UNIVERSITATEA „POLITEHNICA” din BUCUREŞTI
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei

PROIECT 2
S10. STM32F103xx_USB

Profesor coordonator:
Adrian Florin Păun

Grupa 432C
Badea Luana
Zahiu Daniel-Mihai

Bucureşti 2021

CUPRINS

SCURTA DESCREIRE A TEMEI

S10. STM32F103xx_USB. Descriere functionare USB, blocuri USB, endpoints, IN packets, out& setup packets, alte operatii pe USB (transferul controlului, endpoint cu dublu buffer, transfer isosincron, suspendare si reluare evenimente). Scurta descriere a registrilor USB (comun, de intrerupere, USB_FNR, buffer, registrii de endpoints).

INTRODUCERE USB

Perifericul USB implementează o interfață între o magistrală USB 2.0 de mare viteză și **magistrala** APB1.
Sunt acceptate suspendarea / reluarea USB, ceea ce permite oprirea clock-urilor dispozitivului pentru a obtine un consum redus.

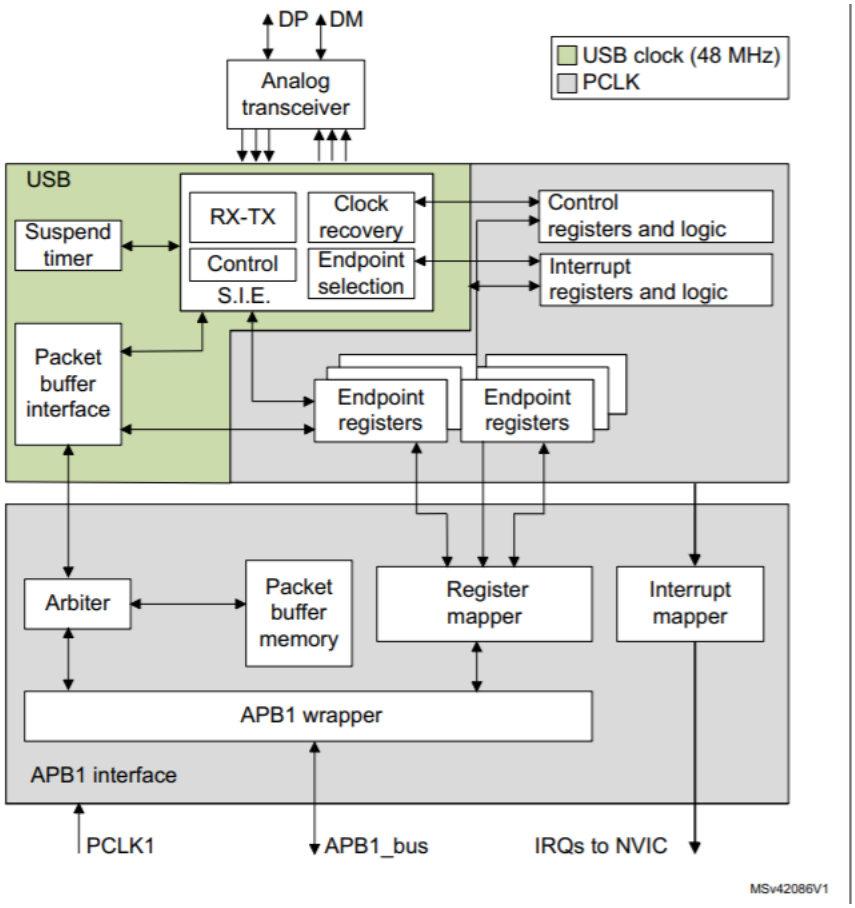
Caracteristici principale ale USB-ului

- Specificație USB versiunea 2.0 compatibilă cu viteză maximă;
- Număr configurabil de endpoints de la 1 la 8;
- Generarea verificării ciclului de redundanță (CRC), codificare/decodare inversă fără revenire la zero (NRZI) și bit-stuffing;
- Suport pentru transferuri izocrone;
- Operații de suspend/resume USB;

Pe dispozitivele cu densitate mică, medie, mare și XL, USB și CAN conferă o memorie SRAM de 512 biti pentru transmisie și recepție de date, astfel încât acestea nu pot fi utilizate simultan (memoria RAM partajată este accesată exclusiv prin CAN și USB).

USB-ul și CAN pot fi utilizate în aceeași aplicație, dar nu în același timp.

Descrierea functionala a USB-ului



SCHEMA DE BLOC A PERIFERICULUI USB

Perifericul USB oferă o conexiune compatibilă USB între computerul gazdă și funcția implementată de microcontroler. Transferul de date între computerul gazdă și memoria de sistem apare printr-un pachet de **memorie buffer** care poate fi accesat direct de către perifericul USB. Dimensiunea acestei **memorii buffer** alocate trebuie să fie în concordanță cu numărul de endpoints utilizate și a dimensiunii maxime a pachetului. Această **memorie** are dimensiunea de 512 octeți și până la 16 endpoints mono-direcționale (sau 8 bidirecționale) pot fi utilizate. Formatarea tranzacției este realizată de hardware, inclusiv generarea CRC și verificarea.

Fiecare endpoint este asociat cu un bloc de descriere care indică locul în care este localizat endpoint-ul, zona de memorie aferentă, cât de mare este sau câți octeți trebuie să fie transmiși.

Când un simbol pentru o pereche validă funcție/**endpoint** este recunoscut de către perifericul USB, are loc transferul de date aferent (dacă este necesar și dacă **endpoint-ul** este configurat). Datele **stocate** de perifericul USB sunt încărcate într-un registru intern de 16 biți și accesul la memorie **se efectuează de către bufferul alocat**. Când toate datele au fost transferate, dacă este necesar, este generat (sau așteptat) pachetul handshake adecvat prin intermediul USB, în funcție de direcția de realizare a transferului.
La sfârșitul procesului, este generată o întrerupere specifică **endpoint-ului**, citindu-se statusul regiștrilor și (/sau) se utilizează diferite comenzi predefinite de răspuns la întrerupere. Microcontrolerul poate determina:

- **endpoint-ul** care trebuie oferit;
- tipul solicitării care a avut loc și dacă există erori (cum ar fi bit stuffing, CRC,protocol, lipsă ACK, overrun, underrun).

Atenție specială este acordată transferurilor izocrone și transferurilor **bulk** cu randament ridicat, implementării unei utilizări a unui buffer dublu, care permite să aibă întotdeauna un buffer disponibil pentru p**erifericul USB** în timp ce microcontrolerul îl folosește pe celălalt.
Unitatea poate fi plasată în modul de consum redus de energie (modul SUSPEND), **scriind în registrul de control, ori de câte ori este necesar**. În acest moment, este evitată orice disipare a puterii statice și **USB clock-ul** poate fi încetinit sau oprit. Detectarea unor evenimente la intrările USB, în timp ce se află în modul de consum redus, trezește dispozitivul asincron. O sursă specială de întrerupere poate fi conectată direct **la un wakeup line** pentru a permite imediat sistemului să repornească **generarea clock-ului** și/sau sa ofere suport pentru pornirea/oprirea instantanee a clock-ului**.**

Descrierea blocurilor USB-ului

Perifericul USB implementează toate caracteristicile legate de interfața USB, care includ următoarele blocuri:

- Motor de interfață serial (SIE): Funcțiile acestui bloc includ: sincronizarea tiparului de recunoaștere, bit-stuffing, generarea și verificarea CRC, verificarea/generarea PID și evaluarea handshake-ului. Trebuie să se conecteze cu transceiver-ul USB-ului și să utilizeze bufferele virtuale furnizate de interfața bufferului de pachete pentru stocarea locală a datelor. Aceast element generează, **de asemenea, semnale în funcție de evenimentele detectate la perifericul USB (cum ar fi Start of Frame (SOF), USB_Reset, erori de date etc.) și**

evenimentele conexe de la endpoint, cum ar fi sfârșitul transmisiei corecte sau sfârșitul recepției corecte a unui pachet; aceste semnale sunt apoi utilizate pentru a genera întreruperi.

- Timer-ul: acest bloc generează un impuls blocat de ceas la începutul cadrului și **detectează o întrerupere globala (de la host) când nu a fost primit niciun trafic timp de 3 ms.**
- Interfață **buffer** de pachete: Acest bloc gestionează memoria locală implementând **un set de buffere** într-un mod flexibil, atât pentru transmisie, cât și pentru recepție. **Poate alege corect buffer-ul** conform cererilor venite de la SIE și le poate localiza în adresele de memorie indicate de registrele endpoint. **Incrementează adresa până la sfârșitul pachetului după fiecare schimbare efectuată, contorizând numărul de octeți schimbați și împiedicând bufferul să depășească capacitatea maximă.**
- Registre legate **de endpoint**: fiecare **endpoint** are un registru asociat care conține **tipul endpoint-ului** și starea sa actuală. Pentru **end-pointurile** mono-direcționale / cu un **singur buffer**, un singur registru poate fi utilizat pentru a implementa două **endpoint-uri** distincte. Numarul registrelor este de 8, permițând până la 16 endpoint-uri monodirecționale **sau până la 7 bidirecționale (în orice combinație). De exemplu, perifericul USB poate fi programat pentru a avea 4 endpointuri bidirectionale și 8 endpoint-uri monoparalale/mono-direcționale.**

- Registre de control: Acestea sunt registrele care conțin informații despre **starea fișierului perifericului USB și de regulă poate impune unele evenimente USB, cum ar fi resume si power down.**
- Registre de întrerupere: acestea conțin interrupt masks și o evidenta **a evenimentelor. Ele pot fi folosite pentru a solicita un motiv de întrerupere, statusul întreruperii sau pentru a sterge o solicitare de intrerupere**.

