UNIVERZITA PARDUBICE
DOPRAVNÍ FAKULTA JANA PERNERA

Aplikace pro prohlížení záznamů na paměťové kartě řidiče a jejich archivaci

Bc. Martin Špatenka

Diplomová práce

2010

Univerzita Pardubice
Dopravní fakulta Jana Pernera
Akademický rok: **2009/2010**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Martin ŠPATENKA**
Studijní program:	**N3708 Dopravní inženýrství a spoje**
Studijní obor:	**Aplikovaná informatika v dopravě**
Název tématu:	**Aplikace pro prohlížení záznamů na paměťové kartě řidiče a jejich archivaci**
Zadávající katedra:	**Katedra informatiky v dopravě**

Zásady pro vypracování:

- Prohloubit teoretické znalosti problematiky evidence doby práce osádek vozidel v mezinárodní i vnitrostátní silniční dopravě, zejména se zaměřením na využití digitálních záznamových zařízení / digitální tachograf a paměťová karta řidiče.
- Osvojit si komunikaci s externím USB zařízením / čtečkou paměťových karet / na úrovni operačního systému i ve vybraném vyšším programovacím jazyce.
- Seznámit se s datovou strukturou zápisu údajů na paměťové kartě řidiče a osvojit si čtení těchto údajů.
- Navrhnout a ve vybraném vyšším programovacím jazyce implementovat samostatně běžící aplikaci, která by uměla načtená data z paměťové karty řidiče analyzovat, resp. zobrazovat v grafické podobě, běžné v oboru silniční dopravy. Zejména se jedná o zobrazení aktuálního stavu najetých hodin řidiče, resp. jeho aktuální pozice v denním/týdenním cyklu. Dále též archivaci a prohlížení historických dat, resp. synchronizaci aktuálních dat z karty s již archivovanými údaji.

Rozsah grafických prací:

Rozsah pracovní zprávy: **50 normostran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **Nařízení Evropského parlamentu a Rady (ES) č. 561/2006, o harmonizaci některých předpisů v sociální oblasti týkajících se silniční dopravy, o změně nařízení Rady (EHS) č. 3821/85 a (ES) č. 2135/98 a o zrušení nařízení Rady (EHS) č. 3820/85.**
2. **Nařízení Rady (EHS) č. 3821/85, o záznamovém zařízení v silniční dopravě.**

Vedoucí diplomové práce: **Ing. Viktor Patras**
Katedra informatiky v dopravě

Datum zadání diplomové práce: **24. listopadu 2009**

Termín odevzdání diplomové práce: **24. května 2010**

prof. Ing. Bohumil Culek, CSc.
děkan

L.S.

doc. Ing. Josef Volek, CSc.
vedoucí katedry

V Pardubicích dne 20. listopadu 2009

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

v Pardubicích dne 15. 5. 2010

Martin Špatenka

Souhrn:

Předmětem této práce je vytvoření programu pro načítání, analýzu a archivaci dat, uložených na paměťové kartě řidiče. Načtená data je možné zobrazit jak tabelárně tak i graficky. V programu jsou dále implementovány algoritmy kontroly dodržování přestávek a dob řízení podle nařízení č. 561/2006.

Klíčová slova:

Paměťová karta řidiče, čipová karta, digitální tachograf

Summary:

The main goal of this work is to create a program for reading, analysing and preserving of data stored in a driver's memory card. The data from the card can be displayed in tabular or graphical way. There are also algorithms for controlling respecting driving breaks and driving durations, implemented in the program, all in accordance with 561/2006 regulation.

Keywords:

Memory Driver Card, Smart Card, Digital Tachograph

Obsah

Seznam obrázků

1 Úvod

Cílem této diplomové práce je vytvoření samostatně běžící desktopové aplikace, která bude schopna načítat a dále pracovat s daty uloženými na čipové kartě řidiče. Ta se využívá převážně v silniční nákladní dopravě jako jedna z karet digitálního tachografu. Karta řidiče a digitální tachograf obecně nahrazují již zastaralé zaznamenávání jednotlivých činností řidiče, které se provádělo na papírová „kolečka“, vkládaná do klasického (analogového) tachografu, na která se vykreslovala křivka těchto činností. Výsledná aplikace může být naprogramována v libovolném vyšším programovacím jazyce a měla by umožňovat načítání, archivaci, zobrazování (grafické i tabelární) a analýzu dat, načtených z karty řidiče. Do analýzy dat se zahrnuje kontrola činností podle nařízení č. 561/2006, které nařizuje řidičům povinné dodržování přestávek a dob řízení.

Jednoduchá aplikace obsahující všechny tyto funkce by byla vhodná pro malé dopravce, případně jednotlivce, které ve svých vozech povinně využívají digitální tachografy a které mají ze zákona nařízenou povinnou archivaci záznamů činností jednotlivých řidičů po dobu nejméně jednoho roku, avšak pořízení profesionálních softwarových řešení pro velké společnosti je pro ně finančně zbytečně nákladné.

Důvodem zvolení tohoto tématu bylo také získání nových zkušeností v oblasti práce s externím zařízením (v tomto případě čtečkou čipových karet a čipovou kartou jako takovou). Dalším důvodem bylo rozšíření teoretických vědomostí záznamových zařízení používaných v silniční dopravě o praktické zkušenosti s funkcionalitami digitálního tachografu a zaznamenávání dat do něj a na čipové karty.

2 Čipové karty

Čipové karty jsou jedním z podruhů plastových karet, které se vyznačují vestavěným integrovaným obvodem (čipem) a zpravidla kapesní velikostí. Tyto karty se vyznačují vysokou bezpečností uložených dat před zneužitím a přitom umožňuji bezpečnou a spolehlivou autorizaci. Problematika čipových karet je popsána v [1]-[9].

2.1 Historie

O vznik čipové karty se zasloužil japonský vynálezce Kunitaka Arimura, který jako první zapustil mikroelektronický obvod do plastikového substrátu. Svůj vynález registroval v roce 1970. Jeho patent byl však zaměřen hlavně na způsob, jak zasadit mikročip do karty a nevyužil funkčních vlastností systému. O čtyři roky později, v roce 1974, si francouzský žurnalista Roland Moreno zaregistroval patent na „nezávislý elektronický objekt s pamětí“. Zde Moreno věnoval více pozornosti funkčním aspektům karty a začal využívat kontrolu přístupu k datům uloženým na kartě, neboli PIN (*Personal Identification Number* – osobní identifikační číslo). Začátkem roku 1980 byla technologie čipových karet už tak vyspělá, že se o ní začala zajímat francouzská vláda a firmy z finančních, zdravotnických i telekomunikačních sektorů, ve kterých se využívají dodnes.

2.2 Rozdělení

Čipových karet je několik typů a dají se rozdělit podle několika různých hledisek:

- podle typu použitého čipu a způsobu komunikace se dělí karty na:
  - kontaktní
  - bezkontaktní
  - hybridní
  - kombinované (duální)
- podle funkce respektive použití karty se dělí na:
  - paměťové
  - s mikroprocesorem (neboli „chytré,“ nebo z anglického názvu *Smart card*)
- podle způsobu komunikace s kartou na:
  - synchronní (např. česká telefonní karta)
  - asynchronní (například SIM karty)
- podle konkrétního komunikačního protokolu – komunikace pomocí „sběrnic“ se dělí na:
  - I2C (tato sběrnice je někdy též označována 2 wire)
  - MicroWire (někdy též značené 3 wire) a dále různé proprietární sběrnice
  - SPI

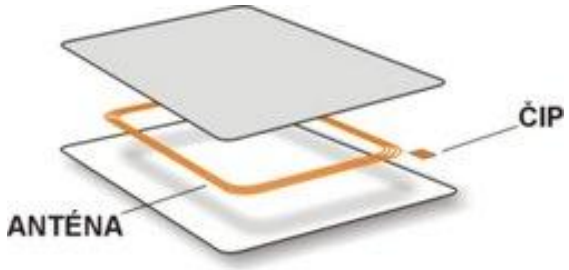
V tomto textu budou dále rozebírány převážně první dva druhy tj. podle způsobu komunikace a podle funkce čipových karet a dále pak asynchronní komunikace s kartou.

2.2.1 Bezkontaktní karty

Bezkontaktní čipové karty jsou v dnešní době nejvíce perspektivním typem karet. Využívá se technologie radiofrekvenční identifikace RFID. Čipová karta obsahuje vestavěnou indukční anténu a čip s integrovaným obvodem, který se nachází v těle karty. Pro lepší mechanickou ochranu je čip s integrovaným obvodem umístěn do drobného bloku, který se připojuje ke koncům antény. Vestavěný integrovaný obvod se skládá ze dvou dílů – bezkontaktního radiofrekvenčního (RF) rozhraní a mikrořadiče. Obvod RF rozhraní je připojený do výstupu antény čipové karty a používá střídavé elektromagnetické pole emitované čtečkou (snímatelem) pro získání napájecí energie pro čip a pro výměnu dat mezi kartou a čtečkou.

Čipová karta nemá vlastní zdroj elektrické energie a je proto zcela závislá na dodávce energie ze čtečky. U bezkontaktních karet se energie přenáší ve formě indukce elektromagnetického pole ze čtečky do antény čipové karty. Energie indukovaná v anténě karty slouží k napájení čipu. Zajímavým problémem je zpětný přenos informace od karty ke čtečce. Oblíbeným omylem je představa, že karta aktivně vysílá. Ve skutečnosti karta používá princip zátěžové modulace (*load modulation*), kdy odebírá větší nebo menší množství energie z elektromagnetického pole čtečky a reprezentuje tak bity 0 a 1. Čtečka detekuje změny odběru a „přijímá“ tímto způsobem data „vysílaná“ kartou.

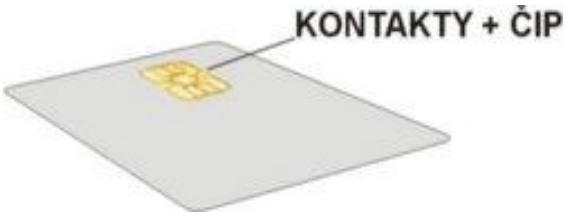
Bezkontaktní čipové karty fungují při vzdálenosti od 10 cm do 1 m v závislosti od pracovní frekvence čtečky a nepotřebuji přesné centrování, což zaručuje jejich stabilní práci, pohodlí při využití a vysokou přenosovou schopnost, která činí až 848 Kbitů/sekundu.



