VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2018 Bc. Vít Miškařík

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

KONVERTOR SYNCHRONIZAČNÍHO KÓDU MTC A LTC
MTC AND LTC SYNC CODE CONVERTER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Vít Miškařík
AUTHOR

VEDOUCÍ PRÁCE — Ing. Ondřej Krajsa, Ph.D.
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Audio inženýrství**
Ústav telekomunikací

Student: Bc. Vít Miškařík ***ID:*** 164609
Ročník: 2 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Konvertor synchronizačního kódu MTC a LTC

POKYNY PRO VYPRACOVÁNÍ:

Vytvořte obousměrný hardwarový konvertor synchronizačního kódu MTC a LTC se zabezečením proti krátkodobým výpadkům kódu LTC.

DOPORUČENÁ LITERATURA:

[1] The Complete MIDI 1.0 Detailed Specification, 3rd ed. MIDI Assoiciation, document version 96.1, 1996.

[2]Rumsey, F., McCormick, T. Sound and Recording, 6th ed. Focal Press, 2009. ISBN 978-0-24-052163-3

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 21.5.2018

Vedoucí práce: Ing. Ondřej Krajsa, Ph.D.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Konvertor synchronizačních kódů LTC na MTC a naopak je důležité zařízení pro synchronizaci všech zařízení používaných pro úpravy, střih a jinou práci se zvukovými a video materiály. Většina těchto digitálních zařízení podporuje synchronizaci podle MTC, zatímco na analogovém pásu je údaj o čase zaznamenán ve formě LTC. Právě zde nachází uplatnění konvertor těchto synchronizačních kódů. Návrhu a realizaci obousměrného konvertoru těchto synchronizačních kódů se věnuje tato práce.

KLÍČOVÁ SLOVA

Synchronizační kódy, převodník, konvertor, MIDI, SMPTE, LTC, MTC, Lineární časový kód, MIDI časový kód, BMC, návrh, konstrukce

ABSTRACT

Converter of sync codes MTC to LTC is very important device to synchronize all devices which are being used for editing and other work with audio and video. Most of digital equipment can sync to MTC but LTC timecode is recorded od audio or videotape. Therefore this converter is needed to sync all connected devices to recorded timecode. This thesis is about designing and constructing the converter between these timecodes.

KEYWORDS

Sync codes, converter, MIDI, SMPTE, LTC, MTC, Linear timecode, MIDI timecode, BMC, Biphase mark coding, design, construction

MIŠKAŘÍK, Vít. *Konvertor synchronizačního kódu MTC a LTC*. Brno, 2018, 58 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Ondřej Krajsa, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Konvertor synchronizačního kódu MTC a LTC“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Ondřeji Krajsovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czechia

http://www.six.feec.vutbr.cz

Výzkum popsaný v této diplomové práci byl realizovaný v laboratořích podpořených projektem Centrum senzorických, informačních a komunikačních systémů (SIX); registrační číslo CZ.1.05/2.1.00/03.0072, operačního programu Výzkum a vývoj pro inovace.

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007-13
OP Výzkum a vývoj pro inovace

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Tato práce se věnuje návrhu a konstrukci konvertoru synchronizačních kódů MTC a LTC, který je nezbytný pro synchronizaci digitálních studiových zařízení při práci s analogově zaznamenaným audio a video materiálem.

V první části je uveden popis MIDI standardu, rozbor formy jeho zpráv a hardwarové řešení MIDI portů, samostatně je důkladněji rozebrán typ zpráv pro přenos informací o reálném čase využívaných kódem MTC (Midi timecode). Ve druhé části je popsán standard SMPTE, jeho generování a módy, chování při pohybu na časové ose, využití ve vysílacích studiích a formy jeho přenosu a záznamu, z nichž jedna je kód LTC (Linear/Longitudinal timecode), který je detailněji popsán v části další. Nedílnou součástí problematiky LTC je jeho zakódování do zvukové stopy pomocí kódu BMC (Biphase mark code), jemuž je věnována samostatná následující kapitola.

V části poslední je popsán samotný návrh konvertoru těchto synchronizačních kódů. Tato část je rozdělena na tři podkapitoly, z nichž první se věnuje hardwarové stránce návrhu, druhá popisuje softwarovou stránku – program pro použitý mikroprocesor. Třetí část se zabývá ověřením funkčnosti a přesnosti zrealizovaného konvertoru.

1 TEORIE - ROZBOR POTŘEBNÝCH ČÁSTÍ (STANDARDŮ)

1.1 Midi

MIDI (Musical Instrument Digital Interface) – Digitální rozhraní pro hudební nástroje je technický standard, který popisuje komunikační protokol, digitální rozhraní a konektory pro vzájemnou komunikaci mezi širokou škálou elektronických hudebních nástrojů, počítači a dalšími zvukovými zařízeními. Každý MIDI port dokáže přenášet informace v šestnácti různých kanálech, kde každý může být přiřazen jinému zařízení.

Přes MIDI jsou posílány zprávy o událostech, které znamenají změnu některého z hudebních parametrů (stisk klávesy, ohýbání tónu, dynamika). Dalšími typy zpráv jsou řídící signály pro změnu parametrů jako je hlasitost, panorama, vibrato, signály pro odbavování událostí (např. v divadlech) a řídící hodinové signály pro nastavení a synchronizaci tempa mezi různými zařízeními. Tyto zprávy jsou posílány MIDI kabely do ostatních zařízení, kde mohou ovládat např. generování zvuků či jiné funkce. Nejjednodušším příkladem použití MIDI je propojení MIDI kontroleru, např. klaviatury, který ovládá přehrávání zvuků zvukovým modulem. MIDI data mohou být také zaznamenána hardwarovým či softwarovým zařízením zvaným sekvencer, který bývá použit k úpravě zaznamenaných dat a jejich následnému přehrávání.

Výhodami MIDI je malá velikost MIDI souborů, jednoduchost jejich změn a široký výběr dostupných syntezátorů a digitálně nasamplovaných zvuků. Před vývojem MIDI nebylo možné komunikovat mezi elektronickými hudebními nástroji různých výrobců. S příchodem MIDI všechna MIDI-kompatibilní zařízení mohou být připojená ke kterémukoliv jinému MIDI-kompatibilnímu zařízení, bez ohledu na rozdílného výrobce těchto zařízení.

MIDI technologie byla standardizována v roce 1983 skupinou reprezentantů hudebního průmyslu a nyní je udržována Asociací výrobců MIDI zařízení – MMA (MIDI Manufacturers Association). Všechny verze MIDI standardu jsou vyvinuty MMA v Los Angeles a MIDI výborem Asociace průmyslu hudební elektroniky – AMEI (Association of Musical Electronic Industry) v Tokiu. V roce 2016 MMA založilo Asociaci MIDI – TMA (The MIDI Association), která celosvětově sdružuje všechny, kdo pracují s MIDI. [1] [2]

1.1.1 Technická specifikace

MIDI zprávy sestávají z 8-bitových slov (bajtů), které jsou sériově posílány rychlostí 31,25 kbit/s. Tato rychlost byla zvolena díky rovnosti přesnému podílu frek-

vence 1 MHz, což je frekvence, na níž pracovala většina dřívějších mikroprocesorů. První bit každého slova identifikuje, zda jde o stavový bajt (MSB=1) či datový bajt (MSB=0), následován je sedmi bity informace. Ke každému bajtu je přidán startovací a ukončovací bit pro označení jedné zprávy, pro jeden MIDI bajt je tedy potřeba přenést 10 bitů.

Jeden MIDI port dokáže přenášet informaci v šestnácti nezávislých kanálech. Kanály jsou číslovány od 1 do 16, ale odpovídající binární vyjádření je 0-15. Připojená zařízení dokáží „naslouchat“ – přijímat data pouze z jednoho kanálu a úplně ignorovat data na ostatních kanálech (mód „Omni Off“) nebo přijímat data ze všech kanálů ignorujíc adresu kanálu (mód „Omni On“). Zařízení mohou být monofonní (současně je možné přehrávat pouze jeden tón) nebo polyfonní (může být současně přehráváno více tónů až do počtu daného limitem polyfonie, další tóny mohou být přehrány až po přijetí signálu uvolnění některého tónu či jeho doznění). Přijímací zařízení mohou být obvykle nastavena do všech čtyřech kombinací těchto módů.

Midi zpráva je instrukce, kterou je ovládán některý z parametrů přijímacího zařízení. MIDI zpráva se skládá ze stavového bajtu, který označuje typ zprávy, následovaného až dvěma datovými bajty přenášejícími parametry. MIDI zprávy mohou být:

- Kanálové zprávy – posílány jsou pouze po jednom ze šestnácti kanálů a mohou být přijaty pouze zařízeními naslouchajícími na tomto kanálu
- Systémové zprávy – jsou přijímány všemi zařízeními.

Všechna data, která nemají vztah k přijímajícímu zařízení (odlišný kanál), jsou zahozena. Existuje pět typů zpráv:

- Kanálové hlasové zprávy
- Kanálové zprávy režimu
- Společné systémové zprávy
- Zvláštní systémové zprávy
- Systémové zprávy reálného času

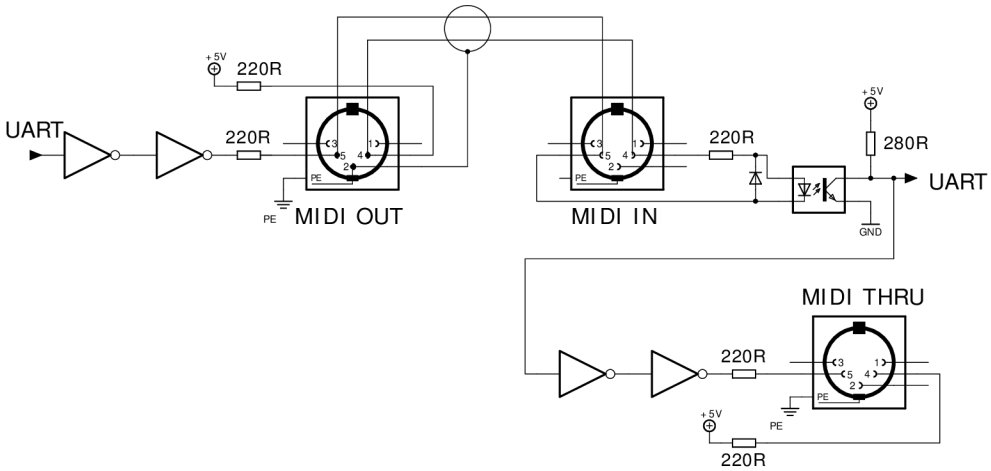
Kanálové hlasové zprávy přenáší po jednom kanále hudební data v reálném čase, například zpráva „Note-on“ – klávesa stisknuta obsahuje informace, která klávesa byla stisknuta, s jakou dynamikou a číslo kanálu, zpráva „Note-off“ – klávesa uvolněna obsahuje číslo klávesy a zpráva „control change“ – změna kontroleru nese číslo kontroleru a jeho hodnotu pro změnu parametrů zařízení. Kanálové zprávy režimu obsahují zprávy „Omni on/off“, změnu módu na polyfonní či monofonní, resetování všech parametrů či vypnutí všech znějících tónů. Systémové zprávy neobsahují číslo kanálu a jsou přijímány všemi zařízeními připojenými do systému. MIDI časový kód je příkladem Společných systémových zpráv. Systémové zprávy reálného času jsou využívány k synchronizaci, zahrnují MIDI clock.

Zvláštní systémové zprávy – SysEx (System Exclusice) jsou hlavním důvodem

flexibility a životnosti MIDI standardu. Dávají totiž výrobcům možnost vytvořit vlastní zprávy, které přináší daleko větší možnosti ovládání jejich zařízení ve srovnání se standardními MIDI zprávami. SysEx zprávy jsou adresovány konkrétnímu zařízení v systému. Každý výrobce má unikátní identifikátor, který je obsažen v SysEx zprávách a dokáže pomoci, aby zprávu přijalo pouze konkrétní zařízení, jemuž je zpráva určena. [1] [2]

Po elektrické stránce je MIDI specifikováno jako plně izolovaná proudová smyčka. Midi výstupní port poskytuje napájení +5V přes předřadný rezistor 220Ω na pin 4 DIN konektoru. Pin 4 vstupního portu je přes rezistor 220Ω připojen na LED diodu optočlenu. Proudová smyčka se uzavírá přes pin 5 a další rezistor 220Ω ve vysílacím zařízení, velikost proudu je přibližně 5mA. Navzdory použití standardních metalických propojovacích kabelů není mezi MIDI zařízeními žádné vodivé spojení, signálová cesta je galvanicky oddělená optočlenem. Správně navržená MIDI zařízení jsou imunní vůči zemním smyčkám a jiným interferencím. Přenosová rychlost je 31,25kbit/s, proud protéká při logické nule.

MIDI specifikace definuje na pinu 2 zemnící vodič a stínění chránící signálové vodiče na pinech 4 a 5 před rušením. I když je definováno připojení těchto stínících vodičů na pin 2 a kovový obal, mělo by toto propojení být provedeno pouze ve výstupním konektoru, ve vstupním konektoru by měl být tedy pin 2 nezapojen a izolován. Někteří velcí výrobci zcela neosazují kovové vodivé části na piny 1, 2 a 3 a ponechávají je tak zcela izolované.[2] Příklad zapojení MIDI portu je ukázán na obrázku 1.1.



Obr. 1.1: Provedení MIDI portů včetně propojení zařízení.

1.2 MTC – Midi timecode

MIDI časový kód (MTC) přenáší stejné časovací informace jako SMPTE časový kód pomocí řady „quarter-frame - čtvrtsnímkových“ MIDI zpráv. Ve standardních MTC zprávách nejsou vyčleněny žádné bity pro uživatelské informace, pro přenos těchto informací se využívají SysEx zprávy. Čtvrtsnímkové zprávy jsou přenášeny jako posloupnost osmi zpráv, kompletní informace o aktuální hodnotě času je tedy získána každé dva snímky. Pokud je datový proud MIDI blízko maximální přenosové kapacity, je možné, že MTC zprávy jsou přeneseny s mírným zpožděním, což způsobí mírný jitter (nestálost frekvence). Abychom se vyvarovali tomuto efektu, je velmi vhodné použít úplně samostatný MIDI port pro přenos MTC dat. Pro přenos časové informace, pokud neběží časový kód (např. časová základna v DAW), jsou posílány delší celosnímkové zprávy, v nichž je obsažena celá časová informace v jedné zprávě.[3]

Na rozdíl od standardního SMPTE časového kódu čtvrtsnímkové i celosnímkové zprávy MIDI časového kódu nesou dvoubitový znak pro indentifikaci snímkové frekvence, která může být:

- 24 snímků/s (standardní hodnota pro film)
- 25 snímků/s (standardní hodnota pro video PAL (používáno v hlavně v Evropě, Austrálii, Argentině a Uruguayi) a SECAM))
- 29,97 snímků/s (video SMPTE s vynecháním snímku – drop-frame, NTSC (používáno hlavně v Severní Americe – USA, Kanada, Mexiko, Kolumbie, atd.), ATSC))
- 30 snímků/s (video SMPTE bez vynechání snímku, ATSC)

MTC rozlišuje mezi rychlostí filmu a rychlostí videa pouze rychlostí, s jakou časový kód postupuje, nikoliv informací, která je obsažena v MTC zprávách, tedy 29,97 snímků/s je reprezentováno jako 30 snímků/s s 0,1% poměrem vynechaných snímků.

MTC umožňuje synchronizovat sekvencer nebo DAW (Digital Audio workstation – Zvukové digitální pracoviště) s dalšími zařízeními, které se umí synchronizovat podle MTC. Druhým případem je nutnost synchronizovat sekvencer nebo DAW (DAW či sekvencer jako „slave“ – závislý) podle páskového magnetofonu, který přehrává pás, na němž je zaznamenán SMPTE časový kód. Pro tento případ je nutný převodník SMPTE časového kódu na MTC. Možné je také synchronizovat páskový magnetofon podle MTC (po konverzi na SMPTE), pokud magnetofon podporuje režim „slave“ – závislý na příchozím časovém kódu, kterým jsou řízeny motory pohonu pásky. Tato funkce je ale spíše velmi výjimečná.[3]

MTC má délku 32 bitů, z nichž 24 je využito a zbylých 8 je obvykle rovno 0.

