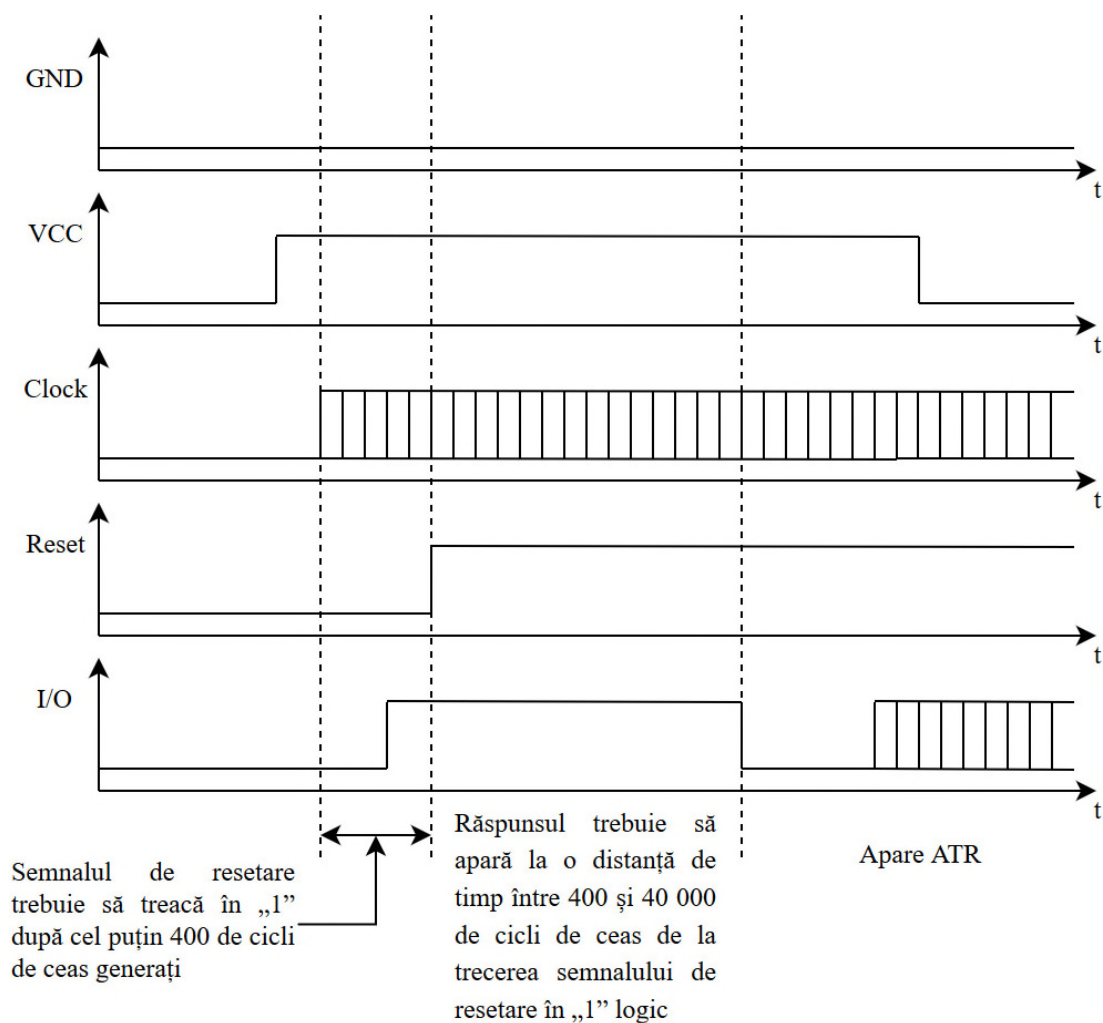


Figura 2.3. Răspunsul la resetare – ATR

În figura 2.3 este prezentată secvența necesară de activare pe care trebuie să o efectueze cititorul pentru a putea primi ATR-ul și momentul de timp aproximativ la care acesta ar trebui să fie recepționat de la dispozitiv, de către cititor.

În plus, în afară de secvența de activare pentru ATR, există o secvență de activare pentru recepționarea de la DUT a unui șir de caractere special, șir unic de identificare a cardului. Secvența este prezentată în figura 2.4. Acest șir de caractere conține informații în legătură cu numărul plăchetei de siliciu, coordonate ce descriu poziția dispozitivului pe plăchetă, un număr de fabricație, date legate de sistemul de inițializare a programului, coduri de RAM, de ROM și alte informații. Acest șir de caractere este folosit pentru identificarea cardurilor pe care se efectuează anumite teste.

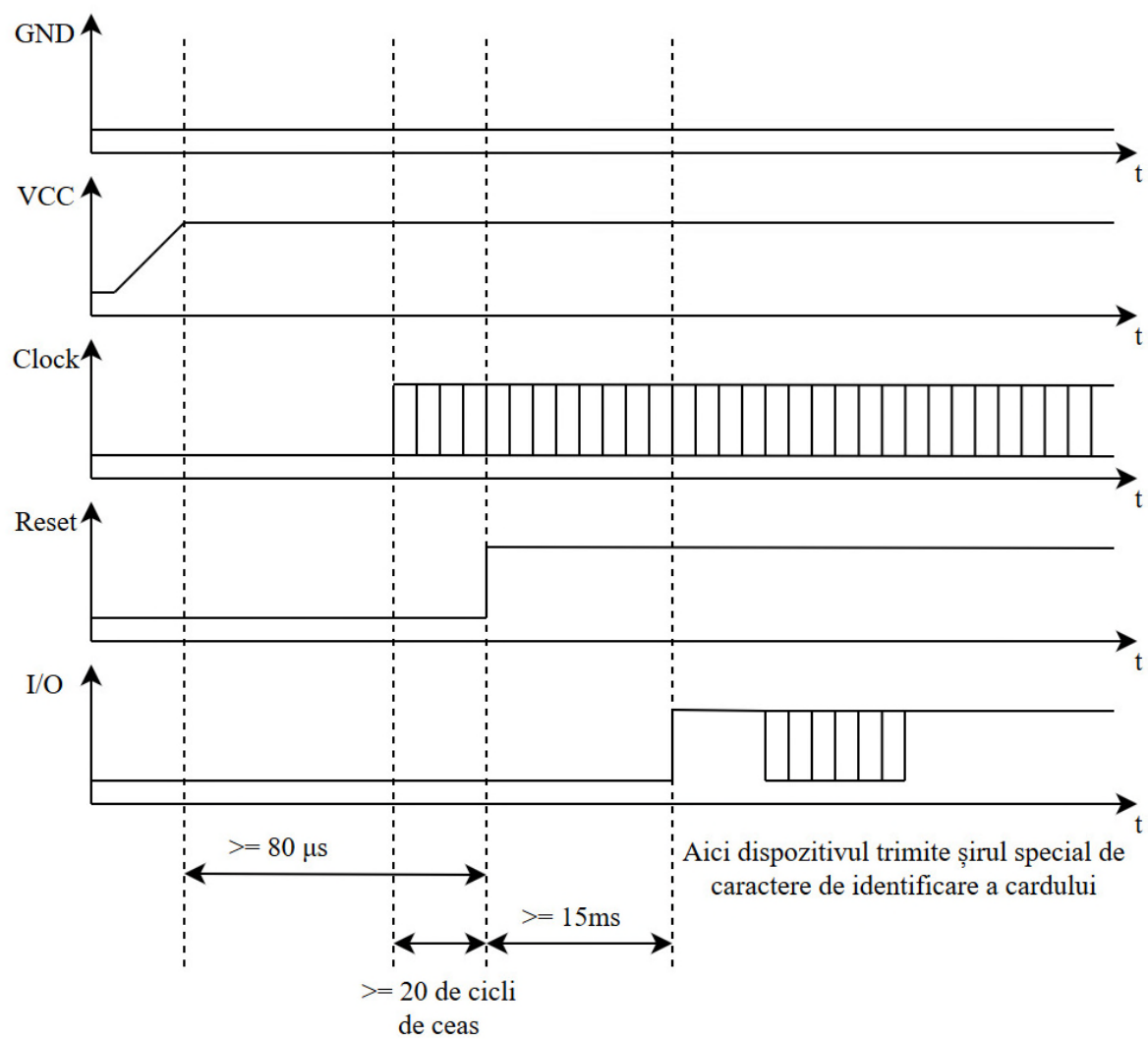


Figura 2.4. Secvența de activare pentru primirea șirului de caractere de identificare a cardului

3. Obiectivul lucrării

În prezent, pentru testarea smartcardurilor Infineon, este folosit un cititor dezvoltat de Micropross, cu denumirea MP65. Cititorul nu mai este fabricat, iar în viitor nu va mai exista niciun sprijin din partea dezvoltatorilor pentru calibrare, sau în cazul unor defecțiuni. Acest lucru constituie o problemă destul de mare într-un departament al cărui scop principal este testarea într-un timp cât mai scurt și într-un mod cât mai eficient a smartcardurilor, interfața către ele fiind tocmai acest cititor.

Astfel, am ajuns la obiectivul acestei lucrări: dezvoltarea unui cititor de carduri care să întrunească cea mai mare parte dintre caracteristicile vechiului cititor. Vom avea nevoie, deci, de un cititor capabil să seteze parametrii funcționali de bază ai unui smartcard, precum tensiunea de alimentare și frecvența de lucru. Acești parametri trebuie să poată fi variați cu un pas suficient de mic pentru ca cititorul să poată fi folosit în scenariile de test definite în echipa noastră. Cititorul trebuie să poată, totodată, să genereze o secvență de activare specifică, definită de standardul internațional, pentru primirea ATR-ului din partea smartcardului, să transmită și să recepționeze date către și de la smartcard, prin intermediul unei interfețe UART. Tot ce se întâmplă în interiorul cititorului trebuie să fie invizibil pentru utilizator, interesul acestuia fiind doar acela de a comunica cu smartcardul și de a seta parametrii cititorului prin intermediul unor comenzi simple.

Pentru realizarea acestui proiect am ținut cont de specificațiile standardelor ISO 7816 pentru smartcarduri.

3.1. Privire de ansamblu asupra soluției propuse

Primul pas în realizarea proiectului a fost o analiză detaliată a tuturor posibilităților existente. Întrebările pe care ni le-am pus au fost: „Ce soluții există deja pe piață pentru problema noastră?”, „Cât de eficiente sunt aceste soluții și care este costul lor?”, „Cum ar arăta o soluție dezvoltată în interiorul echipei?”, „Ce microcontroler ne poate oferi toate resursele de care avem nevoie?”. Într-o primă etapă am adunat informațiile necesare pentru a ne răspunde la aceste întrebări, am făcut o listă cu posibilitățile găsite, am eliminat cele mai puțin favorabile soluții, și am ajuns la trei mari modalități de a rezolva problema:

- Cititoare de carduri deja existente pe piață, soluții gata implementate;
- Folosirea unui microcontroler Infineon drept element central de control și comunicare cu DUT-ul și cu blocurile funcționale auxiliare, o soluție implementată pornind de la o aplicație simplă, proprietate a companiei;
- Folosirea unui microcontroler STM32, integrat pe o placă de dezvoltare Nucleo, și a unui modul auxiliar pentru interfațarea cu DUT-ul.

În continuare voi descrie cele trei soluții, precum și avantajele și dezavantajele pe care le presupune folosirea uneia dintre aceste soluții.

3.2. Privire asupra utilizării unui cititor de carduri disponibil pe piață

Există deja, pe piață, câteva soluții gata implementate și autorizate pentru testarea smartcardurilor, precum a fost și cititorul folosit până în prezent. Una dintre aceste soluții este chiar un înlocuitor

oferit de cei de la Micropross pentru cititorul MP65. Platforma hardware se numește MP300 SC2 și acoperă numeroase scenarii de test, oferă administrarea protocolului T=1 pentru smartcarduri, administrarea sincronizărilor specifice, o bibliotecă de sprijin pentru verificarea și setarea parametrilor necesari. Avantajele pe care le oferă în plus sunt sprijinul primit din partea echipei Micropross, calibrarea periodică, repararea în caz de defectare, certificarea cititorului în testarea de smartcarduri, biblioteca de funcții pe care o pun la dispoziție dezvoltatorii aplicației,

Deși o astfel de platformă oferă lucrurile de care avem nevoie într-un timp foarte scurt (determinat numai de durata procesului de achiziționare), platforma este foarte scumpă, iar, fiind necesare peste o sută de cititoare pentru deservirea proiectelor echipei, ar fi un preț destul de mare de plătit.

3.3. Privire asupra folosirii unui microcontroler Infineon

Primul concept de implementare a proiectului a avut la bază folosirea unui microcontroler Infineon. Microcontrolerul ales se remarcă prin diversitatea de periferice integrate de care dispune: UART pentru transmiterea de date conform standardului, I^2C , GPIO pentru generarea unor semnale simple de comandă și recepționarea unor semnale simple, USB.

Arhitectura procesorului este pe 16 biți, cu memoria organizată liniar cu adresare pe 24 de biți, dispune de până la 16 MB de memorie adresabilă, având în plus 128 B de spațiu de adrese dedicat pentru registrele cu funcții speciale. Instrucțiunile aritmetice și logice sunt pe 16 și, respectiv, 32 de biți, ce pot fi executate într-un timp minim, de un ceas. Microcontrolerul poate folosi moduri eficiente, flexibile de adresare pentru o densitate ridicată a codului precum adresare directă, adresare imediată, adresare indirectă, adresare prin deplasare (cu offset), adresare relativă, adresare pe bit. Conține o zonă dedicată de memorie pentru registrele cu funcții speciale, având și RAM, ROM, NVM. Microcontrolerul are și o memorie cache localizată între UCP și memoriile principale care menține copii ale unor conținuturi de memorie, pentru ca, atunci când este nevoie de aceste date, informația să fie accesată mult mai rapid. Arhitectura acestui microcontroler este asemănătoare cu a unui controler de tip Intel 8051 cu operații structurate după același model. Setul de instrucțiuni este compatibil cu cel al controlerelor din familia 80251.

Fiind un microcontroler utilizat cu precădere în domeniul securității, este dotat și cu componente precum: criptoprocessor, un coprocessor aritmetic pentru calculul rapid în algoritmi de criptare pentru chei publice, un generator de numere pseudo-aleatoare și un generator de numere aleatoare.

Totodată, de interes pentru aplicația propusă este tensiunea de lucru a microcontrolerului: el trebuie să poată genera semnalele necesare către DUT în toată gama de tensiuni de interes pentru dispozitivele noastre – de la 1,62 V până la 5,5 V. Pentru a putea seta frecvența după cum dorim, în schimb, este nevoie de oscilatoare externe.

Pentru gestionarea întreruperilor în cazul microcontrolerului Infineon este folosit un controler ITP care administrează ce acțiuni trebuie să ia UCP pentru a deservi individual întreruperile primite prin intermediul semnalelor de la periferice. Întreruperile sunt atribuite unor noduri de întrerupere, fiecare nod poate fi distins prin nodul lui de întrerupere, și are propriul registru specific de control. Nodurile sunt și ele împărțite în subnoduri pentru a permite o diferențiere mai fină între evenimentele de întrerupere ce au loc. Evenimentele sunt administrate folosind o schemă de

prioritizare care stabilește o structură ierarhică între sursele de întrerupere. Perifericele disponibile sunt atribuite permanent unor noduri de întreruperi, numerotate de la 0 până la maxim 47. Fiecare întrerupere locală poate avea până la patru subnoduri pentru a distinge între acțiunile ce au legătură cu perifericul respectiv – de exemplu, un subnod poate fi folosit pentru a înregistra dacă o listă de transmitere este goală, altul poate indica apariția unei erori de transmitere a datelor.

UART-ul acestui microcontroler oferă posibilitatea comunicației seriale conform standardelor ISO 7816-3. Modulul UART este de interes pentru noi deoarece acesta este modulul prin intermediul căruia putem comunica cu un smartcard.

Microcontrolerul este dispus cu un periferic de tip SSC. Lățimea datelor, direcția de deplasare, polaritatea semnalului de ceas, faza semnalului de ceas sunt parametri programabili. Microcontrolerul poate fi și *master* și *slave* folosind această interfață.

Modulul I^2C oferă comunicație serială și este potrivit pentru aplicații în care simplitatea în utilizare și costul redus sunt mai importante decât viteza cu care sunt transmise datele.

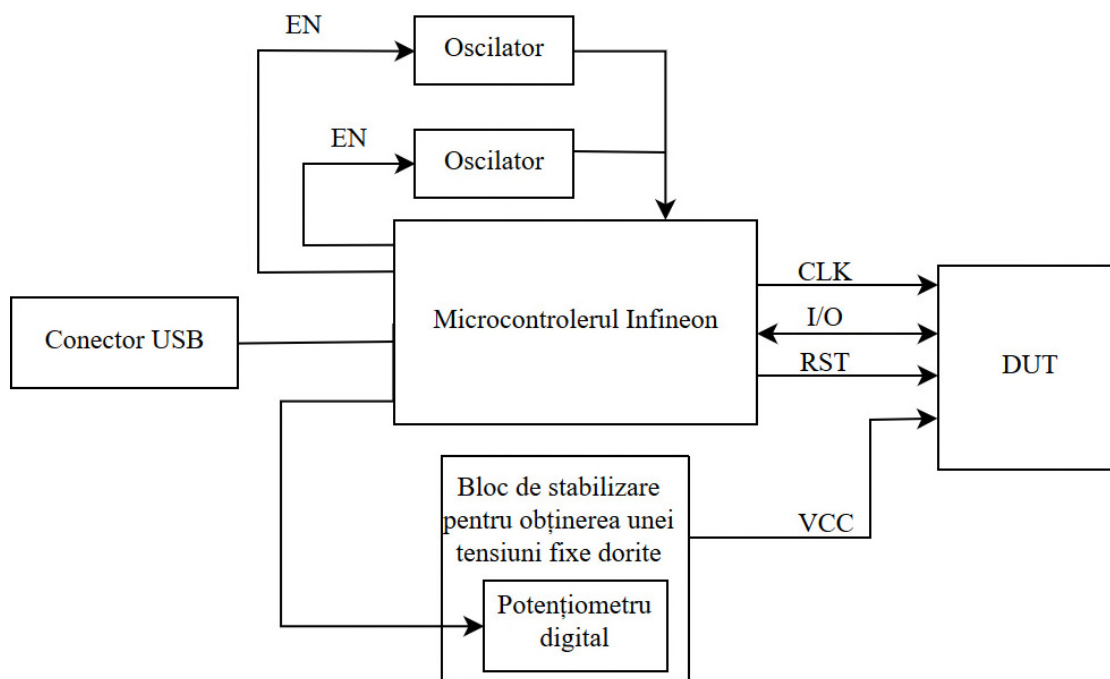


Figura 3.1. Schema bloc simplificată a soluției propuse

Perifericele acestui microcontroler au o deosebită importanță pentru această lucrare deoarece, după implementarea cu succes a primelor funcționalități de cititor pentru smartcarduri, ne propunem să implementăm funcții ce să asigure comunicație pe toate interfețele, pentru ca cititorul să poată fi folosit nu numai pentru comunicarea cu DUT-ul, ci și pentru testarea funcționalității diverselor interfețe ale DUT-ului.

Figura 3.1 reprezintă schema simplificată a conceptului inițial: cu microcontrolerul Infineon ca element central de control al sistemului. Controlerul primește comenzi de la utilizator prin USB, trimite comenzi prin protocolul SPI către un potențiomtru digital a cărui valoare determină tensiunea de la ieșirea unui bloc format dintr-un stabilizator de tensiune și elemente auxiliare,

trimite semnale de activare (EN – *Enable*) către două oscilatoare ce generează semnale de frecvențe diferite. Am ales folosirea a două oscilatoare deoarece acest lucru ne permite obținerea tuturor frecvențelor de care avem nevoie – cu ajutorul modulelor PLL din interiorul controlerului, divizând sau multiplicând frecvențele acestor semnale date de oscilatoare, putem obține toată gama de frecvențe de interes. În acest concept mai exista stabilizatoare pentru obținerea unor tensiuni pentru alimentarea diferitelor domenii de alimentare ale microcontrolerului Infineon, pentru alimentarea oscilatoarelor, LED-uri pentru semnalizarea alimentării.

3.4. Privire asupra folosirii unui microcontroler STM32

Ultima soluție propusă constă în folosirea unui microcontroler STM32H743 ce are la bază un Arm Cortex-M7 pe 32 de biți cu nucleu de dublă precizie FPU pentru generarea semnalelor de control și pentru comunicarea cu DUT-ul, integrat pe o placă de dezvoltare Nucleo.