Obr 1 – Bezkontaktní čipová karta (zdroj: [5])

2.2.2 Kontaktní karty

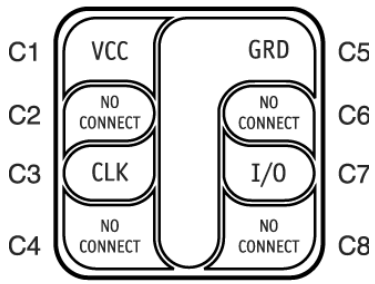
Kontaktní čipové karty se vyznačují tím, že mají na svém povrchu kontaktní plošku s šesti nebo osmi kontakty, jejichž funkce a umístění na čipové kartě je standardizováno normou ISO/IEC 7816-2. Tyto karty musí být pro svou funkčnost vloženy do čtečky karet, kde se její kontakty musí fyzicky dotýkat povrchu této plošky. Jednotlivé kontakty této plošky slouží pro napájení čipu, sériovou komunikaci, přivedení externího taktovacího signálu a programovacího napětí. Důležité jsou dva kontakty rezervované pro budoucí využití, které se již v současnosti používají u některých karet pro alternativní USB rozhraní.



Obr 2 – Kontaktní čipová karta (zdroj: [5])

Kontakty musí mít minimální pravoúhlou oblast povrchu, která není menší než obdélník o rozměrech 2 mm na 1,7 mm. Každý kontakt musí být elektricky izolován od ostatních

kontaktů. Na následujícím obrázku se nachází detail kontaktní plošky s vyznačením a očíslováním jednotlivých kontaktů.

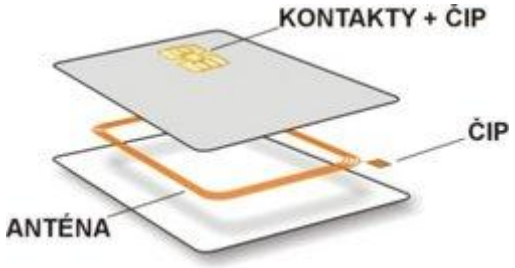


Obr 3 – Kontaktní ploška (zdroj: [9])

C1: VCC je napájecí napětí, které napájí čipy a je obvykle 3 až 5 voltů s maximálně 10% odchylkou
C2: Je buď vyhrazen pro budoucí použití, nebo je požit jako **Reset**, který slouží k odeslání signálu (RST) do integrovaného obvodu, aby jej resetoval. Jedná se o tzv. měkký (*Warm Reset*)
C3: CLK je hodinový signál, který slouží k řízení logiky vestavěných komponent a je také používán jako synchronizace referenční sériové komunikace. Tento pin je používán proto, že mikročip v sobě nemá generátor hodin a potřebuje ho mít jako externí vstup. Čtečka karet poskytuje tento hodinový signál o frekvenci od 5 do 40 MHz
C4: Je vyhrazen pro budoucí použití, u některých současných karet se používá pro USB rozhraní
C5: GRD se používá jako uzemnění
C6: Je nyní vyhrazen pro budoucí použití, ale dříve sloužil pod označením **VPP**, kde byl používán pro vedení vyššího napětí, které bylo nutné k naprogramování EPROM paměti
C7: I/O je pin, který slouží jako sériový vstup/výstup (SIO). Pomocí tohoto pinu dochází k výměně dat mezi čtečkou a kartou (čipem)
C8: Je vyhrazen pro budoucí použití, u některých současných karet se používá pro USB rozhraní

2.2.3 Hybridní karty

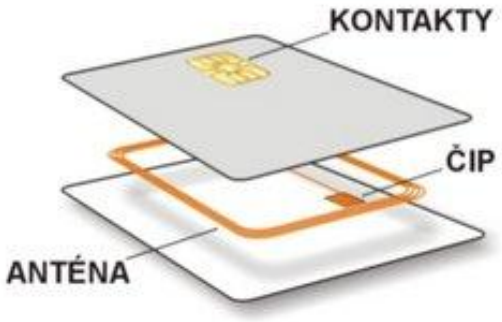
Hybridní karty je termín používající se pro karty, které obsahují dvě nebo více čipových technologií zároveň, jako např. bezkontaktní čip s anténou a kontaktní čip s kontaktním polem (to znamená vše na jedné kartě). Bezkontaktní čip je typicky použit pro aplikace vyžadující rychlé přenosy a kontaktní čip se pak používá v aplikacích vyžadující vysoké zabezpečení. Čipy pro kontaktní a bezkontaktní přenos dat nejsou vzájemně propojeny a díky tomu může jedna karta sloužit pro více aplikací zároveň.



Obr 4 – Hybridní čipová karta (zdroj: [5])

2.2.4 Kombinované (duální) karty

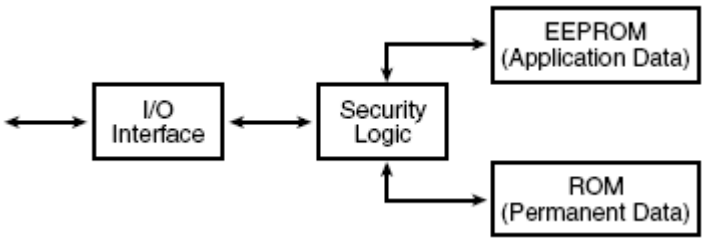
Kombinované karty jsou v podstatě hybridní karty, které však mají pouze jeden čip, ke kterému lze přistoupit buď pomocí kontaktního pole, nebo bezdrátově pomocí zalité antény. Tento druh čipové karty stejně jako předchozí je populární díky snadnému použití a vysokému stupni zabezpečení. Například hromadná doprava je jedním z odvětví, které nejčastěji využívá právě kombinovaných karet. V takovém případě se používá kontaktní pole čipu pro přenos finanční částky do paměti a bezdrátové rozhraní pak pro odečtení jízdného.



Obr 5 – Kombinovaná čipová karta (zdroj: [5])

2.2.5 Paměťové karty

Jedná se o nejběžnější a nejlevnější čipové karty, jenž jsou složeny ze čtyř základních komponent, které jsou zobrazené na následujícím obrázku.

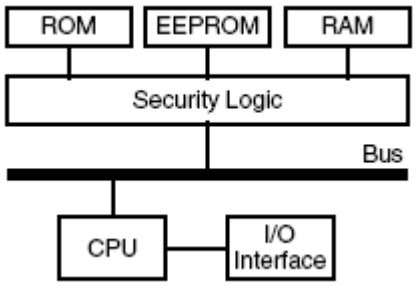


Obr 6 – Základní komponenty pamětné karty (zdroj: [7])

- *I/O interface* – je základní rozhraní sloužící k výměně vstupně/výstupních dat, mezi čtečkou karet a vnitřní bezpečnostní logikou
- *Security Logic* – je bezpečnostní logika kontrolující přístup do vnitřní paměti EEPROM, která umožňuje čtení a zápis dat. Oblast paměti je přístupná pouze po zadání tajného kódu PIN. Tento kód může být poskytován buď prostřednictvím jiné čipové karty anebo je požadováno zadání od držitele karty
- *EEPROM* (*Electrically Erasable Programmable Read-Only Memory*): – je paměť, ve které jsou uložena data, se kterými je dále možné pracovat. Typická velikost EEPROM se pohybuje od 2Kb do 8KB. Data v této paměti mohou být uzamčena pomocí již výše zmiňovaného PIN kódu a tato data je možné měnit. Použití je například u telefonních karet, kde jsou v této paměti uchovávány počty volných jednotek
- *ROM* (Read Only Memory): – je paměť, sloužící např. k uchování čísla karty, nebo jména držitele karty a takto jednou zapsaná data již nelze změnit

2.2.6 Karty s mikroprocesorem

Jak již název napovídá, jedná se o karty, které obsahují mikroprocesor, a proto se jim také říká čipové karty, nebo někdy jsou označovány jako „chytré“ (*Smart Card*). Podobně jako paměťové karty, jsou karty s mikroprocesorem složeny z několika komponent, které jsou přehledně znázorněny v následujícím obrázku.



Obr 7 – Základní komponenty mikroprocesorové karty (zdroj: [7])

- *ROM* – je paměť, ve které je uložen operační systém karty. Operační systém je do této paměti zapsán pouze jednou (obvykle během fáze výroby karty). Velikost paměti ROM se pohybuje od jednotek KB do 32 KB, podle toho, jaký operační systém je na kartě použit
- *EEPROM* – u mikroprocesorových karet jsou v EEPROM paměti uloženy aplikační programy a aplikační data. Tato data nejsou trvalá a často jsou mazána a přepisována. Typická velikost EEPROM paměti se pohybuje od 2 do 32 KB
- *RAM* (*Random Access Memory*) – jedná se o paměť, kterou ke své funkci využívá mikroprocesor a slouží ke spuštění požadovaných funkcí a programů. Tato paměť je vymazána vždy, když je karta odpojena od napájecího napětí. Typická velikost paměti RAM se pohybuje pouze kolem 256 b (=32 B). Tato velikost je dána tím, že paměť RAM zabírá velkou plochu na byte a oblast čipové karty pro umístění čipu je omezena jen na 25 mm^2
- *CPU* (*Central Processing Unit*) – je srdcem mikroprocesorové karty, obvykle je to 8bitový mikroprocesor založený na architektuře CISC s typickou taktovací frekvencí 5 MHz. S nástupem *Java* karet se vývoj CPU pomalu přesouvá směrem k 32-bitové architektuře. Procesor je odpovědný za provádění jednotlivých instrukcí

Mikroprocesorové karty jsou zpravidla dražší než paměťové karty a jejich cena se pohybuje v závislosti na dostupných funkcích. Tyto karty jsou schopny obsluhovat více aplikací a poskytují robustní zabezpečení. Použití těchto karet je zejména v elektronickém bankovnictví, dále pak při řízení přístupů, v cestování a v řadě dalších aplikacích, kde je požadována vysoká bezpečnost.