Perifericul USB este conectat la magistrala APB1 printr-o interfață APB1, care conține următoarele blocuri:

- Pachet de memorie: Aceasta este memoria locală care conține fizic **pachetele buffer.** Aceasta poate fi folosita de interfața Buffet Buffet, care creează structura de date și poate să fie accesata direct de aplicatia software. Dimensiunea pachetului de memorie este de 512 octeți, structurați ca 256 de cuvinte pe 16 biți.
- Arbiter: acest bloc acceptă cererile de memorie provenite de la magistrala APB1 și de la interfața USB. Rezolvă conflictele acordând prioritate

acceselor APB1, în timp ce rezerva întotdeauna jumătate din lățimea de bandă a memoriei pentru a finaliza toate transferurile USB. Aceasta schema time-duplex implementează un SRAM virtual cu două porturi care permit accesul la memorie, în timp ce se întâmplă o tranzitie USB. Transferurile de mai multe cuvinte APB1 de orice lungime sunt permise și de această schemă.

- Register Mapper: Acest bloc colectează diferitele registre la nivel de octeți și la nivel de biți ale perifericului USB într-un set de cuvinte structurat pe 16 biți, adresat de APB1.
- APB1 Wrapper: Acesta oferă o interfață către APB1 pentru memorie și registrii. Aceasta mapează, de asemenea, întregul periferic USB în spațiul de adrese APB1.
- Interrupt Mapper: Acest bloc este utilizat pentru a selecta modul în care pot fi generate intreruperi la anumite posibile evenimente USB și le standardizeaza pe trei linii diferite ale NVIC:

- USB cu prioritate redusă de intrerupere (Canal 20): declanșată de toate evenimentele USB (transfer corect, resetare USB etc.). Firmware-ul trebuie să verifice sursa de întrerupere înaintea efectuarii întreruperii.

- Întrerupere USB cu prioritate ridicată (Canal 19): Declanșată numai de un eveniment de transfer corect pentru transferul izocron și dublu-buffer pentru a atinge cel mai înalt nivel posibil al ratei de transfer.

- Întrerupere USB de wakeup (Canal 42): declanșată de evenimentul de trezire de pe USB Suspend Mode.

Inițializarea Endpoints-urilor

Primul pas pentru a inițializa un endpoint este reprezentat de scrierea valorilor adecvate pentru registrii ADDRn_TX / ADDRn_RX astfel încât perifericul USB să găsească datele care trebuie transmise si care sunt deja disponibile, iar datele care trebuie primite pot fi stocate. Biții EP_TYPE din registrul USB_EPnR trebuie să fie setați în funcție de tipul endpoint-ului, eventual folosind bitul EP_KIND pentru a activa orice caracteristică specială necesară. Pe partea de transmisie, endpoint-ul trebuie să fie activat folosind biții STAT_TX din registrul USB_EPnR, iar COUNTn_TX trebuie sa fie inițializat. Pentru recepție, biții STAT_RX trebuie să fie setați pentru a activa recepția și COUNTn_RX trebuie să fie scris cu dimensiunea bufferului alocat folosind campurile BL_SIZE și NUM_BLOCK.

Endpoint-urile unidirecționale, cu excepția celor izocrone și a celor cu double-buffered endpoints, trebuie să inițializeze numai biți și registre legate de direcția acceptată. Odată ce transmisia și / sau recepția sunt activate, registrul USB_EPnR și locațiile ADDRn_TX / ADDRn_RX, COUNTn_TX / COUNTn_RX (respectiv), nu trebuie modificate de catre software-ul aplicației, deoarece hardware-ul își poate schimba valoarea în acest timp. Când operațiunea de transfer de date este finalizată, aceasta este notificată printr-un eveniment de întrerupere CTR, si poate fi accesata din nou pentru a reactiva o nouă operatiune.

Pachete IN (transmisie de date)

Când primiți un IN token packet, dacă adresa primită corespunde unei configurări valide a unui endpoint, perifericul USB accesează conținutul din ADDRn_TX și COUNTn_TX din interiorul tabelului de intrari al buffer-ului descriptiv al endpoint-ului adresat. Conținutul acestor locații este stocat în registrele sale interne de 16 biți ADR și COUNT. Memoria pachetului este accesată din nou pentru a citi primul cuvânt care trebuie transmis și începe să trimită un DATA0 sau DATA1 PID potrivit bitului DTOG_TX din registrul USB_EPnR. Când PID-ul este finalizat, primul octet din cuvant, citit din memoria buffer, este încărcat în registrul de schimbare de output pentru a fi transmis pe magistrala USB. După ce ultimul octet de date este transmis, CRC calculat este trimis. Dacă endpoint-ul adresat nu este valid, un pachet NAK sau STALL handshake este trimis în locul pachetului de date, în conformitate cu biți STAT_TX din registrul USB_EPnR.

Registrul intern ADDR este utilizat ca un pointer la locația de memorie buffer curenta în timp ce COUNT este utilizat pentru a număra numărul de octeți rămași care trebuie transmiși. Fiecare cuvânt citit din memoria buffer de pachete este transmis prin magistrala USB începând de la cel mai putin semnificativ octet. Memoria buffer de transmisie este citită pornind de la adresa indicată de ADDRn_TX pentru COUNTn_TX / 2 cuvinte. Dacă un pachet transmis este compus dintr-un numărul impar de octeți, numai jumătatea inferioară a ultimului cuvânt accesat va fi utilizata.

La primirea receptionarii ACK de către host, registrul USB_EPnR este actualizat în felul următor: bitul DTOG_TX este comutat, endpoint-ul este invalidat prin setarea STAT_TX=10 (NAK) și bitul CTR_TX este setat. Software-ul trebuie să identifice mai întâi endpoint-ul, care solicită atenția microcontrolerului prin examinarea biților EP_ID și DIR în registrul USB_ISTR.

OUT și SETUP packets (recepție date)

Aceste două pachete sunt manipulate de perifericul USB mai mult sau mai puțin în același mod; diferențele în manipularea pachetelor de configurare sunt detaliate în paragraful următor despre transferuri de control. Când primiți un OUT/SETUP PID, dacă adresa se potrivește cu o adresă validă a unui endpoint, perifericul USB accesează conținutul ADDRn_RX și COUNTn_RX locații din interiorul tabelului de intrării a bufferului descriptor al endpoint-ului adresat. Conținutul ADDRn_RX este stocat direct în registrul său intern ADDR. În timp ce COUNT este acum resetat și valorile câmpurilor de biți BL_SIZE și NUM_BLOCK, care sunt citite din conținutul lui COUNTn_RX sunt utilizate pentru a inițializa BUF_COUNT, un contor intern de 16 biți, care este folosit pentru a verifica starea de overrun a bufferului (toate aceste registre interne nu sunt accesibile prin software). Octeții de date primiți ulterior de perifericul USB sunt împachetați în cuvinte (primul octet primit este stocat ca octet cel mai puțin semnificativ) și apoi transferat la buffer-ul de pachete pornind de la adresa conținută în registrul ADR intern în timp ce BUF_COUNT este decrementat și COUNT este incrementat la fiecare transfer de octet.
În caz de CRC greșit sau alte tipuri de erori (încălcări ale bit-stuff-ului, erori de cadru, etc.), bitii de date sunt de asemenea copiați în bufferul de memorie al pachetelor, cel puțin până la punctul in care se efectueaza detectarea erorilor, dar pachetul ACK nu este trimis și, in acelasi timp, este setat bitul ERR în registrul USB_ISTR. Cu toate acestea, in mod obisnuit nu este necesară nicio acțiune din partea unui software în acest caz: perifericul USB isi face recovery de la erorile de recepție și rămâne pregatit pentru a efectua următoarea linie de comanda. Dacă endpoint-ul adresat nu este valid, un pachet handshake NAK sau STALL este trimis în locul ACK, în funcție de biți STAT_RX din registrul USB_EPnR și nu sunt scrise date în bufferele de memorie de recepție. Locațiile buffer-ului de memorie de recepție sunt scrise pornind de la adresa conținută în ADDRn_RX pentru un număr de octeți corespunzători lungimii pachetului de date primit, CRC inclusiv (adică lungimea sarcinii utile de date + 2) sau până la ultima locație de memorie alocată, definita de BL_SIZE și NUM_BLOCK, oricare dintre acestea survine mai întâi. În acest fel, perifericul USB nu scrie niciodată dincolo de sfârșitul zonei buffer de memorie de recepție alocata.

Când tranzacția este finalizată corect, prin trimiterea pachetului handshake ACK,registrul intern COUNT este copiat înapoi în locația COUNTn_RX din interiorul buffer-ului ce descrie tabel de intrare, lăsând câmpurile bl_size și NUM_BLOCK neafectate, care în mod normal, nu necesită să fie re-scrise, iar registrul USB_EPnR este actualizat în felul următor: bitul DTOG_RX este

comutat, endpoint-ul este invalidat prin setarea STAT_RX = '10 (NAK) și bitul CTR_RX este setat. Dacă tranzitia a eșuat din cauza erorilor sau a depasirii considitiei de overrun, niciuna dintre acțiunile enumerate anterior nu are loc. Aplicatia soft trebuie mai întâi sa identificați endpoint-ul, care solicită atenția microcontrolerului examinând bitii EP_ID și DIR în registrul USB_ISTR. Evenimentul CTR_RX este deservit in principal prin determinarea tipului de tranzitie (bit de configurare în Registrul USB_EPnR); aplicația software trebuie să șteargа flag bitul și să obțină numărul de octeți prin citirea locației COUNTn_RX în interiorul tabelului de intrării a bufferului aflat in legatura cu endpoint-ul ce trebuie procesat. După procesarea datelor primite, software-ul trebuie să stabilească STAT_RX biți la '11 (Valid) în USB_EPnR, permițând actiuni suplimentare. Cat timp biții STAT_RX sunt egali cu '10 (NAK), orice cerere OUT adresată acelui endpoint-ului este NAK, indicând o condiție de control al flow-ului: host-ul USB va încerca din nou actiunea până când reușește. Este obligatorie executarea succesiunii operațiilor menționate mai sus pentru a evita pierderea notificării unei a doua tranzacții adresate aceluiași endpoint imediat după cel care a declanșat întreruperea CTR.