Aby MIDI datový byte byl platný, bit s největší významností musí být roven 0, tzn. ve čtyřech datových bytech je tedy k dispozici maximálně 28 bitů. Stejně jako ve většině časových kódů používaných v audiovizuální technice (např. SMPTE časový kód) i v MTC je zakódován pouze denní čas, který se tedy opakuje každých 24 hodin. Čas je udáván ve formátu hodiny:minuty:sekundy:snímky. Snímková frekvence viz SMPTE může být 24, 25 nebo 30 snímků za sekundu.

Na rozdíl od ostatních časových kódů jednotlivé složky kódu MTC jsou reprezentovány přímo binárně, nikoliv kódovány kódem BCD - binárně kódovaná dekadická hodnota. Každé složce je přiřazen jeden byte: [3]

Byte 0: 0rrhhhhh - kód snímkové frekvence (0 - 3) a hodiny (0 - 23)

- 0: rr=00 - 24 snímků/s
- 1: rr=01 - 25 snímků/s
- 2: rr=10 - 29,97 snímků/s (SMPTE s vynecháním snímku - drop-frame)
- 3: rr=11 - 30 snímků/s

Byte 1: 00mmmmmm - minuty (0 - 59)
Byte 2: 00ssssss - sekundy (0 - 59)
Byte 3: 000fffff - snímky (0 - 29, maximum závisí na snímkové frekvence)

Pro synchronizaci zařízení MTC používá dva základní typy zpráv označované jako „Quarter-Frame – čtvrtsnímkové“ a úplné. Existuje také třetí typ – dodatečné zprávy pro přenos uživatelských bitů. [1]

1.2.1 Čtvrtsnímkové zprávy (Quarter Frame messages)

Pokud běží čas souvisle, 32-bitový údaj o čase je rozdělen na osm 4-bitových částí a jednou za čtvrtinu doby trvání snímku je poslána jedna z těchto částí. V závislosti na snímkové frekvenci je tedy frekvence posílání těchto zpráv 96 - 120 částí za sekundu. Po přijetí všech osmi částí (osm čtvrtsnímkových zpráv) je časová informace kompletní, SMPTE časový kód je tedy aktualizován každé dva snímky. Tyto čtvrtsnímkové zprávy sestávají ze stavového bajtu o hodnotě F1 následovaného jediným datovým bajtem, který obsahuje tři bity pro identifikaci části a čtyři bity s částí údaje o čase. Pokud časová osa běží dopředně, části jsou vysílány postupně od nulté po sedmou, kdy časová informace zakódovaná v jednotlivých částech odpovídá času, v němž je odeslána nultá část, ostatní části jsou odeslány hned posléze. Při běhu časové osy pozpátku jsou části odesílány podle jejich čísel sestupně, opět kódovaný čas odpovídá času odeslání nulté části. [3]

Informace MTC je do jednotlivých částí rozdělena kódováním little-endian (bajt s nejmenší hodnotou je uložen na nejnižší adresu). Rozdělení časové informace do

Číslo části	Datový byte	Popis
0	0000 ffff	snímek - LS bity
1	0001 000f	snímek - MS bit
2	0010 ssss	sekunda - LS bity
3	0011 00ss	sekunda - MS bity
4	0100 mmmm	minuta - LS bity
5	0101 00mm	minuta - MS bity
6	0110 hhhh	hodina - LS bity
7	0111 0rrh	snímková frekvence a hodina - MS bit

Tab. 1.1: Části MTC čtvrtsnímkových zpráv

částí ukazuje tabulka 1.1.

1.2.2 Celosnímkové zprávy (Full Frame messages)

Pokud dojde ke skoku na časové ose, je vyslána jedna zpráva obsahující kompletní informaci o čase pro sesynchronizování všech připojeným zařízení. Tato zpráva má formu speciální SysEx zprávy (system exclusice – systémová výhradní zpráva), která obsahuje celkem 10 bajtů.

Jednotlivé bajty obsahují hodnoty: F0 7F 7F 01 01 hh mm ss ff F7. První bajt o hodnotě F0 označuje SysEx zprávu, druhý bajt 7F indikuje, že jde o univerzální zprávu reálného času, následuje bajt 7F označující globální vysílání této zprávy do celého systému. Čtvrtý bajt 01 značí typ zprávy – časový kód, další bajt 01 identifikuje zprávu jako zprávu s kompletní časovou informací. Dále následují 4 bajty se samotnou informací o čase. Přestože je MIDI v základě kódováno little-endian (nejdříve bajty nejméně významné), tyto čtyři datové bajty jsou v pořadí odpovídajícímu kódování big-endian (nejdříve bajty s největší významností). Zprávu uzavírá bajt s hodnotou F7 – „end of exclusice“ – konec systémové výhradní zprávy.

Po tomto skoku je časový kód zastaven, než je přijata první následující čtvrtsnímková zpráva.[3]

1.3 SMPTE timecode

SMPTE časový kód je soubor standardů pro označení jednotlivých snímků videa či filmu časovým kódem definovaných SMPTE (Society of Motion Picture and Television Engineers – Společnost filmových a televizních inženýrů) ve specifikaci SMPTE 12M. SMPTE revidovalo specifikaci v roce 2008 a rozdělilo ji do dvou částí: SMPTE 12M-1 a SMPTE 12M-2 zahrnující nová vysvětlení a objasnění.

Časové kódy jsou přidávány do video a audio materiálů, mohou být také přizpůsobeny pro synchronizaci hudby. Tyto časové kódy poskytují časové reference pro úpravu a synchronizaci zvukového a video obsahu. Díky použití časového kódu jsou možné moderní úpravy videa na magnetickém pásu. [4]

SMPTE časový kód obsahuje binárně kódované dekadické vyjádření času ve formátu hodina:minuta:sekunda:snímek a dalších 32 bitů pro použití uživatelem. Dále kód obsahuje znaky (příznaky) pro mód „drop-frame – vynechání snímku“, příznaky barevnostních snímků a tři další příznaky „binární skupiny“ používané pro definování použití uživatelských bitů. Formáty variant časových kódů SMPTE jsou odvozeny z lineárního časového kódu.

SMPTE časový kód používá různé snímkové frekvence (24 snímků/s, 25 snímků/s, 29,97 snímků/s a 30 snímků/s). V principu informace o snímkové frekvenci obsažená v SMPTE časovém kódu je definována implicitně, je známa z rychlosti přicházejícího časového kódu z média, ale také z metadat zakódovaných na médiu. Význam několika těchto bitů zahrnujících „colour framing bit“ – bit značící barevné snímky a „drop frame bit“ - bit s informací o vynechání snímku závisí na vlastní datové rychlosti. „Drop-frame“ bit je platný, respektive význam má pouze pro základní snímkovou frekvenci 30 snímků/s.[4]

1.3.1 Nespojitost časového kódu

Časový kód je generován jako souvislý proud sekvence dat. V některých případech je k synchronizaci použit reálný denní čas, ve většině však je generován „virtuální“ čas, který neodpovídá aktuálnímu dennímu času. Po sérii několika nahrávání či po provedení např. střihových úprav může zaznamenaný časový kód sestávat z několika nesouvislých segmentů.

V podstatě je nemožné znát časovou informaci zakódovanou v aktuálním rámci, dokud není tento rámec přijat (přehrán), tedy v době, kdy již má být čas následujícího snímku. Reálné systémy sledují postupnou sekvenci rámců časového kódu a odvozují z ní čas aktuálního snímku. Časové kódy v analogových systémech jsou náchylné k bitovým chybám a výpadkům, většina zařízení pracujících s časovými kódy tedy kontroluje interně konzistenci sekvence dat časového kódu a aplikuje jednoduché algoritmy pro opravu těchto chyb a výpadků. Přesné hranice nespojitých úseků je tudíž nemožné přesně stanovit, dokud není přehráno několik subsekvenčních snímků či nespojitá sekvence snímků těchto úseků s nespojitým časem. Z tohoto důvodu se systémy snaží po většině úprav udržet časový kód zaznamenaného materiálu souvislý, takže se může zaznamenaný materiál po úpravách několikrát přepsat na stejné místo na médiu (pás, videokazeta,...).

Přestože v ryze digitálních systémech je možné potlačit nutnost použití „freewheel“

algoritmu (kromě přijímaného časového kódu si ještě přijímač interně počítá časový kód pro případ krátkodobých výpadků) přidáním zpoždění snímků, abychom umožnili přijímači dekódovat časový kód přednostně před zpracováním snímku, není této možnosti ve většině reálných systémů využito, neboť toto řešení přináší zbytečné zpoždění ve větvi zpracování snímků a navíc je zde stále nutnost kompenzovat chyby v časovém kódu odvozeného z analogově zaznamenaného videa či ze zvukových systémů.[5]

1.3.2 Časový kód s vynecháním snímku – „drop frame“

Časový kód s vynecháním snímku – „drop frame“ vznikl jako kompromis, když bylo vynalezeno barevné NTSC video. Autoři NTSC chtěli zachovat kompatibilitu s již existujícími černobílými televizemi. Pro minimalizaci viditelnosti subnosné na černobílých přijímačích bylo nezbytné udělat barevnostní subnosnou lichým násobkem poloviny frekvence vykreslování řádků, násobitel byl zvolen 495. Při snímkové frekvenci 30 Hz je frekvence vykreslování řádků $30 \cdot 525 = 15750$ Hz. Frekvence barevnostní subnosné by tedy měla být $15750/2 \cdot 495 = 3{,}898125$ MHz. Tato hodnota byla původně zvolena, ovšem testy ukázaly, že na některých černobílých přijímačích se projevuje interference mezi barevnostní subnosnou a nosnou pro zvuk, která má frekvenci 4,5 MHz. Viditelnost tohoto interferenčního obrazce mohla být dobře potlačena snížením násobitele pro subnosnou na 455 (což zvýší interferenční frekvenci z přibližně 600 kHz na přibližně 920 kHz) a uděláním této interferenční frekvence také lichým násobkem frekvence vykreslování řádků. Druhou možností k dosažení stejného výsledku je 0,1% zvýšení frekvence nosné zvuku na 4,5045 MHz, ovšem konstruktéři se rozhodli, kvůli obavám z problémů na již existujících přijímačích, pro snížení frekvence barevnostní subnosné, ale také frekvence vykreslování řádků a snímkové frekvence o 0,1%. Frekvence NTSC barevnostní subnosné má tedy hodnotu 3,57954545MHz (přesně 315/88 MHz), frekvence vykreslování řádků je rovna 15734,27 Hz (přesně 9/572 MHz) a snímková frekvence je 29,97 Hz (přesně 30/1,001 Hz). [4] [6] [7]

Tímto je myšleno, že hodina časového kódu s nominální snímkovou frekvencí 30 snímků/s je při přehrávání rychlostí 29,97 snímků/s o 3,6 s delší než hodina reálného času, což vede k chybě téměř minutu a půl na den přehrávání. K zamezení této chybě byl vymyšlen právě SMPTE časový kód s vynecháním snímku. Navzdory tomu, co říká název tohoto módu, žádný z video snímků není vynechán (přeskočen), naopak část časového kódu je vynechána. Abychom srovnali hodinu časového kódu s hodinou reálného času, časový kód s vynecháním snímku přeskočí snímky 0 a 1 první sekundy každé minuty vyjma minut, jejichž číslo je dělitelné deseti (zbytek po dělení deseti je roven 0). Protože editory umožňující střihové úpravy musí hlídat

změnu fáze barevnostní subnosné mezi sudými a lichými snímky, je lepší přeskakovat páry snímků. Tímto tedy dosáhneme hodnoty 18 vynechaných snímků za každých deset minut a téměř perfektně vykompenzujeme rozdílnou rychlost časového kódu a reálného času, zbylá chyba činí pouze 1ppm, zhruba 2,6 snímku (86,4 ms) za den.

Pro shrnutí časový kód s vynecháním snímku přeskočí 18 z 18000 čísel snímků, což je rovno poměru 1/1000, čímž dosáhmene $30 \cdot 1,001 = 29,97$snímků/s. To je jen nepatrně méně než je skutečná NTSC snímková rychlost $30/1,001 = 29,97\overline{002997}$ snímků/s, což odpovídá vynechání jednoho z 1001 čísla snímků. Rozdíl mezi výsledky je jeden NTSC snímek na 1000000 vynechaných čísel snímků časového kódu, což je zanedbatelné.

Pro názornost uvádíme příklad sekvence, kde jsou přeskočeny čísla snímků (každá minuta kromě desátých):
02:07:59:28
02:07:59:29
02:08:00:02
02:08:00:03
Příklad sekvence pro každou desátou minutu (bez vynechání čísel snímků):
02:09:59:28
02:09:59:29
02:10:00:00
02:10:00:01

Zatímco časový kód bez vynechání snímků je obvykle zobrazován s dělícími znaky v podobě dvojtečky mezi páry číslic (HH:MM:SS:FF), časový kód s vynecháním snímků používá jako dělící znak středník či tečku mezi páry číslic (HH;MM;SS;FF nebo HH.MM.SS.FF) nebo pouze mezi sekundami a snímky (HH:MM:SS;FF nebo HH:MM:SS.FF). Tečka jako dělící znak je zobrazována většinou na VTR (video tape rekorder – zařízení zaznamenávající video na kazetu) a jiných zařízeních, které nemají možnost zobrazovat středník.[4]

1.3.3 Studiové použití a hlavní generátor času

Při použití v televizních studiích je podélný časový kód generován hlavním synchronizačním generátorem celého studia a distribuován do všech zařízení z jednoho místa. Tyto hlavní synchronizační generátory odvozují svou rychlost z atomových hodin nebo využívají síťového času či signálu GPS. Studia obvykle mívají dva nebo tři zdroje přesného času, které se automaticky přepínají při selhání některého z nich. Posledním trendem je zabudování malého SMPTE generátoru řízeného GPS do každé kamery, čímž odpadne nutnost distribuční sítě synchronizačního signálu např. pro

mobilní pracoviště. Podélný SMPTE časový kód je velmi rozšířen pro synchronizaci zvukových zařízení. Snímková frekvence 30 snímků/s je nejčastěji používaná v Americe, Japonsku a v ostatních zemích, které mají distribuční síť elektrické energie pracující na frekvenci 60 Hz a používají televizní standard NTSC. Snímková frekvence 25 snímků/s standardizovaná EBU (European Broadcosting Union - Evropská unie pro vysílání) je používána v celé Evropě, Austrálii a všude jinde, kde je distribuční síť elektrické energie o frekvenci 50 Hz a kde se používají televizní standardy PAL a SECAM.

1.3.4 Formy SMPTE časového kódu

1. **Linear timecode**
   Podrobně bude rozebrán v následující kapitole.
2. **Vertical interval timecode**
   Vertical interval timecode – VITC je zapsán přímo do vertikálních zatemňovacích intervalů každého snímku videa. Velkou výhodou tohoto kódu je jeho umístění přímo ve snímku, což umožňuje jeho čtení, i když páska není zrovna v pohybu.
3. **CTL timecode**
   CTL timecode (control track longitudinal) – podélný časový kód v řídící stopě označuje SMPTE časový kód vložený do řídící stopy videokazety.
4. **Visible time code**
   Visible time code – viditelný časový kód neboli též burnt-in timecode (BITC) – vpálený časový kód je označení pro časový kód, který je přímo zapsán do obrazu, tudíž je přímo viditelný lidským okem.
5. **Film labels**
   Jako příklad uvedeme Keykode, což jsou nápisy (kódy) vytištěné přímo na kraji filmového pásu.