2.3 Operační systém

Operační systém čipových karet, nebo někdy také nazývaný jako *COS – Card operating system*, je uložen v paměti ROM a obvykle zabírá méně než 16 KB. Mezi základní funkce operačního systému patří:

- Řízení a výměna dat mezi kartou a čtečkou karet a to především v oblasti protokolů
- Správa souborů a dat uchovávaných v paměti čipové karty

- Řízení přístupu k informacím a funkcím (např. výběr souborů, čtení, zápis a aktualizace dat)
- Řízení bezpečnosti a spolehlivosti karty, šifrovací algoritmus a zejména zachování integrity dat
- Řízení různých fází životního cyklu karet

2.3.1 Souborový systém

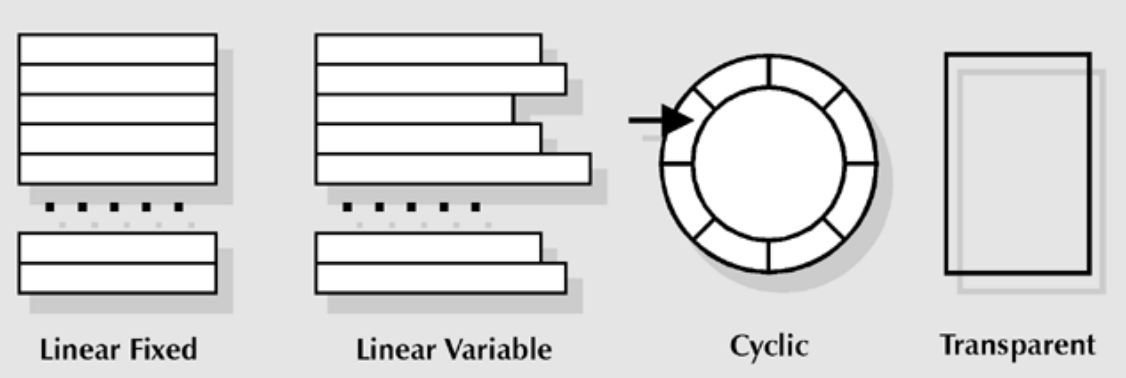
Systém souborů, které jsou použity pro čipové karty, mají hierarchický stromový souborový systém podobný jako je třeba na PC. Každý soubor je zde jednoznačně určen pomocí *ID*, které je většinou tvořeno 2 B. Souborový systém podporuje tři základní typy souborů:

1. **Master file (MF):** tento soubor je jediný svého druhu a představuje kořenový adresář souborů a dále může obsahovat soubory následujících typů. Jeho *ID* je 0x3F00
2. **Dedicated file (DF):** je adresář a může obsahovat jak další adresáře, tak i elementární soubory, přístup k adresářům nemusí být pouze pomocí *ID*, ale muže být také pomocí názvu souboru
3. **Elementary file (EF):** jedná se o elementární soubor, který může obsahovat data, ale i soubor s řídicími informacemi. Základní typy elementárních souborů jsou tyto:
    a. *Working file* – pracovní soubor slouží pro ukládání dat aplikace
    b. *Public file* – veřejný soubor umožňuje přístup k datům bez omezení
    c. *Application control file* – soubor řízení aplikace umožňuje přístup pouze pro čtení
    d. *Internal secret file* – vnitřní tajné soubory obsahují data, která nejsou přístupná mimo integrovaný obvod

2.3.2 Datová struktura elementárních souborů

Datová struktura definuje, jakým způsobem jsou data v souboru uložena. Elementární soubor umožňuje uložení dat pomocí čtyř struktur:

1. *Linear fixed* – je struktura obsahující záznamy fixní velikosti, ke kterým se přistupuje pomocí čísla záznamu
2. *Linear variable* – tento typ struktury obsahuje záznamy proměnlivé (variabilní) délky. Přístup k záznamům je stejný jako u *Linear fixed*, tedy pomocí čísla záznamu
3. *Cyclic* – záznamy mají fixní velikost a datová struktura tvoří obousměrně zřetězený seznam (kruh), ve kterém má každý prvek ukazatel na svého předchůdce a následníka. V hlavičce souboru s touto strukturou je uložen ukazatel na začátek kruhového seznamu
4. *Transparent* – tento typ nemá pevně danou strukturu a ukládání dat se provádí pomocí jednotlivých bajtů. Přístup k datům se provádí pomocí relativní adresy, která poukazuje na začátek bloku dat



Obr 8 – Typy datových struktur elementárních souborů (zdroj: [8])

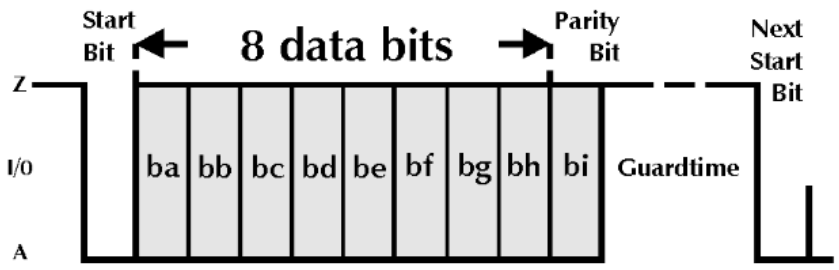
2.4 Komunikační protokoly

Norma ISO 7816-3 definuje dva protokoly: protokol T = 0 a T = 1. Protokol T = 0 je asynchronní *znakově orientovaný protokol*, u kterého musí být obdrženo potvrzení pro každý odeslaný bajt. Naopak T = 1 je asynchronní *blokově orientovaný protokol*, kde potvrzování dat probíhá trochu jiným způsobem. Oba tyto protokoly se využívají pro komunikaci mezi čtečkou a čipovou kartou. Komunikace je poloduplexní, což znamená, že komunikace v určitém časovém okamžiku může probíhat pouze v jednom směru (buď od čtečky ke kartě, nebo od karty k čtečce).

2.4.1 Protokol T = 0

Protokol T = 0 byl navržen k optimalizaci účinnosti komunikace mezi kartou a čtečkou. Právě z důvodů optimalizace bylo zpracování možných chyb a uplatnění tohoto protokolu navrženo tak, aby se minimalizovalo množství přenášených dat mezi čtečkou a kartou a tím došlo ke zkrácení doby transakcí.

Jak již bylo zmíněno, jedná se o *znakově orientovaný protokol* (někdy též označovaný jako bajtově orientovaný protokol), kde znak je tvořen přenášeným bajtem a dále pak startovním a paritním bitem. Pomocí paritního bitu lze zjistit chybu při přenosu jednotlivých bajtu. U toho protokolu se provádí kontrola tzv. liché parity dat, to znamená, že paritní bit se doplní podle počtu jedniček obsažených v přenášeném bajtu tak, aby součet všech jedniček tvořil liché číslo. V případě, že by po doručení znaku nebyl součet jedniček liché číslo, došlo by k detekci chyby a následně by došlo k odeslání tohoto znaku znovu. Na obrázku 9 je znázorněn jeden znak včetně startovacího i paritního bitu. Startovací bit se používá pro synchronizaci rámce znaku. Na tomto obrázku je dále *Guardtime*, který odděluje jednotlivé znaky od sebe. Doba vysílání jednoho znaku musí být nejméně 12 etu (*elementary time unit* = základní časová jednotka, která odpovídá hodnotě 372/frekvence hodinového signálu).



Obr 9 – Grafické znázornění znaku (zdroj: [8])

Datové struktury používané k výměně dat mezi čtečkou a čipovou kartou se nazývají *TPDU* (*transmission protocol data units*). Struktury TPDU v protokolu T = 0 se liší od těch, které jsou použity v protokolu T = 1. TPDU pro protokol T = 0 se skládají ze dvou odlišných struktur:

- **Příkaz** (*Command*), který je odeslán z čtecího zařízení do karty
- **Odpověď** (*Response*), která je odeslána z karty do čtečky

Příkazová zpráva se skládá z hlavičky a dat, které jsou poté odeslány do *ICC* (*integrated circuit card* = karta s integrovaným obvodem – čipová karta). Hlavička příkazové zprávy se skládá z pěti částí a každá část je tvořena jedním bajtem zapsaným v hexadecimálním tvaru. Jednotlivé bajty hlavičky jsou následující:

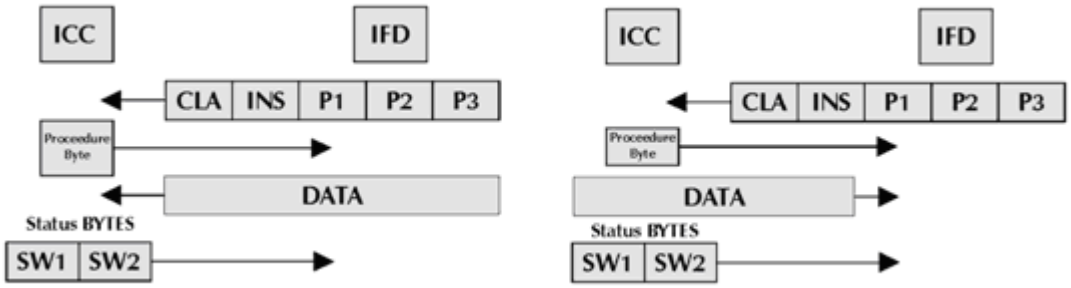
1. **CLA:** instrukční třída. Hodnota CLA definuje sadu aplikačně-specifické instrukce
2. **INS:** instrukční kód. Určuje konkrétní pokyn v rámci sady instrukcí
3. **P1:** parametr 1 – rozšiřující CLA a INS
4. **P2:** parametr 2 – rozšiřující CLA a INS
5. **P3:** parametr 3 – určuje délku bloku dat

Parametry P1 a P2 jsou závislé na konkrétní instrukci (aplikační úrovně). Poskytují ovládací prvek nebo adresování parametrů různých aplikačně specifických instrukcí (např. v instrukci pro výběr souboru *Select file*) se P1 používá k označení, jak se soubor bude vybírat (pomocí ID, názvu nebo cesty) a P2 nabízí další upřesnění. Parametr P3 určuje počet bajtů, které mají být předávány během provádění INS instrukce. Pokud je P3 nastaven na 0 a je požadováno čtení dat z karty, přenese se 255 B. Pokud by však mělo dojít k zápisu a P3 bude nastaven na 0, nedojde k zápisu žádného B dat.

Pro každý příkaz TDPU odeslaného z čtecího zařízení bude odeslána odpověď TPDU kartou. Odpověď obsahuje tři povinné části a jednu nepovinnou část (všechny části jsou jako u hlavičky příkazu jednobajtové):

1. **ACK:** označuje obdržení příkazu kartou
2. **NULL:** tento bajt je používaný pro řízení toku přes kontakt C7 – I/O. Je to signál, který posílá karta čtecímu zařízení, aby jej informoval o tom, že stále ještě probíhá zpracování požadovaného příkazu. Čtecí zařízení tedy musí čekat na dokončení probíhajícího příkazu než bude moci znovu vyslat další příkaz
3. **SW1:** stav reakce na aktuální příkaz
4. **SW2:** (nepovinné) také vyjadřuje stav reakce na aktuální příkaz

Bajt ACK je reakcí na INS bajt z příkazu TPDU. Jestliže se odpověď ACK neodešle do čtecího zařízení v určité době, čtecí zařízení může zahájit RST sekvence k restartování protokolu mezi čtečkou a kartou. Tomuto lze zabránit tím, že čtečka obdrží alespoň jeden bajt NULL z karty. Bajty SW1 a SW2 informují čtečku o výsledku požadovaného příkazu. Standardními návratovými hodnotami, které informují o úspěšném vykonání příkazu, jsou pro SW1 = 0x90 a SW2 = 0x00. Na následujících dvou obrázcích jsou graficky znázorněny příkaz a odpověď pro odesílání a příjem dat.