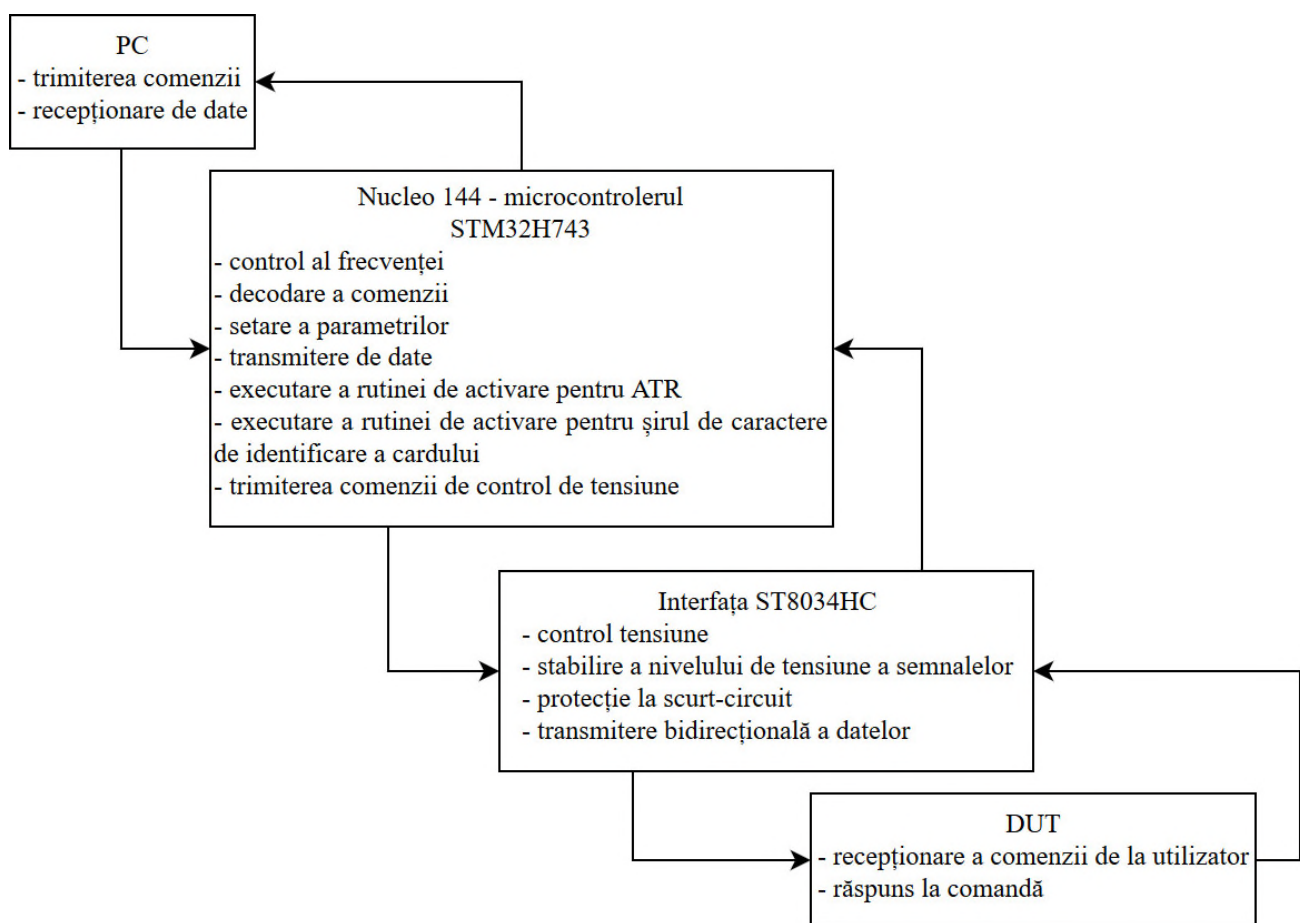


Figura 3.2. Diagrama conceptuală

Acest procesor – Cortex-M7 – deține cele mai înalte performanțe din familia de procesoare Cortex-M. Arhitectura procesorului este de tip Harvard ce se remarcă prin stocarea în memorie separată a instrucțiunilor și a datelor [12]. Microcontrolerul folosește pipeline superscalar pe șase nivele, cu *branch prediction* (predicție a ramurilor de decizie), are o unitate de calcul în virgulă mobilă (FPU), o memorie cache pentru instrucțiuni cu o capacitate de la 0 la 64 kB, asociativă în două sensuri, una pentru date, asociativă în patru direcții. Asemănător, avem la dispoziție și o memorie TCM pentru instrucțiuni și una pentru date – ambele de maxim 16 MB [8]. O memorie de acest tip reprezintă o regiune de memorie dedicată foarte apropiată de unitatea centrală de procesare. Principalul

beneficiu al ei este acela că unitatea de procesare poate să o acceseze la fiecare ciclu de ceas (timpul de acces în memorie este foarte mic) – de aceea, aici sunt stocate date și instrucțiuni critice din punct de vedere al performanței, precum rutine de întreruperi, date pentru sarcini realizate în timp real, structuri de control ale sistemului de operare. Întreruperile în acest controler pot fi NMI, în plus putem avea până la 240 de tipuri de întreruperi fizice, cu până la 256 de nivele de prioritate asociate. Memoria de program, memoria de date, registrele și porturile I/O sunt organizate în același spațiu liniar de adrese de 4 GB [8]. Octeții respectă, în memorie, convenția de cod *Little Endian*. Octetul căruia îi corespunde cel mai mic indice dintr-un cuvânt este considerat cel mai puțin semnificativ octet al acelui cuvânt, respectiv cel cu cel mai mare indice este considerat cel mai semnificativ octet.

Dispune de numeroase interfețe de comunicare printre care patru unități USART și patru UART ce pot funcționa până la o frecvență de 12.5 Mbit/s, șase unități SPI ce pot rula până la 100 Mbit/s, patru interfețe I^2C ce pot lucra până la 1 MHz, modul USB 2.0 ce operează la viteză maximă, modul Ethernet și multe altele [8].

De interes pentru noi este și modul de *debug* (depanare) posibil cu ajutorul unui port JTAG ce permite programatorului să transfere date în memoria internă non-volatilă a dispozitivului și utilizarea de către utilizator a maxim opt breakpointuri. Acest controler este integrat și el pe placa de dezvoltare Nucleo. Controlerul asigură o putere dinamică de 33 μ W/MHz.

Diagrama conceptuală ce stă la baza proiectului este prezentată în figura 3.2, unde putem observa principalele sarcini ale fiecărui element constructiv în parte.

3.5. Potențiale avantaje și dezavantaje ale soluțiilor propuse

Deși achiziționarea unui cititor de carduri disponibil pe piață, dezvoltat și testat de către o altă echipă, ne poate ușura foarte mult munca și este de departe cea mai rapidă soluție, prezintă dezavantajul costului ridicat și al dependenței constante de echipa dezvoltatorilor – pentru calibrare, pentru modificarea software-ului cu scopul de a introduce noi funcționalități ce nu au fost prevăzute, pentru rezolvarea problemelor, a defecțiunilor.

Inițial, pentru realizarea acestui proiect, am mers pe calea folosirii unui microcontroler Infineon: am început prin a dezvolta schema electrică ce conținea elementele necesare pentru a seta parametrii doriți, am conceput placa imprimată a circuitului. În tot acest demers, ne-am lovit de limitările pe care le presupune această abordare. Numărul de pini de tip GPIO s-a dovedit a fi insuficient pentru nevoile proiectului, interfețele ar fi trebuit să fie folosite în totalitate pentru gestionarea semnalelor necesare pentru ca blocurile funcționale să poată fi utilizate. Acest lucru ar fi anulat de la început posibilitatea de a proiecta un cititor de carduri capabil să comunice pe mai multe interfețe: SPI, I^2C , chiar și SWP, deoarece toate resursele ar fi fost epuizate pentru stabilirea comunicării pe UART – pinii GPIO nu ar mai fi putut fi folosiți cu funcțiile lor alternative, ce stabilesc comunicarea prin protocoalele precizate. Cel mai mare avantaj și, în același timp, dezavantaj al acestui concept este simplitatea: circuitul implementat este compact, funcționalitățile implementate pe microcontroler nu sunt numeroase, dar nu poate fi folosit decât pentru comunicarea prin UART.

În final, am ales să îmbinăm capacitatea pe care o poate avea un cititor de pe piață cu simplitatea unui controler prin alegerea unui dispozitiv STM32H743 ca element central de control al tuturor semnalelor de care avem nevoie pentru proiectarea unui cititor. Deși microcontrolerul Infineon este superior din punct de vedere al securității, am ajuns la concluzia că microcontrolerul STM32H743 este mai potrivit pentru aplicația propusă. Cum cititorul nu trebuie să îndeplinească condiții de securitate, ci să ofere flexibilitate și transparență în setarea parametrilor necesari funcționării cardurilor și a transmiterii datelor, capacitatea mai mică din acest punct de vedere a microcontrolerului STM32H743 nu aduce niciun dezavantaj. Cel mai mare avantaj pe care îl aduce această soluție este flexibilitatea: dacă vom avea nevoie de o anumită funcție de implementat, cu ajutorul acestui cititor putem dezvolta noi algoritmi ce pot rezolva noi sarcini, lucru care nu este atât de ușor de realizat cu un cititor de pe piață. Programul încărcat pe microcontroler poate fi actualizat după nevoile care apar pe parcurs.

4. Mediul de dezvoltare Nucleo

4.1. Microcontrolerul STM32H743

După cum am amintit și în subcapitolul 3.4, microcontrolerul STM32H743 dispune de numeroase interfețe și porturi de intrare/ieșire, și demonstrează, astfel, că este mult mai potrivit pentru aplicația propusă. Multitudinea interfețelor pe care le pune la dispoziție este importantă pentru noi deoarece, în viitor, dorim să proiectăm un cititor capabil să comunice cu DUT-ul pe toate interfețele – pentru a putea acoperi cât mai multe scenarii de test cu un singur dispozitiv. În continuare voi realiza o descriere amănunțită a proprietăților și interfețelor acestui microcontroler.

4.1.1. Memoria

Microcontrolerul dispune de o memorie Flash integrată de până la 2MB care poate fi folosită pentru a stoca programe și date. Memoria Flash este o memorie nonvolatilă cu acces aleator care, atunci când este necesar, poate fi ștearsă și reprogramată. Denumirea sa este o analogie pentru felul în care se poate face operația de ștergere, dintr-o singură acțiune, mulțumită felului ei de organizare.

Memoria Flash a dispozitivului STM32H743 este divizată în blocuri independente, fiecare fiind organizat după cum urmează:

- Un bloc de memorie de 1 MB ce poate fi folosit de către utilizator, bloc ce conține opt sectoare a câte 128 kB;
- 128 kB de memorie Flash de sistem, memorie din care dispozitivul își va lua datele necesare inițializării;
- 2 kB pentru configurări setate de către utilizator.

Pe lângă memoria Flash, microcontrolerul dispune și de memorie integrată de tip SRAM, memorie ce utilizează celule de tip *flip-flop* pentru stocarea fiecărui bit [8].

Pe parcursul vieții produsului, din cauza unor evenimente exterioare precum radiațiile, pot apărea biți invalizi în memorie. Aceștia pot fi detectați și corectați cu ajutorul unui sistem ECC. Implementarea unor algoritmi ECC asigură integritatea datelor la nivel software. Mecanismul folosit are la bază un algoritm numit SECDED – ce poate detecta două erori și corecta o singură eroare [8].

4.1.2. Modulul Reset and Clock Control (RCC)

Modulul RCC administrează generarea tuturor semnalelor de ceas, precum și tăierea semnalelor de ceas din diferite zone (pentru a diminua consumul) și controlul resetărilor sistemului și perifericelor. Oferă o flexibilitate foarte mare de alegere între sursele de ceas, iar, în plus, unele periferice pot lucra cu domenii de ceas diferite [8].

Acest modul poate avea drept intrări următoarele surse de ceas: oscilatoare interne – de 64 MHz, 48 MHz, 4 MHz sau 32 kHz, sau oscilatoare externe – 4 ÷ 50 MHz (generat de la o sursă externă) sau 4 ÷ 48 MHz (generat cu ajutorul unui cristal/rezonator ceramic) [8].

Modulul pune la dispoziție trei PLL-uri, unul pentru ceasul de sistem și două pentru semnalele de ceas din nucleu. Un PLL este un circuit electronic oscilator ce își ajustează frecvența pentru a se potrivi cu aceea a unui semnal de intrare, folosit drept componentă de control a frecvenței.

4.1.3. Pinii GPIO

Un GPIO – *General Purpose Input/Output* – sau o intrare/ieșire de uz general este un pin de semnal digital ce nu are o funcție predefinită alocată și care aparține unui circuit integrat, comportamentul lui – fie că este configurat drept intrare sau ieșire – este controlabil de către utilizator în timp real.

Fiecare pin de tip GPIO al controlerului STM32H743 poate fi configurat prin software drept ieșire (*push-pull* sau *open-drain*, cu sau fără *pull-up* sau *pull-down*), ca intrare (variabilă, cu sau fără *pull-up* sau *pull-down*), sau poate primi o funcție alternativă și să corespundă semnalului de intrare sau de ieșire a unui periferic [8]. Toți pinii au o capacitate mare în curent și un comportament bun chiar și în prezența zgomotului intern, al emisiei electromagnetice și din punct de vedere al consumului de putere.

4.1.4. Gestionarea întreruperilor

Există două tipuri de întreruperi disponibile într-un procesor de tip ARM: cauzate de evenimente externe, de la periferice hardware, sau generate cu ajutorul instrucțiunilor SWI. La apariția unui eveniment de la o sursă hardware care are întreruperea activă, un fanion de stare asociat întreruperii (care poate fi doar citit de către utilizator) este setat, iar procesorul se „pregătește” pentru execuția rutinei de întrerupere.

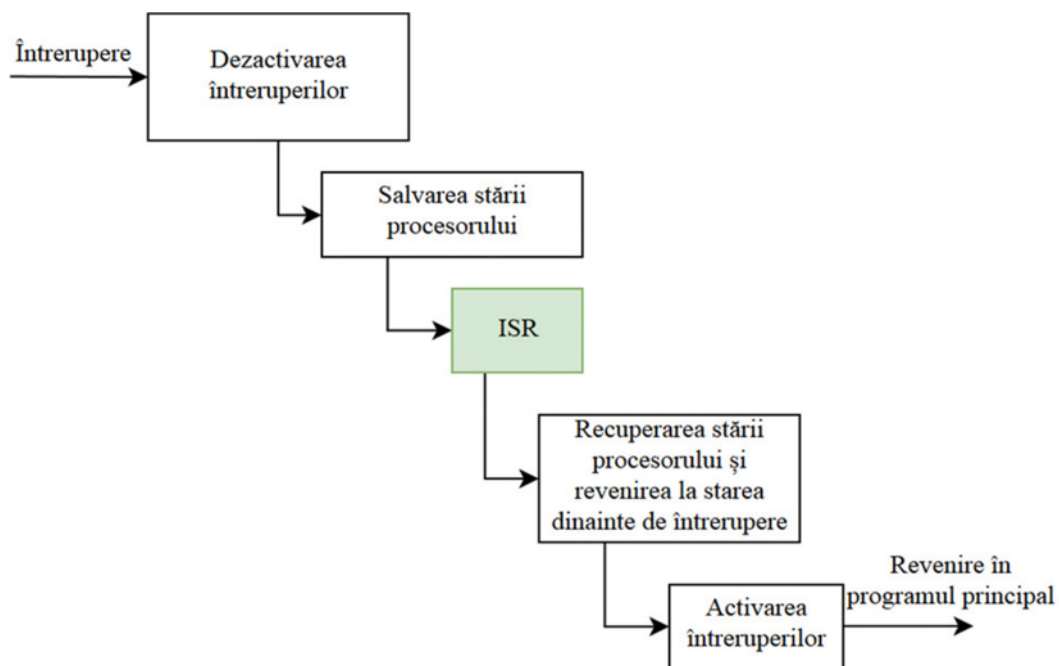


Figura 4.1. Gestionarea unei întreruperi pentru un model non-nested

Procesorul poate deservi întreruperile în două feluri: cu un model *non-nested* de gestionare a întreruperilor, sau cu unul *nested*. Cel mai simplu mod este cel *non-nested*, în care, după apariția unei întreruperi, întreruperile sunt dezactivate până la revenirea în programul principal.

Algoritmul de pregătire pentru a lansa în execuție rutina de tratare a întreruperii în cazul unui model de întreruperi *non-nested* este următorul:

5. UCP salvează conținutul registrelor în stivă – salvează, astfel, starea procesorului dinainte de a executa rutina de întrerupere;
6. Dezactivează/maschează întreruperile care vor veni în timpul deservirii acestei întreruperi;
7. Aduce vectorul de întreruperi pentru întreruperea cu cea mai mare prioritate care se află în așteptare;
8. Lansează în execuție rutina de întrerupere ce pornește de la adresa adusă din vectorul de întreruperi.

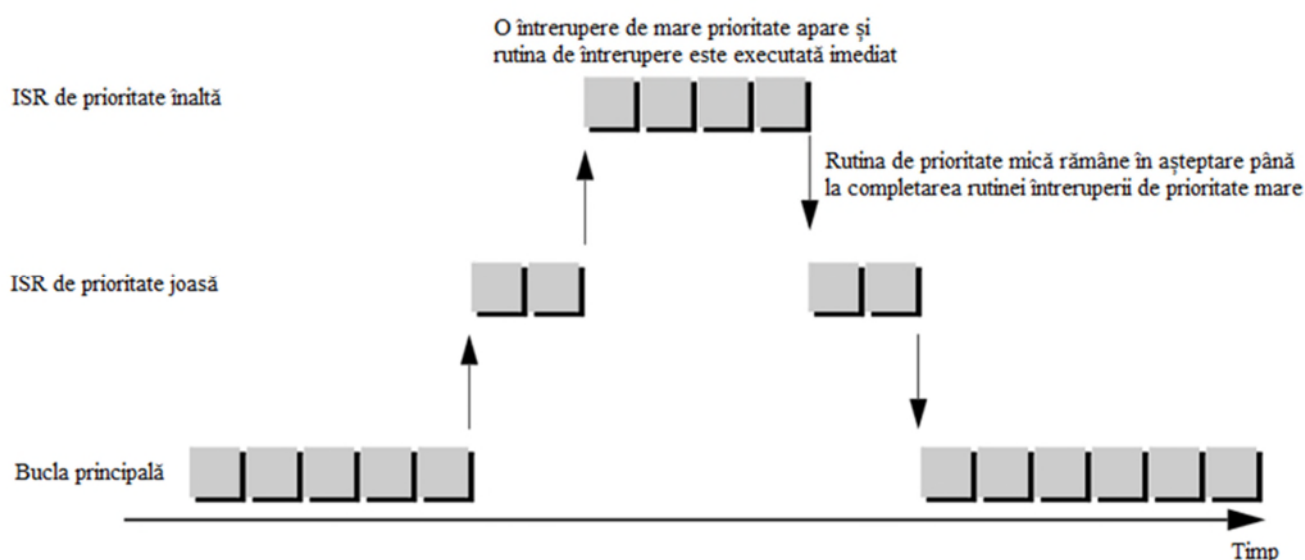


Figura 4.2. Gestionarea întreruperilor în timp cu ajutorul unui model nested [5]

Întreaga secvență de acțiuni pentru gestionarea și revenirea la starea dinainte de întrerupere este descrisă în figura 4.1. Astfel, rutina de întrerupere de tip *non-nested* servește întreruperile individuale în mod secvențial. Un dezavantaj important pentru acest model este latența mare, dar modelul este relativ ușor de implementat și de depanat. Nu este potrivit pentru sistemele integrate complexe.

Într-un sistem de întreruperi de tip *nested* (derivat din substantivul *nest* – cuib, „cuibărit”), o întrerupere este permisă oricând și oriunde, chiar și în timpul unei rutine de deservire a unei alte întreruperi – ISR (*Interrupt Service Routine*) sau RTI (Rutina de Tratare a Întreruperii). Totuși, numai întreruperea cu un nivel mai mare de prioritate va fi deservită. Întreruperea cu a doua cea mai înaltă prioritate va fi deservită abia după ce cea mai mare prioritate a fost rezolvată. În figura 4.2 este prezentat felul în care execuția secvenței principale de cod este întreruptă pentru a executa o rutină corespunzătoare unei întreruperi de prioritate mică, iar, mai apoi, această rutină este întreruptă pentru execuția unei rutine ce corespunde unei întreruperi cu prioritate mai mare, așa cum se desfășoară aceste acțiuni în timp.

Regulile unui astfel de sistem de întreruperi sunt următoarele:

- Toate întreruperile trebuie să aibă atribuit un nivel de prioritate;
- După inițializare, oricărei întreruperi îi este permis să aibă loc oricând și oriunde;
- Dacă o rutină de întrerupere de prioritate joasă este întreruptă de o rutină de întrerupere de prioritate mai mare, atunci cea de prioritate mai mare este executată;
- Dacă o rutină de întrerupere de prioritate mare este întreruptă de o rutină de întrerupere de prioritate mică, atunci cea de prioritate mai mare își va continua execuția;
- Întreruperile ce au același nivel de prioritate vor fi executate în ordinea cronologică a apariției lor.

Modulul NVIC este cel ce se ocupă de gestionarea întreruperilor și excepțiilor, având o latență mult mai mică decât un model *non-nested*. El poate fi utilizat pentru activarea sau dezactivarea unei anumite întreruperi și atribuirea unui nivel de prioritate. Este specific microcontrolerelor ARM.

Microcontrolerul STM32H743 integrează un modul NVIC capabil să gestioneze 16 nivele de prioritate și până la 150 de întreruperi mascabile [13].

4.1.5. Modulul USART

UART nu este un protocol de comunicare precum SPI sau I^2C – este un circuit fizic într-un microcontroler sau un circuit integrat de sine stătător. Principalul obiectiv al UART-ului este să transmită și să recepționeze date seriale. Astfel, prezintă marele avantaj de a avea nevoie doar de două semnale: RX (de recepționare) și TX (de transmitere). Pinii corespunzători acestor semnale trebuie conectați conform figurii 4.5.1, deoarece modulul UART1 din figură va fi receptor pentru datele transmise de UART2, și transmițător pentru datele trimise către UART2, și viceversa.

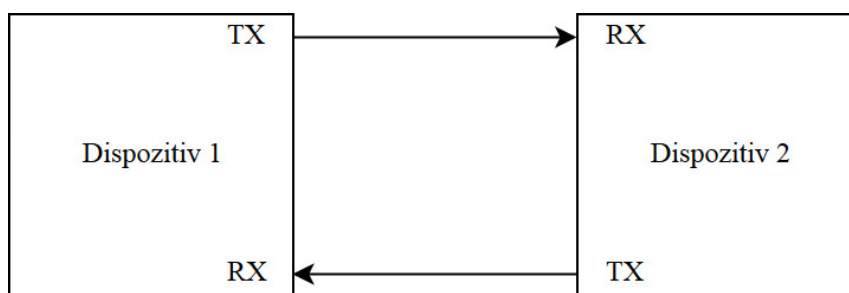


Figura 4.3. Modelul interconectării a două module UART

În comunicarea UART, două dispozitive comunică direct unul cu celălalt: cel care transmite convertește datele paralele primite de la dispozitivul de control – microcontrolerul – într-o formă serială, transmite datele în această formă către dispozitivul UART receptor, care ca converti datele seriale înapoi în date paralele pentru dispozitivul receptor. UART-ul transmite datele asincron, ceea ce înseamnă că nu există niciun semnal de ceas adițional care să sincronizeze datele transmise și să dicteze cum sunt eșantionate datele la recepționare. În schimb, UART-ul transmițător adaugă biți de

start și de stop în pachetul de date transferat. Acești biți definesc începutul și sfârșitul pachetului de date pentru ca UART-u receptor să știe când să înceapă citirea biților [11].

Când detectează un bit de start, UART-ul receptor începe să citească biții primiți cu o frecvență specifică numită *baud rate*. Aceasta este o măsură a vitezei transferului datelor, exprimată în biți pe secundă (bps). Ambele UART-uri – de transmitere și recepție – trebuie să funcționeze la aproximativ același baud rate.

După transmiterea datelor este transmis, în plus, și un bit de paritate, care indică alterarea datelor pe parcursul comunicării. Biții pot fi alterați din cauza radiației electromagnetice, a unor parametri baud rate setați diferit, sau a unor transferuri de date efectuate la mare distanță. După ce primește o fereastră de date, UART-ul numără biții cu valoarea 1 și verifică dacă numărul total este par sau impar [11]. Dacă bitul de paritate are valoarea 0, numărul de biți de 1 ar trebui să fie par, respectiv, dacă bitul de paritate are valoarea 1, numărul de biți de 1 recepționați ar trebui să fie impar. Dacă bitul de paritate se potrivește cu rezultatul numărării, este probabil ca datele să se fi transmis corect. Dacă nu, în schimb, UART-ul va ști că biții au fost alterați [11].

Un modul de tip USART oferă în plus opțiunea de a funcționa în modul sincron. Pentru comunicarea sincronă este necesar în plus semnalul de ceas. Microcontrolerul STM32H743 are integrate patru module USART și patru UART. Modulele USART pun la dispoziție, în plus, modul *Smartcard*, conform cu standardul ISO 7816, un mod de foarte mare interes pentru această lucrare. Toate modulele USART au domeniul de ceas independent de UCP.