1.4 LTC – Linear (longitudinal) timecode

Lineární nebo-li podélný časový kód je kódování dat SMPTE časového kódu do zvukového signálu podle specifikace SMPTE 12M. Tento zvukový signál je obvykle zaznamenáván na magnetické pásy ale i na jiná média. Ke kódování jednotlivých bitů je použito kódování BMC (biphase mark code) – dvoufázový kód známý též jako frekvenční modulace. Bit o hodnotě 0 představuje změnu stavu výstupu jednou za periodu (na začátku periody), bit nesoucí informaci 1 má během jedné periody dvě změny stavu výstupu (na začátku a uprostřed periody). Ke kódování (dekódování) není třeba externí hodinový signál, hodinový signál je vytvořen při kódování

(obnoven z BMC při dekódování). Každý rámec kódu je ukončen tzv. synchronizačním slovem, které má definovanou funkci synchronizace s video obsahem. V každém rámci je obsažen „korekční bit“ – paritní, který zajišťuje sudý počet změn stavu výstupu v rámci. Pokud lineární časový kód přehráváme a reprodukujeme, jeho zvuk je ostrý, nepříjemný, díky obdélníkovému tvaru signálu obsahuje mnoho harmonických složek a bývá použit jako zvukový efekt znázorňující činnost počítačů. [8]

1.4.1 Generování a distribuce

Na video vysílacích pracovištích by LTC generátor měl být řízen společným generátorem hodin pro všechna zařízení, aby byl zajištěn správný přenos barev snímků a synchronizace časování všech zařízení. Při synchronizování vícera digitálních zařízení závislých na hodinovém signálu s video či zvukovým záznamníkem musí všechna zařízení být připojena ke společnému hodinovému signálu, který je odvozen z generátoru hodin pro celé pracoviště. Toho lze dosáhnout použitím generátoru, který generuje jak referenční hodinový signál tak i signál pro synchronizaci s videem, nebo sesynchronizováním hlavního digitálního zařízení s videem a následnou synchronizací ostatních zařízení podle hlavního zařízení.

Každý rámec LTC je tvořený 80 bity, rámců může být v závislosti na snímkové frekvenci 24, 25 nebo 30 za sekundu, frekvence LTC se tedy mění od 960 Hz (binární nuly při snímkové frekvenci 24 snímků/s) do 2400 Hz (binární jedničky při 30 snímcích/s). Toto rozmezí se spolehlivě vejde do běžného audio rozsahu. LTC existuje jako symetrický „balanced“ i nesymetrický „unbalanced“ signál a s ohledem na distribuci ho lze považovat za běžný zvukový signál. Stejně jako pro zvuk lze pro přenos LTC použít standardní audio kabely, konektory, zesilovače a přepojovací panely, může být také galvanicky oddělen běžným audio transformátorem. Také lze k přenosu LTC používat 75Ω video kabely a video zesilovače, ačkoli úbytek napětí způsobený použitím kabelů s impedancí 75Ω může způsobit pokles signálu na úroveň, kterou již některá zařízení nejsou schopna přečíst. Zvláštní péče se musí věnovat zamezení přeslechů z LTC stopy do audio stop.

Zásady práce s LTC:

- Vyvarujte se perkusivním zvukům v blízkosti stopy s LTC
- Nikdy nepoužívejte na LTC signál kompresor, ekvalizér ani redukci šumu
- Povolte „pre-roll“ a „post-roll“ – pohyb pásu před spuštěním a po vypnutí
- Vždy zvolte nejpomalejší zařízení jako hlavní

Pokud je podélný SMPTE časový kód zaznamenán do zvukové stopy, měl by být přehráván s úrovní signálu ve středu rozsahu, vysoké i nízké úrovně způsobí zkreslení signálu.[8]

1.4.2 Datový formát podélného (lineárního) časového kódu

Základní formát lineárního časového kódu je 80-ti bitové datové slovo (rámec), které udává denní čas s přesností na sekundy a číslo snímku v aktuální sekundě. Hodnoty jsou uloženy ve formátu BCD (binárně kódovaná dekadická hodnota), bit s nejmenší významností nejdříve. Rozložení bitů nesoucích informaci o čase ukazuje tabulka 1.2. V rámci je k dispozici také 32 bitů pro uživatelská data, nejčastěji jsou používány pro datum a číslo role s pásem. Rozložení těchto uživatelských bitů ukazuje tabulka 1.3. Další částí datového slova jsou příznaky, jejich rozdělení je v tabulce 1.4. Obsah celého LTC rámce je přehledně zobrazen v tabulce 1.6. [9] [8] [6]

Bit	Definice
0-3	Jednotky snímků
8-9	Desítky snímků
16-19	Jednotky sekund
24-26	Desítky sekund
32-35	Jednotky minut
40-42	Desítky minut
48-51	Jednotky hodin
56-57	Desítky hodin

Tab. 1.2: Bitové pozice LTC časové informace

Bit	Definice
4-7	První binární skupina
12-15	Druhá binární skupina
20-23	Třetí binární skupina
28-31	Čtvrtá binární skupina
36-39	Pátá binární skupina
44-47	Šestá binární skupina
52-55	Sedmá binární skupina
60-63	Osmá binární skupina

Tab. 1.3: Bitové pozice LTC bitových skupin

Desátý bit je příznak pro mód s vynecháním snímků. Pokud je nastaven na 1, čísla snímků nula a jedna jsou přeskočeny v první sekundě každé minuty vyjma desátých. Tím je převedena rychlost 30 snímků/s na 29,97 snímků/s dle NTSC standardu.

Jedenáctý bit – příznak barevných snímků je nastaven na 1, pokud je časový kód synchronizován s barevným video signálem. Číslo snímku pro provedení střihu

musí být dělitelné 2 (pro NTSC a SECAM) nebo 4 (PAL), aby nedošlo k posunu fáze chrominanční subnosné.

Význam bitů 27, 43 a 59 je různý mezi snímkovou frekvencí 25 snímků/s a ostatními. Příznak korekce polarity (pro 25 snímků/s na bitu 59, pro ostatní na bitu 27) zamezuje lichému počtu změn stavu kódu, aby bit 0 každého rámce vždy začínal vzestupnou hranou. Přiznaky uživatelských skupin bitů BGF0 a BGF2 (bity 27 a 43 při 25 snímcích/s a 43 a 59 při ostatních snímkových rychlostech) specifikují formát uživatelských dat. Pokud jsou oba rovny nule, formát není specifikován. Pokud je nastaven BGF0 na 1, označuje formát jako 4 osmibitové znaky (přenášeny kódováním „little-endian“). Kombinace s BGF2 nastaveným na 1 jsou vyhrazeny pro budoucí použití.

Bit 58, který byl v dřívějších verzích specifikace nevyužit, je nyní definován jako BGF1 a označuje, zda je časový kód synchronizován podle externího zdroje (hodnota 1).

Synchronizační slovo mezi bity 64 a 79 obsahuje dvanáct po sobě jdoucích jedniček, které se nemohou objevit nikde jinde v celém rámci. Nejdelší možná sekvence po sobě jdoucích jedniček má délku 10 (bity 9 až 18 včetně).

Synchronizační slovo začíná hodnotami 00 a končí 01, aby bylo možné identifikovat, zda se audiokazeta se záznamem LTC točí dopředu, či nazpět.

30 snímků/s bit	**25 snímků/s bit**	**24 snímků/s bit**	**Definice**
10	-	-	Příznak vynechání snímků
11	11	-	Příznak barevných snímků
27	59	27	Korekce polarity
43	27	43	Příznak binárních skupin – BGF0
58	58	58	Příznak binárních skupin – BGF1
59	43	59	Příznak binárních skupin – BGF2

Tab. 1.4: Bitové pozice LTC příznaků

Bit	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
Hodnota	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1

Tab. 1.5: Bitová pozice a hodnota LTC synchronizačního slova

Bit	Význam
0-3	Jednotky snímků
4-7	Uživatelské bity – skupina 1
8-9	Desítky snímků
10	Příznak vynechání snímků
11	Příznak barevných snímků
12-15	Uživatelské bity – skupina 2
16-19	Jednotky sekund
20-23	Uživatelské bity – skupina 3
24-26	Desítky sekund
27	Příznak – závislé na snímkové frekvenci
28-31	Uživatelské bity – skupina 4
32-35	Jednotky minut
36-39	Uživatelské bity – skupina 5
40-42	Desítky minut
43	Příznak - závislé na snímkové frekvenci
44-47	Uživatelské bity – skupina 6
48-51	Jednotky hodin
52-55	Uživatelské bity – skupina 7
56-57	Desítky hodin
58	Příznak hodin
59	Příznak – závislé na snímkové frekvenci
60-63	Uživatelské bity – skupina 8
64-79	Synchronizační slovo - „0011111111111101“

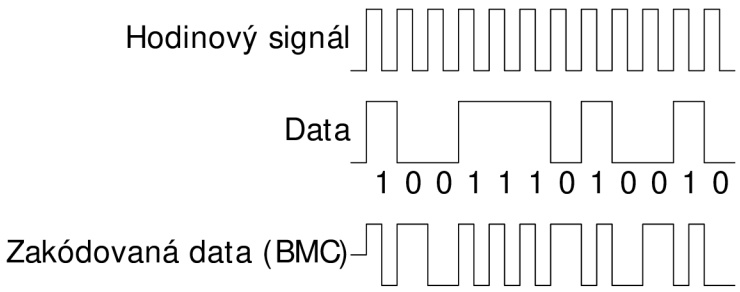
Tab. 1.6: Přehled celého LTC rámce

1.5 Bi-phase mark coding

Bi-phase mark coding (BMC) – kódování dvoustavovými značkami je metoda kódování dat do zvukového signálu nebo jiné přenosové cesty s logickými úrovněmi signálu. Mezi jeho základní vlastnosti patří:

- samostatné časování, není třeba externího hodinového signálu ani nepotřebujeme znát přenosovou rychlost, abychom dokázali dekódovat data
- stejnosměrná složka je nulová, kapacitně oddělený signál je stále dekódován korektně
- robustnost – identifikace bitů záleží na fázových inverzích
- schopnost zpracovat synchronní i asynchronní data
- schopnost pracovat v širokém rozsahu přenosových rychlostí
- schopnost korektního dekódování při měnící se přenosové rychlosti - např. z magnetofonového pásu

Výstupní signál je definovaný jako změna stavu výstupu (změna fáze) na konci každé „periody“ – spíše taktu (odpovídá jednomu bitu). Pokud je tento bit o hodnotě jedna, ve výstupním signálu je další změna úrovně výstupu (změna fáze) uprostřed periody. Tedy souvislý datový proud samých nul při přenosové rychlosti 1kbit/s odpovídá frekvenci výstupního signálu 500 Hz, pokud by tento datový proud obsahoval samé jedničky, výstupní signál by měl frekvenci 1kHz. Běžný signál reprezentující data bývá tedy směsicí dvou střídajících se frekvencí, ukázku takového signálu zobrazuje obrázek 1.2. Tento signál při reprodukování zní velmi nepříjemně až nesnesitelně.



Obr. 1.2: Průběh BMC zakódovaných dat.

BMC je velmi široce rozšířen ve vysílacích studiích pro distribuci a záznam lineárního časového kódu, zvukový signál s tímto kódem je jednoduše zaznamenán na analogovou video či audio kazetu a může být čten jak při normálním přehrávání, tak i při rychlém převíjení kazety. Tento signál bývá také distribuován po celých studiích jako zdroj denního času – stačí připojit čtečku tohoto kódu do příslušného konektoru a správný čas je okamžitě zobrazen. Dalším použitím, kde nalezneme BMC, je

digitální rozhraní SPDIF, ale také je úzce spojen s kódováním v digitálním audio rozhraní AES-EBU.

1.5.1 Synchronní vs. asynchronní

Synchronní data představují souvislý bitový datový proud - každý datový bajt je hned následován dalším. Nejsou zde žádné start ani stop bity (paritní bity mohou být obsaženy v datovém proudu), tudíž musí být použit jiný algoritmus k označení začátku dat. I když přestanou přicházet vstupní data, výstupní datový proud musí stále pokračovat. Synchronizace je zajištěna tzv. „synchronizačním slovem“ , které je vysíláno, i když nepřicházejí na vstup žádná data. Synchronizační slovo musí být jednoznačné a nesmí být zaměnitelné s reálnými daty, tím se snižuje počet možných datových bajtů použitelných jako synchronizační slovo. Pokud uvážíme osmi-bitový systém, například kód 'FF' může být vyhrazen jako synchronizační slovo. Nicméně musíme zajistit, aby data nemohla náhodně nasimulovat synchronizační slovo, např. sekvence po sobě jdoucích '0F' a 'F0' vypadá jako synchronizační slovo.

Lineární časový kód, jak již bylo zmíněno, je navržen tak, aby byl dekódovatelný nejen při proměnných rychlostech přehrávání, ale také při přehrávání jak před, tak i pozpátku. Aby byla zajištěna tato schopnost kódu, je k synchronizaci použito 16-ti bitové synchronizační slovo, které nemůže být zaměněno s legitimní datovou posloupností v 80-ti bitovém rámci, který je vysílán tolikrát, kolik je snímků/s. Podle formy synchronizačního slova můžeme určit směr přehrávání pásu. Asynchronní data jsou nejvíce známá jako non-return to zero (kód bez návratu do nulového stavu) RS232 datový přenos. Data se mohou objevit kdykoliv, což vyruší přenosový kanál z nečinného stavu. V RS232 standardu je tento klidový stav reprezentován logickou hodnotou 1. K označení začátku přenosu dat je využit startovací bit (logická 0) následován řadou datových bitů. Po nepovinných paritních bitech následuje jeden či několik stop bitů o hodnotě 1, které vrátí přenosový kanál do klidového stavu. Každý 8-mi bitový byte zabere tedy pro přenos minimálně 10 bitových period. Protože se startovací bit může objevit kdykoliv, přenosový kanál musí být vzorkován poněkud rychleji, než je jeho nominální přenosová rychlost, obvykle bývá vzorkován 16x rychleji. Tato vlastnost umožňuje následující bity vzorkovat v jejich středu, což zlepšuje šumové vlastnosti.

Nevýhodou asynchronního přenosu je, že frekvence hodinového signálu vysílače i přijímače musí být podobná. Pokud se tyto liší již o 5 %, může se stát, že pozice vzorkování posledního bitu může skočit do předchozího či následujícího bitu, což způsobí chybu přenosu. Toto je běžný problém mikročipů AVR řízených interním nekalibrovaným hodinovým signálem, kalibrace frekvence tedy musí být provedena softwarově, lepší je ale použít externí přesný krystalový oscilátor.

Díky schopnosti BMC samostatného časování nemá nepřesnost hodinového signálu vliv na kódování, správnost dekodovaných dat je garantována. [10]

1.5.2 Kódování a dekódování

Kódování dat BMC je jednoduché, je třeba zajistit změny fáze podle bitové rychlosti a další dodatečnou změnu fáze uprostřed periody, pokud je bit o hodnotě 1. Jak toho dosáhnout záleží na programátorovi. Jednou z možností je použití časového přerušení každou polovinu bitové periody a v metodě obsluhující přerušení rozhodnout, zda má dojít k překlopení výstupu (fázové změně) podle hodnoty příslušného bitu. Další variantou je čistě hardwarové řešení, které využívá několik klopných obvodů.

Dekódování je také poměrně jednoduché v závislosti na tom, zda známe datovou rychlost a jestli bude tato rychlost stabilní. Pokud datovou rychlost známe, pouze v daných intervalech ověřujeme, zda došlo ke změně fáze. Když do 3/4 trvání periody ke změně na vstupu nedošlo, byla přijata hodnota 0, v opačném případě byla přijata polovina bitu o hodnotě 1. Pokud již byla přijata první polovina bitu, můžeme při další změně vstupu zapsat, že byla dekódována hodnota 1, jinak označíme tuto změnu příznakem „první polovina přijata“. Uvažme, že neznáme fázování libovolného datového proudu logických jedniček, je možné, že následující perioda bude bez změny fáze, což jednoznačně identifikuje logickou nulu, předešlá data tedy musí být považována za nedůvěryhodná.