Obr 10 – Odeslání dat na kartu a vrácení dat čtečce (zdroj: [8])

2.4.2 Protokol T = 1

Protokol T = 1 je blokově orientovaný protokol, ve kterém je přesně definovaná kolekce dat (blok) přesouvána jako celek mezi čtecím zařízením a kartou. Data se přenáší pomocí rámce (*frame*). Největší výhodou T = 1 protokolu je schopnost řídit tok dat v obou směrech. U předchozího protokolu bylo řízení pouze na straně ICC. Další výhodou tohoto protokolu je schopnost zřetězovat bloky dat tak, aby libovolně velký blok mohl být proveden jako výsledek jednoho příkazu přenesením příslušného počtu zřetězených rámců v sekvenci. Blokový protokol má také více propracovaný systém řízení chyb. Detekce chyb se provádí buď pomocí *longitudinal redundancy check (LRC)* nebo *cyclic redundancy check (CRC)*. LRC je složitější forma kontrola parity, která se používá v protokolu T = 0 a dokáže zaručeně detekovat chyby jednotlivých bajtů přenášených v bloku. Protokol T = 1 používá tři typy bloků:

- *Information block:* používá se pro výměnu dat mezi aplikačním softwarem na kartě a aplikačním softwarem na straně čtečky
- *Receive ready block:* používá se pro kladné a záporné potvrzení. Kladné potvrzení označuje, že byl správně přijat blok. Negativní potvrzení označuje, že byla zjištěna chyba v přijatém bloku
- *Supervisory block:* používá se k přenosu řídicích informací mezi čtečkou a kartou

Rámec znázorněn na následujícím obrázku se skládá celkem ze tří částí:

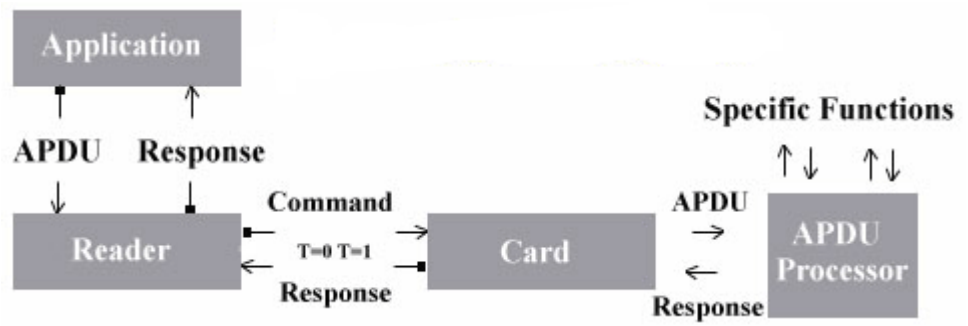
1. *Prologue field:* se skládá celkem ze tří bajtů
    a. **NAD** – adresa uzlu, která se používá k identifikaci adresy zdroje a zamýšlené místo určení bloku
    b. **PCB** – slouží k označení druhu bloku (*Information block, Receive ready block, Supervisory block*)
    c. **LEN** – délka bloku
2. *Information field* (nepovinný) – pole dat, které může být až 254 bajtů dlouhé a může obsahovat APDU viz kapitola 2.4.3.
3. *Epilogue field* – je dlouhé 1 nebo 2 bajty a slouží pro detekci chyb. Jestliže je dlouhé 1 bajt jedná se o *LRC* kontrolu a jestliže 2 bajty potom se jedná o metodu *CRC*

Prologue Field			Information Field	Epilogue Field
Node Address	Protocol Control Byte	Length	Optional	Error Detection LRC or CRC
NAD	PCB	LEN	INF	EDC
1 Byte	1 Byte	1 Byte	0-254 Bytes	1/2 Bytes

Obr 11 – Složení rámce protokolu T = 1 (zdroj: [8])

2.4.3 Struktura zpráv APDU

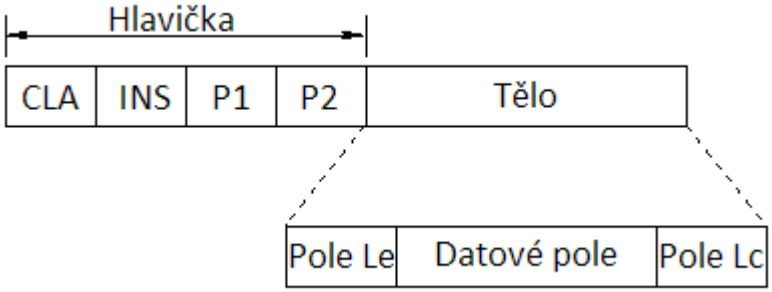
Protokoly T = 0 a T = 1 slouží k podpoře protokolů aplikační vrstvy mezi aplikacemi čipové karty a aplikacemi čtečky. Tyto protokoly aplikací sloužící k výměně datových struktur se nazývají APDU (*Application Processing Data Unit*). Graficky je komunikace a zasílání zpráv mezi aplikacemi znázorněna na následujícím obrázku.



Obr 12 – Komunikační schéma (zdroj: [9])

Struktura APDU podle normy ISO 7816-4 je velmi podobná struktuře TPDU používané v protokolu T = 0. Ve skutečnosti, když je APDU přepravováno protokolem T = 0, prvky APDU přímo překryjí prvky TPDU. Existují dva typy zpráv, které jsou používány:

- *Příkaz APDU* – odesílaný z čtecího zařízení na kartu
- *Odpověď APDU* – odesílaná z karty do čtecího zařízení

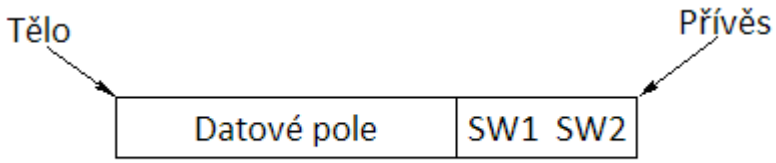


Obr 13 – Struktura APDU příkazu (zdroj: [9])

Příkaz APDU se skládá z hlavičky a těla. Hlavička obsahuje pole CLA, INS, P1 a P2. Stejně jako v protokolu T = 0 CLA a INS určuje třídu aplikace a instrukce. P1 a P2 slouží k zapsání zvláštních pokynu a specifických definic, které jsou dány každé CLA a INS instrukci.

Velikost těla APDU zprávy může být různá a je používáno pro přenos dat do APDU procesoru karty, které jsou součástí příkazu, nebo obsahují sdělení odezvy z karty do čtecího zařízení. Pole LC určuje počet bajtů, které jsou předávány na kartu jako část instrukcí tj. délka datového pole. Datové pole obsahuje informace, které musí být odeslány APDU procesoru karty, pro vykonání příkazu obsaženému v APDU. Pole Le určuje počet bajtů, které budou vráceny kartou do čtecího zařízení v APDU odpovědi. Tělo APDU může mít čtyři různé formy:

- Případ 1: Žádná data nejsou převedena do nebo z karty, takže APDU obsahuje pouze hlavičku
- Případ 2: Žádná data nejsou převedena kartě, ale je požadováno vrácení dat z karty. Těla APDU obsahuje pouze neprázdné pole Le
- Případ 3: Data jsou předávána kartě, ale žádné se nevracejí zpět. Tělo APDU obsahuje LC a datové pole
- Případ 4: Data jsou posílána na kartu a zároveň jsou také požadována z karty. Tělo APDU obsahuje LC, datové i Le pole



Obr 14 – Struktura APDU odpovědi (zdroj: [9])

Odpověď APDU má mnohem jednodušší strukturu než je u příkazu. Odpověď se skládá z těla a přívěsu. Tělo je buď prázdné, nebo jej zahrnuje datové pole – v závislosti na konkrétním příkazu. Délka datového pole je určena hodnotou v poli Le odpovídajícího (předcházejícího) příkazu APDU. Přívěs je tvořen dvěmi poli SW1 a SW2, které informují o stavu odpovědi. Tato pole vrací stavový kód, ve kterém se používá jeden bajt k určení kategorie chyby a druhý se používá k zadání příkazu specifického stavu nebo označuje údaj o chybě.

3 Digitální tachograf

Na základě „Nařízení (EHS) 3821/85“ musí mít všechna vozidla, uvedená do provozu od 1. května 2006 s celkovou povolenou hmotností nad 3,5 tuny resp. s více než devíti sedadly, nainstalován digitální tachograf a musí jej používat. Každý řidič, který řídí vozidlo vybavené takovýmto přístrojem, musí používat kartu řidiče. Všechna data karty řidiče tachografu musí být v pravidelných intervalech stažena, vyhodnocena a poté archivována.

Digitální tachograf umožňuje zaznamenání a uložení informací týkajících se činnosti řidiče a některých dat o vozidle, jako je například rychlost vozidla, délka trasy. Data jsou zaznamenána v paměti tachografu po dobu nejméně jednoho roku. Při ukládání těchto dat do paměti přístroje se zároveň některá data ukládají i na vloženou kartu řidiče.