Interfața UART este cea folosită pentru conectarea microcontrolerului smartcard la cititor. În cazul unui microcontroler de securitate, așa cum este un smartcard, această interfață prezintă o particularitate importantă: folosește doar trei semnale, I/O, RST și CLK. Semnalul I/O este folosit și pentru RX, și pentru TX, comunicația fiind de tip *half-duplex*. Semnalul de ceas este generat de către cititor și este folosit pentru a stabili frecvența de comunicare a dispozitivului smartcard. Microcontrolerul STM32H743 oferă posibilitatea configurării anumitor interfețe USART cu care este dotat în modul „Smartcard”, mod ce permite folosirea unor pini pentru I/O și pentru semnalul de ceas, CLK, în loc de a avea pini RX și TX, precum în UART-ul clasic.

4.1.6. Modulul SPI

SPI este o interfață utilizată în general pentru a transmite date între microcontroler și periferice mici precum registre de deplasare, senzori, carduri. Folosește semnal de ceas și două linii de date, și o linie pentru selectarea dispozitivului cu care vrem să comunice microcontrolerul. Interfața poate fi configurată fie ca *master*, fie ca *slave* și este capabilă să opereze și în modurile multi-master sau multi-slave [10].

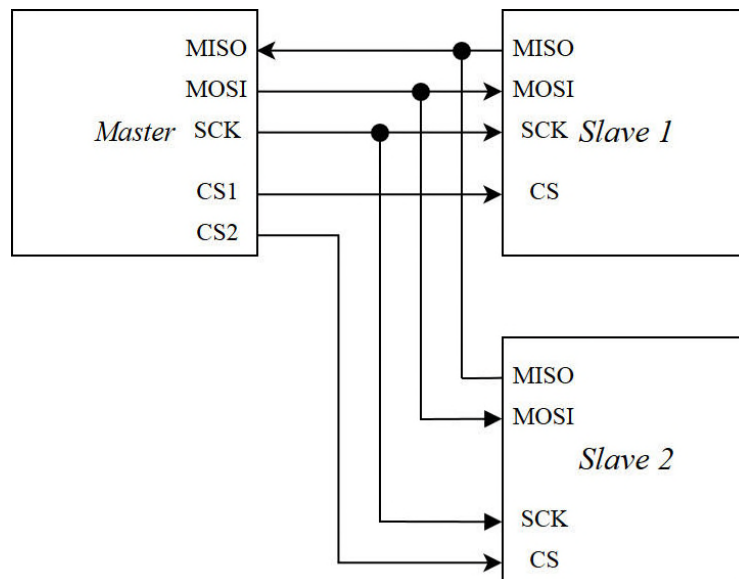


Figura 4.4. Modelul interconectării dispozitivelor pentru comunicarea prin SPI

Patru pini sunt dedicați comunicației prin protocolul SPI [3]:

- MISO: Master In / Slave Out – acest pin este folosit pentru a transmite datele în modul *slave* și pentru a recepționa datele în modul *master*;
- MOSI: Master Out / Slave In – pinul este folosit pentru a transmite datele în modul *master* și pentru a le recepționa în modul *slave*;
- SCK: semnalul de ceas serial de ieșire pentru *master* și de intrare pentru *slave*;
- SS sau CS: Slave Select sau Chip Select, pin ce poate fi folosit în mai multe feluri în funcție de setările folosite: selectează un dispozitiv de tip *slave* pentru comunicare, sincronizează fereastra de date sau detectează un conflict între mai multe dispozitive *master*.

În figura 4.4 ste prezentată interconectarea dintre un dispozitiv de tip master și două dispozitive slave, cu pinii configurați corespunzător. Microcontrolerul folosit dispune de până la șase interfețe SPI (SPI1, SPI2, SPI3, SPI4, SPI5 și SPI6) ce permit comunicarea cu o frecvență de până la 150 Mbit/s.

4.1.7. Modulul I^2C

Inter-Integrated Circuit (I^2C) este un protocol de comunicare construit pentru a permite mai multor circuite integrate digitale de tip *slave* să comunice cu unul sau mai multe controlere de tip *master*. Are nevoie doar de două linii de semnal pentru a face schimb de informație.

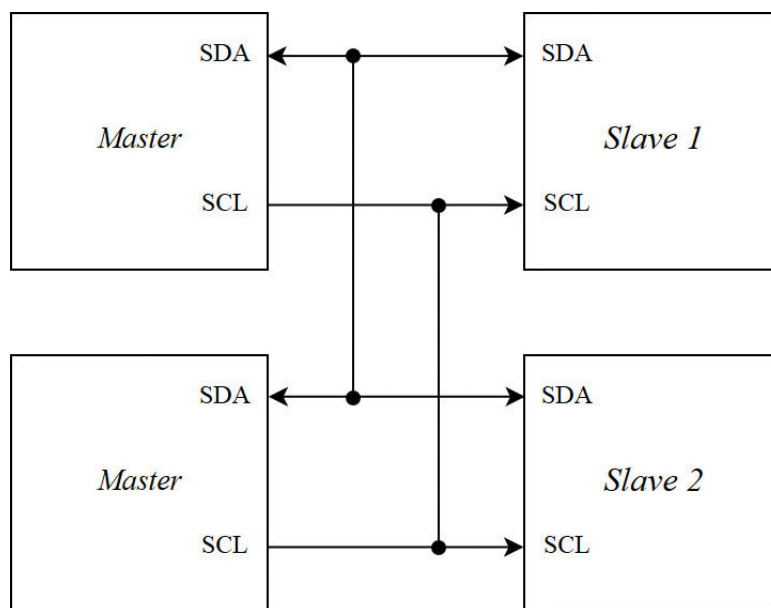


Figura 4.5. Modelul interconectării dispozitivelor pentru comunicarea prin I²C

Acest protocol nu are dezavantajele prezentate de UART: comunicarea asincronă, deci necesitatea unei înțelegeri între transmițător și receptor cu privire la rata de transmisie, începutul și sfârșitul transmisiei datelor, sau pe cele ale protocolului SPI: necesitatea unui număr mare de pini pentru transmiterea datelor [14]. În plus, oferă posibilitatea folosirii mai multor dispozitive de tip *master*. În figura 4.5 este reprezentat un model de interconectare a unor module ce comunică prin acest protocol.

Microcontrolerul STM32H743 oferă posibilitatea folosirii acestui protocol atât în modul *master* cât și în modul *slave*, cu o capacitate *multimaster*, în trei moduri de funcționare: modul standard (cu o frecvență de lucru de până la 100 kHz), modul rapid (până la 400 kHz) și cel rapid-plus (până la 1 MHz).

4.1.8. Modulul SWP

Microcontrolerul STM32H743 poate fi folosit și drept *master* în comunicarea prin SWP – prin intermediul SWPMI - *Single Wire Protocol Master Interface*. Acest protocol presupune transmiterea a două semnale pe un același fir: un semnal de la master către slave, transmis printr-o modulare digitală în domeniul tensiune, și un semnal de la slave către master, transmis prin modulare digitală în curent.

4.2. Arhitectura plăcii Nucleo

Plăcile de dezvoltare STM32 Nucleo-144 oferă o modalitate flexibilă pentru implementarea conceptelor noi și construirea prototipurilor, prin a pune la dispoziție performanța și consumul redus ale microcontrolerului STM32. Programarea microcontrolerului STM32H743 nu necesită folosirea niciunui modul programator în plus, deoarece modulul ST-LINK/V2-1 este deja integrat în conceptul plăcii. Modulul este un circuit construit pentru programarea și depanarea microcontrolerelor STM32. În figura 4.6 este prezentată diagrama bloc a plăcii de dezvoltare, într-un model simplificat.

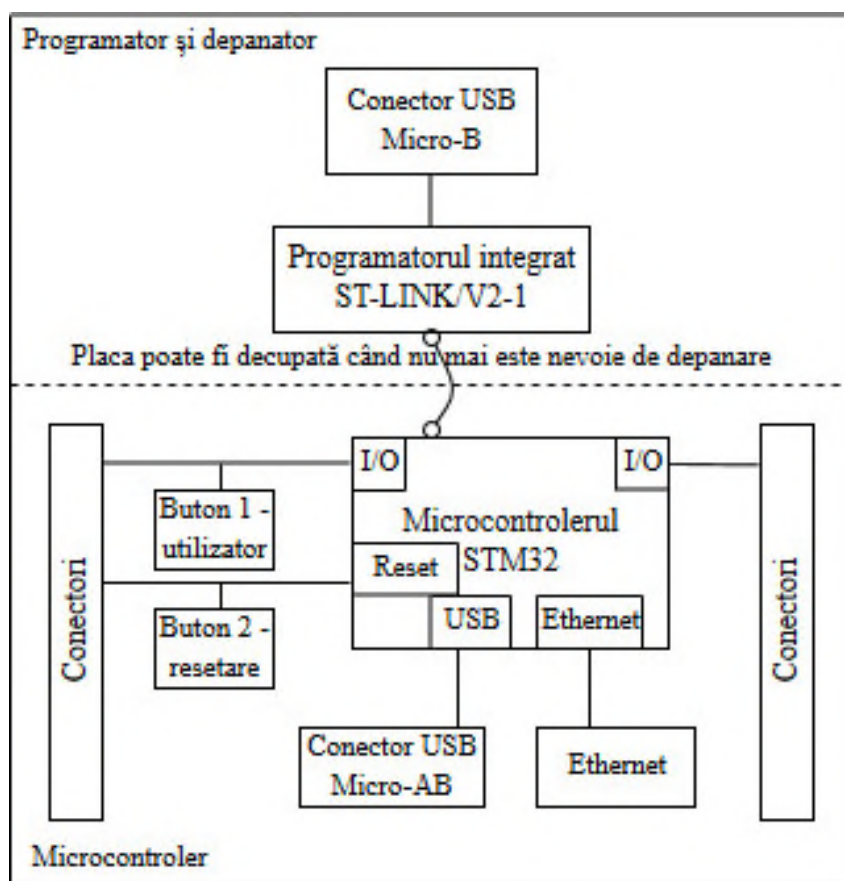


Figura 4.6. Diagrama bloc a plăcii de dezvoltare Nucleo – model simplificat

Placa de dezvoltare conține o multitudine de conectori ce fac legătura cu majoritatea pinilor, un conector USB, un conector Ethernet, LED-uri pentru utilizator, pini de selecție și de măsurare. Placa poate fi alimentată cu ajutorul conectorului USB al modului ST-LINK/V2-1 sau de la o sursă de alimentare externă. Oferă o ieșire pentru alimentare la 5 V și una pentru alimentare la 3,3 V, pe care le vom folosi pentru alimentarea plăcii proiectate de noi, ca interfață între microcontrolerul STM și DUT. Pentru configurarea frecvenței semnalului de ceas, microcontrolerul STM32H743 primește un semnal de ceas extern de la modulul programator ST-LINK, cu frecvența de 8 MHz. Pentru comunicarea cu modulul programator este folosită interfața USART 3.

4.3. Interfața ST8034HC

Dispozitivul ST8034HC este o interfață analogică de preț redus pentru comunicarea sincronă și asincronă cu smartcardurile ce operează la 5 V, 3 V și 1,8 V.

Interfața are nevoie de un semnal CS pentru activare. Dispozitivul de interfață va trece semnalele de interes printr-un schimbător de nivel (*level shifter*) pentru a le aduce la nivelul de tensiune dorit. Pentru ca nivelul de tensiune să fie setat la o anumită valoare, sunt necesare două semnale de control transmise de către microcontrolerul STM32H743, VCCSEL1 și VCCSEL2, conform tabelului 4.1.

Tabelul 4.1. Configurarea tensiunii de alimentare/nivelului de tensiune pentru semnalele transmise către DUT [6]

VCC_SEL1	VCC_SEL2	VCC – către DUT
„0”	X – <i>don't care</i>	1,8 V
„1”	„1”	5 V
„1”	„0”	3 V

Interfața va primi semnalul de ceas generat de microcontrolerul STM32H743 și va aduce toate semnalele la același nivel, cel dorit de utilizator. Totodată, interfața pune la dispoziție un pin de detecție a prezenței cardului în cititor – atunci când cardul este inserat în conectorul special pentru carduri un contact este adus mecanic la punctul de masă, de potențial „0”. Atunci, pinul va primi valoarea „0” logic, iar interfața va detecta inserția cardului.

Interfața are mai multe domenii de alimentare: alimentarea interfeței în sine, VDD(INTF), alimentarea pentru LDO, VDDP, și alimentarea de control logică, VDD. Oferă și o detecție de scurt-circuit între contactele dinspre card: atunci când se produce un scurt între VCC și oricare alt contact, interfața se dezactivează ca urmare a detecției acestei erori.

Semnalele ce trebuie să ajungă la DUT vor trece prin această interfață cu ajutorul pinilor de intrare/ieșire dinspre microcontroler: RSTIN pentru semnalul de resetare, I/OUC pentru semnalul I/O, XTAL1 (în cazul nostru, deoarece vom avea un semnal de ceas extern care va fi generat de microcontroler) pentru semnalul de ceas, eventual AUXUC1 și AUXUC2 (pentru contactele auxiliare ale cardului), și cu ajutorul pinilor de intrare/ieșire dinspre DUT: VCC, RST, CLK, I/O, AUX1 și AUX2.

Interfața pune la dispoziție și posibilitatea divizării frecvenței de ceas, care nu este de interes în această lucrare deoarece semnalul de ceas va fi generat și controlat numai de controlerul STM32H743.

5. Ansamblul fizic propus pentru dezvoltarea proiectului

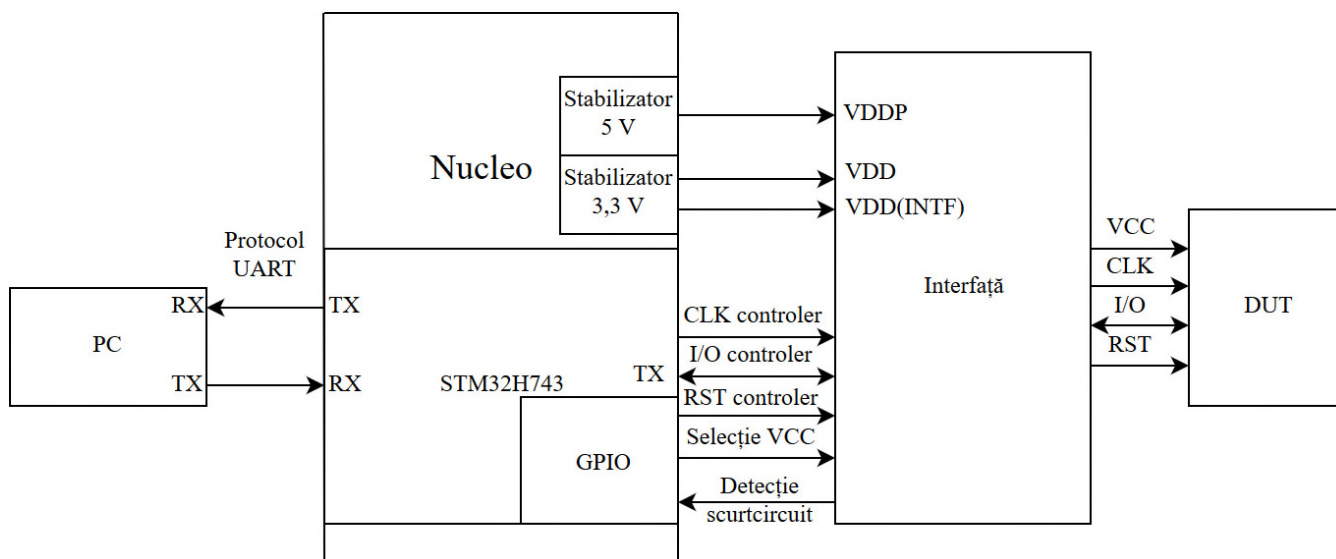


Figura 5.1. Diagrama generală a ansamblului construit

Pentru implementarea unui cititor de carduri am ales soluția folosirii plăcii de dezvoltare Nucleo, cu microcontrolerul STM32H743 integrat. Pentru a crea interfața cu DUT-ul, am ales folosirea unui dispozitiv ce funcționează ca o interfață analogică între microcontrolerul principal - STM32H743 – și dispozitivul ce trebuie testat. În figura 5.1 este descris conceptul hardware ce stă la baza acestui proiect, și felul în care datele sunt transmise până la DUT.

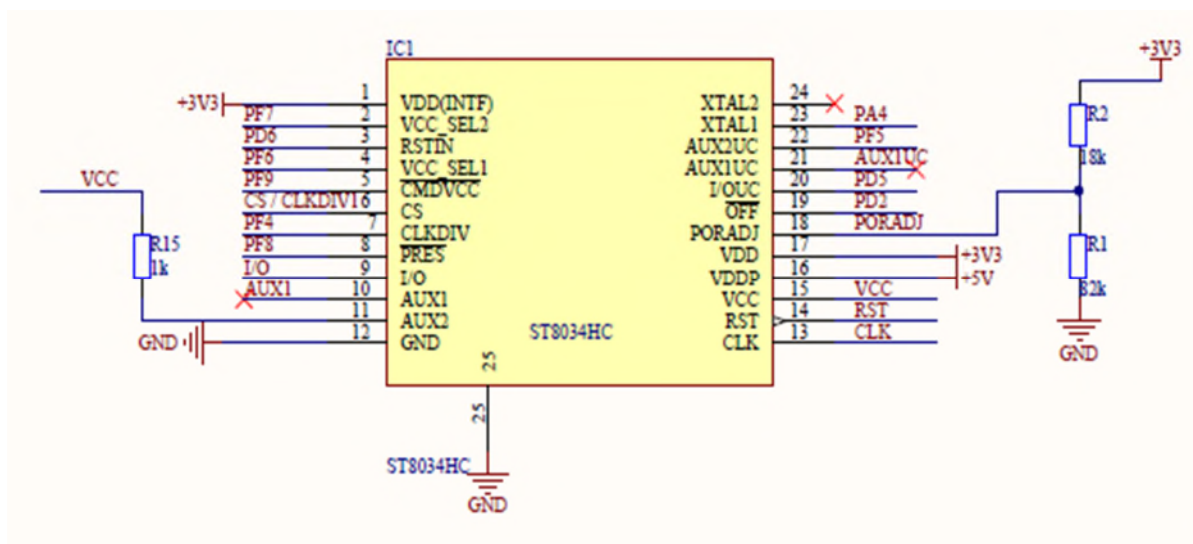


Figura 5.2. Interconectarea interfeței cu pinii corespunzători ai microcontrolerului

Datele sunt transmise de la PC la microcontrolerul principal folosind interfața UART, pentru a fi, mai apoi, interpretate de către microcontrolerul STM32H743. Interfața este alimentată cu ajutorul stabilizatoarelor de pe placa de dezvoltare Nucleo, conform specificațiilor. Semnalele CLK, I/O, RST, toate la nivelul de 3,3 V, trec prin interfață pentru a ajunge la nivelul de tensiune dorit către DUT. Linia I/O este bidirecțională, nivelul dinspre controlerul STM fiind menținut la 3,3 V.

Conform cu trăsăturile smartcardurilor, după cum am stabilit în capitolul 1, comunicația dintre cititor și smartcard se efectuează *half-duplex*, cu ajutorul unei singure linii, I/O (TX), în loc de două, cum este caracteristic comunicării prin UART. Selecția valorii tensiunii de alimentare pentru DUT este efectuată cu ajutorul unor pini de tip GPIO ai controlerului STM. Semnalele de ieșire din interfață vor trece, mai departe, către DUT, pentru stabilirea comunicării.

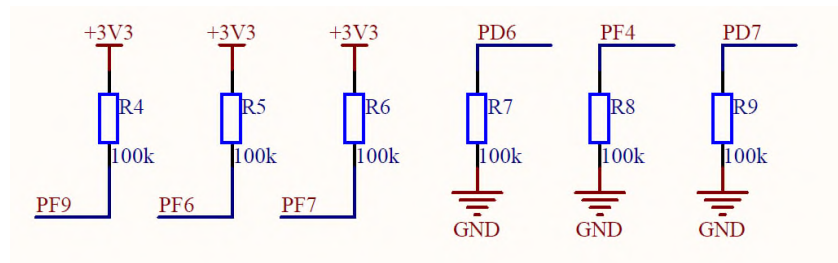


Figura 5.3. Rezistențele de pull-up și de pull-down

Pentru a putea integra interfața analogică în proiectul nostru, am proiectat o schemă electrică ce conține interfața, condensatoare pentru filtrarea zgomotului de pe liniile de alimentare, rezistoare pentru *pull-up* și *pull-down* în situațiile în care este necesar, un conector special pentru introducerea smartcardului. Conectorii folosiți în această schemă vor fi identici cu cei folosiți în schema plăcii Nucleo, iar placa va avea aceeași dimensiune pentru a putea fi conectată pe placa Nucleo cu ajutorul conectorilor menționați. Întreg proiectul este realizat în programul Altium Designer pentru construirea schemelor și desenarea cablajului imprimat.

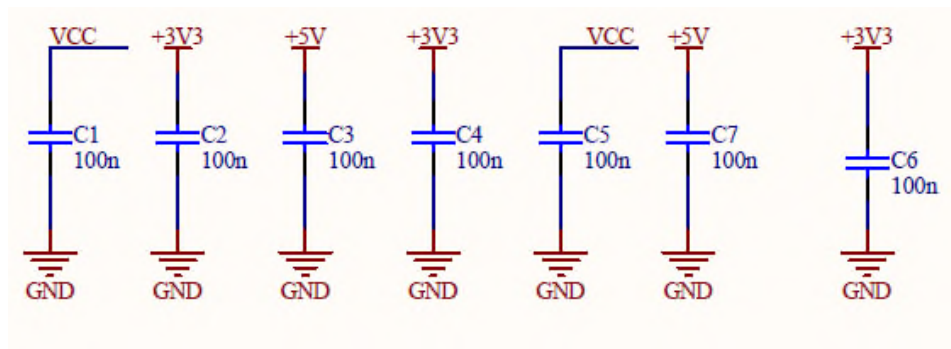


Figura 5.4. Condensatoarele de filtrare

Inițial, am pornit de la o placă de dezvoltare ce are ca element central interfața analogică, iar mai apoi am conceput o schemă electrică mai potrivită pentru aplicația noastră. În figura 5.2 se poate observa simbolul folosit pentru interfața analogică și conectivitatea acesteia cu pinii microcontrolerului principal. De exemplu, pinul VCC_SEL2 (pinul cu numărul „2”) al interfeței este conectat cu PF7 (Portul GPIO „F”, pinul 7). În figura 5.3 sunt prezentate rezistențele de *pull-up* și de *pull-down* folosite, așa cum este prevăzut și în specificație. În figura 5.4 sunt prezentate condensatoarele de filtrare pentru alimentările necesare.

Am realizat schema astfel încât să poată fi adaptată, cu ajutorul unui șunt, și pentru folosirea interfeței analogice ST8034HN, care are un singur pin diferit de ST8034HC – în loc de CS, permite folosirea a două semnale pentru divizarea frecvenței semnalului de ceas.