Pokud neznáme bitovou rychlost, je dekódování o něco komplexnější. V tomto případě nepotřebujeme znát pouze délku aktuální periody, ale také periody předchozí. Pokud je délka těchto dvou stejná (od 0,75 do 1,5 násobku), můžeme konstatovat, že jsme přijali stejnou hodnotu jako v předchozím stavu, ale nevíme jakou. Pokud je délka aktuální periody delší než 1,5 násobek předchozí, můžeme jednoznačně říct, že jsme přijali logickou nulu. Pokud je tato délka kratší než 0,75 násobek předchozí délky, můžeme konstatovat, že byla přijata první polovina logické jedničky. Pouze po tomto rozhodnutí můžeme jednoznačně dekódovat pokračující datový tok. Po jednoznačné identifikaci datového toku se již rozhodování o hodnotě bitů děje automaticky a to i v případě, že dojde ke změně datové rychlosti, pokud je změna menší než přibližně šestina bitové periody na jednu periodu. [10]

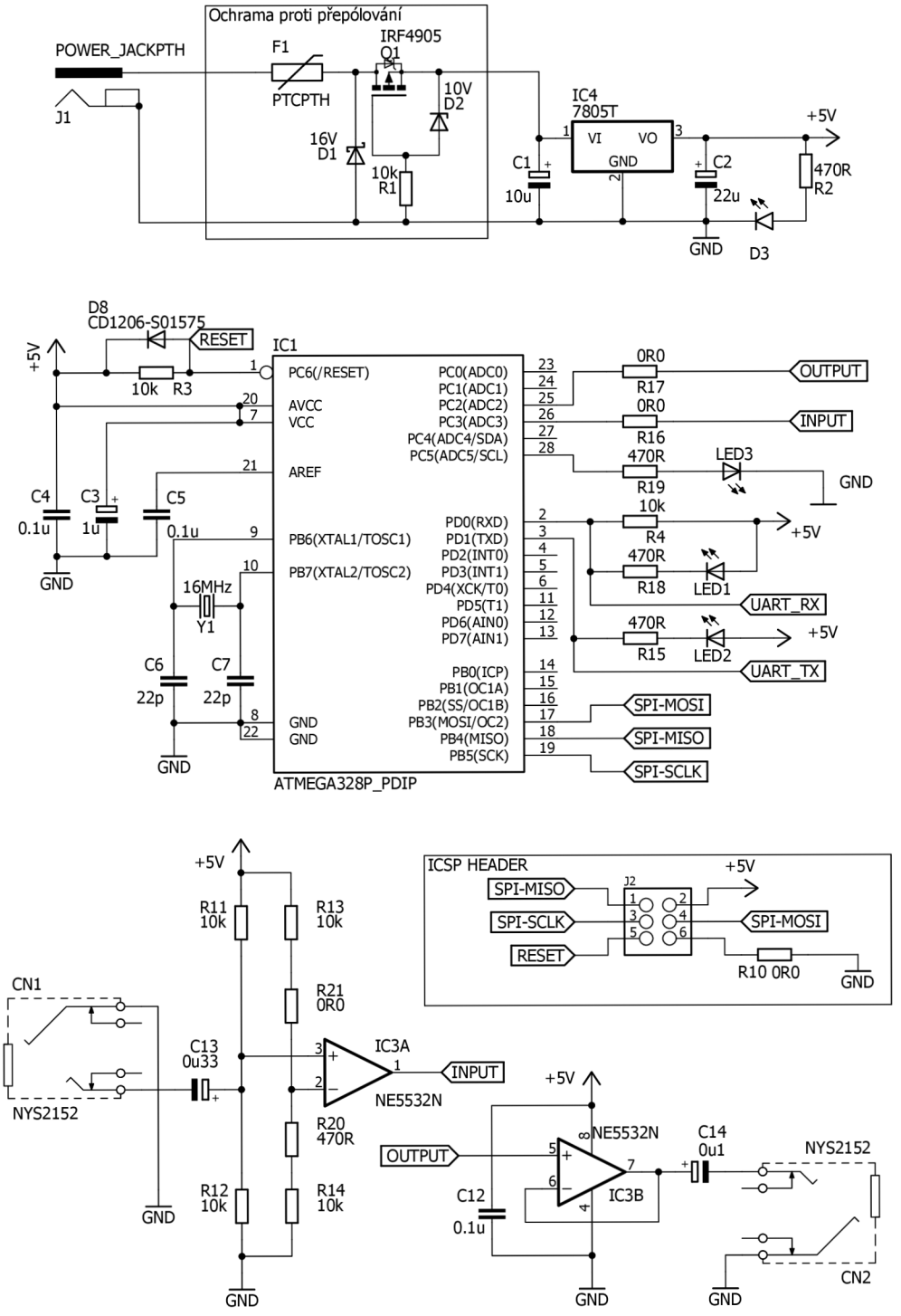
2 NÁVRH KONVERTORU SYNCHRONIZAČNÍCH KÓDŮ

2.1 Návrh hardwaru konvertoru

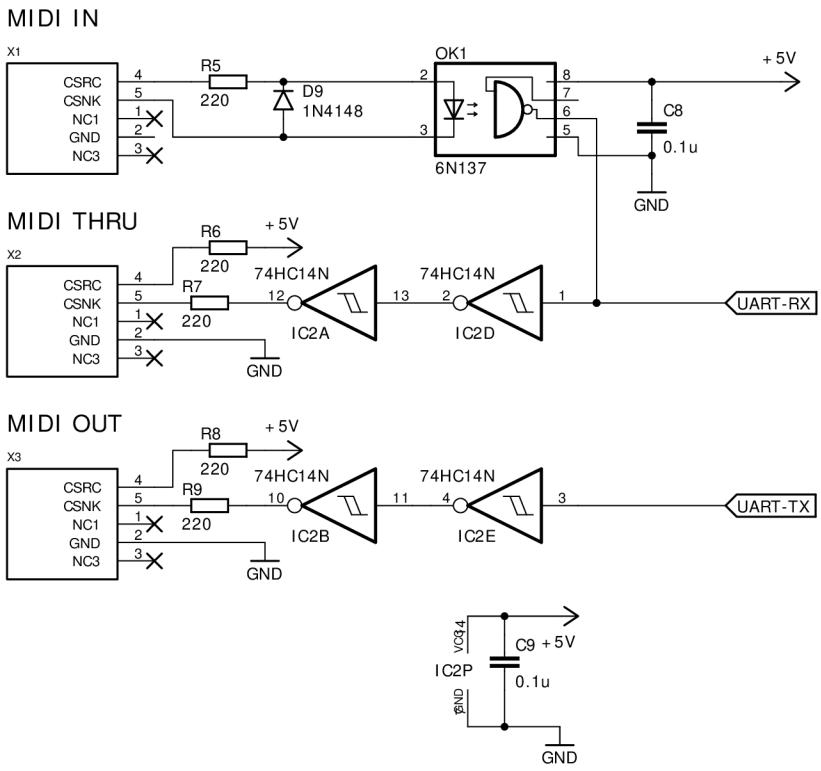
Při návrhu hardwaru převodníku jsem vycházel z projektu MTCDisplay – Zobrazovače aktuálního MTC času.[11] Stejně jako autor tohoto projektu jsem zvolil mikroprocesor Atmel ATMega328P. Původní důvod výběru právě tohoto mikroprocesoru byla možnost jeho programování v Arduino vývojovém prostředí. Od této myšlenky jsem ale velmi rychle ustoupil a zvolil programování použitého mikroprocesoru v jazyce C, neboť s ním již mám zkušenosti z minulosti, ale také pro jeho větší přehlednost. Návrh programu byl tedy rychlejší a efektivnější.

Zapojení celého obvodu vychází z katalogového zapojení mikroprocesoru Atmel ATmega328P [12], zdrojem hodin pro mikroprocesor je připojený 16 MHz krystal. Analogový vstup jsem navrhl jako nesymetrický, po stejnosměrném oddělení kondenzátorem je vstupní signál stejnosměrně posunut o polovinu napájecího napětí, se kterou je pak porovnáván operačním zesilovačem v zapojení jako komparátor, na jehož výstupu je tedy signál o hodnotě 0 V nebo kladné napájecí napětí. Obdobně je realizován analogový výstup, výstup mikroprocesoru je pouze zopakován operačním zesilovačem s jednotkovým zesílením a kondenzátorem stejnosměrně oddělen, čímž se střed výstupního signálu opět posune do nuly. Zapojení MIDI portů jsem navrhl přesně podle specifikace MIDI normy [1], k mikroprocesoru jsou připojeny na výstup a vstup UART sériového rozhraní (obrázek 2.2).

Upravený obvod zobrazovače aktuálního MTC času [11] jsem nejprve zapojil v nepájivém poli, abych otestoval, zda obvod funguje, jak bylo zamýšleno. Zde došlo k několika změnám, aby byl zvýšen komfort programování a následného reálného používání – přidány LED diody signalizující aktivitu na UART rozhraní (příjem a odesílání MIDI zpráv) a jedna LED dioda ovládaná mikroprocesorem (nalezení synchronizace v příchozím LTC signálu). Další změnou bylo přidání dvou rezistorů R20 a R21 zapojených sériově k R13 a R14 pro jemné dostavení komparační úrovně na záporném vstupu operačního zesilovače IC3A (zamezení překlápění komparátoru vlivem šumu na vstupu). Poslední změnou bylo snížení kapacity vazebních kondenzátorů C13 a C14 (experimentálně jsem zjistil spolehlivější dekódování příchozího signálu při propojení LTC výstupu s LTC vstupem). Výslednou podobu zapojení celého obvodu ukazuje obrázek 2.1.

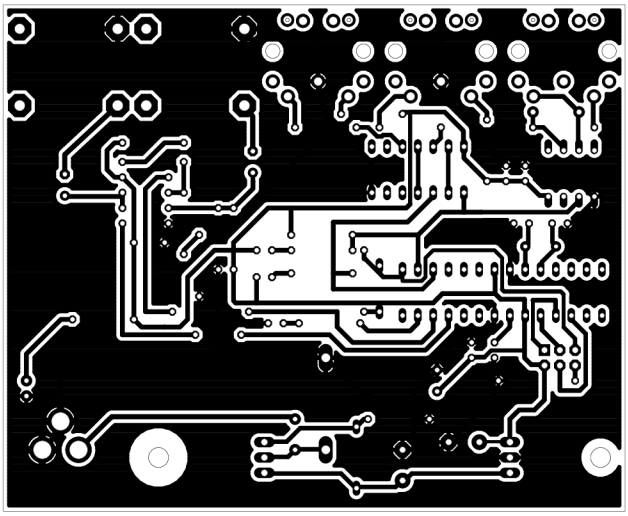


Obr. 2.1: Schéma hlavní části návrhu hardwaru konvertoru.

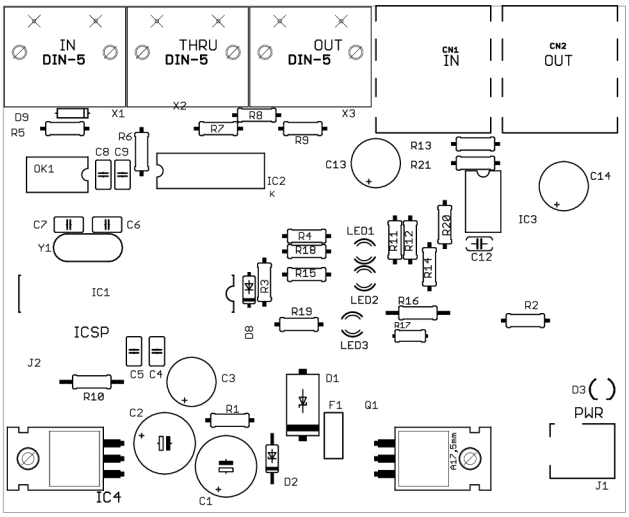


Obr. 2.2: Schéma části návrhu zobrazující zapojení MIDI portů.

Návrh plošného spoje jsem provedl v návrhovém softwaru Eagle. Desku plošného spoje se podařilo navrhnout pouze jako jednostrannou za použití čtyř drátových propojek (v plánu rozmístění součástek znázorněny rezistory R10, R16, R17, R21 s hodnotou 0 Ω - 0R0). Výsledný obrazec desky plošného spoje je zobrazen na obrázku 2.3, rozmístění součástek ukazuje obrázek 2.4. Seznam použitých součástek je uveden v tabulce 2.1. Výslednou podobu osazené desky plošného spoje zobrazuje obrázek 2.5, finální montáž osazené DPS do krabičky je možné vidět na obrázku 2.6. Zařízení je napájeno ze zdroje stejnosměrného napětí o velikosti 6 až 9 V, který se připojuje přes napájecí konektor v panelu přístroje (kladný pól na středu konektoru).



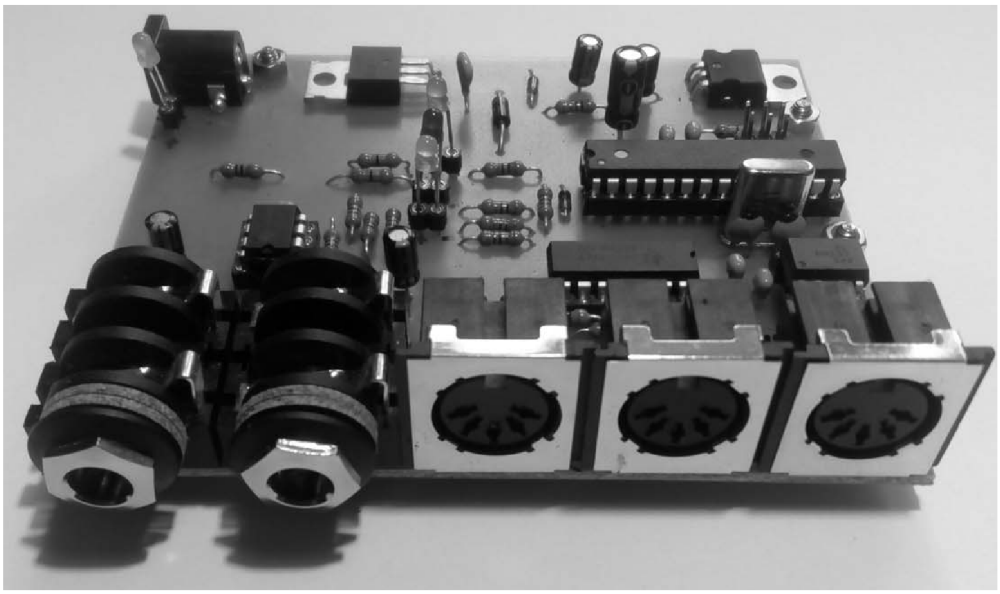
Obr. 2.3: Deska plošného spoje ze strany spojů