Transferul Controlului

Transferurile de Control se fac dintr-o tranzitie de configurare, urmată de zero sau mai multe data stages, toate avand aceeași direcție, urmată de o etapă de stare (un transfer de zero-bytes în direcția inversa). Tranzacțiile de configurare sunt gestionate numai de endpoint-urile de control și sunt foarte asemănătoare cu cele OUT (recepție de date), cu excepția faptului că valorile bitilor DTOG_TX și DTOG_RX ai registrelor de tip endpoint adresate sunt setate la 1 și, respectiv, 0, pentru a inițializa transferul de control, și atât STAT_TX cât și STAT_RX sunt setate la '10 (NAK) pentru a permite software-ului să decidă dacă ulterior tranzacțiile trebuie să fie IN sau OUT, în funcție de conținutul de setup. Un endpoint de control trebuie sa verifice bitul SETUP în registrul USB_EPnR la fiecare eveniment CTR_RX pentru a distinge tranzacțiile normale de cele de configurare. Un dispozitiv USB poate determina numărul și directia etapelor de date prin interpretarea datelor transferate în etapa de setup, și este necesară pentru a bloca tranzacția în caz de erori. Pentru a face acest lucru, în toate etapele de date in in afara de ultima, direcția neutilizată ar trebui să fie setată să se oprească, astfel încât, dacă gazda inversează direcția de transfer prea curând, devine un STALL ca o etapă de stare.
În timp ce activați ultima etapă de date, direcția opusă trebuie setată pe NAK, astfel încât, dacă gazda inversează direcția de transfer (pentru a efectua stadiul de stare) imediat, este păstrată in așteaptă finalizarea operațiunii de control. Dacă operația de control este finalizată cu succes, software-ul va schimba NAK la

VALID, în caz contrar se va schimba cu STALL. În același timp, dacă etapa de stare va fi o ieșire, bitul STATUS_OUT (EP_KIND în registrul USB_EPnR) ar trebui să fie setat, astfel încât ar genera o eroare în cazul în care o tranzacție de stare este efectuată cu not-zero data. Când tranzacția de stare este deservită, aplicația șterge bitul STATUS_OUT și setează STAT_RX la VALID (pentru a accepta o nouă comandă) și STAT_TX la NAK (pentru a întârzia o posibilă etapă de stare imediat după următoarea configurare).
Deoarece specificația USB afirmă că un pachet de configurare nu poate fi interceptat cu un handshake diferit de ACK, logica USB nu permite unui endpoint de control să răspundă cu un pachet NAK sau STALL la un token de configurare primit de la gazdă.
Când biții STAT_RX sunt setați la '01 (STALL) sau '10 (NAK) și un token de configurare este primit, USB acceptă datele, efectuarea transferurilor de date necesare și trimite înapoi un handshake ACK. Dacă acel endpoint are încă o solicitare CTR_RX emisă anterior si recunoscuta inca de aplicație (adică bit-ul CTR_RX este încă setat dintr-o solicitare anterioara finalizata deja), USB rejecteaza tranzacția de configurare și nu răspunde cu niciun pachet handshake, indiferent de starea sa, simulând o eroare de recepție și forțând host-ul să trimita token-ul de configurare din nou. Acest lucru se face pentru a evita pierderea notificării unei tranzacție SETUP adresată aceluiași endpoint imediat după tranzacție, care a declanșat întreruperea CTR_RX.

Endpoints cu buffer dublu

Toate tipurile diferite de endpoints definite de standardul USB reprezintă diferite modele de trafic, și descriu cerințele tipice ale diferitelor tipuri de operațiuni de transferuri de date. Când porțiuni mari de date trebuie transferate între PC-ul gazdă și USB, bulk endpoint-ului este cel mai potrivit model. Acest lucru se datorează faptului că host-ul prioritizeaza actiunile bulk, astfel încât să umple toată lățimea de bandă disponibilă în cadru, maximizand rata de transfer atât timp cât funcția USB este gata să se ocupe de o tranzacție bulk adresată
ei. Dacă funcția USB este încă ocupată cu tranzacția anterioară, la sosirea următoarei comenzim aceasta va răspunde un handshake NAK și PC-ul gazdă va emite din nou aceeași tranzacție până când funcția USB este gata să o gestioneze, reducând rata de transfer datorită latimii de bandă ocupată de retransmisii. Din acest motiv, o caracteristică dedicată numită "bufferelor duble" poate fi utilizată cu bulk endpoints.

Când este activat double-buffering-ul, secvențierea comutării datelor este utilizată pentru a selecta care buffer este utilizat de perifericul USB pentru a efectua transferurile de date necesare, utilizând ambele zone de pachete de memorie

"transmisie" și "recepție" pentru a gestiona buffer swapping pe fiecare tranzacție de succes pentru a avea întotdeauna un buffer complet ce poate fi utilizat de către aplicație, în timp ce perifericul USB îl umple pe celălalt.

Deoarece managementul swapped buffer-ului necesită utilizarea tuturor celor patru tabele de descriere a locatiilor bufferelor care găzduiesc pointerul și lungimea bufferelor de memorie alocate, registrele USB_EPnR utilizate pentru implementarea bulk endpoints-urilor cu buffer dublu sunt obligate să fie folosite in mod unidirecțional. Prin urmare, o singură pereche de biți STAT trebuie setată la o valoare diferit de '00(Disabled): STAT_RX dacă double buffered endpoint este activat pentru recepție, STAT_TX dacă double buffered endpoint este activat pentru transmisie. În caz în care este necesar să aveți activate double buffered endpoint-urile atât pentru recepție, cât și pentru transmisie, trebuie utilizate două registre USB_EPnR. Pentru a exploata caracteristica de double-buffering și pentru a atinge cea mai mare rată de transfer posibilă, structura de control al debitului, descrisă în capitolele anterioare, trebuie modificată astfel incat pentru a comuta starea punctului final la NAK numai atunci când apare un buffer conflict între software-ul utilizat si perifericul USB, în loc să o facă la sfârșitul fiecărui tranzacție reusita. Bufferul de memorie care este utilizat în prezent de perifericul USB este definit de bitul DTOG legat de direcția endpoint-ului: DTOG_RX(bitul 14 al registrului USB_EPnR) pentru "recepția" double buffered endpoint sau DTOG_TX (bit 6 din registrul USB_EPnR) pentru "transmisia" double buffered endpoint. Deoarece în registrul USB_EPnR există doi biți DTOG, dar numai unul este utilizat de periferice USB pentru date și buffer sequencing (datorită constrângerii unidirecționale cerute de caracteristica double-buffering) altul poate fi folosit de software-ul aplicației pentru a arăta ce buffer este utilizat în prezent. Acest nou buffer flag se numește SW_BUF.
În tabelul următor este explicata corespondența dintre bitii registrului USB_EPnR si definiția DTOG / SW_BUF este explicată, pentru cazurile de double-buffered endpoints ("transmisie" și "recepție")

Buffer flag	'Transmission' endpoint	'Reception' endpoint
DTOG	DTOG_TX (USB_EPnRbit 6)	DTOG_RX (USB_EPnRbit 14)
SW_BUF	USB_EPnR bit 14	USB_EPnR bit 6

Memoria bufferului care este utilizat în prezent de perifericul USB este definit de

buffer flag-ului DTOG, în timp ce bufferul utilizat în prezent de software-ul aplicației este identificat de buffer flag-ul SW_BUF. Relația dintre valoarea buffer flag-ului și pachetul buffer utilizat este aceeasi în ambele cazuri, și este listat în tabelul următor.

Endpoint Type	DTOG	SW_BUF	Packet buffer used by the USB peripheral	Packet buffer used by the application software
IN	0	1	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.	ADDRn_TX_1 / COUNTn_TX_1 Buffer description table locations.
	1	0	ADDRn_TX_1 / COUNTn_TX_1 Buffer description table locations	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.
	0	0	None (1)	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.
	1	1	None (1)	ADDRn_TX_0 / COUNTn_TX_0 Buffer description table locations.
OUT	0	1	ADDRn_RX_0 / COUNTn_RX_0 Buffer description table locations.	ADDRn_RX_1 / COUNTn_RX_1 Buffer description table locations.
	1	0	ADDRn_RX_1 / COUNTn_RX_1 Buffer description table locations.	ADDRn_RX_0 / COUNTn_RX_0 Buffer description table locations.
	0	0	None (1)	ADDRn_RX_0 / COUNTn_RX_0 Buffer description table locations.
	1	1	None (1)	ADDRn_RX_1 / COUNTn_RX_1 Buffer description table locations.