Deoarece pinii AUX1 și AUX2 nu sunt utilizați pentru cardurile noastre, am folosit unul dintre semnale pentru a semnaliza către microcontrolerul STM dacă s-a petrecut un scurt-circuit. După cum am menționat în subcapitolul 4.3, dacă dispozitivul de tip interfață detectează un scurt-circuit între VCC și un alt contact al cardului, va produce o dezactivare cauzată de această eroare, ca protecție. Astfel, folosind această informație, vom lega pinul de ieșire din interfață – AUX2, folosind o rezistență pentru *pull-up*, la contactul VCC dinspre card. Astfel, vom evita detecția unui scurt-circuit și vom putea, în același timp, măsurând valoarea pinului de intrare – AUXUC2, să determinăm în cadrul aplicației noastre dacă s-a produs un scurt-circuit atunci când nu mai primim niciun răspuns de la card. Divizorul rezistiv format din R1 și R2 este conectat astfel pentru a oferi o tensiune de prag de referință, așa cum este menționat în specificațiile interfeței analogice.

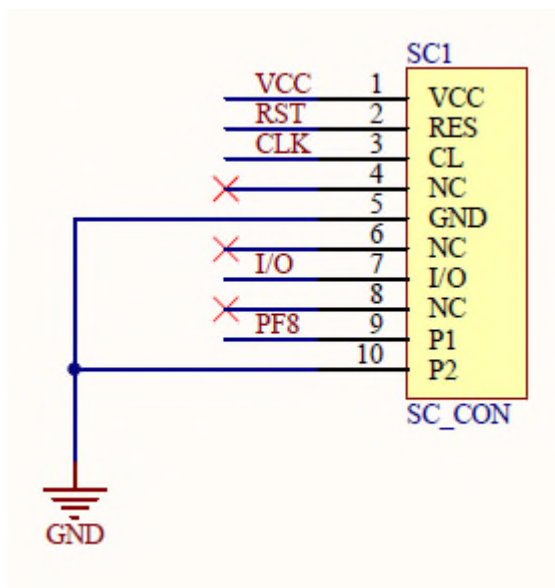


Figura 5.5. Conectorul pentru smartcard

Conectorul pentru smartcard este prezentat în figura 5.5, iar LED-urile pentru semnalizarea anumitor potențiale sau evenimente sunt prezentate în figura 5.6. Conectorul pentru smartcard este dispus și cu un mijloc de a detecta dacă smartcardul este introdus sau nu în el. Acest lucru este realizat cu ajutorul unui contact îmîns mecanic către punctul de masă atunci când cardul este introdus în cititor. Am legat acest pin la pinul $\overline{PRES}$ al interfeței analogice, care se va dezactiva dacă smartcardul nu este inserat în cititor, dar și la un pin al microcontrolerului STM32H743, pentru a putea verifica prezența cardului și, eventual, pentru a o semnala mai departe.

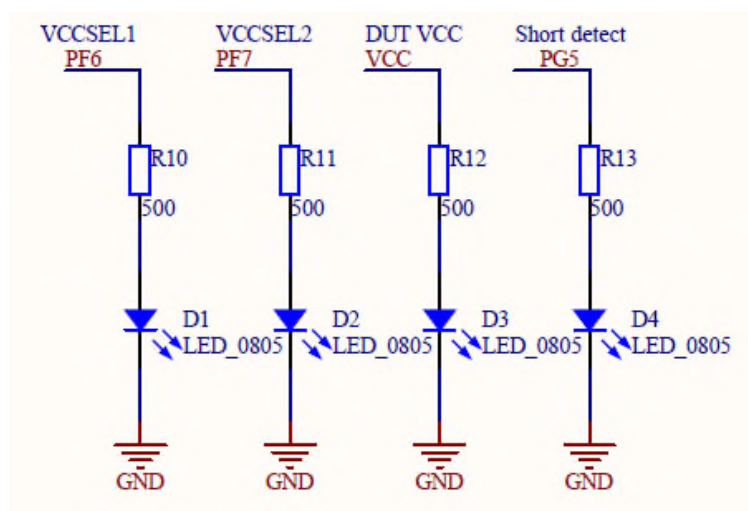


Figura 5.6. LED-uri pentru semnalizare

Deoarece proiectul se află la primul prototip, iar depanarea este foarte importantă în acest punct al dezvoltării, am integrat în schema electrică mulți pini de test pentru monitorizarea semnalelor ce ajuns la DUT și nu numai.

Schema electrică generală, completă, împreună cu PCB-ul, sunt prezentate în anexa A.

Merge bagat tabelul cu clk div din interfata pe undeva

6. Programarea microcontrolerului STM32H743

6.1. Conceptul programului

Pentru început, am realizat o schemă conceptuală a algoritmului ce urmează a fi implementat. Diagrama este prezentată în figura 6.1.

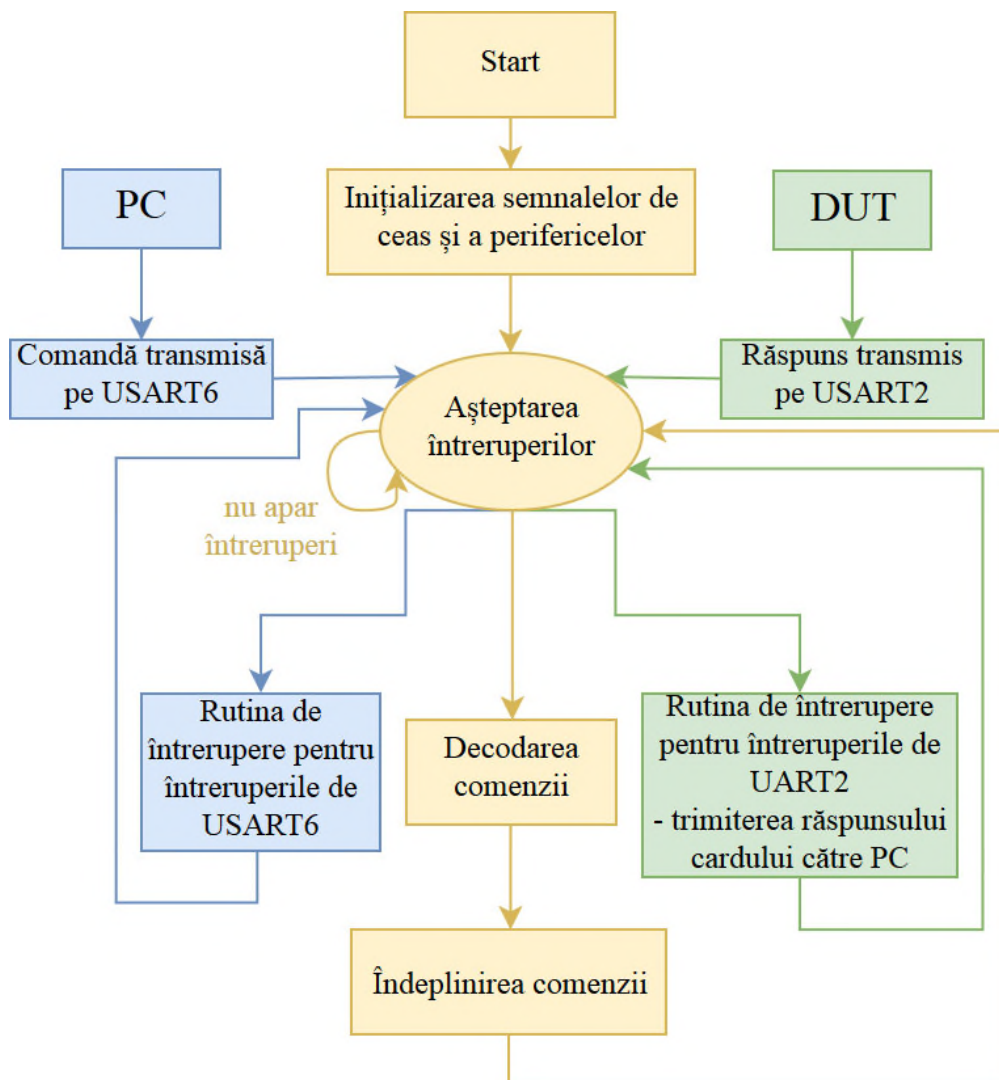


Figura 6.1. Diagrama conceptuală a programului

Pentru început, în program, este inițializat HAL-ul, este configurat sistemul de ceas, sunt inițializate toate perifericele cu configurările dorite: GPIO, USART2 în modul de funcționare „Smartcard”, USART6 și sunt activate întreruperile. Urmează apoi bucla infinită, care marchează faptul că microcontrolerul așteaptă continuu pentru primirea unei întreruperi: o comandă primită prin USART6 de la PC, sau un răspuns anume primit prin USART2 de la DUT. Recepționarea acestor mesaje se efectuează prin întreruperi configurate cu același nivel de prioritate.

Atunci când se modifică valoarea logică a bitului de întrerupere de recepționare pe USART6, programul va sări și el către rutina de întrerupere pentru interfața USART6. Va pune datele comenzii într-un vector de stocare, pentru a fi mai apoi interpretate în bucla principală.

Dacă întreruperea semnalizată are loc pe interfața USART2, programul va sări la rutina de întrerupere corespunzătoare, va pune datele primite de la DUT într-un vector și le va transmite prin USART6 către PC. Microcontrolerul, ca orice cititor, nu interpretează datele de la card, ci doar stabilește comunicarea și trimite datele mai departe pentru a fi interpretate.

6.2. Comunicarea cu cardul și structura comenzilor

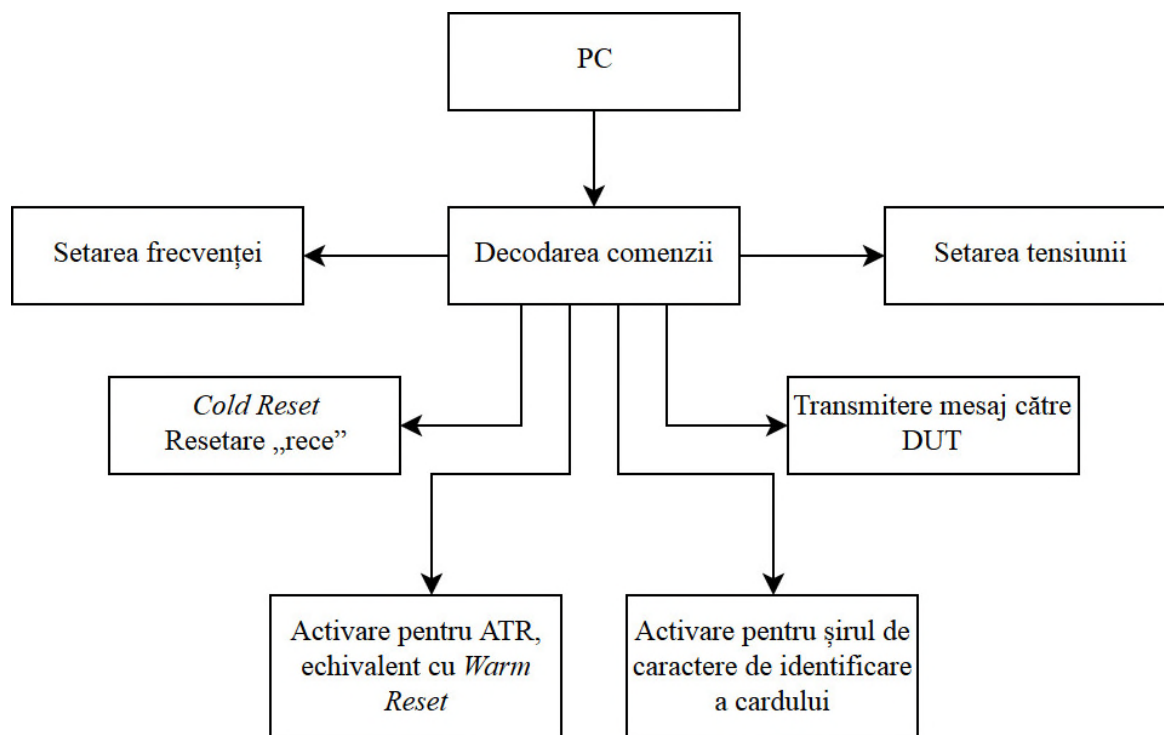


Figura 6.2. Diagrama conceptuală a implementării comenzilor

Comunicarea dintre utilizator (inginerul de testare) și DUT (smartcard) este realizată, după cum am amintit în capitolul 4 al acestei lucrări, prin interfețe de tip UART cu anumite particularități. Rolul cititorului este acela de a oferi o interfață între utilizator și smartcard, comunicarea putând fi considerată de tipul *master – slave*, în sensul în care smartcardul nu ar trebui să răspundă înainte de a primi o comandă de la cititor. Cititorul este responsabil să activeze smartcardul și să transmită în mod transparent comenzile de la utilizator la card și datele/răspunsul de la card la utilizator.

Fiecărei comenzi transmise de către utilizator îi va corespunde un cod al comenzii, pentru ca cititorul să poată interpreta dacă acea comandă impune setarea/schimbarea unor parametri, transmiterea unui șir de caractere către DUT, executarea secvenței de activare specifică pentru primirea ATR-ului, sau a celei pentru primirea șirului de caractere de identificare a cardului. Microcontrolerul STM32H743 va fi capabil să decodeze o comandă transmisă de către utilizator și să decidă care este sarcina pe care trebuie să o îndeplinească.

În figura 6.2 este prezentată o diagramă conceptuală a principalelor funcționalități pe care le poate realiza microcontrolerul după decodarea comenzii primite.

Limbajul de programare folosit pentru scrierea aplicației este C, iar mediul de programare se bazează pe Eclipse. Pentru dezvoltarea aplicației am pornit de la HAL-ul oferit de echipa STMicroelectronics. HAL – *Hardware Abstraction Layer* – este, așa cum sugerează și denumirea, un nivel de abstractizare între programator și funcționalitățile la nivel hardware ale dispozitivului. Bibliotecile de tip HAL pun la dispoziție funcții cu instrucțiuni ce presupun schimbarea directă a unei valori dintr-un registru, definițiile din cod ale porturilor, ale pinilor, ale tuturor constantelor necesare, rutine de configurare a interfețelor, de setare a anumitor parametri și așa mai departe.

Tabelul 6.1. Structura comenzilor de bază implementate

Comanda	Codul comenzii
Setare a frecvenței semnalului de ceas	01 xx
Activare pentru ATR	02 00
Activare pentru șirul de caractere de identificare a cardului	03 00
Transmitere a unui mesaj către DUT	04 00
Setare a tensiunii de alimentare a cardului	05 xx
<i>Cold Reset</i> – Resetare „Rece”	

Scopul HAL-ului este acela de a oferi o modalitate mai intuitivă de a programa un microcontroler: programatorul nu trebuie să vadă întotdeauna ce biți trebuie să seteze într-un registru pentru a activa o anumită funcționalitate a unui pin de intrare/ieșire, sau să scrie la nivel de hardware instrucțiunile necesare pentru a defini un anumit comportament al unei interfețe [9].

Nivelul de abstractizare oferă, prin includerea unor biblioteci în program, funcții ce inițializează perifericele, funcții de transfer de date, de setare a unor parametri, de modificare a valorii logice a unui pin de intrare/ieșire.

Pentru a implementa funcționalitățile despre care am vorbit, am stabilit un set de instrucțiuni pe care microcontrolerul va fi capabil să le deservească. Ele sunt prezentate în tabelul 6.1. Toate valorile din tabel sunt reprezentate în cod hexadecimal. În cazul în care va fi transmisă o comandă care nu va fi recunoscută de acest algoritm, microcontrolerul nu va realiza, în mod implicit, nicio operație.

În plus, pentru comenzile de setare a tensiunii și a frecvenței, valoarea celui de-al doilea byte al comenzii determină care este valoarea frecvenței sau a tensiunii dorite. Frecvența semnalului de ceas poate fi setată de la valoarea de 500 kHz până la 10 MHz, cu un pas de 500 kHz, deoarece, pentru aplicațiile de test existente în prezent, această rezoluție este suficientă. Pentru variația frecvenței semnalului de ceas general am folosit PLL-urile puse la dispoziție de controlerul folosit. Parametrul ce trebuie transmis pentru obținerea fiecărei frecvențe drept al doilea byte al comenzii este explicat în tabelul 6.2.

Tabelul 6.2. Comanda de schimbare a frecvenței semnalului de ceas

Comanda de setare a frecvenței semnalului de ceas		
Primul byte	Al doilea byte	Frecvența corespunzătoare
01	00	500 kHz
	01	1 MHz
	02	1,5 MHz
	03	2 MHz
	04	2,5 MHz
	05	3 MHz
	06	3,5 MHz
	07	4 MHz
	08	4,5 MHz
	09	5 MHz
	0A	5,5 MHz
	0B	6 MHz
	0C	6,5 MHz
	0D	7 MHz
	0E	7,5 MHz
	0F	8 MHz
	10	8,5 MHz
	11	9 MHz
	12	9,5 MHz
	13	10 MHz

Structura comenzilor pentru schimbarea tensiunii de alimentare de lucru este prezentată în tabelul 6.3. Asemănător cu structura comenzii de schimbare a frecvenței, primul byte va determina tipul comenzii, iar al doilea va corespunde valorii parametrului ce trebuie modificat. Decodarea comenzii se va face în bucla principală a programului. Tensiunea de lucru poate avea 4 valori, precum este precizat și în tabel: 0 V, 1,8 V, 3 V, și 5 V. Trebuie menționat faptul că tensiunea de lucru pe care o stabilim pentru card este valabilă și pentru tensiunea de alimentare, și pentru amplitudinile semnalelor transmise către card.

Tabelul 6.3. Comanda de schimbare a tensiunii de lucru

Comanda de setare a tensiunii de alimentare		
Primul byte	Al doilea byte	Tensiunea corespunzătoare
05	00	0 V
	01	1,8 V
	02	3 V
	03	5 V

Tabelul 6.4. Implementarea pentru interpretarea comenzii primite

Comanda efectuată	Implementarea în limbaj de programare
Comanda de setare a frecvenței	<pre> switch (cmd.cmd_type) { case 1: setFrequency(cmd.cmd_param); pRXDataUart6[contor_buffer_u6rx-2] = 0; pRXDataUart6[contor_buffer_u6rx-1] = 0; break; case 2: ATR_Activation(); HAL_Delay(50); pRXDataUart6[contor_buffer_u6rx-2] = 0; pRXDataUart6[contor_buffer_u6rx-1] = 0; break; case 3: Card_Info_Activation(); HAL_Delay(100); pRXDataUart6[contor_buffer_u6rx-2] = 0; pRXDataUart6[contor_buffer_u6rx-1] = 0; break; case 4: HAL_SMARTCARD_Transmit_IT(&hsmartcard2, dummy, 2); // pentru transmiterea unui mesaj oarecare, pentru inceput pRXDataUart6[contor_buffer_u6rx-2] = 0; pRXDataUart6[contor_buffer_u6rx-1] = 0; HAL_Delay(100); endTransmitstartReceiveSmartcard(); HAL_Delay(100); break; case 5: setVoltage(cmd.cmd_param); pRXDataUart6[contor_buffer_u6rx-2] = 0; pRXDataUart6[contor_buffer_u6rx-1] = 0; break; case 6: ColdReset(); pRXDataUart6[contor_buffer_u6rx-2] = 0; pRXDataUart6[contor_buffer_u6rx-1] = 0; break; default: __NOP(); break; } </pre>
Comanda de activare pentru ATR	
Comanda de activare pentru șirul de identificare a cardului	
Comandă pentru transmiterea unui mesaj către DUT	
Comandă pentru setarea tensiunii de lucru	
Comandă pentru Cold Reset	
Implicit, nu este efectuată nicio operație	

Pentru implementarea funcției de schimbare a tensiunii am folosit datele din tabelul 4.1, pentru a transmite semnalele corespunzătoare de control, iar construcția comenzii este explicată în tabelul 6.3.

Inițial, frecvența de lucru este de 3,5714 MHz, o frecvență standard pentru comunicarea cu smartcarduri – corespunzătoare acestei frecvențe sunt baud rate-ul de 9600 bps și a factorului de divizare 372.

După stabilirea conceptuală a structurii comenzilor, am trecut la implementarea lor în limbajul de programare. Secvența de cod ce interpretează primul byte al comenzii primite de la utilizator este prezentată în tabelul 6.4.

6.3. Secvența de activare pentru ATR

Recepționarea comenzilor de la PC la microcontroler și transmiterea datelor de la microcontroler la PC se efectuează prin intermediul modulului USART6, cu ajutorul unui cablu convertor de la USB la UART de tip TTL-232R. Pentru transmiterea datelor de la PC am folosit programul terminal HTerm. Semnalele au fost transmise către DUT prin intermediul interfeței USART2, configurată în modul de „Smartcard”. Pentru semnalul RST am folosit un pin de tip GPIO ca ieșire.

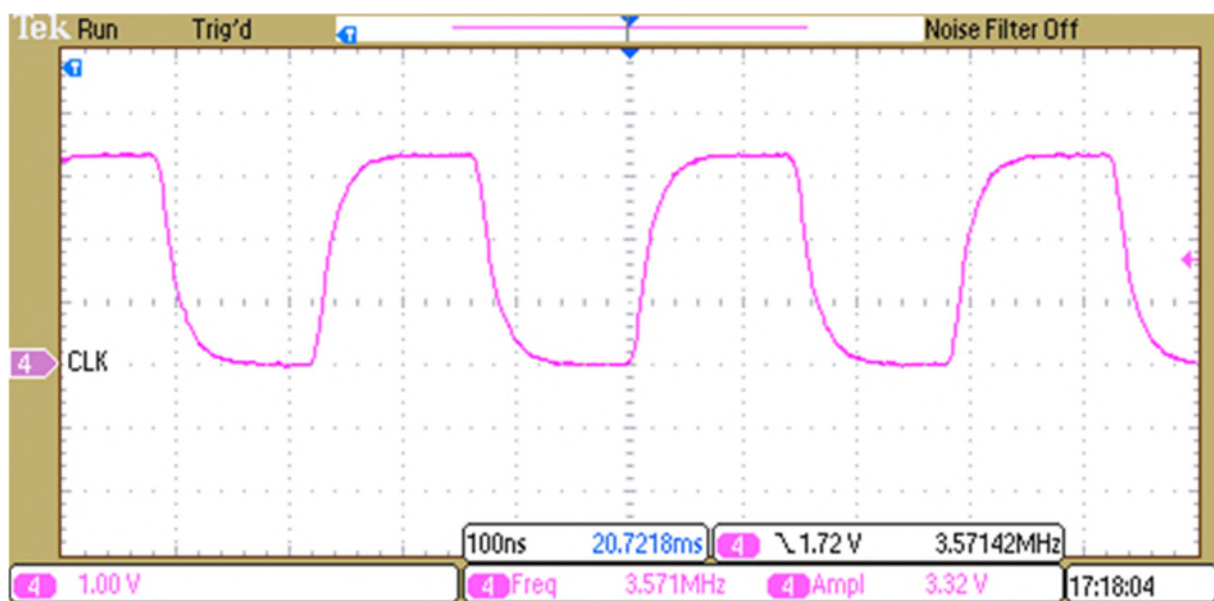


Figura 6.3. Frecvența semnalului de ceas de lucru implicită

Frecvența de lucru predefinită în aplicația noastră este prezentată printr-o captură de osciloscop, în figura 6.3, după inițializarea întregului sistem de generare a semnalelor de ceas al dispozitivului.