Obr. 2.4: Rozmístění součástek na DPS, pohled ze strany součástek

Tab. 2.1: Seznam použitých součástek

Název	**Hodnota**	**Počet**	**Pouzdro**	**Označení**
Dioda	1N4148	2		D8, D9
DIN - 5pinů		3	DIN 5P TP90	X1, X2, X3
Jack konektor	NYS2152	2	NYS2152	CN1, CN2
Kondenzátor	22 pF	2	C025-025X050	C6, C7
Kondenzátor	0,1 µF	5	C025-025X050	C4, C5, C8, C9, C12
Kondenzátor elyt.	0,1 µF	1	E5-10,5	C14
Kondenzátor elyt.	0,33 µF	1	E5-10,5	C13
Kondenzátor elyt.	1 µF	1	E5-10,5	C3
Kondenzátor elyt.	10 µF	1	E5-10,5	C1
Kondenzátor elyt.	22 µF	1	E5-10,5	C2
Krabička	KP06	1	KP06	
Krystal	16 MHz	1	HC49U	Y1
LED		4	3mm	D3, LED1, LED2, LED3
Logický obvod	74HC14N	1	DIL14	IC2
Mikroprocesor	ATMEGA328P	1	DIL28-3	IC1
Napájecí konektor	HEB 21BB	1	2,1mm	J1
Oboustranný kolík	3x2	1	S2G10, 2,54mm	J2
Operační zesilovač	NE5532N	1	DIL08	IC3
Optočlen	6N137	1	DIL08	OK1
Proudová ochrana	0,25 A	1	PTC	F1
Drátová propojka	0 Ω	4	0207/10	R10, R16, R17, R21
Rezistor	220 Ω	5	0207/10	R5, R6, R7, R8, R9
Rezistor	470 Ω	5	0207/10	R2, R15, R18, R19, R20
Rezistor	10 kΩ	7	0207/10	R1, R3, R4, R11, R12, R13, R14
Schottkyho dioda	16 V	1	C4111-15	D1
Stabilizátor 5V	7805T	1	TO220H	IC4
Tranzistor	IRF4905	1	TO220BH	Q1
Zenerova dioda	10 V	1	DO35Z10	D2



Obr. 2.5: Osazená DPS

Obr. 2.6: Výsledná podoba zrealizovaného konvertoru

2.2 Návrh programu pro použitý mikroprocesor

Při návrhu programu pro mikroprocesor konvertoru jsem původně chtěl čerpat z již hotových knihoven zabývajících se časovými kódy [13] [14] [15] a programu pro mikroprocesor použitý v projektu MTCDisplay [11]. V podstatě všechny části potřebné pro tvorbu programu by měly být k dispozici z těchto zdrojů, bohužel zmíněné knihovny nejsou tvořeny přímo pro mikroprocesory AVR, bylo by nutné je tedy modifikovat. Další ze zmíněných možných zdrojů - projekt MTCDisplay [11] obsahoval pouze velmi malou část z potřebných funkcí, konkrétně uměl pouze přijímat MIDI timecode ve formě čtvrtsnímkových zpráv (příjem celosnímkových zpráv nebyl v tomto projektu implementován). S ohledem na časovou náročnost studia těchto knihoven pro jejich nutnou úpravu v souvislosti s implementací do vybraného mikroprocesoru jsem se ale rozhodl jít cestou vytvoření vlastního programu, který bude určen přímo pro zvolený model mikroprocesoru a bude zohledňovat všechny požadavky na funkcionalitu.

K vytvoření programu pro použitý mikroprocesor jsem zvolil programovací jazyk C a vývojové prostředí Atmel Studio 7.0, které je přímo určeno k psaní programů nejen pro mikroprocesory ATMega. Velkou výhodou přímého určení tohoto vývojového prostředí pro zvolený typ mikroprocesoru je možnost přímého nahrávání přeložených zdrojových kódů do mikroprocesoru, debugování a v neposlední řadě také našeptávací funkce při psaní např. názvů vnitřních registrů procesoru, což velmi usnadňuje práci s dokumentací k použitému procesoru. Pro nahrávání přeloženého programu do mikroprocesoru jsem použil programátor USBASP, který ovšem není přímo podporován zvoleným vývojovým prostředím. Spolu s pomocným programem AVRDUDE jej ale lze do vývojového prostředí přidat jako externí nástroj.[18] I přes použití tohoto programátoru bylo tedy možné provádět zápis programu do mikroprocesoru přímo z panelu nabídek vývojového prostředí.

Ještě před samotnou tvorbou programu pro mikroprocesor bylo nutné změnit jeho základní nastavení – zdroj hodin přepnout z interního 8 MHz oscilátoru na připojený externí 16MHz krystal a změnit hodnotu prescaleru na 1, aby frekvence hodin mikroprocesoru byla skutečně požadovaných 16 MHz. Tato operace se provádí přepisem tzv. „fuse bitů“, v nichž jsou uloženy právě tyto základní nastavení. Na základě zmíněných požadavků je nutné změnit hodnoty „fuse registrů“ na: Nízký=0xFF, Vysoký=0xD9, Rozšířený=0xFF.[12][19] Změna hodnot těchto registrů není v kombinaci s použitým programátorem možná přímo z Atmel studia, je nutné je přepsat instrukcí z příkazového řádku. Po provedení popsaných změn již tedy nic nebrání v tvorbě samotného programu.

Prvním krokem v našem návrhu je pomyslné rozdělení celého programu na dva na sobě nezávislé celky – směr konverze z LTC na MTC a konverze z MTC na LTC.

Z toho pramení i rozdělení proměnných v paměti na dvě skupiny, z nichž každá náleží právě jednomu směru konverze. Rozlišení proměnných je provedeno přidáním prefixu „LM_“ u proměnných souvisejících se směrem konverze z LTC na MTC a „ML_“ pro opačný směr převodu. Díky uspořádání dat v LTC datovém slově (údaje času vždy na začátku bytu [8]) se jeví jako velmi praktické uložení celého LTC datového slova (pro příjem i odesílání) do jedné proměnné – pole 10 bytů. Práce s takto uloženými daty bude daleko jednodušší než ukládání do jednotlivých proměnných. Ve společné hlavní části programu je provedena pouze inicializace proměnných na počáteční hodnotu, definování funkce jednotlivých pinů, nastavení režimů čítačů/časovačů, inicializace příjmů a odesílání přes UART rozhraní, globální povolení přerušení a signalizace dokončení inicializačního procesu. Pro inicializaci UART rozhraní jsem použil metodu USART_Init(unsigned char ubrr).[12][20] Jejím parametrem je hodnota podle požadované přenosové rychlosti v baudech. V této metodě je nastavena přenosová rychlost, zapnut příjem a odesílání přes UART rozhraní a nastaven datový formát přenosu dle MIDI specifikace – tedy 8 bitů a 1 stop bit.[1]

Po úspěšném dokončení inicializace jsou stále v cyklu prováděny následující operace: ovládání signalizační LED podle stavu nalezení synchronizace na LTC signál (proměnná *LM_syncFound*), pro směr konverze LTC na MTC nastavení hodnot komparačního registru OCR0A čítače/časovače 0 podle zjištěné snímkové frekvence (proměnná *LM_rate*), pro směr konverze MTC na LTC nastavení hodnot komparačních registrů OCR1AH a ORC1AL čítače/časovače 1 podle zjištěné snímkové frekvence (proměnná *ML_rate*) a v případě aktivních příznaků *LM_flag_obnovaInformace* či *ML_flag_obnovaInformace* obnova časové informace v proměnné pro její odesílání. U obnovy časové informace pro směr konverze z MTC na LTC je nutné kromě přepisu informace o čase v LTC datovém slově provést také součet bitů o hodnotě jedna, který musí být sudý. Pokud tomu tak není, musíme nastavit patřičný bit, aby byla tato podmínka splněna. Všechny ostatní operace spojené s dekódování a kódováním LTC i příjmem a odesíláním MIDI zpráv jsou prováděny v rutinách obsluhujících jednotlivá přerušení. Nejvhodnější hodnoty komparačních registrů čítače/časovače 1 pro správné časy změn na LTC výstupu podle snímkové frekvence ukazuje tabulka 2.2. Nalezení nejvhodnějších hodnot těchto registrů bylo provedeno v programu MS Excel seřazením všech kombinací podle nejmenší odchylky od celého čísla.

Nejvhodnější hodnoty komparačního registr čítače/časovače 0 pro správné časy odesílání MIDI čtvrtsnímkových zpráv na výstup podle snímkové frekvence ukazuje tabulka 2.3. Jejich nalezení bylo provedeno v tabulkovém programu MS Excel vyhledáním co nejbližšího počtu přerušení za jednu zprávu pro všechny snímkové frekvence.

Navržený algoritmus hlavní části programu ukazuje vývojový diagram na ob-

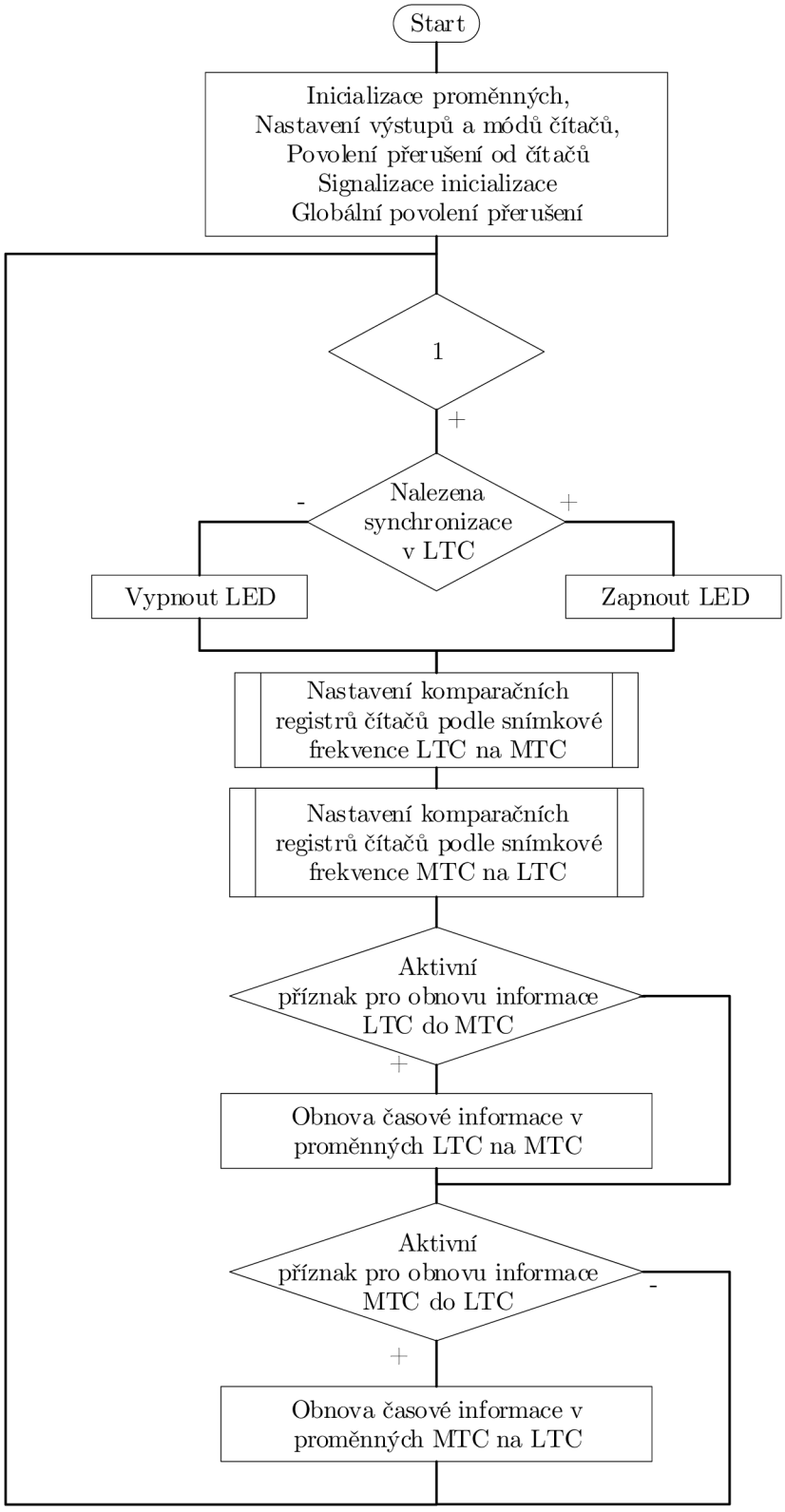
rázku 2.7. Popis proměnných použitých v části provádějící konverzi MTC na LTC je uveden v tabulce 2.4, popis proměnných použitých v části provádějící konverzi LTC na MTC je uveden v tabulce 2.5.

Tab. 2.2: Hodnoty komparačních registrů čítače 1 pro kódování LTC

Snímková frekvence [FPS]	30	30 df	25	24
Počet bitů ve slově (jeden snímek)	80	80	80	80
Počet událostí za slovo – samé jedničky	160	160	160	160
Počet událostí za 1s	4800	4795	4000	3840
Fosc [MHz]	16	16	16	16
Prescaler	1	1	1	1
Hodnota komparačního registru MS	**5**	**9**	**160**	**71**
Hodnota komparačního registru LS	**13**	**13**	**15**	**16**
Dekadická hodnota 16bit komp. registru	3333	3337	4000	4167
Počet přerušení za 1/2 periody (bitu)	**1,0001**	**0,9999**	**1**	**0,9999**
Počet přerušení za 1s	4800	4795	4000	3840

Tab. 2.3: Hodnoty komparačního registru čítače 0 pro odesílání MTC čtvrtsnímkových zpráv

Snímková frekvence [FPS]	30	30 df	25	24
Počet zpráv za snímek	4	4	4	4
Počet událostí za 1s	120	119,88	100	96
Fosc [MHz]	16	16	16	16
Prescaler	8	8	8	8
Hodnota pro vynulování čítače	**204**	**204**	**245**	**255**
Počet přerušení za jednu zprávu	**81,7**	**81,8**	**81,6**	**81,7**
Počet přerušení za 1s	9804	9804	8163	7843



Obr. 2.7: Vývojový diagram hlavní části programu.

Tab. 2.4: Proměnné pro konverzi MTC na LTC, jejich datový typ a význam

Proměnná	**Datový typ**	**Význam**
ML_midiState	unsigned char	rozlišení stavu pro příjem MIDI zpráv
ML_parseHelp	unsigned char	pomocná proměnná pro příjem QFM s LS částí čísla snímku
ML_parseSekundy	unsigned char	pomocná proměnná pro příjem QFM s LS částí aktuální sekundy
ML_parseMinuty	unsigned char	pomocná proměnná pro příjem QFM s LS částí aktuální minuty
ML_parseHodiny	unsigned char	pomocná proměnná pro příjem QFM s LS částí aktuální hodiny
ML_kolikatyByte	unsigned char	iterační proměnná pro příjem celosnímkové MIDI zprávy
ML_prvniPulka	unsigned char	rozlišení poloviny a konce periody
ML_syncFound	unsigned char	signalizace nalezení synchronizace
ML_runningDetection	int	iterační proměnná pro detekci zastavení příchozích dat
ML_transmit	unsigned char	hodnota aktuálního bitu k odeslání
ML_rate	unsigned char	snímková frekvence
ML_hodiny	unsigned char	hodiny
ML_minuty	unsigned char	minuta
ML_sekundy	unsigned char	sekundy
ML_snimek	unsigned char	sekundy
ML_LTCframe [10]	unsigned char	sekundy
ML_bitByte	unsigned char	číslo aktuálního bitu v bytu LTC slova k odeslání
ML_byte	unsigned char	číslo aktuálního bytu LTC slova k odeslání
ML_flag_obnovaInformace	unsigned char	příznak pro obnovu času v paměti

Tab. 2.5: Proměnné pro konverzi LTC na MTC, jejich datový typ a význam

Proměnná	**Datový typ**	**Význam**
LM_pocetProDekodovani	int	iterační proměnná pro měření času mezi změnami na vstupu
LM_prvniPulka	unsigned char	rozlišení poloviny a konce periody
LM_posledniByte	unsigned char	pro hledání synchronizačního slova, méně významný byte
LM_posledniByteMS	unsigned char	pro hledání synchronizačního slova, významnější byte
LM_syncFound	unsigned char	signalizace nalezení synchronizace
LM_freeWheel	int	iterační proměnná pro detekci výpadku příchozích dat
LM_lostFrames	unsigned char	počet snímků bez nalezení synchronizačního slova
LM_rate	unsigned char	snímková frekvence
LM_hodiny	unsigned char	hodiny
LM_minuty	unsigned char	minuta
LM_sekundy	unsigned char	sekundy
LM_snimek	unsigned char	sekundy
LM_detekovany_bit	unsigned char	hodnota přijatého bitu
LM_flag_bit	unsigned char	příznak přijetí bitu
LM_LTCframe [10]	unsigned char	LTC datové slovo
LM_bitByte	unsigned char	číslo aktuálně přijímaného bitu v bytu LTC slova
LM_byte	unsigned char	číslo aktuálně přijímaného bytu LTC slova
LM_kolikatyQFM	unsigned char	číslo aktuálního QFM k odeslání
LM_pocetProQFM	unsigned char	iterační proměnná pro odeslání QFM ve správný čas
LM_flag_obnovaInformace	unsigned char	příznak pro obnovu času v paměti

