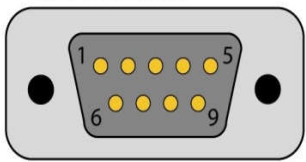


Aplicație C# pentru controlul modului radio FM RDA5807M

Modulul RDA5807M se alimentează la 3V, comunică prin interfața I2C la frecvențe de până la 400 kHz și oferă semnal audio stereo pe impedanță de 32 Ω . Aplicațiile standard în care este folosit prevăd controlul acestuia cu un microcontroler, dar eu am folosit în acest proiect o aplicație C# form care, prin intermediul portului serial, emulează protocolul I2C.

Standardul RS232C definește caracteristicile electrice ale unei interfețe dintre un echipament numeric, denumit Data Terminal Equipment (DTE) și un modem, denumit Data Communications Equipment (DCE).

DB9M Connector		RS232 Pin Out		Semnificație	
		Pin #	Signal		
		1	DCD	Data Carrier Detect	
		2	RX	Recepție date	
		3	TX	Emisie date	
		4	DTR	Data Terminal Ready (terminal de date gata?)	
		5	GND	Ground	
		6	DSR	Data Set Ready (terminal de date gata)	
		7	RTS	Request to Send (cerere de emisie)	
		8	CTS	Clear to Send (gata de emisie)	
		9	RI	Ring Indicator	

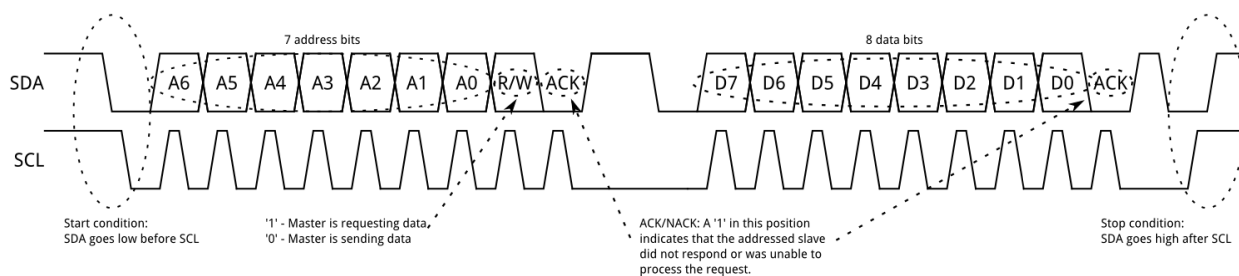
Datele circulă prin liniile Tx și Rx, iar perechile de semnale RTS/CTS și DTR/DSR au rolul de a implementa un protocol hardware de comunicație (handshake).

Interfața RS232 poate funcționa asincron sau sincron. În cazul comunicației asincrone (fără tact) transmițătorul emite mai întâi un bit de start prin care anunță receptorul că urmează un caracter. După bitul de start transmițătorul pune pe linia Tx caracterul, începând cu bitul cel mai semnificativ. Fiecare bit este menținut pe linie o durată bine stabilită de timp. Receptorul eșantionează linia Tx la momente de timp care se situează cât mai aproape de centrul momentului de timp afectat fiecărui bit. Acest lucru se întâmplă numai dacă emițătorul și receptorul lucrează cu aceeași viteză (au același baud rate). Codul folosit pentru caracter este codul ASCII (7 biți). După biții caracterului urmează bitul de paritate (opțional), iar ultimii biți sunt cei de stop, care dau timp receptorului să asambleze într-un cuvânt biții recepționați și să se pregătească pentru recepționarea unui nou caracter.

Nivelele logice sunt de +5 -> +25V pentru 0 logic și -5 -> -25V pentru 1 logic, ceea ce duce la incompatibilitate în interfațarea directă cu dispozitive cu nivele TTL (de exemplu interfața UART de la microcontrolere) dar există circuite specializate care realizează conversia de nivel (MAX232).