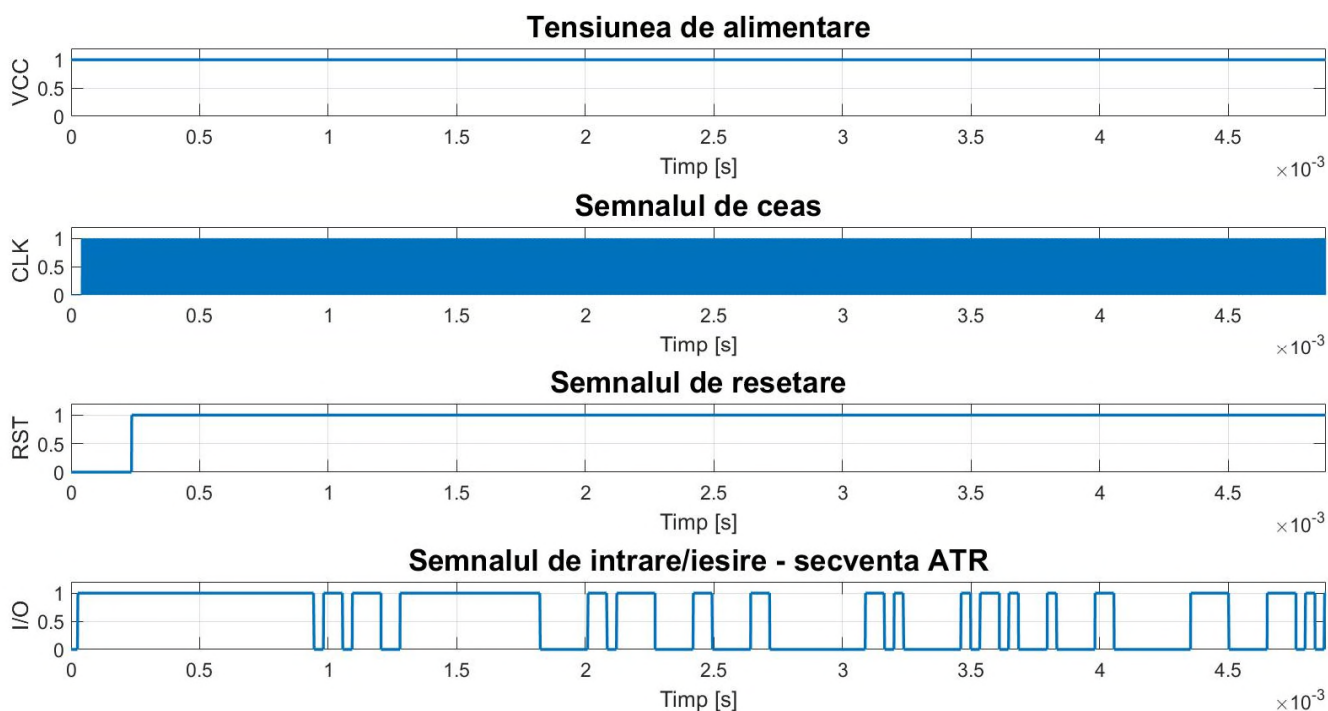


Figura 6.4. Secvența de activare pentru ATR și răspunsul primit

Pentru început, am implementat o funcție pentru generarea secvenței de activare pentru primirea ATR-ului, așa cum am prezentat în subcapitolul 2.3. Inițial, pentru monitorizarea semnalelor transmise către DUT, am folosit un analizor logic Saleae cu opt canale. Am obținut, după testarea algoritmului, rezultatele din figura 6.4, vizualizate cu ajutorul programului Saleae Logic, versiunea 1.2.18. Pentru a putea obține figuri într-un format optim, ușor de vizualizat, am exportat datele într-un fișier cu extensia „.mat”, compatibil cu mediul de programare Matlab, și am construit figurile cu ajutorul programului Matlab. Formele de undă sunt prezentate în timp, iar valorile de amplitudine nu au nicio semnificație fizică, nu au unitate de măsură, ele reprezintă doar nivelele logice „0” și „1”. Codul folosit este descris în anexa D.

Deoarece pot exista situații în care analizorul logic nu este de încredere pentru măsurătorile de care avem nevoie în aplicația noastră (poate detecta praguri ale semnalelor de mai multe ori în aceeași zonă – din cauza zgomotului, chiar dacă semnalul trece o singură dată de la un nivel logic la celălalt) am realizat și o captură de osciloscop pentru a observa ATR-ul. Figura 6.5 prezintă răspunsul la resetare al unui card Infineon, introdus în ansamblul creat.

Primul semnal din figură este cel de ceas, al doilea este I/O și cu ajutorul lui poate fi vizualizat răspunsul la resetare, iar cel de-a treilea este un bus pentru decodarea mesajului, prezentând fiecare byte corespunzător fiecărui set de tranziții din ATR.

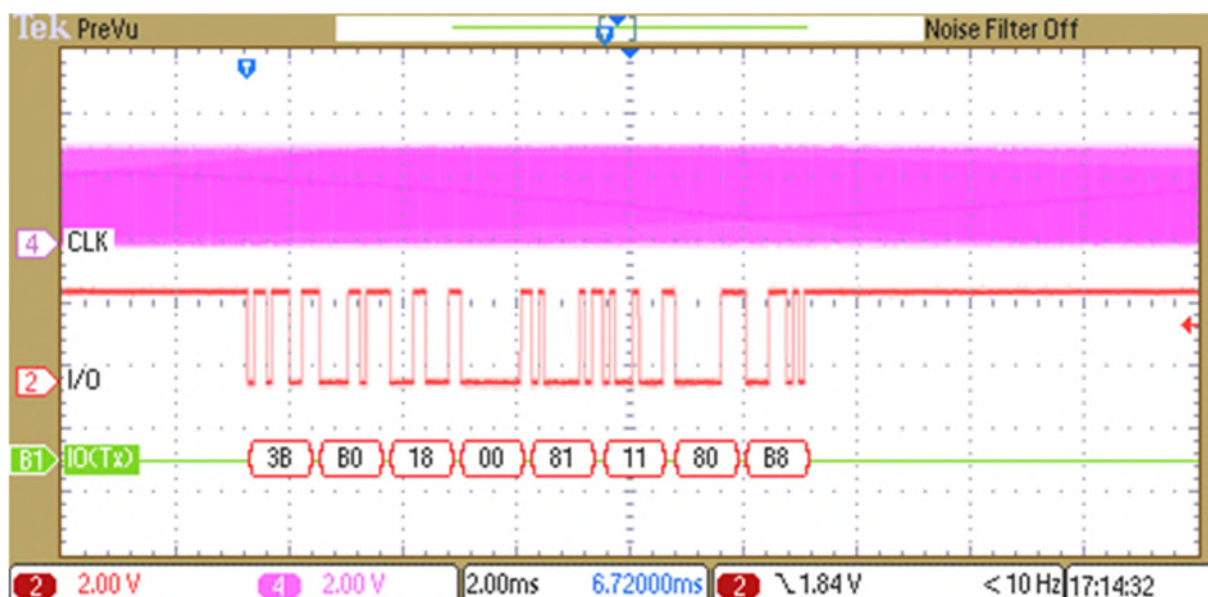


Figura 6.5. ATR-ul unui dispozitiv Infineon, după secvența de resetare generată de microcontrolerul STM

6.4. Secvența de activare pentru șirul de caractere de identificare a cardului

În același fel am procedat și pentru secvența de activare pentru primirea informațiilor de identificare a cardului, ce a fost detaliată în capitolul 2. Astfel, am obținut semnalele prezentate în figura 6.6. Deoarece semnalul de ceas, așa cum a fost exportat cu ajutorul programului oferit de Saleae, avea cele mai multe eșantioane (cele mai mult tranziții din „0” în „1” și invers), a fost foarte dificil de afișat și a fost exclus din această figură.

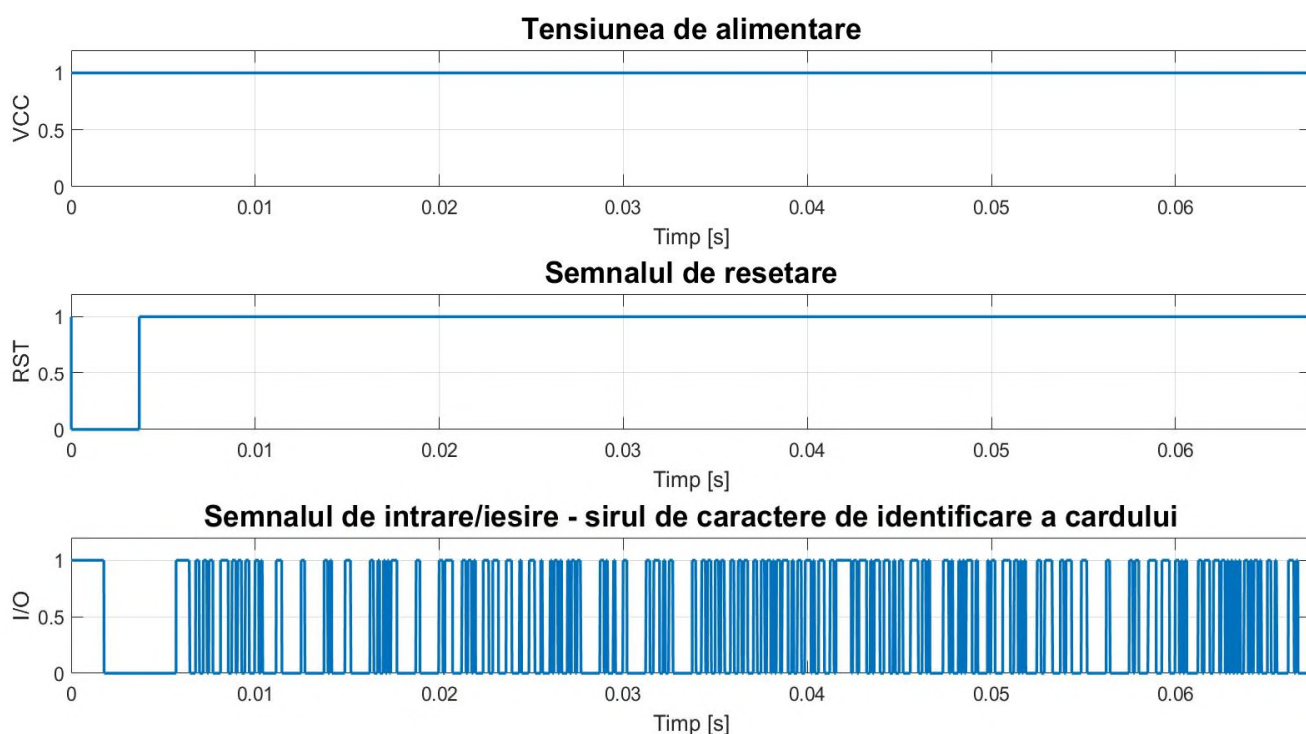


Figura 6.6. Șirul de caractere de identificare a cardului

6.5. Schimbarea frecvenței semnalului de ceas

În continuare, am implementat funcția de selecție a frecvenței semnalului de ceas, cu structura prezentată în tabelul 6.2. Funcția implementată pentru configurarea frecvenței semnalului de ceas este prezentată în anexa C.

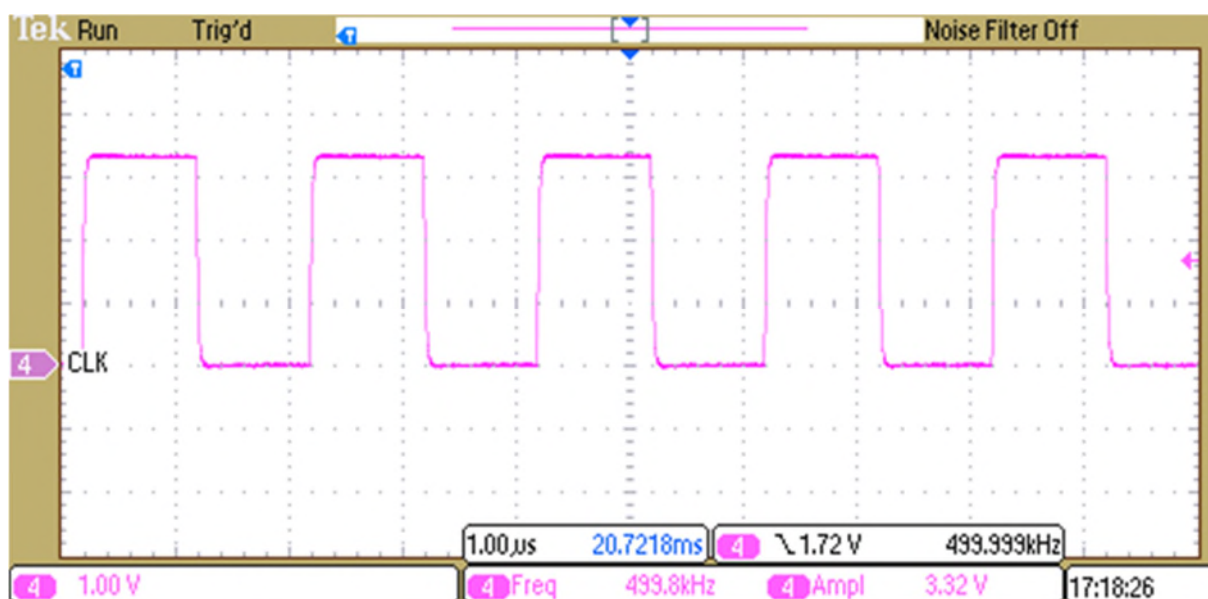


Figura 6.7. Frecvența semnalului de ceas, configurată la 500 kHz

După implementarea comenzii de setare a frecvenței semnalului de ceas, am monitorizat comportamentul acestui semnal tot cu ajutorul analizorului Saleae. Analizorul logic este conceput pentru a monitoriza semnale digitale înregistrate pornind de la un anumit stimul, de start, pentru achiziția datelor. Pentru a putea vizualiza mai clar formele de undă obținute, pentru a măsura mai precis caracteristicile electrice ale semnalului, și anume frecvența și tensiunea, am folosit osciloscopul pentru a vedea cum se modifică frecvența semnalului la trimiterea comenzii de la PC.

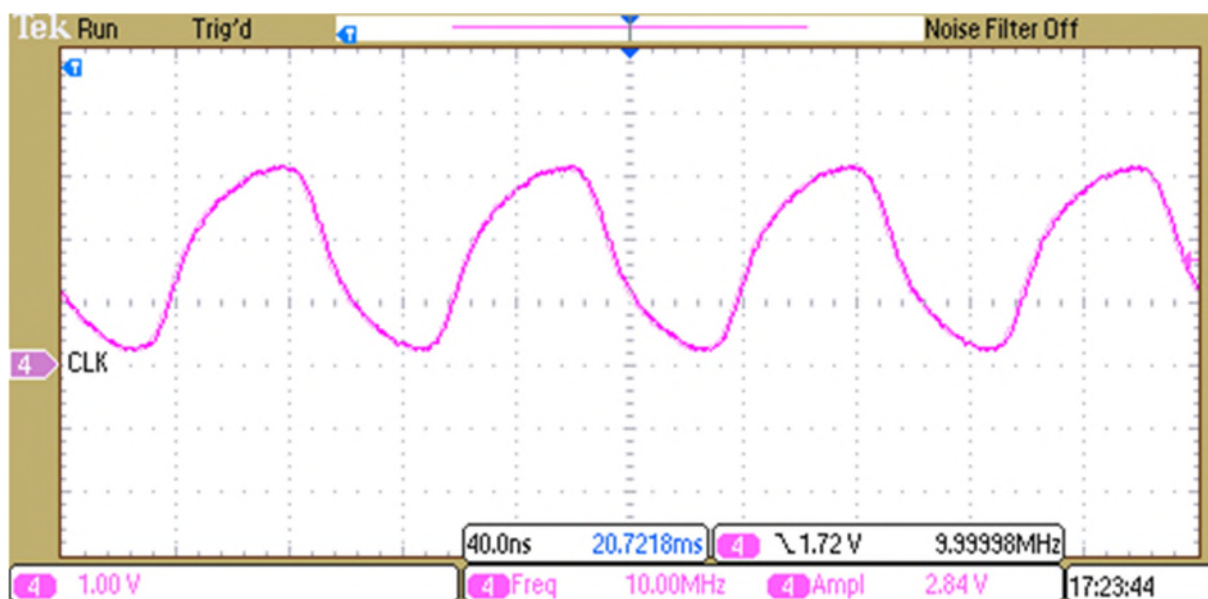


Figura 6.8. Frecvența semnalului de ceas, configurată la 10 MHz

În figura 6.7 este reprezentat semnalul de ceas, la frecvența de 500 kHz, după trimiterea comenzii „01 00” prin USART6 către microcontroler. Microcontrolerul STM32H743 lucrează cu semnale logice cu amplitudinea de 3,3 V, iar amplitudinea semnalului de ceas este, după cum se poate observa și în figură, foarte apropiată de 3,3 V.

În figura 6.8 este prezentat semnalul de ceas, configurat cu frecvența de 10 MHz. Cu ajutorul figurii putem vedea cum, la o frecvență atât de mare, devine mai ușor de observat efectul capacitiv al întregului ansamblu.

Pentru a putea trimite comenzi către microcontroler am folosit HTerm, un program-terminal de emulare pentru interfețe seriale. În figura 6.9 este prezentat programul, jos poate fi observată comanda transmisă pe interfața serială către USART-ul microcontrolerului (*Transmitted Data*), iar în fereastra *Received Data* este regăsit ATR-ul, trimis de card și transmis către PC în mod transparent de către microcontroler.

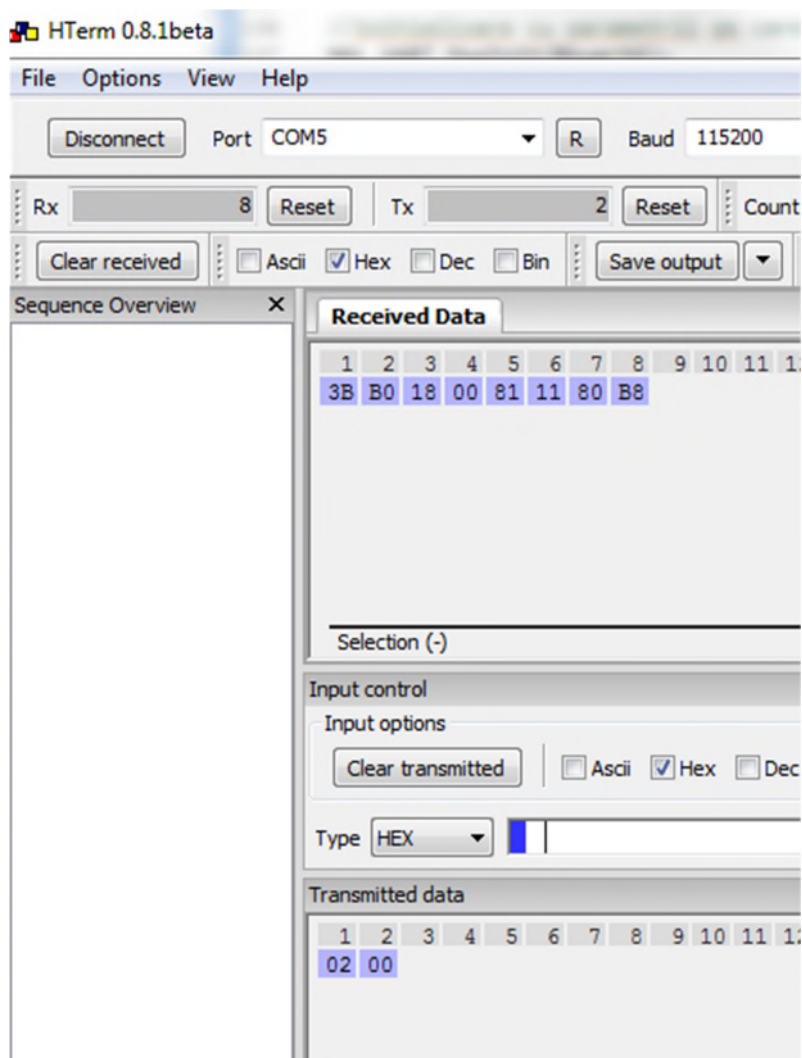


Figura 6.9. ATR-ul primit de către PC, vizualizat cu ajutorul Hterm

6.6. Schimbarea tensiunii de lucru

După implementarea funcționalității de schimbare a tensiunii de alimentare și, implicit, a amplitudinii semnalelor de lucru, am testat această funcție cu ajutorul plăcii de dezvoltare STEVAL

– IPT007V1. Această placă are ca element central interfața analogică prezentată în subcapitolul 4.3 și oferă atât puncte de testare pentru semnalele de interes, cât și un conector special pentru introducerea smartcardului. Funcția implementată pentru configurarea tensiunii este prezentată în anexa C.

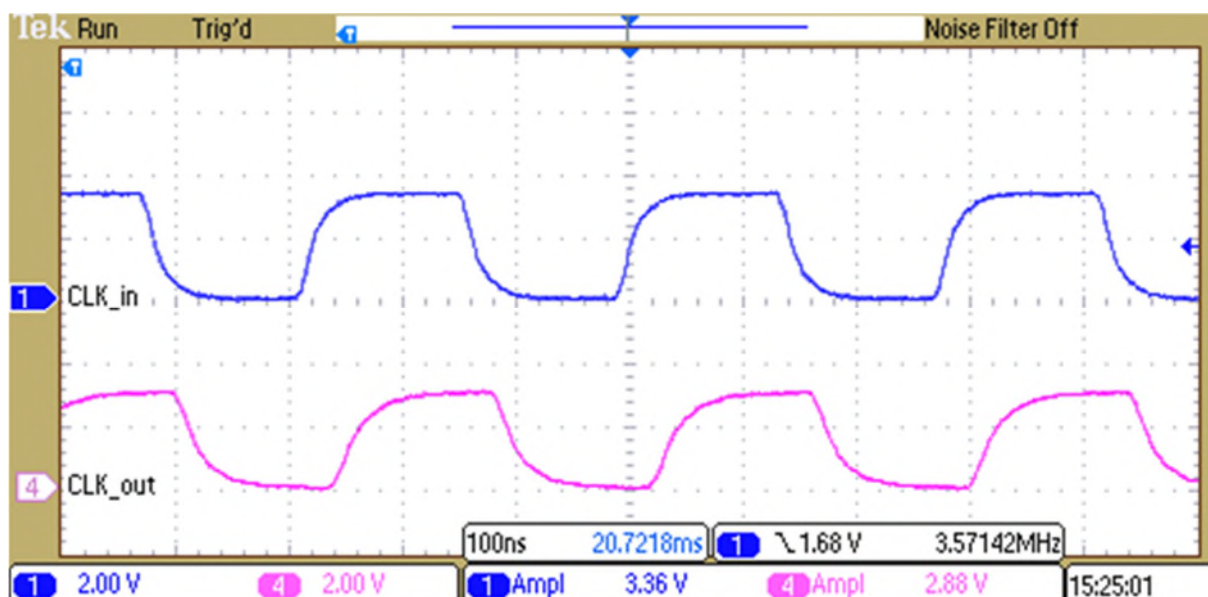


Figura 6.10. Tensiunea configurată, pentru început, la 3 V

În figura 6.10 este prezentată configurarea inițială a tensiunii de lucru, la 3 V. Semnalul CLK_in este semnalul de intrare în interfața analogică, înainte de trecerea prin *level shifter*, iar semnalul CLK_out este semnalul de ieșire, cu o valoare a amplitudinii de aproximativ 3 V.