Funcția de double-buffering pentru un bulk endpoint este activată de:

- Setarea câmpului de biți EP_TYPE la '00 în registrul său USB_EPnR, pentru a defini bulk endpoint-ul;
- Setarea bitului EP_KIND la '1 (DBL_BUF), în același registru.

Software-ul aplicației este responsabil pentru inițializarea biților DTOG și SW_BUF conform cu primul buffer care va fi utilizat; acest lucru trebuie făcut având în vedere proprietatea specială toggle-only pe care acesti doi biți o au. Sfârșitul primei tranzacții care are loc după ce s-a stabilit DBL_BUF, declanșează controlul special al flow-ului de double-buffered bulk endpoints, care este utilizat pentru toate celelalte tranzacții adresate acestui endpoint până când DBL_BUF rămâne setat. La sfârșitul fiecarei tranzacții, bitul CTR_RX sau CTR_TX al endpointului adresat registrului USB_EPnR este setat, în funcție de direcția activată. În același timp, bitul DTOG aflat în registrul USB_EPnR este hardware toggled, făcând schimbarea bufferului periferic USB complet independenta fata de software.

Software-ul aplicației poate înlocui întotdeauna controlul special al flow-ului implementat pentru double-buffered endpoints, scriind un statut explicit diferit de " 11 (Valid) în pereche de biți STAT din registrul USB_EPnR aferent.

Transferuri izocronice

Standardul USB acceptă periferice cu viteză maximă care necesită date fixe și exacte, definind acest tip de trafic ca fiind "Izocronic". Exemple de date tipice sunt: eșantioane audio, fluxuri video comprimate și, în general, orice un fel de date eșantionate care au cerințe stricte pentru acuratețea frecvenței livrate. Atunci când un endpoint este definit ca fiind "izocron", în timpul fazei de enumerare, host-ul alocă în cadru lățimea de bandă necesară și livrează exact un pachet de intrare sau ieșire pentru fiecare cadru, în funcție de direcția endpoint-ului. Pentru a limita cerințele de lățime de bandă, nu este posibilă retransmiterea tranzacțiilor eșuate pentru traficul Izocronic; acest lucru duce la faptul că o tranzacție izocronă nu are o fază de handshake și nici un pachet ACK nu este așteptat sau trimis după pachetul de date. Din același motiv, transferurile Izocronice nu suportă secvențierea comutării datelor și folosește întotdeauna DATA0 PID pentru a porni orice pachet de date.
Comportamentul Izocronic pentru un punct final este selectat prin setarea biților EP_TYPE la '10 în registrul USB_EPnR; deoarece nu există o fază de handshake, singurele valori valide pentru perechile de biți STAT_RX/STAT_TX sunt '00 (Disabled) și '11 (Valid), orice altă valoare va produce rezultatele care nu sunt conforme cu standardul USB.
Bufferul de memorie utilizat de perifericul USB este definit de bitul DTOG corespondend direcției endpoint-ului (DTOG_RX pentru endpoint-urile izocrone "recepție" , DTOG_TX pentru endpoint-urile izocrone "transmisie", ambele în registrele aferente USB_EPnR) conform tabelului de mai jos.

Endpoint type	DTOG bit value	Packet buffer used by the USB peripheral	Packet buffer used by the application software
IN	0	ADDRn_TX_0 / COUNTn_TX_0 buffer description table locations.	ADDRn_TX_1 / COUNTn_TX_1 buffer description table locations.
	1	ADDRn_TX_1 / COUNTn_TX_1 buffer description table locations.	ADDRn_TX_0 / COUNTn_TX_0 buffer description table locations.
OUT	0	ADDRn_RX_0 / COUNTn_RX_0 buffer description table locations.	ADDRn_RX_1 / COUNTn_RX_1 buffer description table locations.
	1	ADDRn_RX_1 / COUNTn_RX_1 buffer description table locations.	ADDRn_RX_0 / COUNTn_RX_0 buffer description table locations.

Suspendare si reluare evenimentele

O scurtă descriere a unei proceduri de suspendare tipice este furnizată mai jos, axată pe aspectele legate de caracteristicile USB ale softwareului utilizat, care răspunde la notificarea SUSP a perifericul USB:
1. Setați bitul SUSP în registrul USB_CNTR la 1. Această acțiune activează suspend mode în perifericul USB. De îndată ce suspend mode este activat, verificarea la recepție SOF este dezactivata pentru a evita orice alta intrerupere SUSPP care ar putea fii emisa în timp ce USB-ul este suspendat.
2. Eliminați sau reduceți orice consum static de energie în diferite blocuri ale perifericului USB..
3. Setați bitul lp_mode în registrul USB_CNTR la 1 pentru a elimina consumul de energie statică în transmițătoarele USB analogice, dar menținându-le capabile să detecteze activitatea de reluare.
4. Opțional opriți oscilatorul extern și dispozitivul PLL pentru a opri orice activitate din interiorul dispozitivului.

Când apare un eveniment USB în timp ce dispozitivul este în suspend mode, reluarea procedurii trebuie invocată pentru a restabili clock-urile nominale și pentru a redobandi comportamentul USB normal. O atenție deosebită trebuie acordată pentru a va asigura că acest proces nu durează mai mult de 10ms când evenimentul de wake-up este o secvență de resetare USB (consultați " Universal Serial Bus Specificație" pentru mai multe detalii). Începutul unei secvențe de reluare sau resetare, în timp ce perifericul USB este suspendat, șterge bitul LP_MODE în registrul USB_CNTR asincron.

In continuare este prezentata o listă de acțiuni pe care o procedure de resume ar trebui să o urmeze:
1. Opțional porniți oscilatorul extern și / sau dispozitivul PLL.
2. Stergeti bitul FSUSP din registrul USB_CNTR.
3. Dacă evenimentul de declanșare a procedurii de resume trebuie identificat, biți RXD și RXD din Registrul USB_FNR pot fi utilizati în conformitate cu tabelul de mai jos, in care sunt enumerate și acțiune software în toate cazurile. Dacă este necesar, sfârșitul secvenței de resume sau resetare poate să fie detectata prin monitorizarea stării biților menționați mai sus, verificând când ajunge la configurația "10", care reprezintă starea de bus inactiv.

[RXDP,RXDM] status	Wakeup event	Required resume software action
“00”	Root reset	None
“10”	None (noise on bus)	Go back in Suspend mode
“01”	Root resume	None
“11”	Not allowed (noise on bus)	Go back in Suspend mode

Un dispozitiv poate solicita ieșirea din suspend mode ca răspuns la anumite evenimente care nu au legatura direct cu protocolul USB (de exemplu, o mișcare a mouse-ului trezește întregul sistem). În acest caz, secvența de resume poate fi pornită prin setarea bitului de resume din registrul USB_CNTR la '1 și se resetează la 0 după un interval cuprins între 1 mS și 15 mS ((acest interval poate fi temporizat folosind întreruperi SOF, care apar cu o perioadă de 1ms atunci când ceasul de sistem rulează la frecvența nominală). Odată ce bitul de reluare este sters, secvența va fi completată de PC-ul host și sfârșitul acestuia poate fi monitorizat din nou folosind bitii RXD și RXD în registrul USB_FNR.

Registrele USB

Registrele periferice USB pot fi împărțite în următoarele grupuri:

- Registre comune: registre de întrerupere și Control
- Registrele endpoint-urilor: configurarea și starea endpoint-urilor
- Buffer descriptor table: locația memoriei pachetelor utilizate pentru localizarea bufferelor de date

Toate adresele de registru sunt exprimate pe post de compensări (offset) prin respectarea adreselor de baza a registrilor periferici USB 0x4000 5C00, cu excepția locațiilor tabelului buffer descriptor, care începe la adresa specificată de registrul USB_BTABLE. Registrele periferice pot fi accesate prin jumătăți de cuvinte (16 biți) sau cuvinte (32 biți).

Registrele comune

Aceste registre afectează comportamentul general al modului de operare al perifericului USB, întreruperea manevrarii/manipularii, adresa dispozitivului și oferirea accesului la numărul actual al cadrului actualizat de PC-ul gazdă.

Registrul de control USB (USB_CNTR)

Adresa offset: 0x40
Valoarea de reset: 0x0003

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RESETM	SOFM	ESOFM	Reserved			RESUME	FSUSP	LP_MODE	PDWN	FRES
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bit 15 **CTRM**: correct transfer interrupt mask
0: Correct Transfer (CTR) Interrupt dezactivat.
1: CTR Interrupt activat, o cerere de întrerupere este generată atunci când bitul corespunzător din registrul USB_ISTR este setat.

Bit 14 **PMAOVRM**: zona de memorie a pachetelor interrupt overrun/underrun
0: PMAOVR Interrupt dezactivat.
1: PMAOVR Interrupt activat, o cerere de întrerupere este generată atunci când bitul corespunzător din registrul USB_ISTR este setat.

Bit 13 **ERRM**: Error Interrupt Mask
0: ERR Interrupt dezactivată.
1: ERR Interrupt activat, o cerere de întrerupere este generată atunci când bitul corespunzător din registrul USB_ISTR este setat.

Bit 12 **WKUPM**: Wakeup Interrupt Mask
0: Wkup Interrupt dezactivat.
1: Wkup Interrupt activat, o cerere de întrerupere este generată atunci când bitul corespunzător din registrul USB_ISTR este setat.

Bit 11 **SUSPM**: Suspend mode interrupt mask
0: Suspend Mode request (SUSP) Interrupt dezactivată.
1: SUSP Interrupt activat, o cerere de întrerupere este generată atunci când bitul corespunzător din Registrul USB_ISTR este setat.