2.2.1 Konverze z LTC na MTC

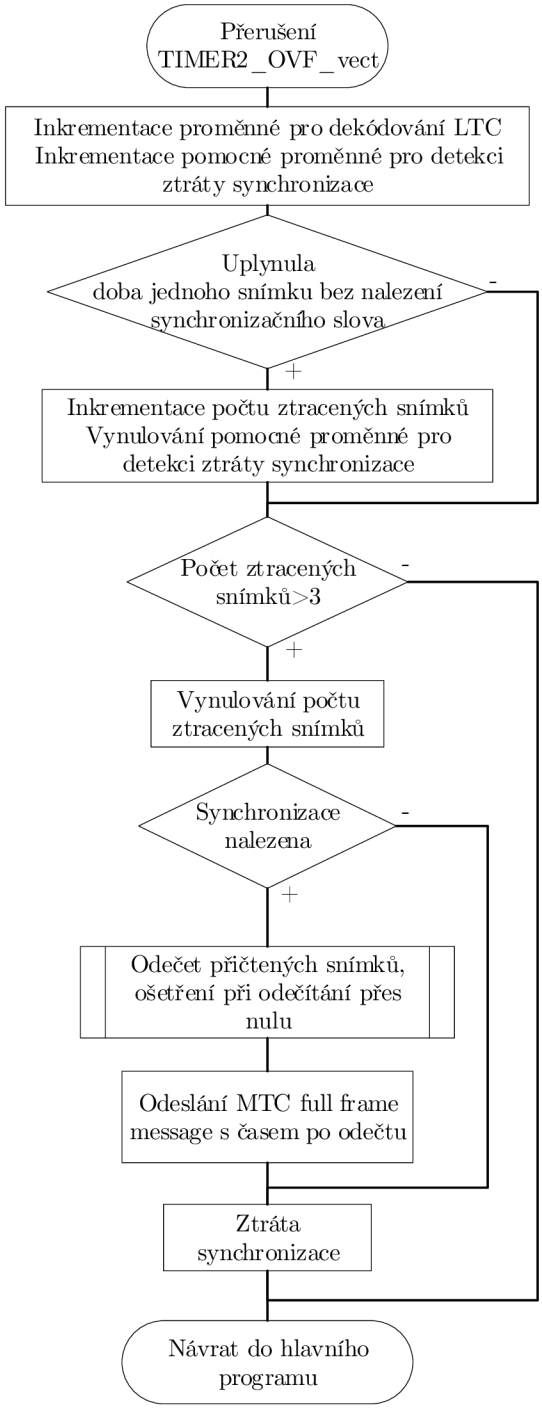
Část programu zajišťující konverzi z Lineárního časového kódu na MIDI časový kód sestává z celkem tří podprogramů – rutin obsluhujících přerušení. Jedná se o přerušení při přetečení čítače 2, přerušení při rovnosti hodnoty čítače 0 s komparačním registrem a přerušení při změně hodnoty na pinu – vstupu LTC. Přerušení při změně

hodnoty na pinu je pro tuto aplikaci obzvlášť výhodné tím, že nerozlišuje, zda jde o vzestupnou, či sestupnou hranu. Tyto tři podprogramy zajišťují měření doby mezi změnami na vstupu, dekódování vstupního LTC a odesílání MIDI čtvrtsnímkových zpráv. Jejich úkolem je zjišťování hodnoty příchozího bitu (dekódování BMC), hledání synchronizačního slova, hlídání ztráty synchronizace, zápis přijatých bitů na správné místo v paměti, zjištění snímkové frekvence příchozího signálu. Dále musí ve správnou dobu odesílat správné zprávy na MIDI rozhraní a v případě výpadku či porušení vstupního signálu musí obnovovat aktuální čas do obnovení synchronizace, či zastavit posun času po skutečném zastavení příchozího signálu. Příjem a dekódování příchozího LTC budou obstarávat rutiny obsluhy přerušení přetečení čítače 2 a změny hodnoty na vstupním pinu. Návrh algoritmu obsluhy přerušení přetečení čítače 2 ukazuje obrázek 2.8, návrh algoritmu obsluhy přerušení změny hodnoty vstupního pinu ukazuje obrázek 2.9.

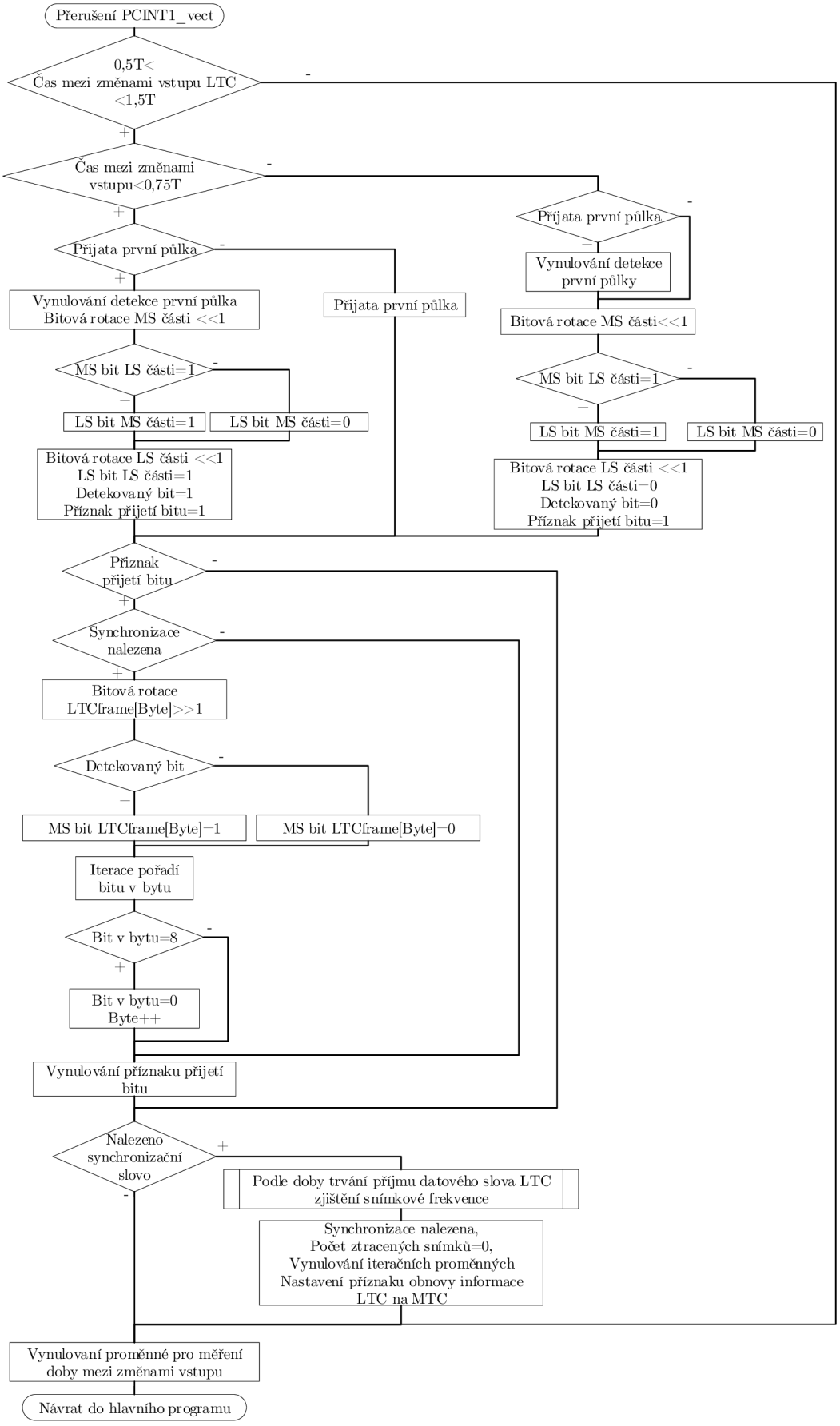
Pro dekódování LTC signálu se nabízely dvě možnosti – neustálé samplování vstupu a následné zpracování uložených dat nebo měření doby mezi změnami na vstupu. Kvůli charakteru vstupního signálu a na první pohled rychlejší a jednodušší implementaci druhé možnosti jsem již při návrhu hardwaru počítal s použitím právě této metody dekódování dat na vstupu. Vzhledem ke znalosti vstupního signálu (respektive jeho možnému rozsahu od 24 snímků/s do 30 snímků/s) není nutné dekodér navrhovat univerzální. Tímto dojde k velkému ulehčení v podobě změření doby mezi změnami vstupu a její zařazení do intervalu pro detekci jedničky (tedy 1/160 doby trvání snímku) nebo nuly (1/80 doby trvání snímku). Čítač/časovač 2 použitý právě pro měření této doby má maximální hodnotu 255, tedy přeteče a vyvolá 62500 přerušení za 1s, tzn. přerušení každých 16µs. Vypočtené počty přerušení mezi změnami stavu vstupu jsou pro všechny snímkové frekvence uvedeny v tabulce 2.6.

Tab. 2.6: Výpočet počtu přerušení mezi změnami vstupu pro dekódování LTC

Snímková frekvence [FPS]	30	30 df	25	24
Fosc [MHz]	16	16	16	16
Prescaler	1	1	1	1
Maximální hodnota čítače	256	256	256	256
Počet přerušení za 1s	62500	62500	62500	62500
Počet změn vstupu ve slově (nuly)	80	80	80	80
Počet změn vstupu za 1s (nuly)	2400	2398	2000	1920
Počet přerušení mezi změnami (nuly)	**26,04**	**26,07**	**31,25**	**32,55**
Počet změn vstupu ve slově (jedničky)	160	160	160	160
Počet změn vstupu za 1s (jedničky)	4800	4795	4000	3840
Počet přerušení mezi změnami (jedničky)	**13,02**	**13,03**	**15,63**	**16,28**



Obr. 2.8: Vývojový diagram obsluhy přerušení čítače 2.



Obr. 2.9: Vývojový diagram obsluhy přerušení změny stavu pinu.

Při každém přerušení z čítače 2 dojde k inkrementaci proměnné, jejíž hodnota je řídící pro detekci jedničky či nuly. Na začátku přerušení PCINT1 [12] jsou vyfiltrovány časy pod polovinou a nad 1,5 násobkem periody (hodnota proměnné *LM_pocetProDekodovani* mezi 10 a 35). Následně podle zjištění, zda je doba menší než 0,75 násobek periody (hodnota proměnné *LM_pocetProDekodovani* 20), je rozhodnuto, zda čas odpovídá přenosu jedničky či nuly. V případě jedničky ještě testujeme nastavení proměnné *LM_prvniPulka*. Pokud není nastavena, přijali jsme první půlku bitu a proměnnou nastavíme. Pokud již byla nastavena, právě jsme dekódovali druhou polovinu bitu a do proměnné *LM_dekodovanyBit* zapíšeme 1. V případě rozhodnutí o přijetí bitu s hodnotou nula do proměnné *LM_detekovanyBit* zapíšeme 0. Dále provedeme bitovou rotaci proměnných *LM_posledniByte* a *LM_posledniByteMS* a přijatý bit zapíšeme na uvolněné místo. Po přijetí bitu také nastavíme příznak přijetí bitu *LM_flag_bit*. Rozhodování o hodnotě přijatého bitu realizuje kód 2.1

Výpis 2.1: Detekce hodnoty přijatého bitu v obsluze přerušení PCINT1.

```
843     if (LM_pocetProDekodovani<=20) //rozsah 10 az 20
844     {
845       if (LM_prvniPulka==1)
846       {
847         LM_prvniPulka=0;
848         LM_posledniByteMS<<=1;
849         if ((LM_posledniByte&0b10000000)==0b10000000) //
               posun pro zjisteni synchronizacniho slova
850         {
851           LM_posledniByteMS|=0b00000001;
852         }
853         else if ((LM_posledniByte&0b10000000)==0b00000000
               )
854         {
855           LM_posledniByteMS&=0b11111110;
856         }
857         LM_posledniByte<<=1;
858         LM_posledniByte|=0b00000001;
859         detekovany_bit=1;
860         LM_flag_bit=1;
861       }
862       else
863       {
864         LM_prvniPulka=1;
```

```
865       }
866     }
867     else // rozsah 20 az 35
868     {
869       if (LM_prvniPulka==1)
870       {
871         LM_prvniPulka=0;
872       }
873       LM_posledniByteMS<<=1;
874       if ((LM_posledniByte&0b10000000)==0b10000000)
875       {
876         LM_posledniByteMS|=0b00000001;
877       }
878       else if ((LM_posledniByte&0b10000000)==0b00000000)
879       {
880         LM_posledniByteMS&=0b11111110;
881       }
882       LM_posledniByte<<=1;
883       LM_posledniByte&=0b11111110;
884       detekovany_bit=0;
885       LM_flag_bit=1;
886     }
```

Pokud byl přijat bit (*LM_flag_bit*=1) a bylo nalezeno synchronizační slovo (*LM_syncFound*=1), dojde k zapsání přijatého bitu do struktury *LM_LTCframe* na pozici podle proměnných *LM_bitByte* a *LM_byte*. Inkrementujeme tyto proměnné a ověříme, zda nejsme na konci slova. Dále je třeba ověřit, zda proměnné *LM_posledniByte* a *LM_posledniByteMS* jsou rovny synchronizačnímu slovu. Pokud platí rovnost, právě bylo nalezeno synchronizační slovo a začátek dalšího slova. Pomocí aktuální hodnoty proměnné *LM_freeWheel* zjistíme čas od posledního synchronizačního slova a dle této doby určíme snímkovou frekvenci. Zjištění snímkové frekvence realizuje kód 2.2. Dále nastavíme proměnnou *LM_syncFound*, vynulujeme *LM_bitByte* a *LM_byte*, nastavíme příznak obnovy informace v paměti *LM_flag_obnovaInformace* a vynulujeme proměnné pro detekci ztráty synchronizace. V samotném závěru této rutiny vynulujeme proměnnou pro zjištění času od předchozí změny a vše je připraveno na příjem dalšího bitu.