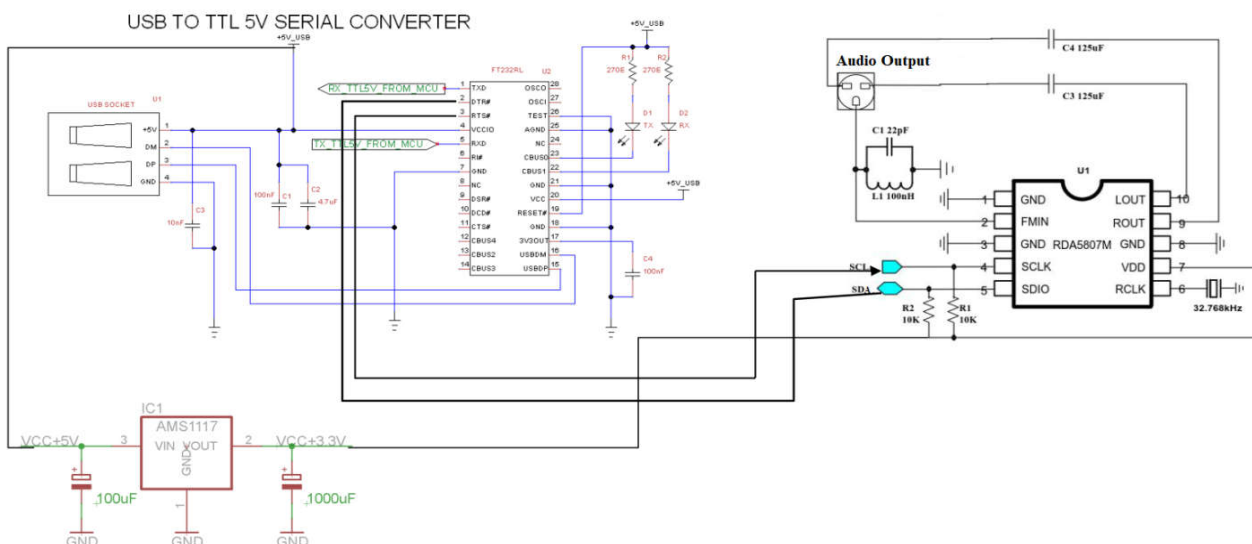
Deoarece interfața serială nu mai este foarte mult utilizată în echipamentele de calcul moderne uzuale (PC, laptop) producătorii au renunțat să mai includă acest port. În schimb, se găsesc adaptoare USB-UART, care oferă semnale logice de nivel TTL, de exemplu modulul USB to Serial care include circuitul FT232R.

transmise, condițiile de start și de stop, biții de citire/scriere și biții de ACK/NACK (acknowledge). Într-o comunicație master-slave linia SDA este bidirecțională (pentru a permite atât citirea cât și scrierea), iar SCL este unidirecțională (masterul asigură clock-ul). Comunicația este inițiată de master prin generarea condiției de start (SDA trece din HIGH în LOW în timp ce SCL este în starea HIGH), urmată de adresa dispozitivului slave. Dacă bitul 0 al adresei a fost setat 0, masterul va executa o scriere, în caz contrar va face o citire. Fiecare dispozitiv slave compară adresa primită cu cea proprie, iar cel vizat pune linia SDA în starea LOW pe durata unui bit (ACK), apoi master-ul trimite unul sau mai mulți octeți de date. După fiecare octet, masterul așteaptă ACK de la slave. După terminarea transmiterii datelor, masterul inițiază secvența de stop adică trece linia SDA din LOW în HIGH în timp ce SCL este în HIGH.



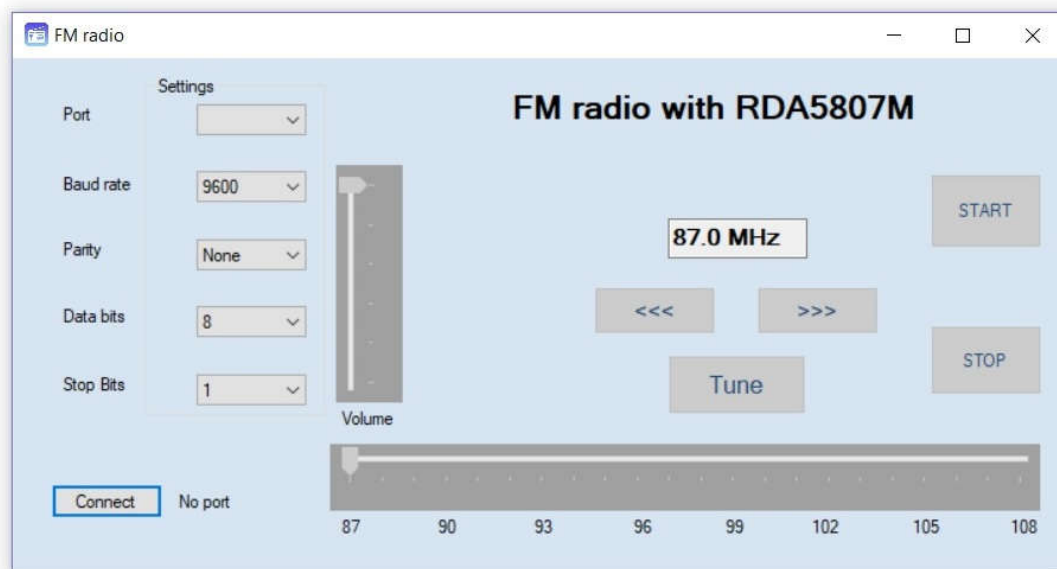
Se observă că tranziția dintre biții trimiși trebuie să se realizeze numai atunci când SCL este în starea LOW, deoarece orice tranziție între stările HIGH-LOW în momentul în care SCL este HIGH este interpretată drept semnal de control (condiție de start sau de stop).

Modulul USB to Serial oferă și o tensiune de 3.3V, dar din păcate, aceasta nu oferă decât 50 mA insuficient pentru alimentarea modului radio, de aceea am folosit un stabilizator de tipul AMS1117 care preia tensiunea de 5V și îl alimentează cu 3.3V. Conexiunea RTS este legată la intrarea de clock a modului radio (SCL), iar DTR la intrarea de date (SDA).