Bit 10 **RESETM**: USB reset interrupt mask
0: RESET Interrupt dezactivată.
1: RESET Interrupt activat, o cerere de întrerupere este generată atunci când bitul corespunzător din Registrul USB_ISTR este setat.

Bit 9 **SOFM**: Start of frame interrupt mask
0: SOF Interrupt dezactivat.
1: SOF Interrupt activat, o cerere de întrerupere este generată atunci când bitul corespunzător din registrul USB_ISTR este setat.

Bit 8 **ESOFM**: Expected start of frame interrupt mask
0: Expected Start of Frame (ESOF) Interrupt dezactivat.
1: ESOF Interrupt activat, o cerere de întrerupere este generată atunci când bitul corespunzător din Registrul USB_ISTR este setat.

Biți 7:5 .

Bit 4 **RESUME**: Resume request
Microcontrolerul poate seta acest bit pentru a trimite un semnal de reluare host-ului. Trebuie să fie activat, conform specificațiilor USB, pentru nu mai puțin de 1 ms și nu mai mult de 15 ms după care PC-ul gazdă este gata să conducă secvența de reluare până la capăt.

Bit 3 **FSUSP**: Force suspend
Software- ul trebuie să stabilească acest bit atunci când este primită întreruperea SUSP, care este emisă atunci când traficul nu este primit de perifericul USB pentru 3 mS.
0: nici un efect.
1: Introduceți modul de suspendare. Clock-urile și disiparea statică a puterii în transmițătorul analogic sunt lăsate neafectate. Dacă suspendarea consumului de energie este o cerință (dispozitiv alimentat cu bus) , software-ul de aplicație ar trebui să stabilească bitul LP_MODE după FSUSP așa cum este explicat mai jos.

Bit 2 **LP_MODE**: Low-power mode
Acest mod este utilizat atunci când constrângerile de putere în suspend mode necesită ca disiparea intregii puteri statice sa fie evitată, cu excepția celei necesare pentru alimentarea rezistorului extern de tracțiune. Acest conditie trebuie introdusă atunci când aplicația este gata să oprească toate clock-urile de sistem sau sa reduca frecvența acestora pentru a îndeplini cerințele de consum de energie ale conditiei de suspendare ale USB-ului. Activitatea USB-ului în timpul modului de suspendare (eveniment WKUP) resetează asincron acest bit (poate fi resetat și de software).
0: No Low-power mode.
1: Enter Low-power mode

Bit 1 **PDWN**: Power down
Acest bit este utilizat pentru a opri complet toate piesele analogice legate de USB dacă este necesar sa fie dezactivat complet perifericul USB din orice motiv. Când acest bit este setat, perifericul USB este deconectat de la transmițătoare și nu poate fi utilizat.
0: Exit Power Down.
1: Enter Power down mode

Bit 0 **FRES**: Force USB Reset
0: Clear USB reset.
1: forțați o resetare a perifericului USB, exact ca o resetare a semnalizării pe USB. În Perifericul USB este ținut în starea de RESET până când software-ul șterge acest bit. O întrerupere de tipul "USB-RESET" este generată, dacă este activată.

Înregistrare stare întrerupere USB (USB_ISTR)

Adresa offset: 0x44
Valoarea de reset: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR	PMA OVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	Reserved			DIR	EP_ID[3:0]			
r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0				r	r	r	r	r

Acest registru conține starea tuturor surselor de întrerupere care permit software-ului să determine ce evenimente au provocat o solicitare de întrerupere.
Partea superioară a acestui registru conține biți unici, fiecare reprezentând un anumit eveniment. Acești biți sunt setați de hardware atunci când are loc evenimentul aferent; dacă bitul corespunzător din registrul USB_CNTR este setat, este generată o cerere generică de întrerupere. Rutina de întrerupere, examinând fiecare bit, va efectua toate acțiunile necesare și, în final, va goli biții deserviți. Dacă oricare dintre ele nu sunt șterse, întreruperea este considerată a fi în continuare în pending, iar linia de întrerupere va fi menținută din nou ridicată. Dacă mai mulți biți sunt setați simultan, doar o singură întrerupere va fi generată.

Bitul CTR este setat de hardware imediat ce un endpoint a reușit sa finalizeze o tranzacție, generând o cerere de întrerupere generică dacă bitul corespunzător din USB_CNTR este setat. O condiție de întrerupere dedicată endpoint-ului este activată independent de bitul CTRM din registrul USB_CNTR. Ambele condiții de întrerupere rămân active până când software-ul șterge bitul în așteptare din registrul USB_EPnR (bitul CTR este de fapt, un bit read-only).

Bit 15 **CTR**: transfer corect
Acest bit este setat de hardware pentru a indica faptul că un endpoint a finalizat cu succes o tranzacție; folosind bitii DIR și EP_ID software-ul poate determina ce endpoint a solicitat întrerupea. Acest bit este doar pentru citire.

Bit 14 **PMAOVR**: zona de memorie a pachetelor overrun/underrun
Acest bit este setat dacă microcontrolerul nu a reușit să răspundă la timp la o solicitare de memorie USB. Perifericul USB gestionează acest eveniment în felul următor: în timpul recepției un pachetul ACK handshake nu este trimis, în timpul transmisiei este forțată o eroare bit-stuff pe stream-ul de transmisie; în ambele cazuri, host-ul va încerca din nou tranzacția. Întreruperea PMAOVR nu ar trebui să apară în timpul operațiilor normale.

Bit 13 **ERR**: eroare
Acest flag este setat ori de câte ori a apărut una dintre erorile enumerate mai jos:

- *NANS: nici un răspuns. Timeout-ul pentru un răspuns din partea host-ului a expirat.*
- *CRC: eroare de verificare a redundanței ciclice. Unul dintre CRC-urile primite, fie în token, fie în date, a fost greșit.*
- *BST: Bit Stuffling Error. O eroare fost detectata oriunde în PID, date, și / sau CRC.*
- *FVIO: încălcarea formatului de încadrare. A fost primit un cadru non-standard (EOP nu este în locul potrivit, wrong token sequence, etc.).*

Software-ul USB poate ignora de obicei erorile, deoarece perifericul USB și PC host-ul gestioneaza retransmisia în caz de erori într-un mod complet transparent. Această întrerupere poate fi utila în timpul fazei de dezvoltare software, sau pentru a monitoriza calitatea transmisiei peste magistrala USB, pentru a semnala posibile probleme utilizatorului (de exemplu, conector liber, mediu prea zgomotos, conductor rupt în cablul USB, etc.). Acest bit este read/write, dar numai '0 poate fi scris și scriind '1 nu are nici un efect.

Bit 12 **WKUP**: Wakeup
Acest bit este setat la 1 de hardware atunci când, în timpul modului de suspendare, este detectată o activitate care activeaza perifericul USB. Acest eveniment șterge asincron bitul LP_MODE în registrul CTRL și activează linia USB_WAKEUP, care poate fi utilizată pentru a notifica restul dispozitivul (de exemplu, unitatea de wakeup) despre începutul procesului de resume. Acest bit este read/write, dar numai '0 poate fi scris și scrierea lui '1 nu are efect.

Bit 11 **SUSP**: cerere de suspend mode
Acest bit este setat de hardware atunci când nu a fost primit niciun trafic pentru un timp de 3ms, indicând o solicitare de suspend mode din partea magistralei USB. Verificarea stării de suspendare este activată imediat după orice resetare USB și este dezactivat de hardware atunci când modul de suspendare este activ (FSUSP=1) până la sfârșitul secvenței de resume. Acest bit este read/write, dar numai '0 poate să fie scris și scrierea '1 nu are nici un efect.

Bit 10 **RESET**: solicitare de resetare USB
Este setat când perifericul USB detectează un semnal de resetare USB activ la intrările sale. Perifericul USB, ca răspuns la o resetare, resetează doar protocolul intern al state machine-ul, generând o întrerupere dacă bitul de permisie RESETM din registrul USB_CNTR este setat. Recepția și transmisia sunt dezactivate până când bitul de resetare este șters. Toate registrele de configurare nu se reseteaza: microcontrolerul trebuie să șteargă în mod explicit aceste registre (acest lucru se intampla pentru a se asigura că intreruperea resetării poate fi livrată în siguranță și orice tranzacție urmată imediat de o resetarea poate fi finalizată). Adresa funcției și registrele endpoint-urilor sunt resetate printr-un eveniment de tipul USB reset. Acest bit este read/write, dar numai '0 poate să fie scris și scrierea '1 nu are nici un efect.

Bit 9 **SOF**: Start of frame
Acest bit semnalează începutul unui nou frame USB și este setat când sosește un pachet SOF prin magistrala USB. Rutina de întrerupere a serviciului poate monitoriza evenimentele SOF pentru a avea un eveniment de sincronizare de 1ms la host-ul USB și pentru a citi în siguranță registrul USB_FNR care este actualizat la recepția pachetelor SOF (acest lucru ar putea fi util pentru aplicațiile izocrone). Acest bit este read/write, dar numai '0 poate să fie scris și scrierea '1 nu are nici un efect.
Bit 8 **ESOF**: Expected start of frame
Acest bit este setat de hardware atunci când este așteptat un pachet SOF, dar nu a fost primit. Host-ul trimite un pachet SOF la fiecare ms, dar în cazul în care hub-ul nu-l primește în mod corespunzător, Suspend Timer-ul semnaleaza această problema de întrerupere. Dacă sunt generate trei întreruperi ESOF consecutive (adică trei Pachetele SOF sunt pierdute) fără nici un trafic care are loc între ele, o întrerupere SUSP este generata. Acest bit este setat chiar și atunci când pachetele lipsă de tip SOF apar în timp ce Suspend Timer-ul nu este încă blocat. Acest bit este read/write, dar numai '0 poate să fie scris și scrierea '1 nu are nici un efect.