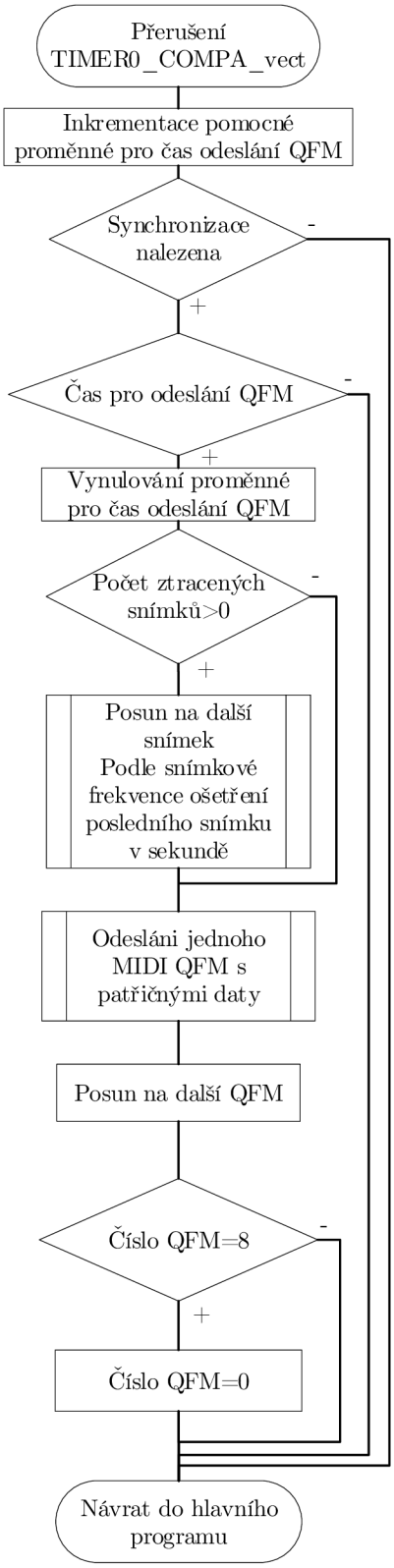
Výpis 2.2: Zjištění snímkové frekvence v obsluze přerušení PCINT1.

```
910     if (LM_posledniByte==0b11111101 && LM_posledniByteMS
           ==0b00111111) //detekce synchronizacniho slova
911     {
912       if (LM_freeWheel>=2075) //presne 2083,3  //pro
             zjisteni frame ratu z delky datoveho slova
913       {
914         LM_rate=FPS30;
915         if (((LM_LTCframe[1]>>2)&0x01)==1) // drop frame
916         {
917           LM_rate=FPS30D;
918         }
919         if (LM_freeWheel>=2400)//presne 2500
920         {
921           LM_rate=FPS25;
922           if (LM_freeWheel>=2550)//presne 2604;
923           {
924             LM_rate=FPS24;
925           }
926         }
927       }
928       LM_syncFound=1;
929       LM_lostFrames=0;
930       LM_bitByte=0;
931       LM_byte=0;
932       LM_pocetProQFM=0;
933       LM_flag_obnovaInformace=1; //vynulovani iteracnich
             promennych a nastaveni priznaku pro prepis
             casove informace
934       LM_freeWheel=0;
935     }
```

V obsluze přerušení od čítače/časovače 2 je kromě inkrementace proměnných *LM_pocetProDekodovani* a *LM_freeWheel* také hlídání nalezení synchronizačního slova v daném časovém intervalu. Tento interval byl stanoven na 41,7 ms (hodnota proměnné *LM_freeWheel* 2610), což odpovídá délce datového slova při snímkové frekvenci 24 snímků/s. Při překročení této doby bez nalezení synchronizačního slova dojde k inkrementaci *LM_lostFrames* a vynulování *LM_freeWheel*. Pokud je hodnota proměnné *LM_lostFrames* větší než nula, při odesílání MIDI čtvrtsnímkové

zprávy dochází k internímu posunu času (Free wheel algoritmus), čímž jsou ošetřeny případné krátkodobé výpadky vstupního signálu. Maximální dobu, kterou lze považovat za výpadek, jsem stanovil na 167 ms, to odpovídá čtyřem snímkům při snímkové frekvenci 24 snímků/s. Pokud je tedy překročen tento nastavený mezní čas, dojde podle snímkové frekvence k odečtu počtu snímků, který byl přičten free wheel algoritmem. Samozřejmostí je ošetření odečítání do záporných čísel – posun na konec předcházející časové jednotky. Po snížení časové informace o danou hodnotu je odeslána celosnímková zpráva s časem, kdy byla ztracena synchronizace, tento čas je tedy považován za skutečný čas zastavení transportu na časové ose. Další nezbytnou operací při překročení daného času je nastavení *LM_syncFound* na nulu, aby došlo k zastavení odesílání MIDI čtvrtsnímkových zpráv.

Poslední částí náležející směru konverze z LTC na MTC je obsluha přerušení rovnosti hodnoty čítače/časovače 0 s komparačním registrem. Vývojový diagram této části programu je zobrazen na obrázku 2.10. Opět, stejně jako u předchozího časovače, při každém vyvolaném přerušení dochází k inkrementaci pomocné proměnné pro měření času (*LM_pocetProQFM*). Pokud je nalezena synchronizace ve vstupním signále (*LM_syncFound=1*) a nastane doba pro odeslání čtvrtsnímkové zprávy (*LM_pocetProQFM*=82 podle tabulky 2.3), je vynulována tato pomocná proměnná a zjištěno, zda je počet snímků bez nalezení synchronizačního slova větší než nula. Pokud ano, inkrementujeme počet snímků a ověříme dle snímkové frekvence, zda se nejedná o poslední snímek v sekundě. V případě posledního snímku inkrementujeme sekundy a vynulujeme číslo snímku. Nyní podle *LM_kolikatyQFM* odešleme čtvrtsnímkovou zprávu s patřičnými daty a inkrementujeme tuto proměnnou pro odeslání další zprávy.



Obr. 2.10: Vývojový diagram obsluhy přerušení čítače 0.

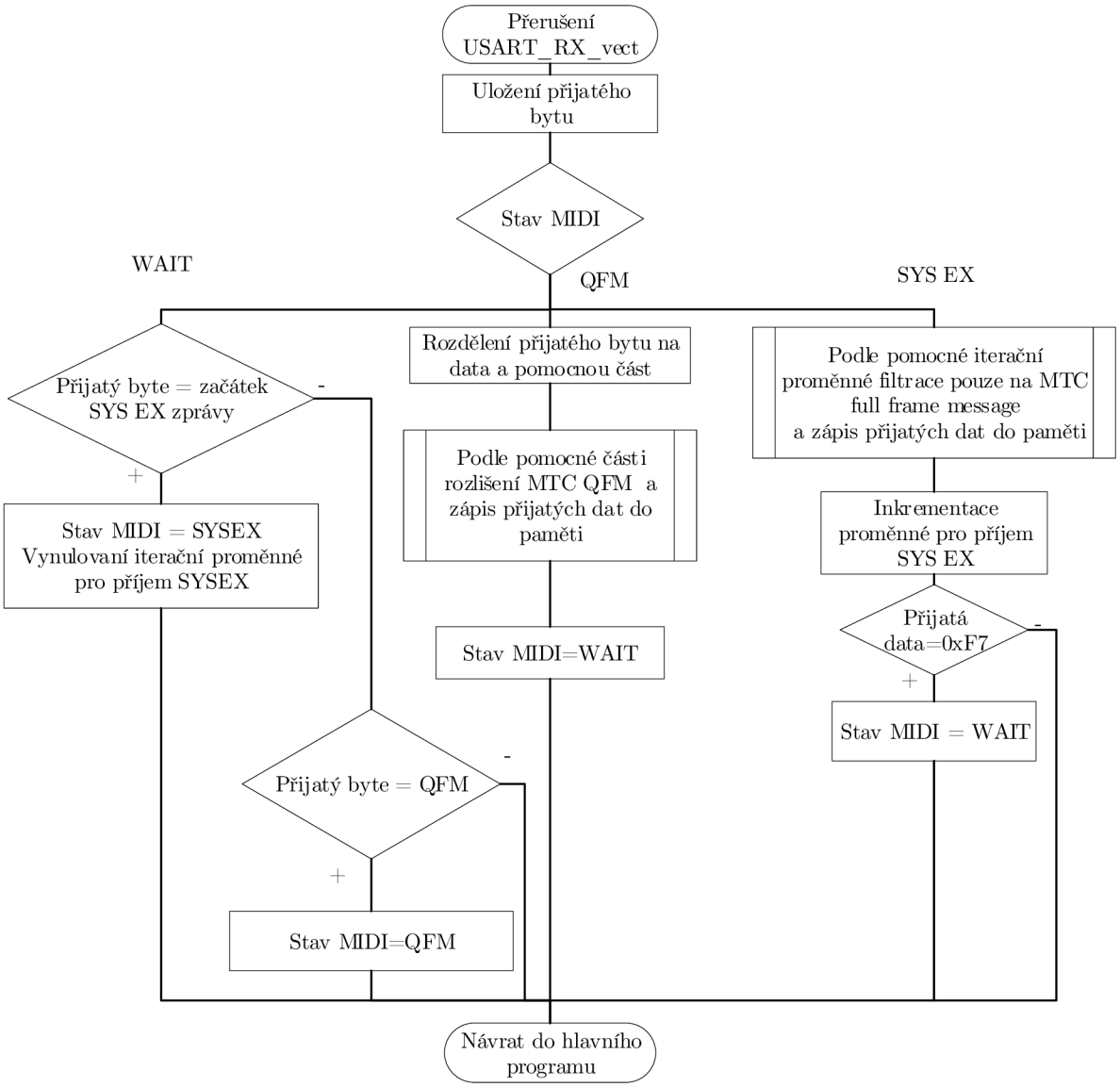
2.2.2 Konverze z MTC na LTC

Část programu zajišťující konverzi z MIDI časového kódu na Lineární časový kód sestává ze dvou podprogramů - rutin obsluhujících přerušení. Jedná se o přerušení při přijetí bytu přes UART rozhraní a přerušení při rovnosti hodnoty čítače 1 s komparačním registrem. Tyto dva podprogramy zajišťují přijímání a dekódování MIDI zpráv a kódování a odesílání LTC signálu. Jejich úkolem je kromě správného dekódování příchozích zpráv také detekce zastavení jejich příchodu, formování LTC rámce podle aktuálních dat a kódování dat a jejich zápis na výstup ve správný čas. Příjem a dekódování obstarává pouze rutina obsluhující přerušení při příjmu dat přes UART rozhraní, zatímco kódování a odesílání LTC signálu zajišťuje rutina přerušení rovnosti čítače 1 s nastavenou hodnotou. Návrh algoritmu obsluhy přerušení přijetí bytu přes UART rozhraní ukazuje obrázek 2.11, návrh algoritmu obsluhy přerušení rovnosti čítače 1 s nastavenou hodnotou je zobrazen na obrázku 2.12.

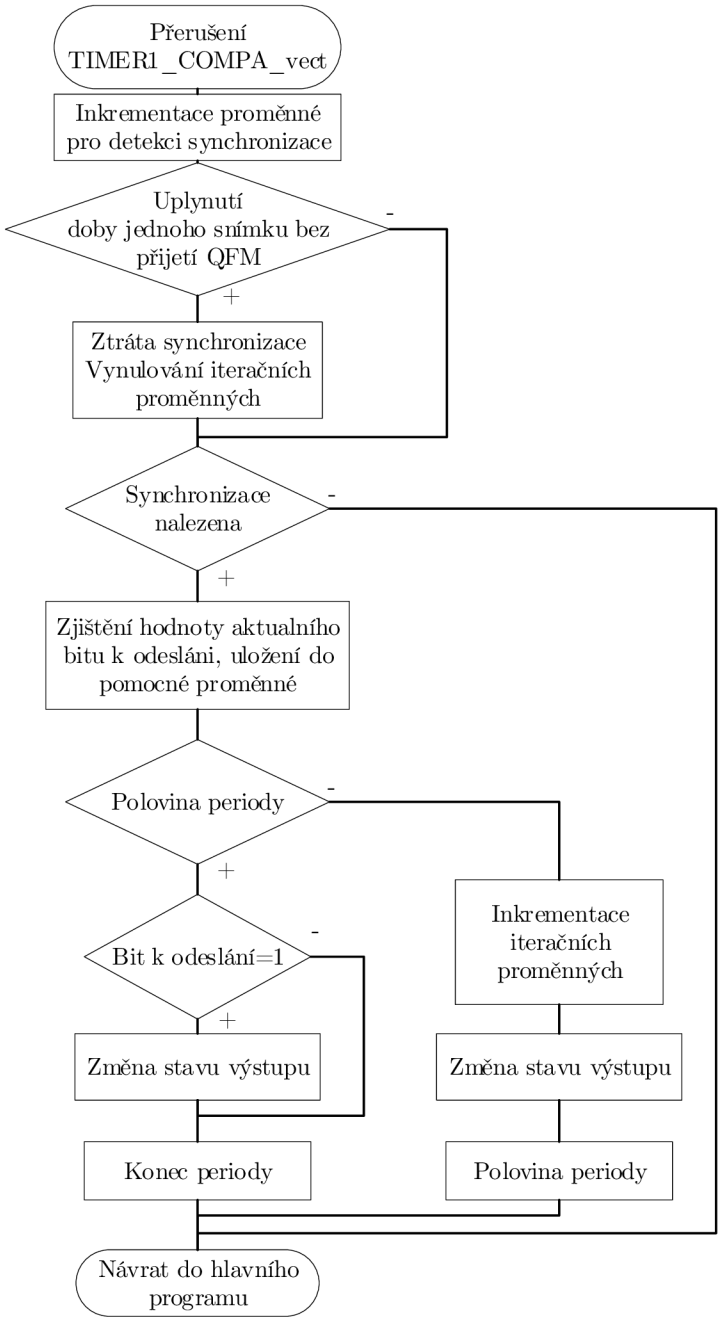
Po přijetí celého bytu přes UART je vyvoláno přerušení „USART_RX“.[20] Prvním krokem v obsluze tohoto přerušení je uložení přijatého bytu do lokální proměnné *prijato*. Dekódování přijatých bytů do MIDI zpráv je řízeno stavovou proměnnou *ML_midiState*.[21] Pokud je její hodnota rovna jedné (stav MIDI_WAIT), je zjištěno, zda hodnota přijatého bytu odpovídá bytu identifikujícímu MIDI čtvrtsnímkovou zprávu. Pokud byla tato identifikována, je do řídící proměnné *ML_midiState* zapsána hodnota 2 (MIDI_QFM) a při přijetí dalšího bytu je jeho hodnota zapsána do proměnné pro patřičný časový údaj podle čísla této čtvrtsnímkové zprávy, které je přijato v horní polovině tohoto přijatého bytu. Pokud nebyl identifikován byte značící čtvrtsnímkovou zprávu, otestujeme, zda přijatý byte neznamená začátek Systémové exkluzivní zprávy. Pokud ano, nastavíme stavovou proměnnou *ML_midiState* na hodnotu 3 (MIDI_SYSEX). Při přijetí dalších bytů testujeme, zda jde o zprávu MIDI časového kódu, pokud ano, zapíšeme přijatý časový údaj do paměti. Po přijetí bytu značícího konec systémové exkluzivní zprávy přejdeme opět do stavu MIDI_WAIT, tedy čekání na příchod další zprávy. Při příjmu čtvrtsnímkových zpráv s čísly 0 a 4, jejichž příchod značí začátek snímku, je nastaven příznak pro obnovu informace v paměti *ML_flag_obnovaInformace*, dojde k nastavení proměnné *ML_syncFound* značící nalezení synchronizace na příchozí signál (spuštění transportu na časové ose) a vynulování proměnné *ML_runningDetection* pro měření času do začátku dalšího snímku, tedy pro detekci zastavení transportu.

Nyní máme vyřešen příjem a dekódování dat na UART rozhraní, zbývá přijatá data správně zformovat do LTC datového slova a ve správný čas je postupně posílat na výstup. To je realizováno v rutině obsluhující přerušení čítače/časovače 1 „TIMER1_COMPA“.[12] Toto přerušení je vyvoláno přesně 160krát za LTC datové slovo (80 bitů, všechny o hodnotě 1) pro všechny snímkové frekvence. Tato

nezávislost je dána správným nastavením hodnoty komparačních registrů v hlavní části programu podle přijaté snímkové frekvence tak, aby doba mezi přerušeními odpovídala 1/160 doby trvání snímku.



Obr. 2.11: Vývojový diagram obsluhy přerušení přijetí bytu na UART rozhraní.