Modulul FT232 va fi master, iar modulul radio slave. Implementarea I2C în acest proiect nu este completă, deoarece linia SDA ar trebui să fie bidirecțională. În consecință, dispozitivul slave nu va putea să transmit semnalul ACK și nu va putea transmite informații către master. Din aceeași cauză radioul nu poate oferi funcția de scan, deoarece nu poate transmite către master frecvența postului recepționat. O posibilă soluție pentru transformarea liniei SDA într-o linie bidirecțională ar fi folosirea conexiunii de tip input CTS sau DSR și a unor buferi de tip 74hc06, care să izoleze direcțiile de transmisie, adică o implementare asemănătoare porturilor de uz general (input/output) din microcontrolere.

Aplicația C# este dezvoltată în IDE-ul Microsoft Visual Studio Community 2017.



Partea din stânga a ferestrei oferă controale de tip *ComboBox* pentru selecția portului serial și a setărilor acestuia. Pentru acest proiect trebuie selectat doar portul potrivit, celelalte controale afectează doar liniile Tx și Rx, care nu sunt folosite. După selectarea portului se apasă butonul *Connect*, moment în care se activează butonul *START*. La apăsarea butonului *START* radioul pornește pe frecvența de 87 Mhz și devin active controalele de tip *TrackBar* volum și stabilirea frecvenței, precum și butoanele de reglare fină (*Down*, *Up*). Pentru acordul pe un post, se glisează trackbar-ul pe frecvența dorită, eventual se folosesc și butoanele pentru reglare fină, care ajustează frecvența cu pasul de 0.1 Mhz, apoi se apasă butonul *Tune*. Din trackbar-ul *Volume* se poate regla nivelul auditei iar pentru oprire se apasă butonul *STOP*, care va dezactiva controalele radioului.

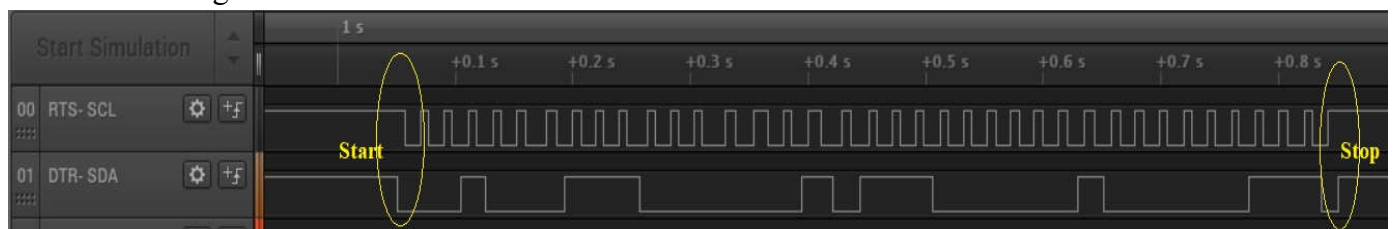
În regiunea de cod "I2C" din fișierul *Form1.cs* se află definițiile metodelor folosite pentru protocolul I2C:

- cele patru metode prin care se trec cele două linii în 0 logic și 1 logic - *sclLow()*, *sclHigh()*, *sdaLow()*, *sdaHigh()*. Se observă că logica folosită este negativă, de exemplu, pentru *sclLow*->ComPort.RtsEnable = true.
- metoda *startI2C()* care cuprinde secvența *sdaHigh-sclHigh-sdaLow*
- metoda *stopI2C()* cu secvența *sclLow-sdaLow-sclHigh-sdaHigh*
- metoda *waitAck()*
- metoda *write()* care trimite cei 8 biți către modulul radio

Adresa modului radio este 0x11 iar regiștri interni au dimensiunea de 16 biți. Pentru a transmite o comandă validă trebuie urmați pașii:

- se transmite secvența de start *startI2C()*
- se scrie octetul adresei + bitul de scriere 00010001 + 0(bit scriere) => 00100010 (0x22)
- se “așteaptă” confirmarea - *waitAck()*
- se scrie adresa registrului care trebuie accesat
- se trimite primul octet al registrului-cel mai puțin semnificativ
- se transmite *waitAck()*
- se trimite al doilea octet al registrului-cel mai semnificativ
- se transmite *waitAck()*
- se transmite secvența de stop *stopI2C()*

Secvențele de start și stop sunt vizibile foarte clar în capturile pe care le-am realizat cu analizorul logic.



Pentru a porni radioul trebuie accesat registrul 02H, care cuprinde pe lângă bitul de oprire/pornire și biți pentru selectare mod mono/stereo, bass boost, reset, etc. În registrul 03H biții 15:6 sunt folosiți pentru setarea frecvenței, bitul 4 pentru acordarea pe frecvență, biții 3:2 pentru selectarea benzii. Biții 3:0 din registrul 05H sunt folosiți pentru setarea volumului.

Bibliografie

1. Yury Magda - Complete practical measurement systems using a PC, Elektor International Media, 2009
2. <https://forum.arduino.cc/index.php?topic=221368.0>
3. http://cxem.net/tuner/files/tuner84_RDA5807M_datasheet_v1.pdf (datasheet RDA 5807M)