Biți 7: 5 Rezervat.

Bit 4 **DIR**: direcția tranzacției
Acest bit este scris de hardware în funcție de direcția tranzacției realizata cu succes,care a generat cererea de întrerupere.
Dacă bitul DIR = 0, bitul CTR_TX este setat în registrul USB_EPnR aflat in legatura cu endpoint-ul de întrerupere.Tranzacția de întrerupere este de tip OUT (date transmise de perifericul USB către PC-ul host).
Dacă DIR bit=1, CTR_RX bit sau ambele CTR_TX / CTR_RX sunt setate în registrul USB_EPnR aflat in legatura cu endpoint-ul de întrerupere. Tranzacția de întrerupere este de tip OUT (date primite de perifericul USB de la PC-ul gazdă) sau două tranzacții în așteptare așteaptă a fi prelucrate.
Aceste informații pot fi utilizate de software-ul aplicației pentru a accesa biții USB_EPnR aflati in legatura cu tranzacția trigger-ului, deoarece reprezintă direcția, avnd întreruperea în așteptare. Acest bit este doar pentru citire.

Bits 3: 0 **EP_ID[3: 0**]: Endpoint Identifier
Acești biți sunt scrisi de hardware în funcție de numărul endpoint-ului, care a generat cererea de întrerupere. Dacă sunt în așteptare mai multe tranzacții cu endpoint-uri, hardware-ul scrie identificatorul endpoint-ului care este in legatura cu endpoint-ul care are cea mai mare prioritate definită în urmatorul mod: sunt definite două seturi de endpoints, în ordinea priorității: izocrone și double-buffered endpoints sunt luate în considerare mai întâi și apoi celelalte endpoint-uri sunt examinate. Dacă mai mult de un endpoint din același set solicită o întrerupere, biții EP_ID din registrul USB_ISTR sunt atribuite în funcție de cel care solicita cel mai mic registru de endpoint, EP0R având cea mai mare prioritate urmată de EP1R și așa mai departe. Software ul aplicației poate atribui un registru pentru fiecare endpoint în conformitate cu această schemă de prioritate. Acesti biți sunt doar citite.

USB frame number register (USB_FNR)

Adresa offset: 0x48
Valoarea de resetare: 0x0XXX unde X este nedefinit in felul urmator:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDP	RXDM	LCK	LSOF[1:0]		FN[10:0]										
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 15 **RXDP**: primire date + statusul liniei

Acest bit poate fi folosit pentru a observa starea datelor primite, plus upstream port data line. El poate fi utilizat în timpul rutinelor de end-of-suspend pentru a ajuta la determinarea evenimentului de wakeup.

Bit 14 **RXDM**: primite date – line status
Acest bit poate fi folosit pentru a observa starea datelor primite minus upstream port data line. El poate fi utilizat în timpul rutinelor de end-of-suspend pentru a ajuta la determinarea evenimentului de wakeup.

Bit 13 **LCK**: Locked
Acest bit este setat de hardware atunci când au fost primite cel puțin două pachete SOF consecutive după încheierea unei condiții de resetare USB sau după încheierea unuei secvențe USB reset. Odată blocat, temporizatorul de cadru rămâne în această stare până la o resetare USB sau pana cand un eveniment USB suspend are loc.

Biți 12:11 **LSOF[1: 0]**: SOF pierdut
Acești biți sunt scrisi de hardware atunci când este generată o întrerupere ESOF, numărand numărul de pachete SOF consecutive pierdute. La recepția unui pachet SOF, acești biți sunt goliti.

Biți 10:0 **FN[10:0]**: Frame Number
Acest câmp de biți conține numărul cadrului de 11 biți conținuti în ultimul pachet SOF primit. Numărul cadrului este incrementat pentru fiecare cadru trimis de host și este util pentru transferuri izocrone. Acest câmp de biți este actualizat la generarea unei întreruperi SOF.

Adresa dispozitivului USB (USB_DADDR)

Adresa offset: 0x4C
Valoarea de resetare: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								EF	ADD6	ADD5	ADD4	ADD3	ADD2	ADD1	ADD0
								rw	rw	rw	rw	rw	rw	rw	rw

Biți 15:8 Reserved

Bit 7 **EF**: Activarea funcției

Acest bit este setat de software pentru a activa dispozitivul USB. Adresa acestui dispozitiv este conținut în următorii biti ADD [6: 0]. Dacă acest bit este la '0 nicio actiune nu este efectuata, indiferent de setările registrelor USB_EPnR.

Biți 6: 0 **ADD[6: 0]**: adresa dispozitivului
Aceşti biți conțin adresa funcției USB atribuită de PC-ul gazdă în timpul procesul de enumerare. Atât acest câmp, cât și câmpul Endpoint Address (EA) din câmpul asociat in registrul USB_EPnR trebuie să corespundă cu informațiile conținute într-un token USB pentru a gestiona o tranzacție la endpoint-ul necesar.

Adresa buffer table-ului (USB_BOOTABLE)

Adresa offset: 0x50
Valoarea de resetare: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BTABLE[15:3]													Reserved		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Biți 15: 3 **BTABLE[15: 3]**: Tabel buffer
Aceşti biți conțin adresa de pornire a tabelului buffer de alocare din interiorul pachetului dedicat de memorie. Acest tabel descrie locația fiecarui endpoint buffer și dimensiunea acestuia și trebuie să fie aliniat la o limită de 8 octeți(3 biți mai puțin semnificativi sunt întotdeauna '0). La începutul fiecărei adresari catre acest dispozitiv, perifericul USB citește elementul acestui tabel legat de endpoint-ul adresat, pentru a obține locația de pornire a buffer-ului și dimensiunea buffer-ului.

Biți 2: 0 Reserved, forțat de hardware la 0.

Registre specifice punctului final

Numărul acestor registre variază în funcție de numărul de endpoints pe care perifericul USB este proiectat să se ocupe. Perifericul USB acceptă până la 8 endpoint-uri bidirecționale. Fiecare dispozitiv USB trebuie să suporte un endpoint de control a cărui adresă (EA bits) trebuie fi setata la 0. Perifericul USB se comportă într-un mod nedefinit dacă sunt mai multe endpoint-uri activate având aceeași valoare. Pentru fiecare endpoint, un registru USB_EPnR este disponibil pentru a stoca informațiile specifice acestuia**.**

USB Endpoint n registru (USB_EPnR), n=[0..7]

Adresa offset: 0x00 la 0x1C
Valoarea de resetare: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR_RX	DTOG_RX	STAT_RX[1:0]		SETUP	EP TYPE[1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX[1:0]		EA[3:0]			
rc_w0	t	t	t	r	rw	rw	rw	rc_w0	t	t	t	rw	rw	rw	rw

Ele sunt, de asemenea, resetate atunci când o resetare USB este primită de la magistrala USB sau forțată prin biți FRES în registrul CTLR, cu excepția biților CTR_RX și CTR_TX, care sunt păstrati neschimbati pentru a evita lipsa unei notificări corecte a pachetelor urmată imediat de un USB reset. Fiecare endpoint are registrul USB_EPnR unde n este identificatorul endpoint-ului. Ciclurile de citire-modificare-scriere pe aceste registre ar trebui evitate deoarece între citire și operațiunile de scriere, unii biți ar putea fi setați de hardware și următoarea scriere ar trebui modificata înainte ca CPU să aibă timp să detecteze schimbarea. În acest scop, toti biții afectati de această problemă au o valoare "invariantă" care trebuie utilizată ori de câte ori modificarea nu este necesară. Se recomandă modificarea acestor registre cu un set de instrucțiuni în care toți biții, care pot fi modificați numai de hardware, sunt scrisi cu valoarea lor "invariantă".

Bit 15 **CTR_RX**: Transfer corect pentru recepție
Acest bit este setat de hardware atunci când o tranzacție de ieșire/configurare este finalizată cu succes pe acest endpoint; software-ul poate șterge doar acest bit. Dacă bitul CTRM în registrul USB_CNTR este setat în consecință, o condiție de întrerupere generică este generată împreună cu condiția de întrerupere asociată endpoint-ului, care este întotdeauna activată. Tipul de tranzacție care a avut loc, OUT sau SETUP, poate fi determinată din bitul de configurare.

Bit 14 **DTOG_RX**: Data Toggle, pentru transferuri de recepție
Dacă endpoint-ul nu este Izocronic, acest bit conține valoarea așteptată a bitului de comutare a datelor (0=DATA0, 1=DATA1) pentru următorul pachet de date care urmează să fie primit. Hardware comută acest bit, când handshake-ul ACK este trimis la host-ul USB, după o recepție de pachete de date având o valoare PID de date care se potrivește; dacă endpoint-ul este definit ca unul de control, hardware-ul șterge acest bit la recepția unei adresari SETUP PID catre acest endpoint.