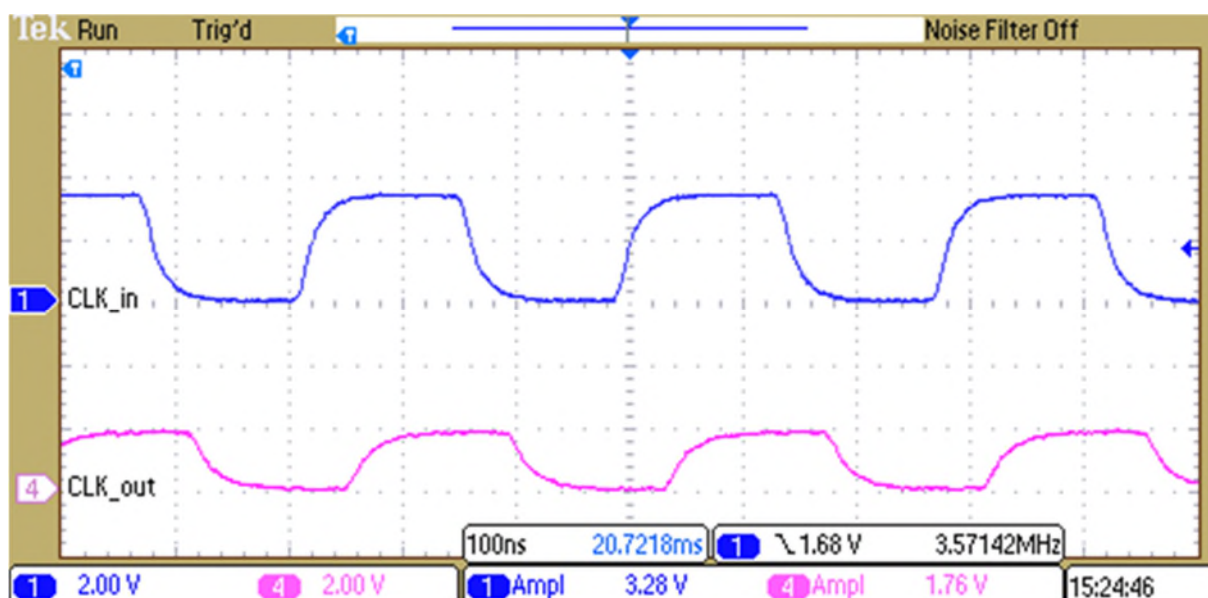


Figura 6.11. Configurarea tensiunii de lucru la 1,8 V

În figura 6.11 tensiunea este configurată la 1,8 V. după cum putem observa, amplitudinea semnalului de ceas de ieșire din interfața analogică are o valoare apropiată de 1,8 V.

Pentru configurarea tensiunii la 5 V, rezultatele sunt prezentate în figura 6.12.

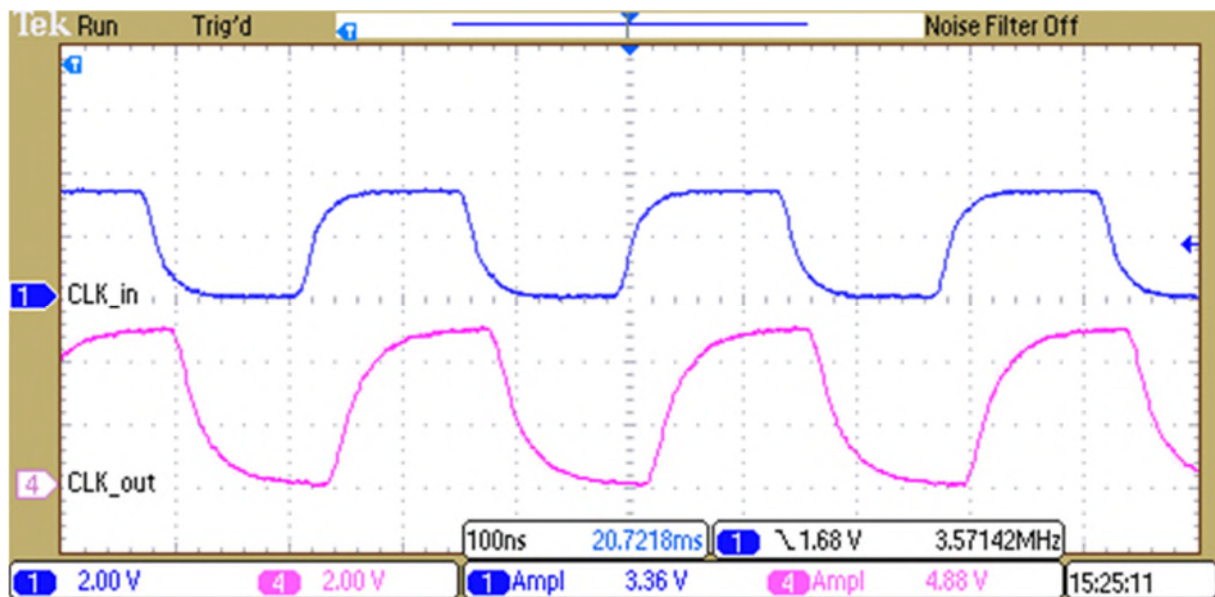


Figura 6.12. Configurarea tensiunii de lucru la 5 V

Interfața analogică este capabilă să detecteze dacă smartcardul nu este introdus în conectorul special, iar în cazul în care cardul este absent, va opri toate ieșirile către DUT. Pentru a putea implementa funcționalitatea de *Cold Reset*, am implementat această funcționalitate și în software, prin schimbarea valorii logice a semnalului ce intră prin $\overline{\text{CMDVCC}}$. Dacă valoarea logică a acestei linii este „0”, atunci interfața analogică va opri – va aduce la masă – semnalele de ieșire, dinspre smartcard. Astfel, la trimiterea comenzii corespunzătoare tensiunii de 0 V din tabelul 6.3. Rezultatul este descris în figura 6.13.

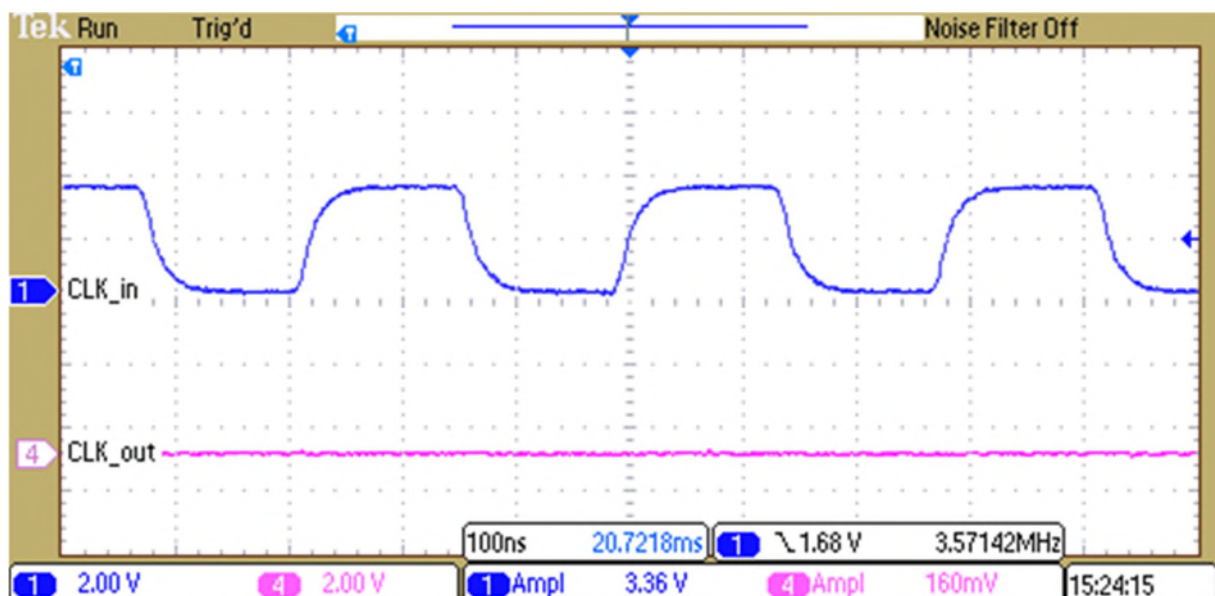


Figura 6.13. Configurarea tensiunii de lucru la 0 V - oprirea trecerii semnalelor către DUT

7. Rezultate și probleme întâmpinate

Pentru caracterizarea ansamblului de test am vrut să vedem care este întârzierea pe care o introduce în mod real interfața analogică folosită. Pentru a măsura această întârziere, am conectat semnalele de interes de intrare și de ieșire în și din interfață la osciloscop, și anume semnalul de ceas și cel de I/O.

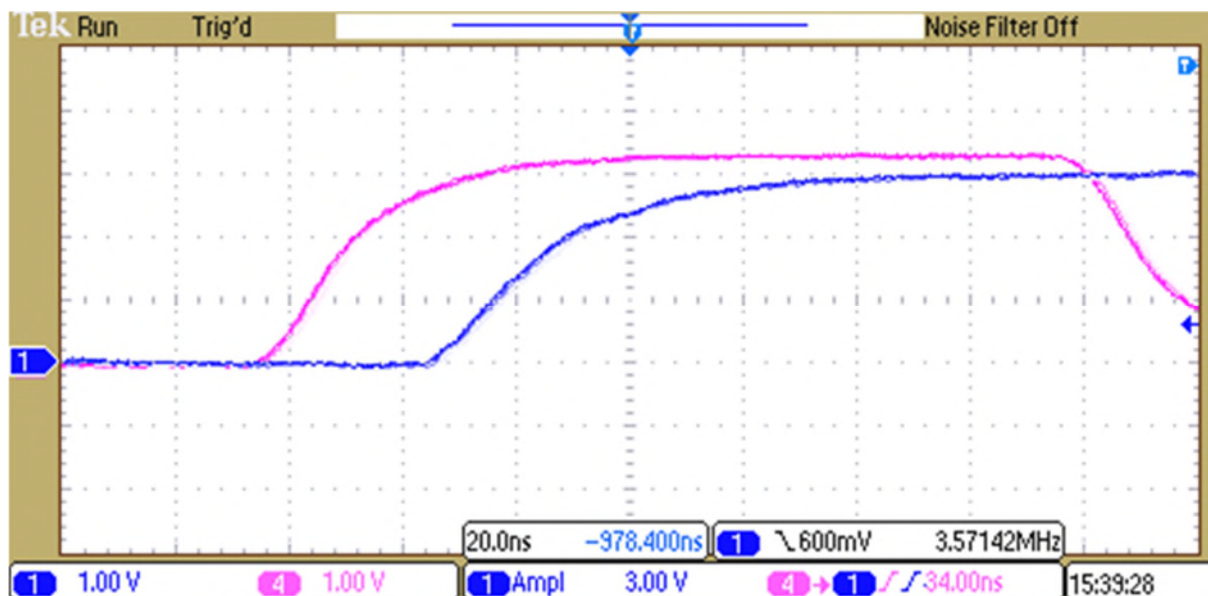


Figura 7.1. Întârzierea introdusă de interfața analogică între semnalul de ceas de intrare și cel de ieșire

În figura 7.1 poate fi observată întârzierea introdusă pe semnalul de ceas, semnalul 1 fiind cel de ieșire, iar 4 cel de intrare.

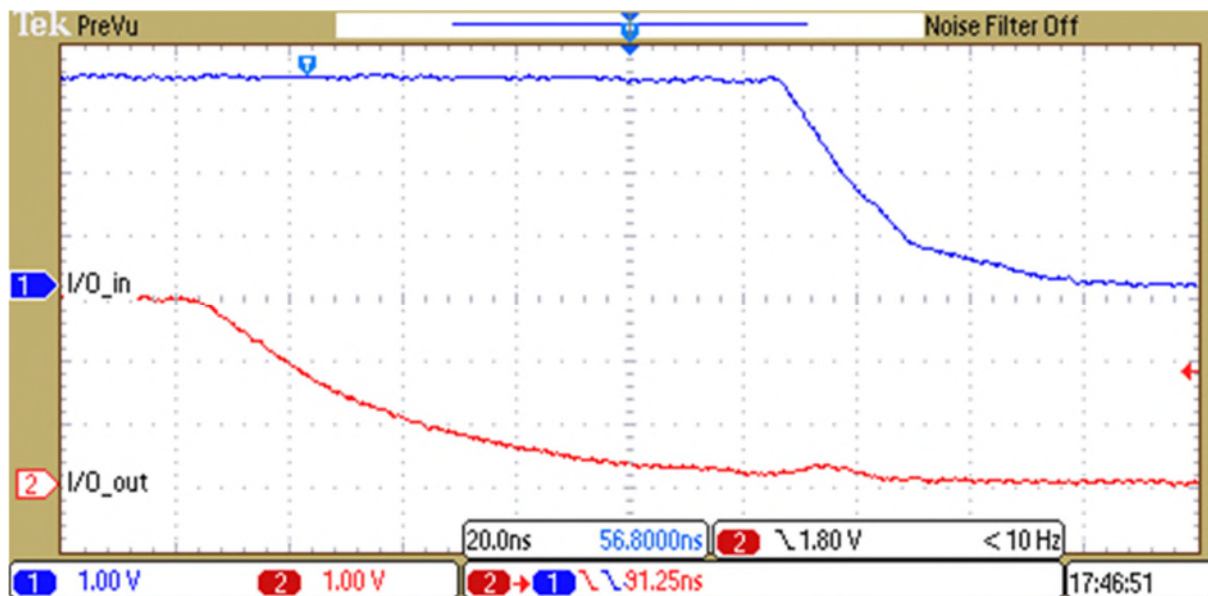


Figura 7.2. Întârzierea dintre semnalul I/O la intrarea în interfață și cel de la ieșire

Următorul pas a fost efectuarea măsurătorilor și pentru întârzierea obținută la primirea ATR-ului. În figura 7.2 este prezentată întârzierea dintre semnalul de intrare în interfața analogică și cel de ieșire,

măsurată la prima tranziție a semnalului, de pe parcursul transmiterii ATR-ului. Trebuie să avem în vedere că pentru această figură am considerat semnalul I/O_in semnalul I/O dinspre microcontroler, iar I/O_out semnalul dinspre DUT.

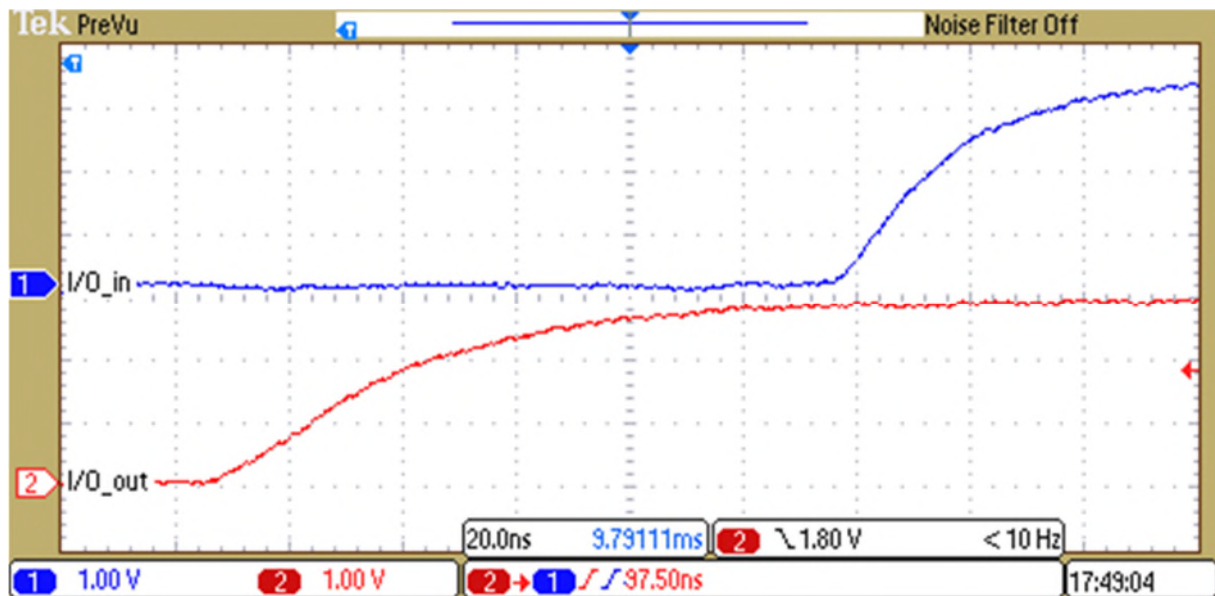


Figura 7.3. Întârzierea introdusă pe semnalul I/O la sfârșitul transmisiunii ATR-ului

În figura 7.3 este prezentată întârzierea măsurată la ultima tranziție a semnalului – sfârșitul transmisiunii. Deși poate fi observat un decalaj de câteva nanosecunde între aceste două întârzieri, acesta nu influențează interpretarea datelor și transmiterea cu succes a mesajului.

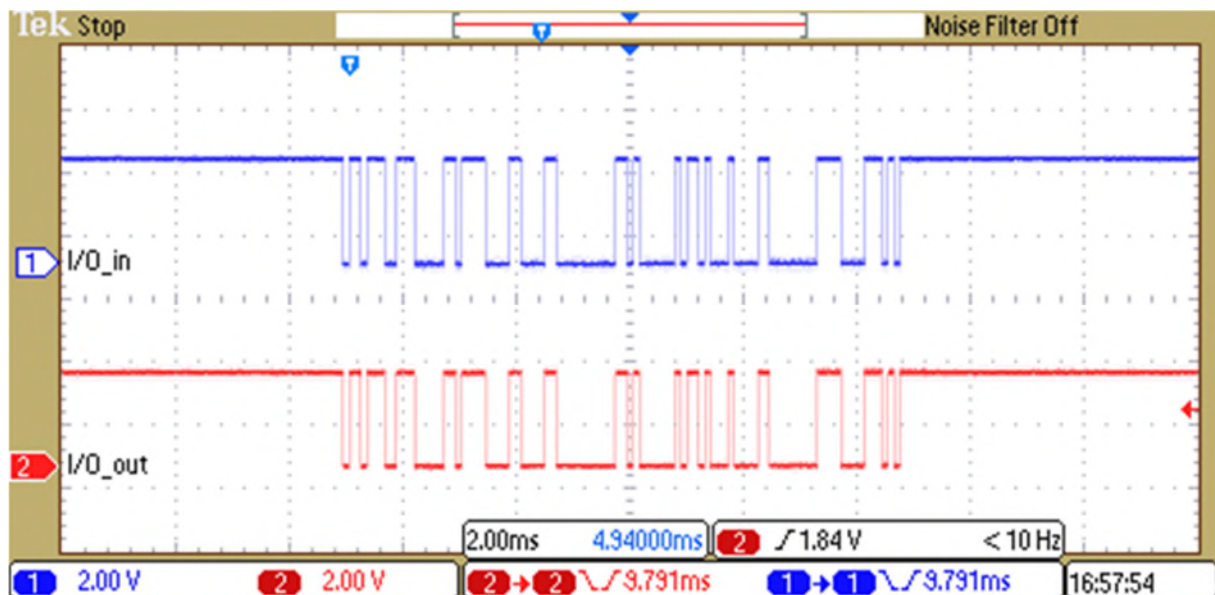


Figura 7.4. Durata răspunsului la resetare

În figura 7.4 am măsurat durata totală de timp a unui răspuns la resetare atât la intrarea interfeței analogice, cât și la ieșirea acesteia.

Alte rezultate obținute pe parcursul lucrării sunt prezentate în anexa B.

Implicit, atât în partea software, cât și prin construcția plăcii de dezvoltare Nucleo, comunicarea dintre PC și microcontrolerul STM32, trimiterea datelor și recepționarea lor se efectuează prin interfața UART3. Comunicarea cu smartcardul a fost implementată cu ajutorul interfeței UART2. Problema de care ne-am lovit a fost aceea că, la schimbarea frecvenței sursei de ceas a lui UART2, UART3 nu mai înțelegea mesajul pe care îl transmitem: după cum am explicat în subcapitolul 4.1, două module UART se vor „înțelege” dacă se stabilește comunicația cu aceeași frecvență și baudrate în ambele părți, cu mici toleranțe.

Prin construcția sistemului intern ce ceas al microcontrolerului, interfața UART2 primește semnalul de ceas de referință din aceeași sursă ca și UART3. De aceea, de fiecare dată când schimbăm parametrii stabiliți de cititor precum frecvență și baudrate pentru UART2 – pentru a comunica cu smartcardul, vom schimba și frecvența semnalului de ceas primit de UART3. Deci, când vom recepționa/transmite date de la PC trebuie să ne asigurăm că frecvența și baudrate-ul sunt setate corespunzător, pentru a ne putea „înțelege” cu UART3. Pentru a nu schimba de fiecare dată parametrii din programul de comunicare serială HTerm, am ales folosirea unui UART cu o sursă de ceas independentă de sursa lui UART2. Pentru a folosi această interfață, și anume UART6, am avut nevoie de un adaptor TTL – USB serial TTL-232R.

Frecvența de lucru a microcontrolerului STM este mult mai mare decât cea oferită către periferice. De aceea, de fiecare dată când am implementat o tranziție pe un pin GPIO configurat ca ieșire și, imediat, tranziția inversă, am observat că programul este mult mai rapid și, fizic, tranzițiile nu sunt realizate. Pentru a realiza tranzițiile corect și complet, am introdus un timp de așteptare de la o tranziție până la efectuarea următoarei tranziții.

O altă problemă întâmpinată a fost gestionarea întreruperilor pentru interfața UART2 – pentru smartcard. Deoarece pentru comunicarea cu un card UART-ul are anumite particularități precum folosirea unei singure linii pentru RX și TX (comunicare de tip *full-duplex*), există un risc mai mare de conflict între datele transmise și cele recepționate. De fiecare dată când cititorul transmite ceva, el trebuie să iasă din modul de recepționare (semnalul I/O configurat în așteptare la valoarea de „1” logic), iar întreruperea pentru recepționare pentru această interfață trebuie oprită. În același timp, este absolut necesară comunicarea cu ajutorul întreruperilor: unele comenzi transmise către DUT implică procesarea unor informații sau efectuarea unor rutine lungi, chiar așteptarea unor perioade de timp îndelungate până la primirea unui rezultat. De aceea, nu putem seta cu exactitate un timp în care să așteptăm primirea unui răspuns de la card. Întreruperile sunt utile în acest caz deoarece cititorul poate rămâne într-o stare de inactivitate până la primirea unui stimul extern, moment în care va intra în rutina de deservire a întreruperii. Deoarece pentru implementarea conceptului nostru este nevoie de recepționarea mesajelor din surse diferite, alternativa folosirii întreruperilor, aceea a interogării continue sau alternative a surselor (*polling*), nu este de dorit. Dacă am recepționa toate mesajele prin interogarea surselor am risca să pierdem date primite de la una dintre surse atunci când o interogăm pe cealaltă.