Obr 15 – Digitální tachograf (zdroj: [10])

3.1 Funkce digitálního tachografu

Záznamové zařízení musí zajistit následující funkce [13]:

- monitorování vkládání karty do čtecího zařízení karet a její vyjímání
- měření rychlosti a ujeté vzdálenosti
  - rychlost se měří v rozsahu 0–220 km/h
  - vzdálenost se měří s rozlišením 0,1 km nebo jemnějším
- měření času – datum a čas je zaznamenáván ve formě UTC (*Coordinated Universal Time* = koordinovaný světový čas)
- monitorování činnosti řidiče:
  - jízda – označovaná symbolem a je nastavována řidiči, jestliže se vozidlo pohybuje
  - práce – označovaná symbolem a nastavuje se, jestliže vozidlo zastaví na déle než 1 minutu
  - pohotovost – označovaná symbolem a je nastavována druhému řidiči, když se vozidlo pohybuje
  - přestávka/odpočinek – označovaná symbolem
- monitorování provozního stavu – jsou-li v záznamovém zařízení vloženy dvě platné karty řidiče, nastavuje se režim „posádka“. V každém jiném případě se navolí stav řízení vozidla „samotný řidič“
- údaje vkládané řidičem ručně:
  - vložení místa kde pracovního doba dne začíná nebo končí

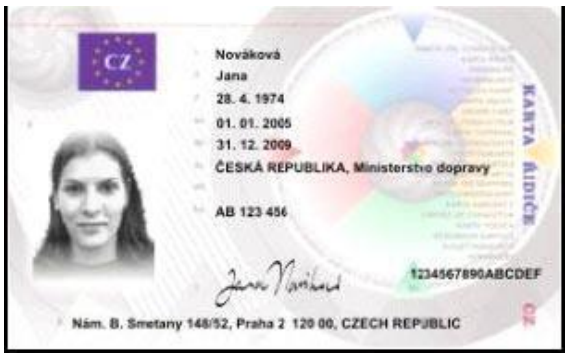
    - ručně vkládané údaje o činnostech řidiče
    - záznam zvláštních podmínek
- využívání možnosti zámků podniků – umožňuje správu zámků podnikem k omezení přístupu k údajům v podnikovém režimu a to pouze pro tento podnik
- monitorování kontrolních činností – musí monitorovat činnosti zobrazování, tisk a stahování dat z celku ve vozidle nebo karty, které jsou prováděny v kontrolním režimu
- zjišťování událostí a závad
    - událost – mimořádná činnost zjištěná záznamovým zařízením, která může pocházet z pokusu o podvod
    - závada – mimořádná činnost zjištěná záznamovým zařízením, která může pocházet z chybné funkce nebo z poruchy zařízení
- vestavěné zkoušky a autotesty – záznamové zařízení musí samo zjistit vlastní závady v průběhu vestavěných zkoušek a autotestů
- načítání z paměti údajů – záznamové zařízení musí být schopno načíst jakékoliv údaje uložené v jeho paměti údajů
- zaznamenávání a ukládání do paměti údajů – údaje uložené do paměti údajů nesmí být ovlivněny přerušením elektrického napájení z vnějšího zdroje a musí být uloženy po dobu nejméně 12 měsíců
- načítání z karet tachografu
- zaznamenávání a ukládání dat na karty tachografu
- zobrazování údajů – záznamové zařízení musí být schopno zobrazovat údaje z vlastní paměti údajů nebo karet tachografu na displej, který musí mít minimálně 20 znaků
- tisk – záznamové zařízení musí být schopno vytisknout údaje z vlastní paměti údajů nebo karet tachografu v podobě následujících šesti výtisků
    - denní výtisk činnosti řidiče z karty
    - denní výtisk činnosti řidiče z celku ve vozidle
    - výtisk událostí a závad z karty
    - výtisk událostí a závad z celku ve vozidle
    - výtisk technických údajů
    - výtisk překročení povolené rychlosti
- dávání výstrahy – záznamové zařízení musí dát výstražné znamení řidiči při zjištění jakékoliv události nebo závady
- stahování dat na externí média – v případě potřeby musí zařízení umožnit stáhnout údaje z paměti údajů nebo z karty řidiče na externí médium pro uložení dat prostřednictvím kalibračního nebo stahovacího konektoru

3.2 Karty digitálního tachografu

Digitální tachograf umožňuje práci se čtyřmi druhy čipových karet. Tyto karty jsou kontaktní procesorové čipové karty podporující oba protokoly, které byly popsány v kapitole 2.4. Jednotlivé karty se od sebe liší způsobem jejich využití a dále pak držitelem, kterému byly vydány. Popis karet je převzat z [11].

3.2.1 Karta řidiče

Karta řidiče je vydána řidičům, kteří jsou držitelem příslušného řidičského oprávnění pro řízení takového vozidla, které bude předmětem používání. Karta obsahuje informace týkající se identifikace karty, identifikace držitele karty, informace o řidičském průkazu držitele karty, údaje o použitých vozidlech, údaje o řidičových činnostech, místa, kde časy výkonu denní práce začínají nebo končí, údaje o událostech, údaje o závadách, údaje o kontrolních činnostech, údaje o použití karty, údaje o specifických podmínkách. Karta musí být během jízdy zasunuta v přístroji, na kterou se zaznamenává průběh jízdy. Tyto záznamy slouží následně kontrolním orgánům ke kontrole dodržování bezpečnostních přestávek řidičů i ke kontrole způsobu jízdy (dodržování rychlosti). Karta řidiče se vydává k platnému řidičskému průkazu a číslo řidičského průkazu je uvedeno přímo na kartě řidiče. Proto pokud dojde k výměně řidičského průkazu, musí dojít i k výměně karty řidiče. Karta řidiče se vydává na dobu nejdéle pěti let.



Obr 16 – Karta řidiče (zdroj: [12])

3.2.2 Karta podniku

Podniková karta je vydána majiteli nebo provozovateli vozidla, který si zároveň musí opatřit technické a programové vybavení pro stahování dat. Karta obsahuje identifikační údaje firmy a umožňuje odemčení dat z vozidlového přístroje pro stahování, prohlížení na displeji i vytištění. Pomocí podnikové karty lze data také uzamknout proti neoprávněnému stahování. Při pořízení vozidla je třeba neodkladně podnikovou kartu použít, neboť data je možno stahovat teprve od okamžiku, kdy se provozovatel vozidla touto kartou identifikoval v paměti přístroje! Při prodeji vozidla jsou data dosavadního majitele uzamčena a nový majitel se identifikuje svojí kartou. Provozovatel vozidla může vlastnit libovolný počet karet a všechny karty lze používat do všech vozidel provozovatele. Podnikové karty jsou přenosné a způsob jejich používání je v kompetenci provozovatele vozidel.

Obr 17 – Karta podniku (zdroj: [12])

3.2.3 Karta dílny

Dílenská karta je vydávána výrobcům tachografů, zkušebnám, opravnám a kalibračním střediskům, schváleným členským státem EU. Karta je vydána pro konkrétní autorizovanou osobu, jejíž identifikační data jsou na kartě uložena. Každá karta dílny má přidělen jedinečný PIN, který musí být po celou dobu zabezpečen proti zneužití. Doba platnosti je jeden rok. Karta dílny umožňuje zkoušení, kalibraci a stahování dat z vozidlového přístroje, přitom ale musí být používána tak, aby data nebyla poškozena nebo zničena. Karta nesmí být odnášena z prostoru schváleného pracoviště.

Obr 18 – Karta dílny (zdroj: [12])

3.2.4 Kontrolní karta

Tato karta je vydávána pracovníkům Policie ČR, Celní správy ČR a dalším orgánům, pověřeným kontrolou pracovních režimů řidičů. Jsou na ní uložena identifikační data osoby, která je oprávněna provádět kontrolu a umožňuje přístup k údajům uloženým v paměti vozidlového přístroje nebo na kartě řidiče. Pomocí karty kontrolního orgánu lze data prohlížet na displeji, načítat a tisknout. Na kartě jsou ukládána data o kontrolních úkonech, tj. datum a čas prováděné kontroly a způsob výstupu dat (displej, stažení, tisk).

Obr 19 – Kontrolní karta (zdroj: [12])

4 Analýza

V této kapitole budou nejprve zmíněny jednotlivé požadavky na výsledný program a dále pak bude podrobněji popsána struktura dat na kartě řidiče a přístup k této struktuře.

4.1 Požadavky na výsledný program

Výsledkem této práce by měla být samostatně běžící desktopová aplikace, na kterou jsou kladeny následující požadavky a měla by obsahovat tyto funkce:

- Program bude schopen načíst a dekódovat data uložená na kartě řidiče
- Načtená a dekódovaná data bude možné zobrazit v grafickém i tabelárním režimu
- Při připojení více čtecích zařízení bude možný výběr vždy jednoho aktivního
- Po stažení dat bude možné tato data archivovat na paměťové médium v počítači
- Archivovaná data v počítači bude možné později zobrazovat a analyzovat
- Načítaná data z karty bude možné sesynchronizovat s již dříve uloženými daty v paměti počítače
- Načtená data bude možné ručně doplňovat o vysvětlující záznamy (zůstávají pouze v paměti počítače)
- Nad načtenými daty bude možné provést kontrolu spojitosti těchto dat
- Program bude schopen provést kontrolu dat podle nařízení č. 561/2006 [14]. Přitom bude sledováno zejména dodržování následujících požadavků na řidiče:
    - Denní doba řízení nesmí přesáhnout 9 hodin. Nejvýše dvakrát za týden může být prodloužena na 10 hodin.
    - Týdenní doba řízení nesmí přesáhnout 56 hodin a nesmí být překročena maximální týdenní pracovní doba stanovená ve směrnici 2002/15/ES.
    - Celková doba řízení nesmí přesáhnout 90 hodin za období dvou po sobě následujících týdnů.
    - Po čtyřech a půl hodinách řízení musí mít řidič nepřerušenou přestávku nejméně 45 minut, pokud mu nezačíná doba odpočinku. Tato přestávka může být nahrazena přestávkou v délce nejméně 15 minut, po níž následuje přestávka v délce nejméně 30 minut.
    - V průběhu každých 24 hodin po skončení předchozí denní nebo týdenní doby odpočinku musí mít řidič novou denní dobu odpočinku. Je-li denní doba odpočinku v průběhu těchto 24 hodin alespoň 9 hodin, ale kratší než 11 hodin, považuje se dotyčná denní doba odpočinku za zkrácenou.
    - Mezi dvěma týdenními dobami odpočinku smí mít řidič nanejvýš tři zkrácené denní doby odpočinku.

4.2 Struktura a přístup k datům na kartě řidiče

V této kapitole bude popsaná struktura dat uložených na kartě řidiče a jednotlivé příkazy, které jsou potřeba k jejich získání resp. k načtení a následnému dalšímu použití.

K získání přístupu k jednotlivým souborům se používají APDU zprávy, které obsahují příkaz k výběru souboru nebo-li *Select file*. Pokud je k souboru takto získán přístup mohou se

data načíst pomocí dalších ADPU zpráv, které tentokrát obsahují příkazy k načítání binárních dat (*Read Binary*). Na následujícím obrázku je znázorněna struktura souborů, která je uložena na kartě řidiče.