În cazul în care endpoint-ul utilizează caracteristica de double-buffering, acest bit

este utilizat pentru a sprijini interschimbarea pachetelor buffer.
Dacă endpoint-ul este Izocronic, acest bit este utilizat numai pentru a suporta schimbarea packet buffer , deoarece nu se utilizează comutarea datelor pentru acest tip de endpoints și se transmite numai pachetul DATA0. Hardware comută acest bit imediat după sfârșitul recepției pachetului de date, deoarece nu se folosește nici un handshake pentru transferurile izocronice.
Acest bit poate fi, de asemenea, comutat de software pentru a inițializa valoarea sa (obligatoriu atunci când endpoint-ul nu este unul de control) sau pentru a forța utilizarea anumite date de comutare/packet buffer. Când aplicația software scrie '0, valoarea DTOG_RX rămâne neschimbată, iar cand scrie '1 face comutarea valorii de biți. Acest bit este de tip read/write, si poate fi comutat doar prin scrierea '1.

Biți 13: 12 **STAT_RX [1: 0]**: biți de stare, pentru transferuri de recepție
Acești biți conțin informații despre starile endpoint-ului, care sunt enumerate în tabelul 173. Acești biți pot fi comutați de software pentru a inițializa valoarea lor. Când software-ul aplicației scrie "0, valoarea rămâne neschimbată, în timp ce scrierea "1 face comutarea valorii de biți.
Hardware-ul setează biții STAT_RX la NAK atunci când a avut loc un transfer corect (CTR_RX=1) corespunzător unei tranzacții OUT sau SETUP adresate acestui endpoint, deci software-ul are timp pentru a elabora datele primite înainte de a recunoaște o nou tranzacție.
Double-buffered bulk endpoints implementează un control special al fluxului de tranzacții, care controlează starea bazată pe condiția de disponibilitate a buffer-ului.
Dacă endpoint-ul este definit ca izocron, statutul său poate fi doar "VALID " sau" DISABLED", deci hardware-ul nu poate schimba starea endpoint-ului după o tranzacție de succes. Dacă software-ul stabilește biții STAT_RX la "STALL" sau " NAK " pentru un endpoint Izocronic, comportamentul USB-ului nu este definit.
Acesti biti este de tip read/write, si pot fi comutati doar prin scrierea '1.

Bit 11 **SETUP**: tranzacție de setup finalizată
Acest bit este read-only și este setat de hardware atunci când ultima tranzacție finalizată este un SETUP. Acest bit își schimbă valoarea numai pentru endpoint-urile de control. Trebuie examinat, în caz de succes a tranzactiei primite, (CTR_RX event), pentru a determina tipul de tranzacție ce a avut loc. Pentru a proteja rutina de întrerupere a serviciului de modificări in bitii de SETUP datorită următorilor tokens ce vor fi primite, acest bit este păstrat frozen în timp ce bitul CTR_RX este la 1; starea sa se modifica atunci când CTR_RX este la 0. Acest bit este doar pentru citire.

Bits 10:9 **EP_TYPE[1: 0]**: Tipul endpoint-ului
Acești biți configurează comportamentul acestui endpoint așa cum este descris în tabelul 174. Endpoint 0 trebuie să fie întotdeauna un endpoint de control și fiecare functie USB trebuie să aibă cel puțin un endpoint de control care are adresa 0, dar pot exista și alte endpoint-uri de control, dacă este necesar. Numai endpoint-urile de control gestionează tranzacțiile de configurare, care sunt ignorate de endpoint-urile altor tipuri. Nu se poate răspunde la tranzacțiile de configurare cu NAK sau STALL. Dacă un endpoint de control este definit ca NAK, USB-ul nu va răspunde, simulând o eroare de primire, în direcția de primire a tranzactiei de configurare care este primita. Dacă endpoint-ul de control este definit ca STALL în direcția de primire, atunci pachetul de configurare va fi acceptat oricum, transferând date și emiterea întreruperii CTR. Primirea tranzacțiilor OUT este tratată în mod normal, chiar dacă endpoint-ul este unul de control. Bulk endpoints si endpoint-urile de întrerupere au un comportament foarte similar și diferă doar caracteristică disponibilă folosind bitul de configurare EP_KIND.

Bit 8 **EP_KIND**: Tipologia endpoint -ului
Semnificația acestui bit depinde de tipul de endpoint configurat de biții EP_TYPE. Tabelul 175 rezumă diferitele semnificații.
DBL_BUF: acest bit este setat de software pentru a activa funcția de double-buffering pentru acest bulk endpoint.
STATUS_OUT: acest bit este setat de software pentru a indica faptul că o tranzacție status out este așteptata: în acest caz, toate tranzacțiile care conțin mai mult de zero biti de date au răspuns "STALL" în loc de "ACK". Acest bit poate fi folosit pentru a îmbunătăți robustețea aplicarii erorilor de protocol în timpul transferurilor de control și utilizarea acestuia este destinată controlului endpoint-urilor. Când STATUS_OUT este resetat, tranzacțiile OUT pot avea orice număr de biti, după cum este necesar.

Bit 7 **CTR_TX**: Transfer corect pentru transmisie
Acest bit este setat de hardware atunci când o tranzacție IN este finalizată cu succes în acest endpoint; software-ul poate șterge doar acest bit. Dacă bitul CTRM din registrul USB_CNTR este setat în consecință, o condiție de întrerupere generică este generată împreună cu endpoint-ul aferent starii de întrerupere, care este întotdeauna activată. O tranzacție încheiată cu un handshake NAK sau STALL nu seteaza acest bit, deoarece nu există de fapt date transferate, ca și în cazul erorilor de protocol sau a nepotrivirilor de comutare a datelor. Acest bit este de tip read/write, dar numai " 0 poate fi scris.

Bit 6 **DTOG_TX**: comutare de date, pentru transferuri de transmisie
Dacă endpoint-ul nu este izocronic, acest bit conține valoarea necesară a bitului de comutare a datelor (0=DATA0, 1=DATA1) pentru următorul pachet de date care urmează să fie transmis. Hardware comută acest bit când handshake-ul ACK este primită de la host-ul USB, urmând un pachet de date transmisie. Dacă endpoint-ul este definit ca unul de control, hardware-ul setează acest bit la 1 la recepția unui PID de configurare adresat acestui endpoint. Dacă endpoint utilizează caracteristica double-buffer, acest bit este utilizat pentru a sprijini schimbarea pachetelor buffer.
Dacă endpoint-ul este Izocronic, acest bit este utilizat pentru a suporta schimbarea pachetelor buffer, deoarece nu comutarea datelor este utilizată pentru acest tip de endpoint-uri și sunt transmise numai pachetele DATA0. Hardware-ul comută acest bit imediat după încheierea transmisiei pachetelor de date, deoarece nu se folosește niciun handshake pentru transferuri Izocronice. Acest bit poate fi, de asemenea, comutat de software pentru a inițializa valoarea sa (obligatoriu atunci când endpoint nu este unul de control) sau pentru a forța o utilizare specifică de comutare a datelor/packet buffer. Când software-ul aplicației scrie ' 0, valoarea DTOG_TX rămâne neschimbată, iar cand scrie '1 face comutarea valorii de biți. Acest bit este de tip read/write, dar poate fi comutat numai prin scrierea 1.

Biți 5: 4 **STAT_TX [1: 0]:** biți de stare, pentru transferuri de transmisie
Acești biți conțin informații despre starea endpoint-ului, enumerate în tabelul 176. Acesti biți pot fi comutati de software pentru a inițializa valoarea lor. Când software-ul aplicației scrie "0, valoarea rămâne neschimbată, iar cand scrie " 1 face comutarea valorii bitului. Hardware seteaza bitii STAT_TX la NAK, atunci când a avut loc un transfer corect (CTR_TX=1) corespunzător unei tranzacții IN sau SETUP (numai pentru control) adresată acestui endpoint. Apoi așteaptă software-ul pentru a pregăti următorul set de date care urmează să fie transmis. Double-buffered bulk endpoints implementează un control special al fluxului de tranzacții, care controlează starea bazată pe condiția de disponibilitate a buffer-ului. Dacă endpoint-ul este definit ca izocron, statutul său poate fi doar "VALID" sau "DISABLED".
Prin urmare, hardware-ul nu poate schimba starea endpoint-ului după o tranzactie reusita. Dacă software-ul seteaza biții STAT_TX la "STALL" sau " NAK " pentru un endpoint izocron, comportamentul USB-ului nu este definit. Acești biți sunt de tip read/write, dar pot fi comutati prin scrierea " 1.

Biți 3: 0 **EA[3: 0]:** Adresa endpoint-ului
Software-ul trebuie să scrie în acest câmp adresa pe 4 biți utilizată pentru a

identifica tranzacţiile direcţionate către acest endpoint. O valoare trebuie scrisă înainte de a activa endpoint-ul corespunzător.

.

Table 173. Reception status encoding

STAT_RX[1:0]	Meaning
00	**DISABLED:** all reception requests addressed to this endpoint are ignored.
01	**STALL**: the endpoint is stalled and all reception requests result in a STALL handshake.
10	**NAK**: the endpoint is naked and all reception requests result in a NAK handshake.
11	**VALID**: this endpoint is enabled for reception.

Table 174. Endpoint type encoding

EP_TYPE[1:0]	Meaning
00	BULK
01	CONTROL
10	ISO
11	INTERRUPT

Table 175. Endpoint kind meaning

EP_TYPE[1:0]		EP_KIND Meaning
00	BULK	DBL_BUF
01	CONTROL	STATUS_OUT
10	ISO	Not used
11	INTERRUPT	Not used

Table 176. Transmission status encoding

STAT_TX[1:0]	Meaning
00	**DISABLED:** all transmission requests addressed to this endpoint are ignored.
01	**STALL**: the endpoint is stalled and all transmission requests result in a STALL handshake.
10	**NAK**: the endpoint is naked and all transmission requests result in a NAK handshake.
11	**VALID**: this endpoint is enabled for transmission.