Totodată, livrarea unor interfețe analogice ST8034HC este posibilă numai cu întârziere foarte mare, de luni întregi. Pentru a realiza un prototip într-un timp cât mai scurt, am folosit interfețe analogice de tip ST8034HN, cu mici diferențe din punct de vedere hardware. Pentru acest lucru, am realizat placa astfel încât să fie compatibilă cu ambele interfețe analogice.

Concluzii și planuri de dezvoltare viitoare

Principalul obiectiv al acestei lucrări a fost să demonstrăm posibilitatea de a implementa funcționalitățile pe care le are un cititor destinat testării cu ajutorul unui microcontroler capabil să gestioneze semnalele de control necesare pentru comunicarea cu un smartcard. Am pornit de la implementarea unei soluții ce avea în centru un microcontroler Infineon. Odată ce ne-am lovit de limitările hardware valabile pentru aplicația propusă ale acestui controler, am continuat prin a alege un microcontroler STM32H743, mult mai potrivit în nevoilor noastre.

Contribuțiile aduse în cadrul acestui proiect constau în implementarea algoritmului de decodare a comenzilor, a rutinelor de rezolvare a acestor comenzi, implementarea comunicației de bază dintre cititor și smartcard, transmiterea și recepționarea mesajelor folosind întreruperile cu ajutorul NVIC, construirea ansamblului de test alcătuit din placa de dezvoltare Nucleo și placa de evaluare a interfeței analogice. Mai apoi, am creat simbolurile necesare pentru schema electrică și amprente componentelor pentru PCB, am creat schema electrică a unei plăci adaptate nevoilor noastre și formei fizice a plăcii Nucleo, pentru integrarea interfeței analogice în proiect.

Am testat funcționalitățile implementate folosind un smartcard Infineon – pentru a putea observa, cu ajutorul analizorului logic și al osciloscopului, răspunsul la resetare și șirul special de caractere pentru identificarea cardului.

Așadar, pe parcursul dezvoltării acestei lucrări am reușit stabilirea unei comunicații de bază cu un smartcard, schimbarea parametrilor de bază precum frecvența și tensiunea de lucru, implementarea unor funcționalități esențiale pentru testare precum activarea pentru răspunsul la resetare și activarea pentru primirea șirului de caractere de identificare a cardului.

Multitudinea interfețelor de care dispune microcontrolerul STM32H743 și modul lor de funcționare au o importanță crucială pentru implementarea cititorului dorit. Fiecare interfață integrată într-un microcontroler de tip smartcard trebuie testată corespunzător, folosind un cititor special pentru interfața respectivă. În prezent, sunt folosite dispozitive diferite pentru comunicarea prin UART, prin SPI, prin I^2C sau prin SWP. Soluția propusă, folosirea microcontrolerului STM32H743, permite implementarea unor funcții ce pot stabili comunicarea pe oricare interfață, așadar, vom putea folosi un singur cititor pentru comunicarea pe toate interfețele. În continuare, pe lângă extinderea modalităților de utilizare ale acestui cititor, ne propunem implementarea unor rutine pentru fiecare situație ce poate genera o eroare, crearea unui mediu mai plăcut și mai intuitiv de transmitere a comenzilor către microcontroler. Pentru a putea realiza tot ce ne propunem din punct de vedere software, vom realiza un program ce presupune rularea mai multor fire de execuție paralele.

Totodată, ne propunem dezvoltarea unei plăci ce va conține întregul concept al plăcii Nucleo, interfața analogică, stabilizatoare pentru o variație mult mai fină a tensiunii, linii de semnal cât mai optime pentru a obține o acuratețe cât mai mare în efectuarea testelor.

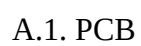
Pentru dezvoltarea acestui proiect am lucrat în limbajul C, într-un mediu de programare oferit de Eclipse, cu mediul de dezvoltare STM32 Cube, am realizat schema electrică și placa imprimată în Altium Designer. Am lucrat cu programul terminal pentru comunicație serială HTerm, am dezvoltat

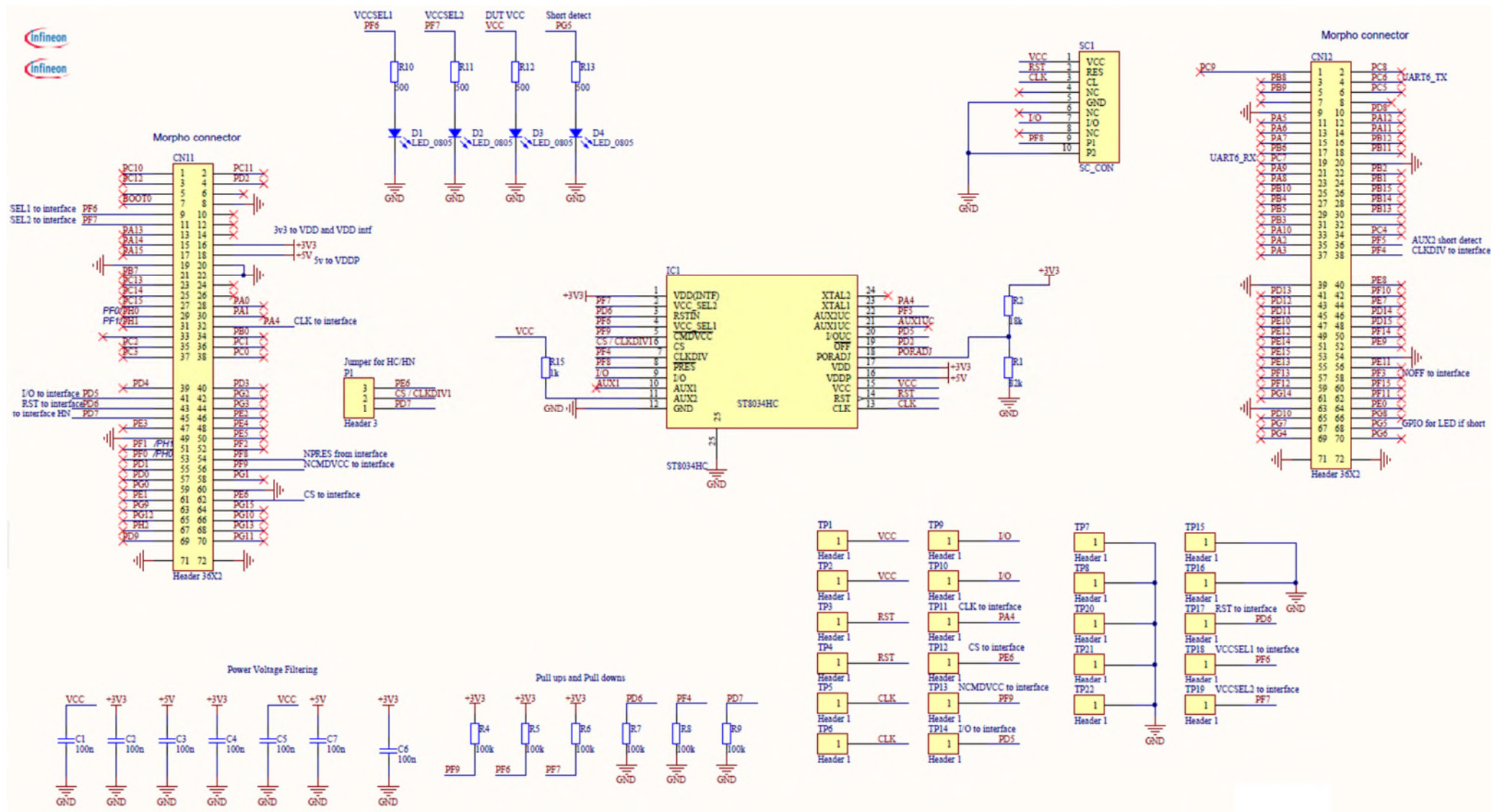
cunoștințele legate de întreruperile într-un microcontroler de tip ARM, aptitudinile de depanare, atât din punct de vedere software, cât și hardware. Am folosit un analizor logic pentru monitorizarea semnalelor logice, un osciloscop pentru măsurarea caracteristicilor electrice. Am afișat graficele folosind datele brute achiziționate de analizor, prin introducerea lor în Matlab.

Bibliografie

- [1] Douglas Hoffman, *Cost Benefits Analysis of Test Automation*, 1999, Software Quality Methods, LLC.
- [2] Wolfgang R., Wolfgang E., *Smart Card Handbook*, Ediția a treia, 2003, John Wiley & Sons Ltd.
- [3] *SPI Block Guide V03.06*, 2003, Motorola, Inc.
- [4] ISO-IEC 7816-3, *Part 3: Identification cards — Integrated circuit cards*, International Organization for Standardization/International Electrotechnical Commission, Partea a III -a, 2006
- [5] AN3496, *A Software Approach to Using Nested Interrupts in HCS08*, 2007, Freescale Semiconductor.
- [6] ST8034HN, ST8034HC, *Datasheet*, 2013, STMicroelectronics.
- [7] AN4800 Application Note, *Smartcard interface based on STM32Cube firmware*, ultima revizie la 18 dec 2015, STMicroelectronics.
- [8] RM0433, *Reference manual*, ultima revizie la 04 aprilie 2019, STMicroelectronics.
- [9] UM2217, *Description of STM32H7 HAL and low-layer drivers*, ultima revizie la 05 aprilie 2019, STMicroelectronics.
- [10] *Serial Peripheral Interface (SPI)*, <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all> pentru functionare SPI , accesat la data 16 iunie 2019.
- [11] *Basics of UART Communication*, Circuit Electronics, <http://www.circuitbasics.com/basics-uart-communication/> , accesat la data 16 iunie 2019.
- [12] *Cortex – M7*, <https://developer.arm.com/ip-products/processors/cortex-m/cortex-m7> , accesat la data 22 iunie 2019.
- [13] *NVIC Interrupts*, <https://developer.arm.com/docs/100165/latest/nested-vectored-interrupt-controller/nvic-functional-description/nvic-interrupts> , accesat la data 22 iunie 2019.
- [14] *Basics of the I2C Communication Protocol*, Circuit Basics, <http://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/>, accesat la data 23 iunie 2019.

Anexa A – Schema electrică pentru integrarea interfeței analogice

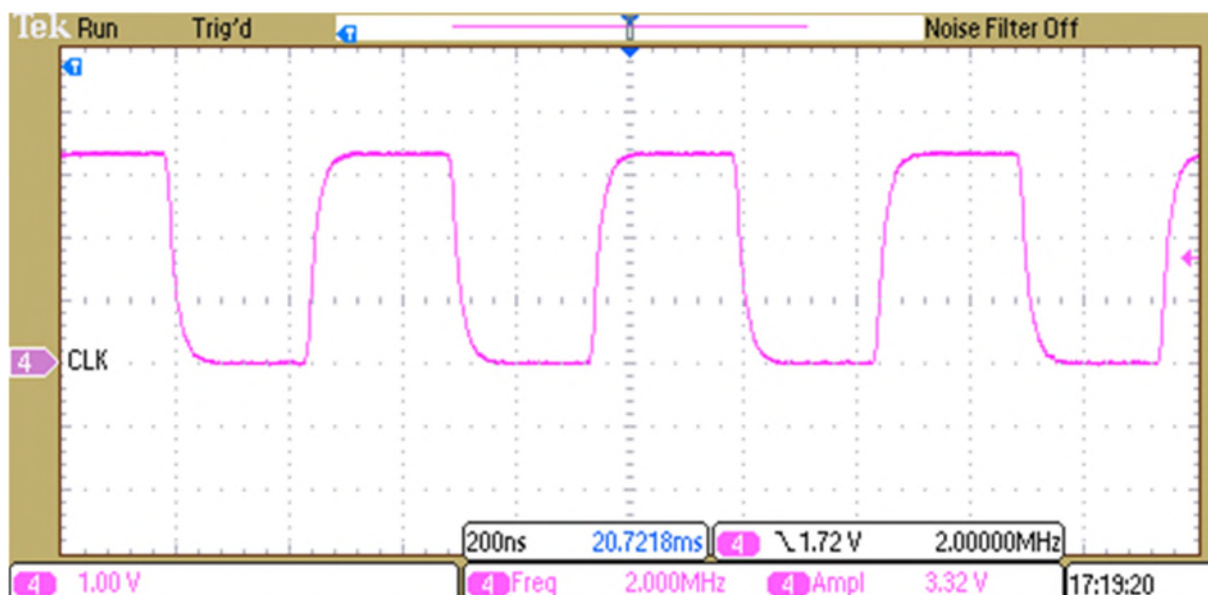




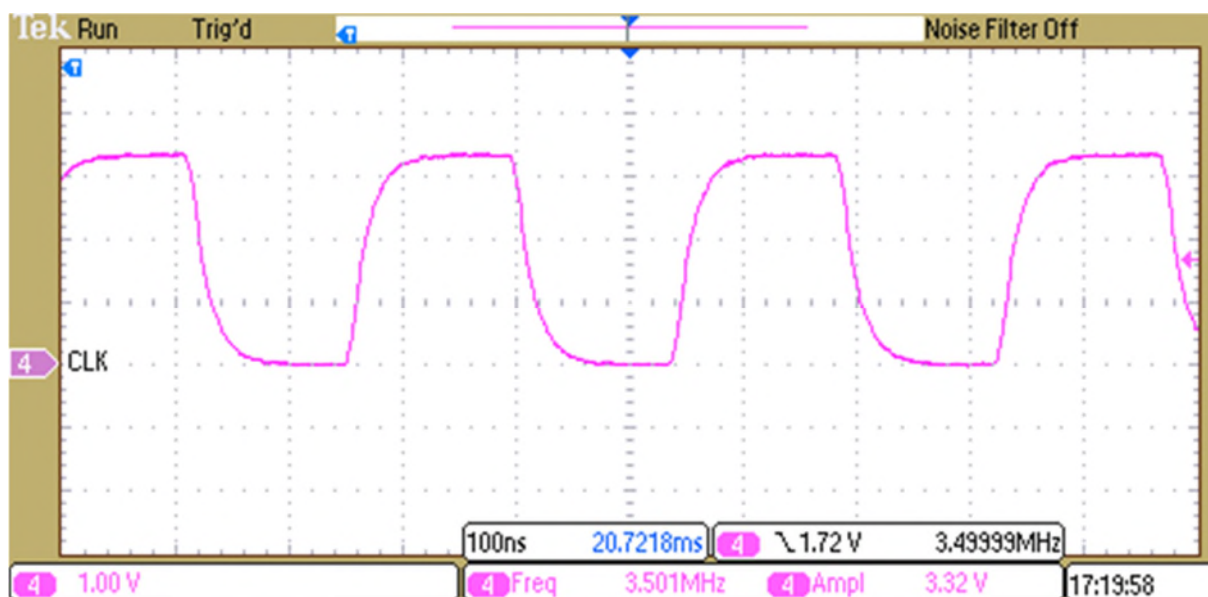
A.2. Schema electrică generală a plăcii pentru integrarea interfeței analogice

Anexa B – Alte rezultate obținute

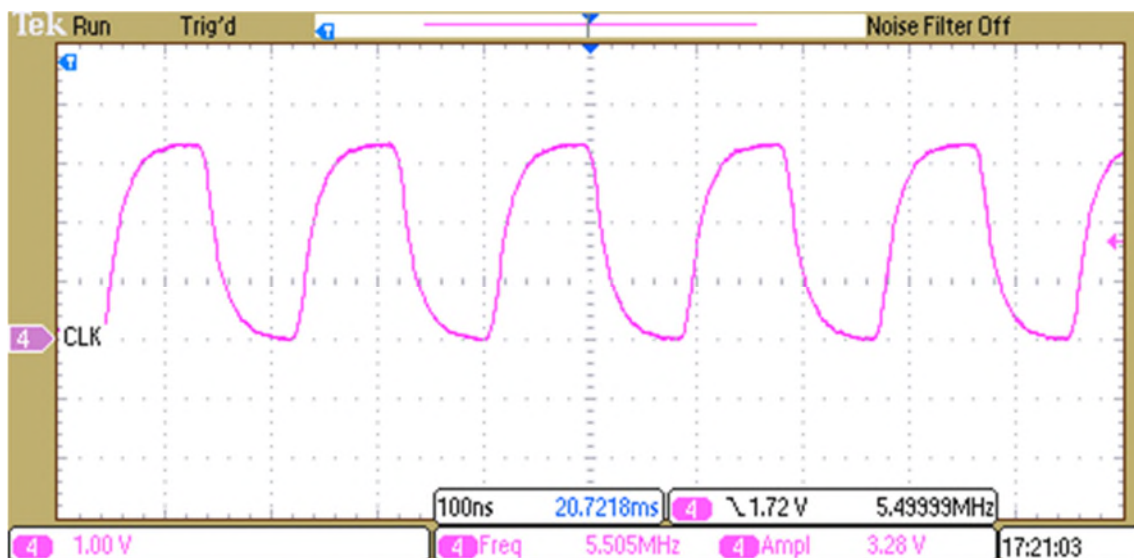
Această anexă cuprinde alte măsurători de frecvență ale semnalului de ceas, pentru celelalte frecvențe configurate. Frecvențele prezentate aici sunt intermediare, nefiind relevant să fie prezentate toate frecvențele ce au fost configurate pe parcursul lucrării.



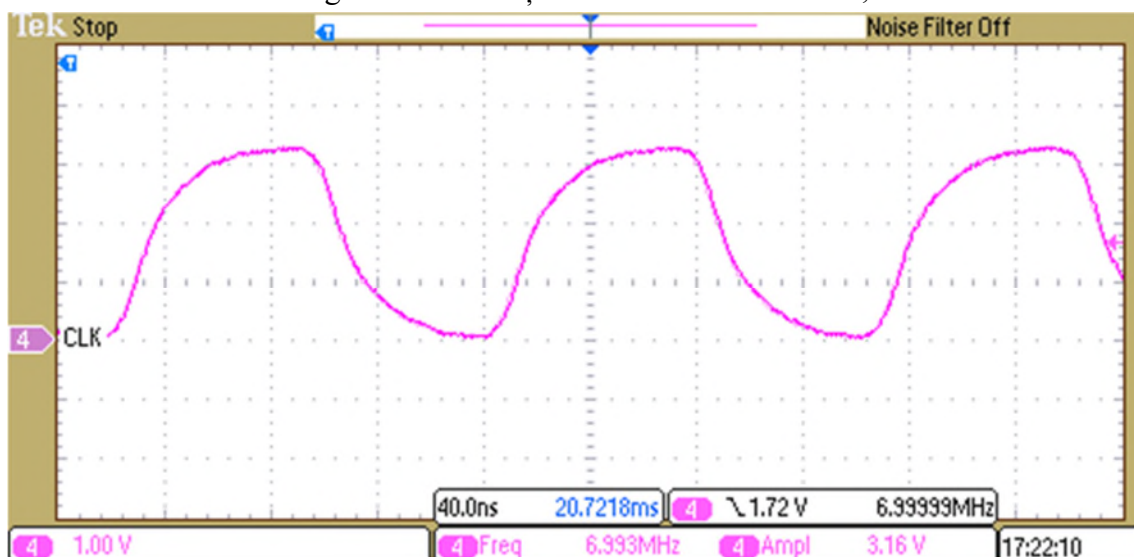
B.1. Configurarea frecvenței semnalului de ceas la 2 MHz



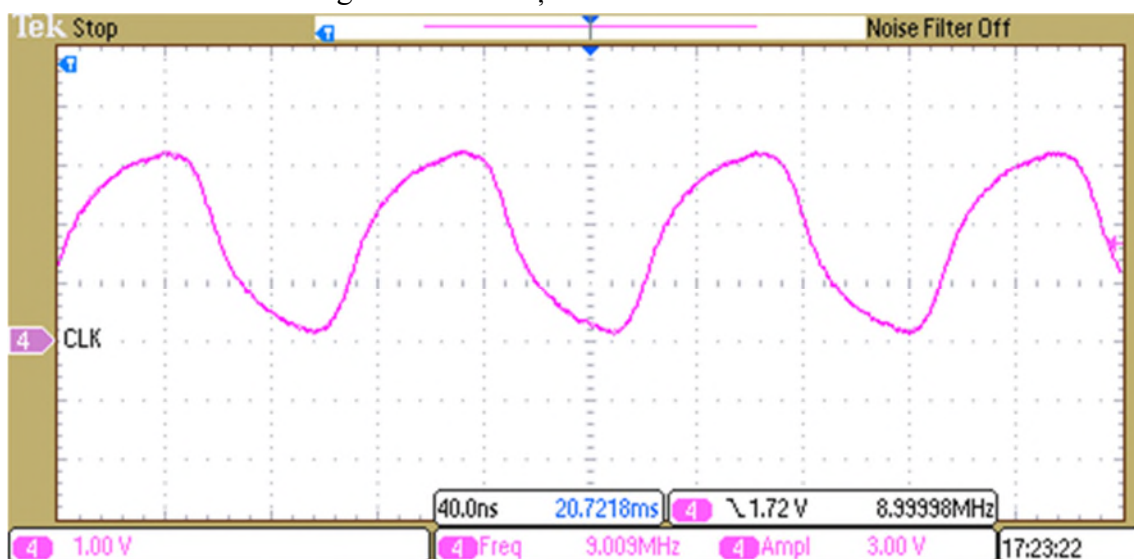
B.2. Configurarea frecvenței semnalului de ceas la 3,5 MHz