		Podmínky přístupu		
Soubor	ID souboru	Čtení	Aktualizace	Kódování
MF	3F00			
├─EF ICC	0002	ALW	NEV	No
├─EF IC	0005	ALW	NEV	No
└─DF Tachograph	0500			
├─EF Application_Identification	0501	ALW	NEV	No
├─EF Card_Certificate	C100	ALW	NEV	No
├─EF CA_Certificate	C108	ALW	NEV	No
├─EF Identification	0520	ALW	NEV	No
├─EF Card_Download	050E	ALW	ALW	No
├─EF Driving_Licence_Info	0521	ALW	NEV	No
├─EF Events_Data	0502	ALW	PRO SM / AUT	No
├─EF Faults_Data	0503	ALW	PRO SM / AUT	No
├─EF Driver_Activity_Data	0504	ALW	PRO SM / AUT	No
├─EF Vehicles_Used	0505	ALW	PRO SM / AUT	No
├─EF Places	0506	ALW	PRO SM / AUT	No
├─EF Current_Usage	0507	ALW	PRO SM / AUT	No
├─EF Control_Activity_Data	0508	ALW	PRO SM / AUT	No
└─EF Specific_Conditions	0522	ALW	PRO SM / AUT	No

Obr 20 – Struktura souboru na kartě řidiče (zdroj: [13])

Na této struktuře jsou nejdůležitější názvy jednotlivých souborů a dále pak jejich *ID*. Na obrázku je dále zobrazen přístup k jednotlivým souborům při čtení a aktualizaci. Zatímco čtení všech souborů je povoleno vždy (*ALW* z anglického slova *always*), tak aktualizace se u některých souborů nemůže provádět nikdy (*NEV = never*) anebo je možná, ale pouze pomocí bezpečného zpracování zpráv (*PRO SM = protected with secure messaging*). Dále je zde uvedeno, zda je některý soubor zakódován či nikoliv. U karty řidiče není žádný soubor kódován.

4.2.1 Select file

Příkaz *Select file* se používá k výběru jednotlivých elementárních souborů *EF*, tak i k výběru adresářů DF. Jakmile byl však vybrán adresář, není již možnost vrátit se zpět do kořenového adresáře MF pomocí příkazu *Select file*. Návrat do kořenového adresáře je možný pouze zasláním signálu RST (signál pro resetování karty). Kvůli tomuto omezení se zpravidla postupuje tak, že se nejprve vybírají elementární soubory v rámci kořenového adresáře a poté teprve v rámci složky DF.

4.2.1.1 Výběr souboru DF

Na kartě řidiče se nachází pouze jeden soubor typu DF a to *DF Tachograph*. Přístup k tomuto typu souboru nelze provést pomocí jeho *ID*, ale provádí se pomocí jeho názvu. Příkazová zpráva má následující strukturu:

Bajt	Délka	Hodnota	Popis
CLA	1	′00h′	
INS	1	′A4h′	
P1	1	′04h′	výběr podle názvu (AID)
P2	1	′0Ch′	neočekává se žádná odezva
Lc	1	′NNh′	počet bajtů odeslaných na kartu (délka AID): ′06h′ pro aplikaci tachografu
#6-#(5+NN)	NN	′XX..XXh′	AID: ′FF 54 41 43 48 4F′ pro aplikaci tachografu

Obr 21 – Struktura příkazu pro výběr podle názvu (zdroj: [13])

4.2.1.2 Výběr souboru EF

Na kartě řidiče se nachází celkem šestnáct elementárních souborů a každý soubor je identifikován jednoznačným *ID*. Jelikož se k výběru souboru používá *ID*, je struktura zprávy odlišná než u předchozího případu. Výběr EF se pokaždé vztahuje k aktuálně vybranému DF, případně pokud nebyl dosud žádný vybrán tak k MF. Pokud tedy budeme požadovat například výběr některého elementárního souboru v rámci *DF Tachograph*, musí být tento nejprve zpřístupněn zasláním předchozího typu zprávy a teprve poté je možné zaslat zprávu tohoto typu k výběru elementárního souboru. Struktura příkazu pro výběr EF je následující:

Bajt	Délka	Hodnota	Popis
CLA	1	′00h′	
INS	1	′A4h′	
P1	1	′02h′	výběr EF při aktuálním DF
P2	1	′0Ch′	neočekává se žádná odezva
Lc	1	′02h′	počet bajtů odeslaných na kartu
#6-#7	2	′XXXXh′	identifikátor souboru

Obr 22 – Struktura příkazu pro výběr podle ID (zdroj: [13])

U obou výše popsaných příkazů k výběru souboru není požadována žádná odezva a v odpovědi dojde k vrácení pouze dvou bajtů SW. Pokud je odeslaný příkaz úspěšný vrací karta zpět 9000. Pokud soubor odpovídající identifikátoru případně názvu souborů není nalezen, je zpět poslaný stav zpracování 6A82. Jestliže je vybraný soubor považován za poškozený (je detekována chyba integrity uvnitř souboru atributů), je zpět poslaný stav zpracování 6400 nebo 6581. Struktura odpovědi má následující tvar:

bajt	délka	hodnota	popis
SW	2	′XXXXh′	stavová slova (SW1, SW2)

Obr 23 – Struktura odpovědi pro výběr souboru (zdroj: [13])

4.2.2 Read Binary

Jak již bylo zmiňováno tento příkaz slouží ke čtení dat z transparentních souborů viz kapitola 2.3.2. Všechny soubory na kartě řidiče jsou transparentní. Příkaz *Read Binary* může být dvojího typu:

1. Příkaz bez bezpečného zpracování zpráv*
2. Příkaz s bezpečným zpracováním zpráv

V tomto textu bude dále rozvedena pouze první možnost tj. příkaz bez bezpečného zpracování zpráv, protože tento typ bude následně využíván v realizovaném programu.

Bajt	Délka	Hodnota	Popis
CLA	1	'00h'	nepožaduje se bezpečné zpracování zpráv
INS	1	'B0h'	
P1	1	'XXh'	offset v bajtech od začátku souboru: nejvýznamnější bajt
P2	1	'XXh'	offset v bajtech od začátku souboru: nejméně významný bajt
Le	1	'XXh'	očekávaná délka dat, počet bajtů ke čtení

Obr 24 – Struktura příkazu pro čtení dat (zdroj: [13])

Velikost dat, které lze načíst pomocí jednoho příkazu *Read Binary* je v rozmezí 0-255 B. Některé soubory na kartě řidiče jsou ale větší než 255 B, a proto je nelze načíst pouze jedním příkazem. Načítání takovýchto souborů probíhá obvykle v cyklu, ve kterém se postupně zvětšuje offset použitý v parametrech P1 a P2 o 256 B.

Na následujícím obrázku je zobrazena odpověď karty na příkaz *Read Binary*:

Bajt	Délka	Hodnota	Popis
#1-#X	X	'XX..XXh'	čtená data
SW	2	'XXXXh'	stavová slova (SW1, SW2)

Obr 25 – Struktura odpovědi čtení dat (zdroj: [13])

Pokud je odeslaný příkaz úspěšný, vrací karta v SW zpět 9000, pokud by před odesláním příkazu nebyl vybrán žádny EF, karta zpět odešle stav zpracování 6986. Jestliže řízení přístupu vybraného souboru dat není uspokojivé, příkaz je přerušen s 6982, který informuje o nedodržení bezpečnostního statusu. V případě, že offset není kompatibilní s velikostí EF (offset > EF velikost EF) je zpět poslaný stav zpracování 6B00 informující o chybných parametrech. Jestliže velikost dat pro čtení není kompatibilní s velikostí EF (offset + Le > velikost EF), je zpět poslaný stav zpracování 6700 nebo 6Cxx, kde „xx“ udává přesnou délku. V případě, že je detekována chyba integrity uvnitř souboru atributů, karta považuje soubor za poškozený a obnovitelný a zpět poslaný stav zpracování je 6400 nebo 6581, pokud chyba integrity je detekována uvnitř uložených dat, karta vrátí požadovaná data a zpět poslaný stav zpracování je 6281.

* U parametru P1 musí být bit 8 nastaven na 0

4.2.3 Struktura dat

V předchozích kapitolách bylo rozebráno jak přistupovat k jednotlivým souborům a jak z nich načítat jednotlivá data. V této kapitole budou postupně rozebrány jednotlivé soubory včetně popisu vnitřních struktur a velikostí, kterou na kartě zabírají. Na obrázku 26 je znázorněna stromová struktura dat uložených na kartě řidiče. Struktura dat je použita z [13].

Soubor/prvek dat	Počet záznamů	Velikost (v bajtech) Min	Velikost (v bajtech) Max	Standardní hodnoty
MF		*11411*	*24959*	
╠═EF ICC		*25*	*25*	
╠═└─CardIccIdentification		*25*	*25*	
╠═├─clockStop		1	1	{00}
╠═├─cardExtendedSerialNumber		8	8	{00..00}
╠═├─cardApprovalNumber		8	8	{20..20}
╠═├─cardPersonaliserID		1	1	{00}
╠═├─embedderIcAssemblerId		5	5	{00..00}
╠═└─icIdentifier		2	2	{00 00}
╠═EF IC		*8*	*8*	
╠═└─CardChipIdentification		*8*	*8*	
╠═├─icSerialNumber		4	4	{00..00}
╠═└─icManufacturingReferences		4	4	{00..00}
╚═DF Tachograph		*11378*	*24926*	
╠═EF Application_Identification		*10*	*10*	
╠═└─DriverCardApplicationIdentification		*10*	*10*	
╠═├─typeOfTachographCardId		1	1	{00}
╠═├─cardStructureVersion		2	2	{00 00}
╠═├─noOfEventsPerType		1	1	{00}
╠═├─noOfFaultsPerType		1	1	{00}
╠═├─activityStructureLength		2	2	{00 00}
╠═├─noOfCardVehicleRecords		2	2	{00 00}
╠═└─noOfCardPlaceRecords		1	1	{00}
╠═EF Card_Certificate		*194*	*194*	
╠═└─CardCertificate		194	194	{00..00}
╠═EF CA_Certificate		*194*	*194*	
╠═└─MemberStateCertificate		194	194	{00..00}
╠═EF Identification		*143*	*143*	
╠═├─CardIdentification		*65*	*65*	
╠═├─├─cardIssuingMemberState		1	1	{00}
╠═├─├─cardNumber		16	16	{20..20}
╠═├─├─cardIssuingAuthorityName		36	36	{20..20}
╠═├─├─cardIssueDate		4	4	{00..00}
╠═├─├─cardValidityBegin		4	4	{00..00}
╠═├─└─cardExpiryDate		4	4	{00..00}
╠═└─DriverCardHolderIdentification		*78*	*78*	
╠═├─cardHolderName		*72*	*72*	
╠═├─├─holderSurname		36	36	{00, 20..20}
╠═├─└─holderFirstNames		36	36	{00, 20..20}
╠═├─cardHolderBirthDate		4	4	{00..00}
╠═└─cardHolderPreferredLanguage		2	2	{20 20}
╠═EF Card_Download		*4*	*4*	
╠═└─LastCardDownload		4	4	
╠═EF Driving_Licence_Info		*53*	*53*	
╠═└─CardDrivingLicenceInformation		*53*	*53*	
╠═├─drivingLicenceIssuingAuthority		36	36	{00, 20..20}
╠═├─drivingLicenceIssuingNation		1	1	{00}
╠═└─drivingLicenceNumber		16	16	{20..20}
╠═EF Events_Data		*864*	*1728*	
╠═└─CardEventData		*864*	*1728*	
╠═└─cardEventRecords	6	*144*	*288*	
╠═└─CardEventRecord	n_1	*24*	*24*	
╠═├─eventType		1	1	{00}
╠═├─eventBeginTime		4	4	{00..00}
╠═├─eventEndTime		4	4	{00..00}
╠═└─eventVehicleRegistration				
╠═├─vehicleRegistrationNation		1	1	{00}
╠═└─vehicleRegistrationNumber		14	14	{00, 20..20}