Tabel descriptor buffer

Deși tabelul descriptor buffer este situat în interiorul memoriei de pachete buffer, intrările sale pot fi considerate registre suplimentare utilizate pentru a configura locația și dimensiunea pachetelor buffer, utilizate pentru a face schimb de date între celula macro USB și STM32F10xxx. Datorită limitării comune a APB asupra adresabilității cuvintelor, toate locațiile pachetelor de momorie sunt accesate de APB folosind adrese aliniate pe 32 de biți, în loc de adresele locațiilor de memorie utilizate de perifericul USB pentru registrul USB_BTABLE și locațiile tabelului descriptor buffer.
În paginile următoare sunt raportate două adrese de locație: cea care urmează să fie utilizată de aplicația software în timp ce se acceseaza packet memory și cea locală în raport cu USB Acces. Pentru a obține valoarea corectă a adresei de memorie STM32F10xxx care trebuie utilizată în software-ul aplicației în timp ce se acceseaza packet memory, locația reală a adresei de memoriei trebuie înmulțită de doua ori sau multiplicata de doua ori. Prima locație a packet memory este localizată la 0x4000 6000. Intrarea din tabelul descriptor buffer asociată registrelor USB_EPnR este descrisa mai jos.

Adresa buffer de transmisie n (USB_Addr_TX)

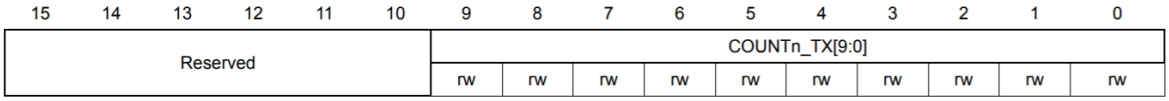
Adresa offset: [USB_TABLE] + n*16
Adresa locală USB: [USB_BTABLE] + n * 8

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRn_TX[15:1]															-
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	-

Biți 15:1 ADDR n_TX[15: 1]: adresa buffer de transmisie
Acești biți indică adresa de pornire a pachetului buffer care conține date care trebuie transmise prin endpointul asociat cu registrul USB_EPnR la următorul simbol adresat acestuia.
Bitul 0 trebuie să fie întotdeauna scris ca " 0 deoarece packet memory este la nivel de cuvânt și toate packet buffer trebuie să fie word-aligned.

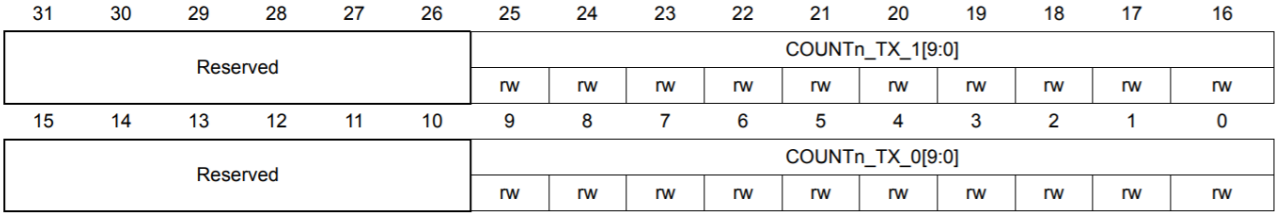
Numărul de biti de transmisie n (USB_COUNTn_TX)

Adresa offset: [USB_BTABLE] + n * 16 + 4
Adresa locală USB: [USB_BTABLE] + n * 8 + 2



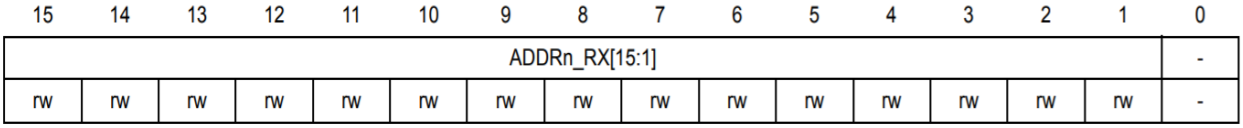
Biți 15:10 *aceşti biți nu sunt utilizați, deoarece dimensiunea pachetului este limitată de specificațiile USB la 1023 octeți. Valoarea lor nu este luată în considerare de USB.*

Biți 9: 0 **COUNTn_TX[9: 0]:** număr de octeți de transmisie
Aceşti biți conțin numărul de octeți care trebuie transmişi prin endpoint-ul asociat cu registrul USB_EPnR la următorul IN token adresat acestuia.



Adresa buffer de recepție n (USB_ADDRn_RX)

Adresa offset: [USB_BTABLE] + n * 16 + 8
Adresa locală USB: [USB_BTABLE] + n*8 + 4



Bits 15: 1 **ADDRn_RX[15:1]**: adresa buffer de recepție
Aceşti biți indică adresa de pornire a pachetului buffer, care va conține datele primite de endpointul asociat cu registrul USB_EPnR la următoarea ieşire / configurare token adresata acestuia.
Bit 0 *Acest bit trebuie să fie întotdeauna scris ca " 0 deoarece pachetul de memorie este word-wide și toate pachetele de buffere trebuie să fie word-aligned.*

Număr de octeți de recepție n (USB_COUNTn_RX)

Adresa offset: [USB_TABLE] + n * 16 + 12
Adresa locală USB: [USB_BTABLE] + n * 8 + 6

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLSIZE	NUM_BLOCK[4:0]					COUNTn_RX[9:0]									
rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r	r	r

Această tabel de locatie este utilizată pentru a stoca două valori diferite, ambele necesare în timpul receptionarii pachetului. Dimensiunea buffer-ului alocat este o parte a descriptorului endpoint-ului și este definit în mod normal în timpul procesului de enumerare în funcție de valoarea parametrului maxPacketSize.

Bit 15 **BL_SIZE**: dimensiunea blocului
Acest bit selectează dimensiunea blocului de memorie utilizat pentru a defini zona buffer alocată.
- Dacă BL_SIZE = 0, blocul de memorie este mare de 2 biti, care este blocul minim permis într-o memorie word-wide. Cu această dimensiune a blocului, dimensiunea buffer alocată variază de la 2 la 62 de biti.
- Dacă BL_SIZE=1, blocul de memorie este mare de 32 de biti, ceea ce permite atingerea maximului lungimii pachetului definit de specificațiile USB. Cu această dimensiune a blocului, dimensiunea buffer alocată variază de la 32 la 1024 biti.

Biți 14: 10 **NUM_BLOCK[4:0]**: Numărul de blocuri
Acești biți definesc numărul de blocuri de memorie alocate acestui packet buffer. Real cantitatea de memorie alocată depinde de valoarea BL_SIZE așa cum este ilustrat în tabelul 177.

Biți 9: 0 **COUNTn_RX[9: 0]:** Numărul de octeți de recepție
Acești biți conțin numărul de octeți primiți de endpoint-ul asociat cu registrul USB EPnR în timpul ultimei tranzacții de ieșire / configurare adresata acestuia.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLSIZE_1	NUM_BLOCK_1[4:0]					COUNTn_RX_1[9:0]									
rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLSIZE_0	NUM_BLOCK_0[4:0]					COUNTn_RX_0[9:0]									
rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r	r	r

Table 177. Definition of allocated buffer memory

Value of NUM_BLOCK[4:0]	Memory allocated when BL_SIZE=0	Memory allocated when BL_SIZE=1
0 (‘00000)	Not allowed	32 bytes
1 (‘00001)	2 bytes	64 bytes
2 (‘00010)	4 bytes	96 bytes
3 (‘00011)	6 bytes	128 bytes
...	...	...
15 (‘01111)	30 bytes	512 bytes
16 (‘10000)	32 bytes	N/A
17 (‘10001)	34 bytes	N/A
18 (‘10010)	36 bytes	N/A
...	...	...
30 (‘11110)	60 bytes	N/A
31 (‘11111)	62 bytes	N/A

Harta registrului USB

Table 178. USB register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**USB_EP0R**	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**USB_EP1R**	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**USB_EP2R**	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**USB_EP3R**	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 178. USB register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10	**USB_EP4R**	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**USB_EP5R**	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**USB_EP6R**	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**USB_EP7R**	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]		SETUP	EP TYPE [1:0]		EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]		EA[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20-0x3F	Reserved																																
0x40	**USB_CNTR**	Reserved																CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RESETM	SOFM	ESOFM	Reserved		RESUME	FSUSP	LPMODE	PDWN	FRES	
	Reset value																	0	0	0	0	0	0	0	0			0	0	0	1	1	
0x44	**USB_ISTR**	Reserved																CTR	PMAOVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	Reserved		DIR	EP_ID[3:0]				
	Reset value																	0	0	0	0	0	0	0	0			0	0	0	0	0	
0x48	**USB_FNR**	Reserved																RXDP	RXDM	LCK	LSOF [1:0]		FN[10:0]										
	Reset value																	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x
0x4C	**USB_DADDR**	Reserved																								EF	ADD[6:0]						
	Reset value																									0	0	0	0	0	0	0	0
0x50	**USB_BTABLE**	Reserved																BTABLE[15:3]															Reserved
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bibliografie

https://www.scribd.com/document/64005548/STM32F103-Datasheet

http://ae.larchitetturadellegno.it/stm32f103-usb-example.html

https://www.scribd.com/document/39167553/STM32F103xx

https://engineeringvolkan.wordpress.com/tag/stm32f103c-usb-hid-uygulamasi/

http://aofn.mo-gio.it/stm32f103-usb-example.html