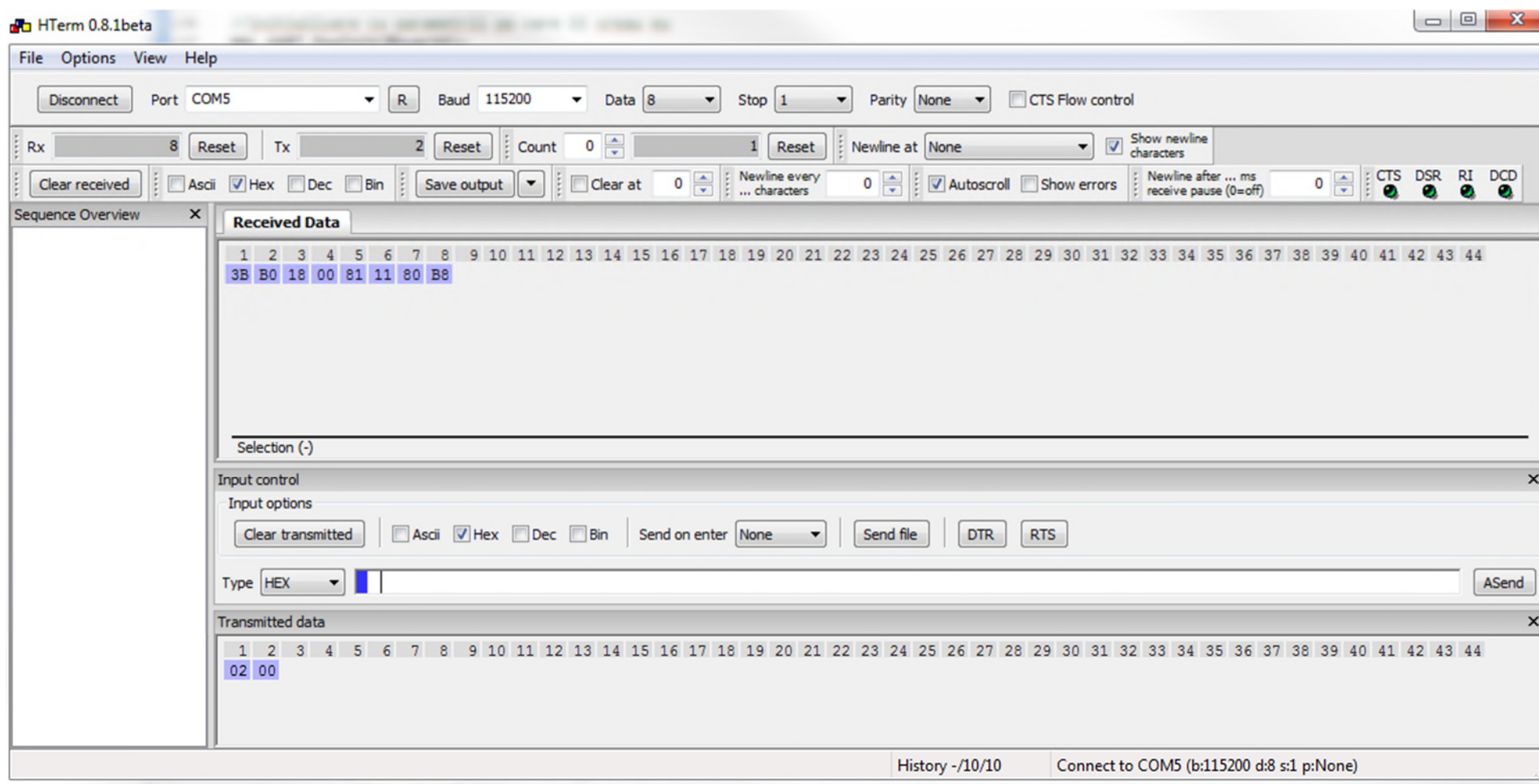
B.3. Configurarea frecvenței semnalului de ceas la 5,5 MHz



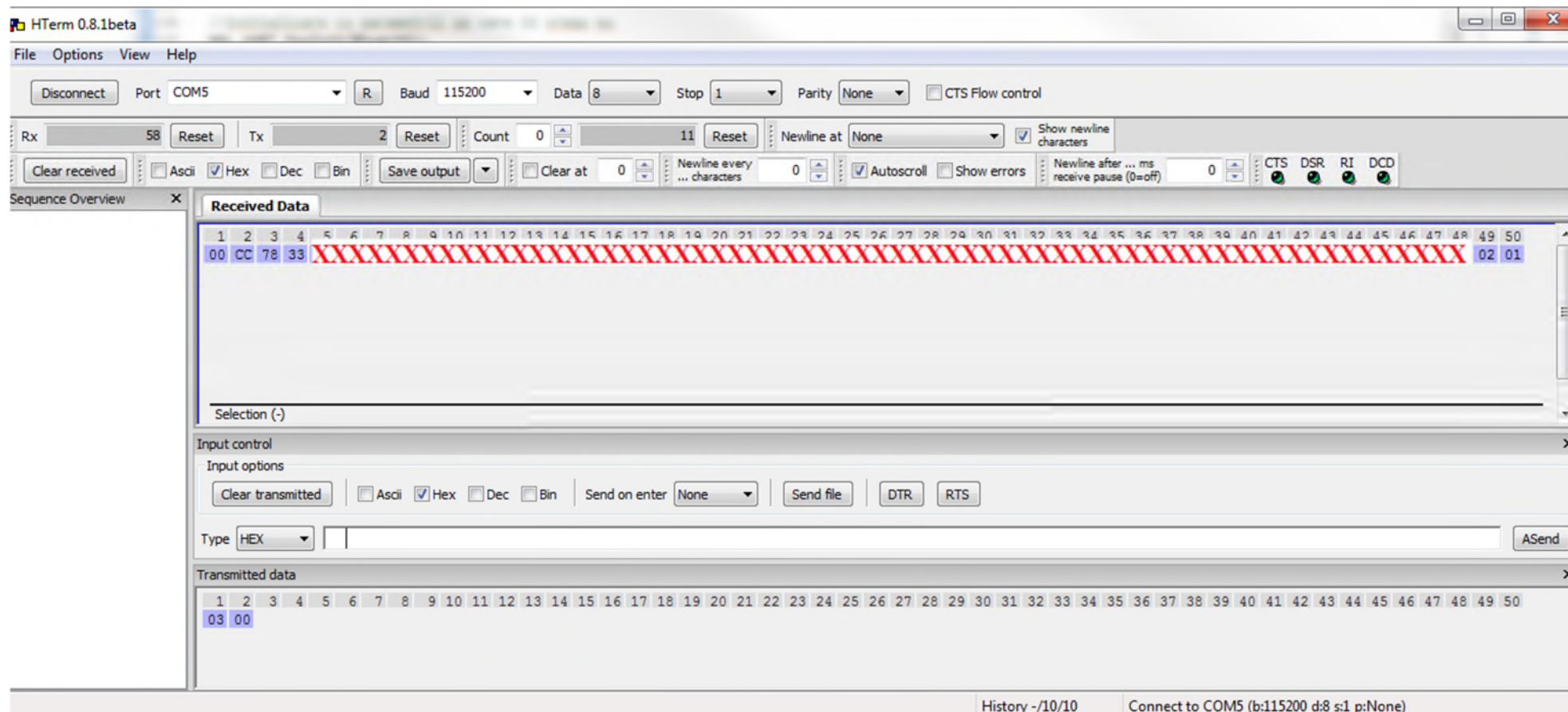
B.4. Configurarea frecvenței semnalului de ceas la 7 MHz



B.5. Configurarea frecvenței semnalului de ceas la 9 MHz



B.6. Transmiterea comenzii de ATR și recepționarea ATR cu ajutorul programului HTerm



B.7. Transmiterea comenzii pentru informațiile de identificare a cardului și răspunsul primit

Anexa C – Implementarea unor funcționalități

Tabelul C.1. Implementarea funcției de configurare a frecvenței

Explicații	Implementare
Declararea variabilelor de care avem nevoie.	<pre>void setFrequency(int freqparam) { RCC_PeriphCLKInitTypeDef UARTFreqConfig; int freq; int prescaler; int baudrate;</pre>
Deinițializarea interfeței pentru setarea parametrilor.	<pre> HAL_SMARTCARD_DeInit(&hsmartcard2); HAL_Delay(100);</pre>
Atribuirea parametrilor ce sunt la fel pentru configurarea tuturor frecvențelor	<pre> UARTFreqConfig.PeriphClockSelection = RCC_PERIPHCLK_USART3 RCC_PERIPHCLK_USART2; UARTFreqConfig.PLL2.PLL2RGE = RCC_PLL2VCIRANGE_3; UARTFreqConfig.PLL2.PLL2VCOSEL = RCC_PLL2VCOWIDE; UARTFreqConfig.PLL2.PLL2FRACN = 0; UARTFreqConfig.Usart234578ClockSelection = RCC_USART234578CLKSOURCE_PLL2;</pre>
Interpretarea celui de-al doilea byte al comenzii pentru configurarea frecvenței. Setarea configurărilor pentru 500 kHz	<pre> switch(freqparam) { case 0: freq = 500000; // Setari pentru 500kHz UARTFreqConfig.PLL2.PLL2M = 1; UARTFreqConfig.PLL2.PLL2N = 20; UARTFreqConfig.PLL2.PLL2Q = 10; prescaler = 16; break;</pre>
Setarea configurărilor pentru 1 MHz	<pre> case 1: freq = 1000000; // Setari pentru 1MHz UARTFreqConfig.PLL2.PLL2M = 1; UARTFreqConfig.PLL2.PLL2N = 25; UARTFreqConfig.PLL2.PLL2Q = 4; prescaler = 25; break;</pre>
Setarea configurărilor pentru 1,5 MHz	<pre> case 2: freq = 1500000; // Setari pentru 1.5MHz UARTFreqConfig.PLL2.PLL2M = 1; UARTFreqConfig.PLL2.PLL2N = 27; UARTFreqConfig.PLL2.PLL2Q = 4; prescaler = 18; break;</pre>
Toate celelalte configurații pentru frecvențele semnalului de ceas	<pre> : : :</pre>
Setarea configurărilor pentru 9 MHz	<pre> case 17: freq = 9000000; // Setari pentru 9MHz UARTFreqConfig.PLL2.PLL2M = 1; UARTFreqConfig.PLL2.PLL2N = 27; UARTFreqConfig.PLL2.PLL2Q = 4; prescaler = 3; break;</pre>

Setarea configurărilor pentru 9,5 MHz	<pre> case 18: freq = 9500000; // Setari pentru 9.5MHz UARTFreqConfig.PLL2.PLL2M = 1; UARTFreqConfig.PLL2.PLL2N = 19; UARTFreqConfig.PLL2.PLL2Q = 8; prescaler = 1; break; </pre>
Setarea configurărilor pentru 10 MHz	<pre> case 19: freq = 10000000; // Setari pentru 10MHz UARTFreqConfig.PLL2.PLL2M = 1; UARTFreqConfig.PLL2.PLL2N = 25; UARTFreqConfig.PLL2.PLL2Q = 2; prescaler = 5; break; </pre>
Implicit, frecvența este setată la 3,571 MHz	<pre> default: freq = 3571428; // Setari pentru 3.5714MHz UARTFreqConfig.PLL2.PLL2M = 1; UARTFreqConfig.PLL2.PLL2N = 25; UARTFreqConfig.PLL2.PLL2Q = 14; prescaler = 2; break; </pre>
Calculul baudrate-ului	<pre> } baudrate = freq / DF; </pre>
Configurarea semnalului de ceas pentru smartcard	<pre> HAL_RCCEx_PeriphCLKConfig(&UARTFreqConfig); </pre>
Reinițializarea interfeței pentru smartcard	<pre> USART2_Smartcard_Init (prescaler, baudrate); } </pre>

Funcția USART2_Smartcard_Init a fost implementată în cadrul acestui proiect pentru atribuirea parametrilor doriți, după primirea unei anumite comenzi. Interfața trebuie să fie deinițializată întotdeauna înainte de schimbarea frecvenței de lucru și inițializată după modificarea tuturor parametrilor necesari, pentru a ne asigura de funcționarea corectă după configurare.

```

void USART2_Smartcard_Init (int prescaler, int baudrate)
{
    hsmartcard2.Instance = USART2;
    hsmartcard2.Init.BaudRate = baudrate;
    hsmartcard2.Init.StopBits = SMARTCARD_STOPBITS_1_5;
    hsmartcard2.Init.Parity = SMARTCARD_PARITY_EVEN;
    hsmartcard2.Init.Mode = SMARTCARD_MODE_TX_RX;
    hsmartcard2.Init.CLKPolarity = SMARTCARD_POLARITY_LOW;
    hsmartcard2.Init.CLKPhase = SMARTCARD_PHASE_1EDGE;
    hsmartcard2.Init.CLKLastBit = SMARTCARD_LASTBIT_DISABLE;
    hsmartcard2.Init.OneBitSampling = SMARTCARD_ONE_BIT_SAMPLE_DISABLE;
    hsmartcard2.Init.Prescaler = prescaler;
    hsmartcard2.Init.GuardTime = 0;
    hsmartcard2.Init.NACKEnable = SMARTCARD_NACK_DISABLE;
    hsmartcard2.Init.TimeOutEnable = SMARTCARD_TIMEOUT_ENABLE;
    hsmartcard2.Init.TimeOutValue = 250;
    hsmartcard2.Init.BlockLength = 0;
    hsmartcard2.Init.AutoRetryCount = 0;
    hsmartcard2.Init.FIFOMode = SMARTCARD_FIFOMODE_DISABLE;
    hsmartcard2.Init.TXFIFOThreshold = SMARTCARD_TXFIFO_THRESHOLD_1_8;

```

```
hsmartcard2.Init.RXFIFOThreshold = SMARTCARD_RXFIFO_THRESHOLD_1_8;
hsmartcard2.AdvancedInit.AdvFeatureInit = SMARTCARD_ADVFEATURE_NO_INIT;
HAL_SMARTCARD_Init(&hsmartcard2);
```

```
}
```

Tabelul C.2. Implementarea funcției de configurare a tensiunii

Explicații	Implementare
Dezactivarea ieșirilor interfeței	<pre>void setVoltage(int voltageparam) { HAL_GPIO_WritePin(GPIOF, NCMDVCC_PF9_Pin, GPIO_PIN_SET); switch (voltageparam) { case 1: //1.8V HAL_GPIO_WritePin(GPIOF, VCC_SEL2_PF7_Pin, GPIO_PIN_SET); HAL_Delay(50); HAL_GPIO_WritePin(GPIOF, VCC_SEL1_PF6_Pin, GPIO_PIN_RESET); break; case 2: //3V HAL_GPIO_WritePin(GPIOF, VCC_SEL1_PF6_Pin, GPIO_PIN_SET); HAL_Delay(50); HAL_GPIO_WritePin(GPIOF, VCC_SEL2_PF7_Pin, GPIO_PIN_RESET); break; case 3: //5V HAL_Delay(50); HAL_GPIO_WritePin(GPIOF, VCC_SEL1_PF6_Pin, GPIO_PIN_SET); HAL_GPIO_WritePin(GPIOF, VCC_SEL2_PF7_Pin, GPIO_PIN_SET); break; default: HAL_GPIO_WritePin(GPIOF, NCMDVCC_PF9_Pin, GPIO_PIN_SET); //0V } HAL_GPIO_WritePin(GPIOF, NCMDVCC_PF9_Pin, GPIO_PIN_RESET); }</pre>
Configurarea tensiunii la 1,8 V	
Configurarea tensiunii la 3 V	
Configurarea tensiunii la 5 V	
Implicit, tensiunea ieșirilor este de 0 V	
Reactivarea interfeței analogice	

Tabelul C.3. Implementarea funcției pentru *Cold Reset*

Explicații	Implementare
Pentru a opri alimentarea către DUT am oprit interfața analogică	<pre> void ColdReset () { HAL_GPIO_WritePin(GPIOF, NCMDVCC_PF9_Pin, GPIO_PIN_SET); HAL_Delay(100); HAL_GPIO_WritePin(GPIOF, NCMDVCC_PF9_Pin, GPIO_PIN_RESET); HAL_GPIO_TogglePin(GPIOF, VCC_SEL2_PF7_Pin); HAL_Delay(100); HAL_GPIO_TogglePin(GPIOF, VCC_SEL2_PF7_Pin); } </pre>
După oprirea interfeței trebuie să fie efectuată o tranziție pe unul dintre pinii de selecție, așa că am negat valoarea logică a pinului pentru ca, în final, să ajung la aceeași tensiune setată, ca înainte de apelul funcției	

Pentru gestionarea întreruperilor am folosit funcția pusă la dispoziție de HAL, HAL_SMARTCARD_IRQHandler, care verifică ce tip de întrerupere pe interfața pentru smartcard a apărut și ia decizia să recepționeze sau să transmită date, să oprească transmiterea și să treacă în modul de recepționare (dacă întreruperea este de transmisiune completă), și altele. Rutina implementată apelează această funcție și este prezentată în continuare:

```

void USART2_IRQHandler(void)
{
    hsmartcard2.RxState = HAL_SMARTCARD_STATE_BUSY_RX;
    HAL_SMARTCARD_IRQHandler(&hsmartcard2);

    if ((hsmartcard2.Instance->CR1 & USART_CR1_RE))
    {
        HAL_UART_Transmit(&huart6, pRXDataUart2+contor_buffer_sc2, 1, 0xFF);
        *(pRXDataUart2+contor_buffer_sc2) = 0;
        contor_buffer_sc2 =
(contor_buffer_sc2+1)%(sizeof(pRXDataUart2)/sizeof(pRXDataUart2[0]));
    }

    hsmartcard2.RxState = HAL_UART_STATE_READY;
}

```

De fiecare dată când microcontrolerul recepționează ceva de la DUT, trebuie să trimită mesajul către PC.

Rutina de întrerupere pentru UART6 este asemănătoare cu cea pentru UART2, în afară de faptul că, după ce datele recepționate sunt puse într-un vector, acele date sunt interpretate în bucla principală – pentru interpretarea comenzii.

Anexa D – Afișarea graficelor cu ajutorul Matlab

Codul sursă implementat pentru afișarea ATR-ului:

```
close all
```

```

clear all
clc

% este incarcata fisierul .mat cu date
load ATR.mat

% pentru canalul 0
j=1; % indexul 1 apartine canalului 0
ch0 = digital_channel_0; % extragerea canalului

CH0 = [];
for i = 1 : 1 : length(ch0);
    CH0 = [CH0; digital_channel_initial_bitstates(j)*ones(ch0(i),1)];
    digital_channel_initial_bitstates(j)~=digital_channel_initial_bitstates(j);
end

% pentru a avea timpul pe axa 0x
Ts=1/digital_sample_rate_hz;
t=0:Ts:(length(CH0)-1)*Ts;

subplot(4,1,3), plot(t, CH0, 'Linewidth', 2), grid
axis( [ 0 length(CH0)*Ts 0 1.2 ])
xlabel ('Timp [s]')
ylabel ('RST')
title ('Semnalul de resetare')
ax = gca;
ax.TitleFontSizeMultiplier = 1.5;
ax.FontSize = 14;
hold on

% pentru canalul 1
j=2; % indexul 2 apartine canalului 1
ch1=digital_channel_1; % extragerea canalului

CH1=[];
for i=1:1:length(ch1);
    CH1=[CH1; digital_channel_initial_bitstates(j)*ones(ch1(i),1)];
    digital_channel_initial_bitstates(j)~=digital_channel_initial_bitstates(j);
end

Ts=1/digital_sample_rate_hz;
t=0:Ts:(length(CH1)-1)*Ts;

subplot(4,1,1), plot(t, CH1, 'Linewidth', 2), grid
axis( [ 0 length(CH1)*Ts 0 1.2 ])
xlabel ('Timp [s]')
ylabel ('VCC')
title ('Vensiunea de alimentare')
ax = gca;
ax.TitleFontSizeMultiplier = 1.5;
ax.FontSize = 14;

% pentru canalul 2 - semnalul de ceas
j=3; % indexul 3 apartine canalului 2
ch2=digital_channel_2; % extragerea canalului

CH2=[];
for i=1:1:length(ch2);
    CH2=[CH2; digital_channel_initial_bitstates(j)*ones(ch2(i),1)];
    digital_channel_initial_bitstates(j)~=digital_channel_initial_bitstates(j);
end

Ts=1/digital_sample_rate_hz;

```

```

t=0:Ts:(length(CH2)-1)*Ts;

subplot(4,1,2), plot(t, CH2), grid
axis( [ 0 length(CH2)*Ts 0 1.2 ])
xlabel ('Timp [s]')
ylabel ('CLK')
title ('Semnalul de ceas')
ax = gca;
ax.TitleFontSizeMultiplier = 1.5;
ax.FontSize = 14;

% pentru canalul 3 semnalul I/O
j=4; % indexul 4 apartine canalului 3
ch3=digital_channel_3; % extragerea canalului

CH3=[];
for i=1:length(ch3);
    CH3=[CH3; digital_channel_initial_bitstates(j)*ones(ch3(i),1)];
    digital_channel_initial_bitstates(j)=~digital_channel_initial_bitstates(j);
end

Ts=1/digital_sample_rate_hz;
t=0:Ts:(length(CH3)-1)*Ts;

subplot(4,1,4), plot(t, CH3, 'Linewidth', 2), grid
axis( [ 0 length(CH0)*Ts 0 1.2 ])
xlabel ('Timp [s]')
ylabel ('I/O')
title ('Semnalul de intrare/iesire - secventa ATR')
ax = gca;
ax.TitleFontSizeMultiplier = 1.5;
ax.FontSize = 14;

```

Pentru afișarea șirului de caractere de identificare a cardului:

```

close all
clear all
clc

% este incarcat fisierul .mat cu date
load info.mat

% pentru canalul 0
j=1; % indexul 1 apartine canalului 0
ch0 = digital_channel_0; % extragerea canalului

CH0 = [];
for i = 1 : 1 : length(ch0);
    CH0 = [CH0; digital_channel_initial_bitstates(j)*ones(ch0(i),1)];
    digital_channel_initial_bitstates(j)=~digital_channel_initial_bitstates(j);
end

% pentru a avea timpul pe axa 0x
Ts=1/digital_sample_rate_hz;
t=0:Ts:(length(CH0)-1)*Ts;

subplot(3,1,2), plot(t, CH0, 'Linewidth', 2), grid
axis( [ 0 length(CH0)*Ts 0 1.2 ])
xlabel ('Timp [s]')
ylabel ('RST')
title ('Semnalul de resetare')

```

```

ax = gca;
ax.TitleFontSizeMultiplier = 1.5;
ax.FontSize = 14;
hold on

% pentru canalul 1
j=2; % indexul 2 apartine canalului 1
ch1=digital_channel_1; % extragerea canalului

CH1=[];
for i=1:1:length(ch1);
    CH1=[CH1; digital_channel_initial_bitstates(j)*ones(ch1(i),1)];
    digital_channel_initial_bitstates(j)~=digital_channel_initial_bitstates(j);
end

Ts=1/digital_sample_rate_hz;
t=0:Ts:(length(CH1)-1)*Ts;

subplot(3,1,1), plot(t, CH1, 'Linewidth', 2), grid
axis( [ 0 length(CH1)*Ts 0 1.2 ])
xlabel ('Timp [s]')
ylabel ('VCC')
title ('Tensiunea de alimentare')
ax = gca;
ax.TitleFontSizeMultiplier = 1.5;
ax.FontSize = 14;

% pentru canalul 3 semnalul I/O
j=4; % indexul 4 apartine canalului 3
ch3=digital_channel_3; % extragerea canalului

CH3=[];
for i=1:1:length(ch3);
    CH3=[CH3; digital_channel_initial_bitstates(j)*ones(ch3(i),1)];
    digital_channel_initial_bitstates(j)~=digital_channel_initial_bitstates(j);
end

Ts=1/digital_sample_rate_hz;
t=0:Ts:(length(CH3)-1)*Ts;

subplot(3,1,3), plot(t, CH3, 'Linewidth', 2), grid
axis( [ 0 length(CH0)*Ts 0 1.2 ])
xlabel ('Timp [s]')
ylabel ('I/O')
title ('Semnalul de intrare/iesire - sirul de caractere de identificare a
cardului')
ax = gca;
ax.TitleFontSizeMultiplier = 1.5;
ax.FontSize = 14;

```