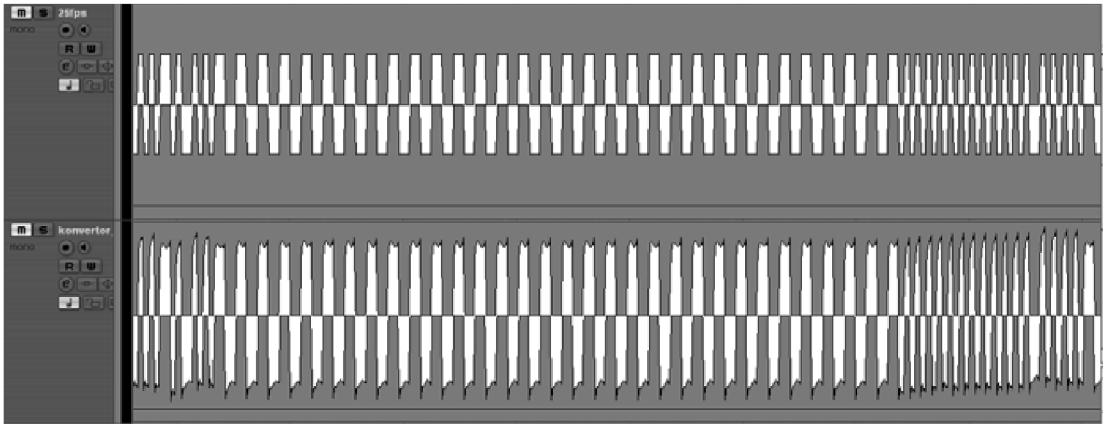
Obr. 2.12: Vývojový diagram obsluhy přerušení čítače 1.

Na začátku obsluhy tohoto přerušení je inkrementována proměnná pro detekci zastavení transportu *ML_runningDetection*. Dále je testováno, zda nedošlo k zastavení transportu. Toho je docíleno srovnáním hodnoty proměnné *ML_runningDetection* s číslem 320 (během jednoho snímku je inkrementována 160krát, odpovídá tedy dvěma snímkům bez přijetí MIDI čtvrtsnímkové zprávy). V případě zapnutého transportu a příchodu MIDI čtvrtsnímkových zpráv je hodnota *ML_runningDetection*

na začátku každého snímku vynulována. Když dojde k zjištění zastavení transportu, vynulujeme proměnné *ML_syncFound* a *ML_runningDetection*. Stejně tak vynulujeme proměnné *ML_bitByte* a *ML_byte* využívané pro definování aktuální polohy – bitu v odesílaném LTC datovém slově. Pokud je nalezena synchronizace, podle hodnot proměnných *ML_bitByte* a *ML_byte* zjistíme aktuální pozici v LTC datovém slově a hodnotu bitu na ní zapíšeme do *ML_transmit*. Nyní zjistíme, jsme-li časově v polovině periody či na konci. Nacházíme-li se v čase poloviny periody, zjistíme, zda je hodnota bitu k odeslání rovna jedné, v případě kladného výsledku změníme stav výstupu. Jsme-li na konci, stav výstupu změníme při obou hodnotách bitu k odeslání a inkrementujeme *ML_bitByte* pro posun na odeslání dalšího bitu. Pokud jsme na konci bytu, inkrementujeme *ML_byte* a vynulujeme *ML_bitByte*.

2.3 Ověření funkčnosti a přesnosti

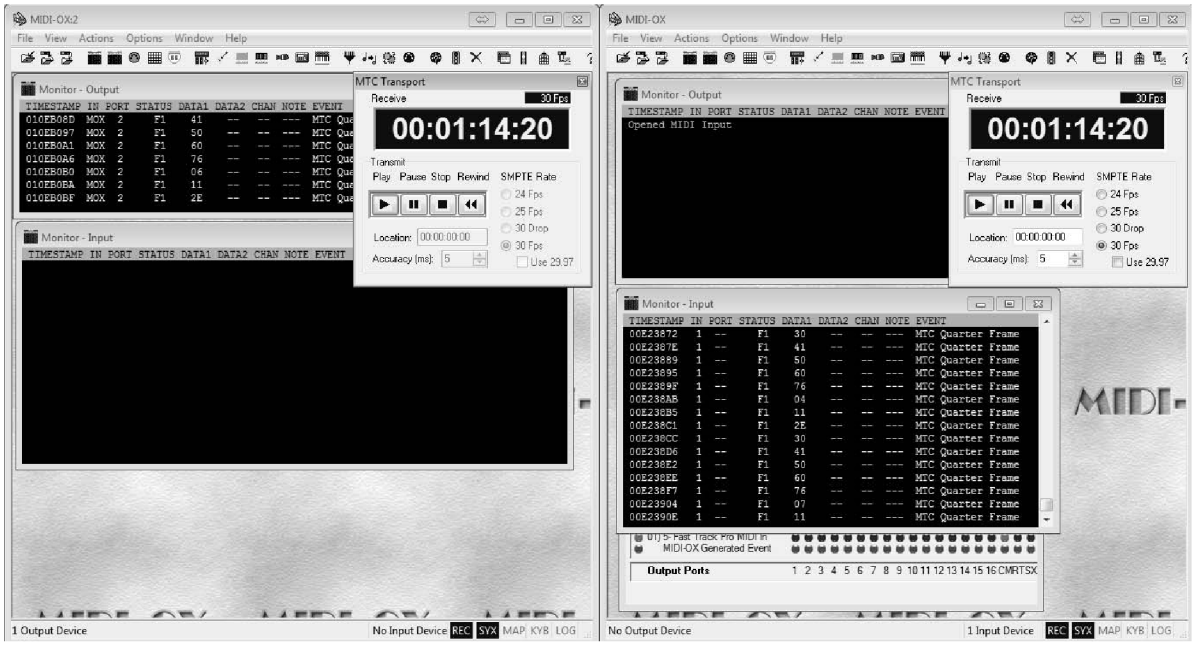
Ověření funkčnosti, přesnosti a požadovaných vlastností jsem prováděl pomocí počítačových programů Cubase 4 a MidiOX (verze 7) a USB zvukové karty s MIDI rozhraním M-AUDIO Fast Track Pro, která je vybavena jak audio vstupem a výstupem, tak i MIDI vstupem a výstupem. Program MidiOX jsem využíval ke generování a příjmu MIDI časového kódu. V programu Cubase byl pomocí plug-in modulu SMPTE generátoru generován testovací Lineární časový kód. Bohužel jsem nenalezl žádný program, který by uměl dekódovat a zobrazit Lineární časový kód z audio vstupu zvukové karty. Jedinou možností otestování správného generování LTC byl tedy jeho záznam v již zmíněném programu Cubase. Po dostatečném přiblížení zaznamenaného signálu jsem vizuálně porovnal jeho průběh s programem vygenerovaným signálem. Průběh obou signálů ukazuje obrázek 2.13.



Obr. 2.13: Srovnání průběhů signálu jednoho LTC datového slova (nahoře generované počítačem, dole výstup z konvertoru)

Nejprve jsem testoval přesnost konverze z LTC na MTC, neboť pro tento test jsem měl k dispozici generátor LTC i přijímač MTC. Testoval jsem dva parametry – přesnost při zapnutém transportu a konečný čas po zastavení transportu. Samotné porovnávání hodnot jsem provedl současným zobrazením oken obou programů (Cubase jako generátor a MidiOX jako přijímač) na jednom monitoru počítače a po zastavení transportu srovnáním zobrazených časů v těchto programech. Po doladění parametrů jsem se dostal na přesnost, kdy po zastavení transportu byly v obou programech zobrazeny stejné časy (testováno při všech snímkových frekvencích 24, 25, 29,97 i 30 snímků/s). Testování přesnosti při zapnutém transportu bylo malinko složitější kvůli neustále se měnícím hodnotám. Tuto potíž jsem vyřešil zaznamenáním aktuálního snímku monitoru pomocí funkce „Print Screen“. Na zaznamenaném snímku již bylo možné přímo porovnat aktuálně zobrazené hodnoty času v obou programech. Zjištěná přesnost byla výborná, zobrazené časy se sobě přímo rovnaly. Tato přesnost se potvrdila i při opakovaných měřeních a všech snímkových frekvencích.

Druhým v pořadí byl test přesnosti konverze z MTC na LTC. Správná podoba výstupního LTC signálu již byla ověřena, ale jediným přijímačem LTC, který jsem měl k dispozici, byl samotný sestavený konvertor. Konverze z LTC na MTC již byla na přesnost ověřena, bylo možné ji tedy použít jako přijímač LTC generovaného částí konvertoru pro směr konverze z MTC na LTC. Výstup LTC byl přímo propojen na vstup LTC, fakticky tedy byla porovnávána hodnota generovaného MTC s hodnotou přijímaného MTC. Tento test probíhal obdobně jako předchozí, pouze byly současně zobrazeny dvě okna programu MidiOX (jedno pro generování a odesílání MTC, druhé pro příjem MTC). Snímek obrazovky z tohoto testu je zobrazen na obrázku 2.14. I zde se přesnost ukázala jako výborná – při zapnutém transportu se oba údaje rovnaly. Pouze při zastavení transportu byl údaj přijímače o dva snímky vyšší než údaj generátoru. Příčinu této nepřesnosti jsem identifikoval v detekci zastavení transportu ve stupních MIDI datech, bohužel se ji ale nepodařilo odstranit.



Obr. 2.14: Snímek obrazovky při testu přesnosti (vlevo generátor, vpravo přijímač).

3 ZÁVĚR

Tato práce je věnována návrhu a realizaci konvertoru synchronizačních kódů MTC a LTC. Nejprve jsou uvedeny všechny důležité informace o jednotlivých částech (standardech) použitých při návrhu konvertoru. Samotný návrh je popsán v další části a je rozdělen do tří kapitol. První z nich popisuje hardwarovou část návrhu, v další je popsán návrh algoritmu a samotné sestavení programu pro použitý mikroprocesor. Poslední kapitola této části je věnována popisu testování funkčnosti realizovaného konvertoru a ověření jeho přesnosti. Na základě těchto testů se neprokázala žádná z mých původních obav z nedostatečného výkonu použitého mikroprocesoru, tudíž nebylo nutné návrh nijak dodatečně upravovat. Při ověření přesnosti byly nalezeny odchylky mezi vstupními a výstupními daty, tyto byly ale softwarově kompenzovány. Časový údaj zakódovaný ve výstupním signálu se rovná údaji v signálu vstupním. Během testování konvertoru jsem nezaznamenal jakoukoliv nestabilitu či jiný nestandardní stav, který by způsobil výpadek výstupních dat.

Na základě těchto zjištění bych v této práci popsaný návrh označil za funkční a zdařilý.

LITERATURA

[1] MIDI Assoiciation. *The Complete MIDI 1.0 Detailed Specification, 3rd ed. MIDI Assoiciation, document version 96.1* 1996 Dostupné z URL: <https://www.midi.org/downloads?task=callelement&format=raw&item_id=92&element=f85c494b-2b32-4109-b8c1-083cca2b7db6&method=download>.

[2] Wikipedia. *MIDI* [online]. 2017 [cit. 2017-12-13]. Dostupné z URL: <https://en.wikipedia.org/wiki/MIDI>.

[3] Wikipedia. *MIDI timecode* [online]. 2017 [cit. 2017-12-06]. Dostupné z URL: <https://en.wikipedia.org/wiki/MIDI_timecode>.

[4] Wikipedia. *SMPTE timecode* [online]. 2016 [cit. 2017-12-05]. Dostupné z URL: <https://en.wikipedia.org/wiki/SMPTE_timecode>.

[5] Wikiaudio.org. *SMPTE time code* [online]. 2009 [cit. 2017-12-12]. Dostupné z URL: <http://en.wikiaudio.org/SMPTE_time_code>.

[6] REES, Phil. *SMPTE EBU timecode by Phil Rees* [online]. 2001 [cit. 2017-12-12]. Dostupné z URL: <http://www.philrees.co.uk/articles/timecode.htm>.

[7] RUMSEY, Francis; McCormick, Tim. *Sound and Recording*. Oxford, Focal Press, Sixth edition, 2009. ISBN 978-0-24-052163-3.

[8] Wikipedia. *Linear timecode* [online]. 2017 [cit. 2017-12-05]. Dostupné z URL: <https://en.wikipedia.org/wiki/Linear_timecode>.

[9] ITU. *Recommendation ITU-R BR.780-2: Time and control code standards, for production applications in order to facilitate the international exchange of television programmes on magnetic tapes* . 2005 Dostupné z URL: <http://www.itu.int/dms_pubrec/itu-r/rec/br/R-REC-BR.780-2-200504-I!!PDF-E.pdf>.

[10] BARNES, Neil. *Biphase-Mark encoding and decoding - a short guide.* . 2007 Dostupné z URL: <http://www.avrfreaks.net/sites/default/files/biphase.pdf>.

[11] AlexDWong. *MTCDisplay* [online]. 2016 [cit. 2017-12-05]. Dostupné z URL: <https://github.com/AlexDWong/MTCDisplay>.

[12] Atmel Corporation. *Atmel-42735B-ATmega328/P_Datasheet_Complete-11/2016* [online]. 2016 [cit. 2018-04-06]. Dostupné z URL:

<http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf>.

[13] Gareus, Robin. *libltc: POSIX-C Library for handling Linear/Logitudinal Time Code (LTC).* . 2012 Dostupné z URL: <http://x42.github.io/libltc/>.

[14] Gareus, Robin. *libltcsmpte: LTC SMPTE Library.* . 2010 Dostupné z URL: <http://ltcsmpte.sourceforge.net/>.

[15] Gareus, Robin. *libtimecode: POSIX-C Library for handling Audio/Video Timecode and Framerate.* . 2012 Dostupné z URL: <http://x42.github.io/libtimecode/>.

[16] Hackaday.io. *Arduino TIMECODE smpte LTC reader generator SHIELD* [online]. 2015 [cit. 2017-12-12]. Dostupné z URL: <https://hackaday.io/project/7694/logs>.

[17] Edaboard.com. *Time-code decoder - SMPTE - BiPhase Mark - clock recovery - full design* [online]. 2016 [cit. 2017-12-12]. Dostupné z URL: <http://www.edaboard.com/thread352841.html>.

[18] Heer Donald. *Using the USBASP Programmer with AVR Studio v7* [online]. 2015 [cit. 2018-04-06]. Dostupné z URL: <https://www.youtube.com/watch?v=5zHI_Gy9ziw>.

[19] Hämmerling Mark. *Engbedded AVR Fuse Calculator* [online]. 2014 [cit. 2018-04-06]. Dostupné z URL: <http://www.engbedded.com/fusecalc/>.

[20] Atmel Corporation. *Atmel-1451C-AVR306-Using-the-AVR-USART-on-tiny AVR-and-megaAVR-devices_Application Note-04/2016* [online]. 2016 [cit. 2018-04-06]. Dostupné z URL: <http://ww1.microchip.com/downloads/en/AppNotes/Atmel-1451-Using-the-AVR-USART-on-tinyAVR-and-megaAVR-devices_ApplicationNote_AVR306.pdf>.

[21] BOOURNS. *C library for quickly creating AVR MIDI devices* [online]. 2012 [cit. 2018-04-06]. Dostupné z URL: <https://github.com/boourns/AVRMIDI>.

[22] SCHIMMEL, Jiří. *Studiová a hudební elektronika.* Brno: Vysoké učení technické v Brně, 2015. ISBN 978-80-214-4452-2. (cs).

[23] TimeLine Vista, Inc. *SMPTE Made Simple* . 1996 Dostupné z URL: <http://faculty.spokanefalls.edu/InetShare/AutoWebs/steveg/SmpteMadeSimple.pdf>.

SEZNAM PŘÍLOH

A OBSAH PŘILOŽENÉHO CD

Na přiloženém CD je v kořenovém adresáři elektronická verze této práce ve formátu PDF. Dále je zde adresář Datasheet, v nemž je uložena dokumentace k použitým integrovaným obvodům. V adresáři DPS jsou uloženy soubory s návrhem desky plošného spoje ve formátu návrhového softwaru Eagle (konvertor.sch a konvertor.brd). V posledním adresáři nazvaném SW je uložen celý projekt „konvertor.sln“ spustitelný v programu Atmel studio. Dále se v tomto adresáři nachází zdrojový kód v jazyce C (soubor „main.c“ a přeložený soubor s instrukcemi pro mikroprocesor ATmega328P „konvertor.hex“).