EF Faults_Data		*576*	*1152*	
CardFaultData		*576*	*1152*	
cardFaultRecords	2	*288*	*576*	
CardFaultRecord	n_2	*24*	*24*	
faultType		1	1	{00}
faultBeginTime		4	4	{00..00}
faultEndTime		4	4	{00..00}
faultVehicleRegistration				
vehicleRegistrationNation		1	1	{00}
vehicleRegistrationNumber		14	14	{00, 20..20}
EF Driver_Activity_Data		*5548*	*13780*	
CardDriverActivity		*5548*	*13780*	
activityPointerOldestDayRecord		2	2	{00 00}
activityPointerNewestRecord		2	2	{00 00}
activityDailyRecords	n_6	5544	13776	{00..00}
EF Vehicles_Used		*2606*	*6202*	
CardVehiclesUsed		*2606*	*6202*	
vehiclePointerNewestRecord		2	2	{00 00}
cardVehicleRecords		*2604*	*6200*	
CardVehicleRecord	n_3	*31*	*31*	
vehicleOdometerBegin		3	3	{00..00}
vehicleOdometerEnd		3	3	{00..00}
vehicleFirstUse		4	4	{00..00}
vehicleLastUse		4	4	{00..00}
vehicleRegistration				
vehicleRegistrationNation		1	1	{00}
vehicleRegistrationNumber		14	14	{00, 20..20}
vuDataBlockCounter		2	2	{00 00}
EF Places		*841*	*1121*	
CardPlaceDailyWorkPeriod		*841*	*1121*	
placePointerNewestRecord		1	1	{00}
placeRecords		*840*	*1120*	
PlaceRecord	n_4	*10*	*10*	
entryTime		4	4	{00..00}
entryTypeDailyWorkPeriod		1	1	{00}
dailyWorkPeriodCountry		1	1	{00}
dailyWorkPeriodRegion		1	1	{00}
vehicleOdometerValue		3	3	{00..00}
EF Current_Usage		*19*	*19*	
CardCurrentUse		*19*	*19*	
sessionOpenTime		4	4	{00..00}
sessionOpenVehicle				
vehicleRegistrationNation		1	1	{00}
vehicleRegistrationNumber		14	14	{00, 20..20}
EF Control_Activity_Data		*46*	*46*	
CardControlActivityDataRecord		*46*	*46*	
controlType		1	1	{00}
controlTime		4	4	{00..00}
controlCardNumber				
cardType		1	1	{00}
cardIssuingMemberState		1	1	{00}
cardNumber		16	16	{20..20}
controlVehicleRegistration				
vehicleRegistrationNation		1	1	{00}
vehicleRegistrationNumber		14	14	{00, 20..20}
controlDownloadPeriodBegin		4	4	{00..00}
controlDownloadPeriodEnd		4	4	{00..00}
EF Specific_Conditions		*280*	*280*	
SpecificConditionRecord	56	*5*	*5*	
entryTime		4	4	{00..00}
SpecificConditionType		1	1	{00}

		Min	Max
n_1	NoOfEventsPerType	**6**	12
n_2	NoOfFaultsPerType	**12**	24
n_3	NoOfCardVehicleRecords	**84**	200
n_4	NoOfCardPlaceRecords	**84**	112
n_6	CardActivityLengthRange	5 544 bajtů (28 dnů * 93 změn činnosti)	13 776 bajtů (28 dnů * 240 změn činnosti)

Obr 26 – Struktura dat (zdroj: [13])

4.2.3.1 EF ICC

Tento soubor obsahuje jednu strukturu dat *CardIccIdentification*, která má v sobě informace týkající se označení karty s integrovaným obvodem

CardIccIdentification ::= SEQUENCE {

clockStop je mód Clockstop dle EN 726-3

cardExtendedSerialNumber je pořadové číslo karty s integrovaným obvodem a výrobní údaj karty integrovaného obvodu dle EN 726-3

cardApprovalNumber je číslo schválení typu karty

cardPersonaliserID je individuální identifikátor karty – dle definice v EN 726-3

}

4.2.3.2 EF IC

Informace uložené v tomto souboru jsou určené k identifikaci integrovaného obvodu karty. Soubor obsahuje jednu strukturu dat:

CardChipIdentification ::= SEQUENCE {

icSerialNumber je pořadové číslo integrovaného obvodu dle EN 726-3

icManufacturingReferences je označení výrobce integrovaného obvodu a výrobního článku dle EN 726-3

}

4.2.3.3 EF Application_Identification

V souboru jsou uložené informace týkající se identifikace žádosti o kartu a jsou uloženy v následující struktuře:

DriverCardApplicationIdentification ::= SEQUENCE {

typeOfTachographCardId udává implementovaný typ karty

cardStructureVersion udává verzi struktury, která je implementována v kartě

noOfEventsPerType je počet událostí od každého typu události, který může karta zaznamenat

noOfFaultsPerType je počet závad od každého typu závady, který může karta zaznamenat

activityStructureLength udává počet bajtů, které jsou k dispozici pro uložení záznamů činnosti

noOfCardVehicleRecords je počet záznamů vozidla, které může karta obsahovat

noOfCardPlaceRecords je počet míst, který může karta zaznamenat

}

4.2.3.4 EF Card_Certificate

Soubor obsahuje certifikát veřejného klíče karty: **CardCertificate** ::= digitální podpis s částečnou obnovou *CertificateContent* podle dodatku 11 „Společný bezpečnostní mechanismus“: podpis (128 bajtů) || zbytek veřejného klíče (58 bajtů) || název certifikačního orgánu (8 bajtů).

4.2.3.5 EF CA_Certificate

Podobně jako předchozí soubor i tento obsahuje certifikát, ale tentokrát se jedná o certifikát veřejného klíče členského státu vydaný Evropským certifikačním orgánem: **MemberStateCertificate**, který má stejné složení jako **CardCertificate**.

4.2.3.6 EF Identification

Tento soubor obsahuje celkem dvě různé struktury dat. První je *CardIdentification*, která obsahuje informace týkající se identifikace karty a druhá *DriverCardHolder-Identification*, která má v sobě uloženy informace týkající se identifikace držitele karty.

CardIdentification ::= SEQUENCE {

cardIssuingMemberState je kód členského státu vydávajícího kartu

cardNumber je číslo karty

cardIssuingAuthorityName je název orgánu vydávajícího kartu

cardIssueDate je datum vydání karty současnému držiteli

cardValidityBegin je datum počátku platnosti karty

cardExpiryDate je datum konce platnosti karty

}

DriverCardHolderIdentification ::= SEQUENCE {

cardHolderName je příjmení a jméno držitele karty řidiče

cardHolderBirthDate je datum narození držitele karty řidiče

cardHolderPreferredLanguage je obvyklý pracovní jazyk držitele karty

}

4.2.3.7 EF Card_Download

Soubor obsahuje datum a čas naposled stažených data z karty (používá se pro jiné účely jako např. kontrola). Toto datum může být aktualizováno libovolným celkem ve vozidle nebo čtečkou karet. Tento datum je uložen v **LastCardDownload**.

4.2.3.8 EF Driving_Licence_Info

V souboru je jedna struktura *CardDrivingLicenceInformation*, která obsahuje informace týkající se dat karty držitele řidičského oprávnění.

CardDrivingLicenceInformation ::= SEQUENCE {

drivingLicenceIssuingAuthority je orgán vydávající řidičský průkaz

drivingLicenceIssuingNation je stát orgánu vydávajícího řidičský průkaz

drivingLicenceNumber je číslo řidičského průkazu

}

4.2.3.9 EF Events_Data

Tento soubor uchovává informace týkající se událostí v souvislosti s kartou držitele. Informace jsou uloženy ve struktuře obsahující celkem šest záznamů z nichž každý záznam umožňuje uložit 6–12 událostí.

CardEventData ::= SEQUENCE SIZE(6) OF {

cardEventRecords SET SIZE(6–12) OF CardEventRecord

}

CardEventData je posloupnost hodnot *EventFaultType* uspořádaná vzestupně hodnotami cardEventRecords (kromě pokusů narušení spolehlivosti záznamů, které jsou seskupeny v posledním souboru posloupnosti)

cardEventRecords je soubor záznamů událostí určité kategorie událostí (záznamové zařízení nebo karta)

CardEventRecord ::= SEQUENCE {

eventType je typ události

eventBeginTime je datum a čas začátku události

eventEndTime je datum a čas konce události

eventVehicleRegistration je registrační číslo vozidla a členský stát registrace vozidla, ve kterém událost nastala

}

4.2.3.10 EF Faults_Data

Soubor je podobný předchozímu. Tentokrát uchovává ve struktuře závady, které se mohou vyskytnout na vozidle. Složení struktury je obdobné, ale tentokrát se skládá ze dvou záznamů, a každý může obsahovat 12–24 závad.

CardFaultData ::= SEQUENCE SIZE(2) OF {

cardFaultRecords SET SIZE(12–24) OF CardFaultRecord

}

CardFaultData je posloupnost záznamů závad záznamového zařízení doprovázená záznamy závad karty

cardFaultRecords je soubor záznamů závad určité kategorie závad (záznamové zařízení nebo karta)

CardFaultRecord ::= SEQUENCE {

faultType je typ závady

faultBeginTime je datum a čas začátku závady

faultEndTime je datum a čas konce závady

faultVehicleRegistration je registrační číslo vozidla a členský stát registrace vozidla, ve kterém závada nastala

}

4.2.3.11 EF Driver_Activity_Data

Tento soubor slouží pro ukládání jednotlivých činností řidiče (práce, řízení, přestávka, pohotovost). Při každé změně činnosti dojde k jejímu zaznamenání, přičemž minimální trvání takovéto činnosti je jedna minuta. Na tomto základě musí být soubor schopen ukládat změny jednotlivých činností každou minutu a to po dobu minimálně 28 dní. Zaznamenávané činnosti se ukládají do následující struktury:

CardDriverActivity ::= SEQUENCE {

activityPointerOldestDayRecord je určení začátku paměťového místa (počet bajtů od začátku řetězce) nejstaršího úplného denního záznamu v řetězci *activityDailyRecords*

activityPointerNewestRecord je určení začátku paměťového místa (počet bajtů od začátku řetězce) nejnovějšího denního záznamu v řetězci *activityDailyRecords*

activityDailyRecords je prostor vhodný k uložení dat činnosti řidiče (struktura dat: *CardActivityDailyRecord*) pro každý kalendářní den, kdy byla karta použita

Přiřazení hodnoty: tento oktetový řetězec je cyklicky plněn záznamy *CardActivityDailyRecord*. Při prvním použití začíná ukládání do paměti údajů na prvním bajtu řetězce. Všechny nové záznamy jsou připojeny na konec předchozího. Když je řetězec plný, ukládání pokračuje na prvním bajtu řetězce nezávisle na přerušení, které je uvnitř datového prvku. Před umístěním dat nové činnosti do řetězce, která nahrazuje starší data činnosti, musí být *activityPointerOldestDayRecord* aktualizovány k vyjádření nového umístění nejstaršího úplného denního záznamu a *activityPreviousRecordLength* tohoto (nového) nejstaršího úplného denního záznamu musí být nastavena na nulu.

}

CardActivityDailyRecord ::= SEQUENCE {

activityPreviousRecordLength je celková délka záznamu předešlého dne v bajtech. Jestliže tento záznam je nejdelší denní záznam, musí být hodnota *activityPreviousRecordLength* nastavena na 0

activityRecordLength je celková délka tohoto záznamu v bajtech

activityRecordDate je datum záznamu

activityDailyPresenceCounter je denní prezentační čítač pro kartu toho dne

activityDayDistance je celková vzdálenost ujetá toho dne

activityChangeInfo je soubor *ActivityChangeInfo* dat toho dne pro řidiče. Může obsahovat maximálně 1440 hodnot (jedna změna činnosti za minutu). Tento soubor vždy obsahuje *activityChangeInfo* pro status řidiče v 00.00

}

ActivityChangeInfo::= Oktetové uspořádání: „scpaatttttttttt“ (16 bitů) kde:

- **‚s‘** Otvor pro kartu (nepoužije se, jestliže ‚p‘ = 1 kromě dále uvedené poznámky):
  - ‚0‘ Řidič
  - ‚1‘ 2. Řidič
- **‚c‘** Status řízení vozidla (v případě ‚p‘ = 0):
  - ‚0‘ Samostatný řidič
  - ‚1‘ Posádka

- **‚c‘** Následující status činnosti (v případě ‚p‘ = 1):
  - ‚0‘ Neznámý
  - ‚1‘ Známý (= ručně vložený)
- **‚p‘** Status karty
  - ‚0‘ Vložena
  - ‚1‘ Není vložena
- **‚aa‘** Činnost (nepoužije se, jestliže ‚p‘ = 1 a ‚c‘ = 0 kromě dále uvedené poznámky):
  - ‚00‘ Přestávka/odpočinek
  - ‚01‘ Pohotovost
  - ‚10‘ Práce
  - ‚11‘ Řízení vozidla
- **‚ttttttttt‘** Čas změny: počet minut od 00h00 daného dne

4.2.3.12 EF Vehicles_Used

Soubor umožňuje ukládat data týkající se vozidel použitých držitelem karty. Následující struktura dokáže uložit 84–200 dob používání vozidla během kalendářního dne:

CardVehiclesUsed ::= SEQUENCE {

vehiclePointerNewestRecord je index posledního aktualizovaného záznamu vozidla

Přiřazení hodnoty: Číslo odpovídající čítači záznamů vozidla začínající ‚0‘ pro první výskyt záznamu vozidla ve struktuře.

cardVehicleRecords je soubor záznamů obsahující informace o použití vozidla SET SIZE (84–200) OF SEQUENCE {

vehicleOdometerBegin je hodnota měřiče ujeté vzdálenosti na začátku doby použití vozidla

vehicleOdometerEnd je hodnota měřiče ujeté vzdálenosti na konci doby použití vozidla

vehicleFirstUse je datum a čas začátku doby použití vozidla

vehicleLastUse je datum a čas konce doby použití vozidla

vehicleRegistration je registrační číslo vozidla a členský stát registrace vozidla

vuDataBlockCounter je hodnota *VuDataBlockCounter* při posledním výpisu doby použití vozidla, který dokáže identifikovat postupně cykly vkládání a vyjímání karty v celku ve vozidle

}

}

4.2.3.13 EF Places

Soubor umožňuje ukládání dat týkající se míst, kde denní pracovní doba začíná nebo končí.

CardPlaceDailyWorkPeriod ::= SEQUENCE {

PlacePointerNewestRecord INTEGER(0..NoOfCardPlaceRecords-1)

PlaceRecords SET SIZE(NoOfCardPlaceRecords) OF PlaceRecord

}

placePointerNewestRecord je index posledního aktualizovaného záznamu o místě

Přiřazení hodnoty: Číslo odpovídající čítači záznamu míst začínající ‚0' pro první výskyt záznamu místa ve struktuře.
PlaceRecords je soubor záznamů obsahující informaci týkající se vložených míst, tento soubor má následující strukturu:

PlaceRecord ::= SEQUENCE {

entryTime je datum a čas vztahující se ke vstupu

entryTypeDailyWorkPeriod je typ vstupu

dailyWorkPeriodCountry je vložený kraj

dailyWorkPeriodRegion je vložený region

vehicleOdometerValue je údaj měřiče ujeté vzdálenosti vztažený k času a vloženému místu

}

4.2.3.14 EF Current_Usage

Soubor ukládající informace o aktuálním použití karty.

CardCurrentUse ::= SEQUENCE {

sessionOpenTime je čas, kdy je karta vložena pro běžné použití. Tato položka se nastavuje na nulu při vyjmutí karty

sessionOpenVehicle je identifikace běžného použití vozidla nastaveného při vložení karty. Tato položka se nastavuje na nulu při vyjmutí karty

}

4.2.3.15 EF Control_Activity_Data

Informace uložené v tomto souboru se týkají poslední kontroly, které byl řidič podroben.

CardControlActivityDataRecord ::= SEQUENCE {

controlType je typ kontroly

controlTime je datum a čas kontroly

controlCardNumber je číslo kontrolní úřední osoby mající vykonat kontrolu

controlVehicleRegistration je registrační číslo vozidla a členský stát registrace vozidla, ve kterém byla kontrola provedena

controlDownloadPeriodBegin a **controlDownloadPeriodEnd** je doba stahování dat v případě stahování

}

4.2.3.16 EF Specific_Conditions

Do tohoto souboru se ukládají informace o specifických podmínkách. Soubor dokáže uložit až 56 struktur následujícího typu:

SpecificConditionRecord ::= SEQUENCE {

entryTime je datum a čas vstupu

specificConditionType je kód identifikující specifickou podmínku

}

5 Program DITA

Program DITA byl pojmenován podle počátečních písmen **di**gitálního **ta**chografu. Celý program je vytvořen v programovacím jazyce C# navržený firmou Microsoft, který umožňuje vytvářet programy pro platformu .NET Framework. Verze .NET Framework, pro kterou je program vytvořen, je 3,5. Program DITA byl naprogramován ve vývojovém prostředí MS Visual Studio 2008 Express Edition, které je zdarma dostupné na internetových stránkách firmy Microsoft.

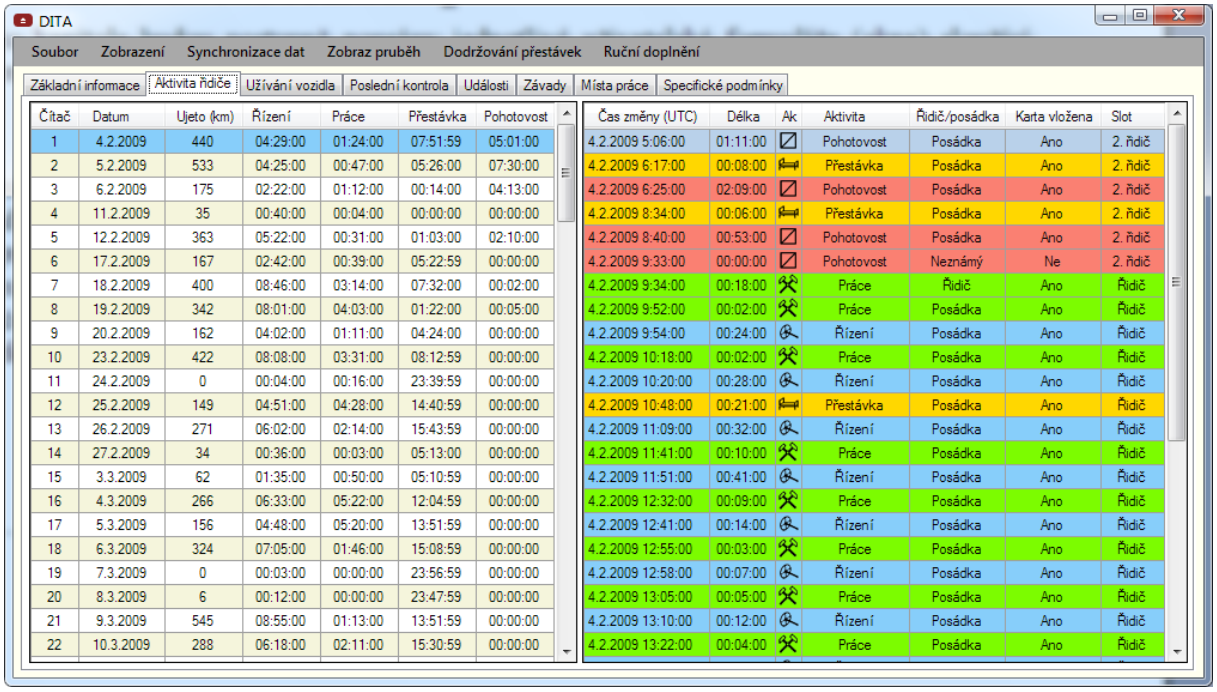
V následujících několika kapitolách bude popsán celý program. Nejprve bude popsáno grafické uživatelské rozhraní (vizuální stránka programu), dále pak budou popsány nejdůležitější třídy používané v tomto programu. Jako poslední částí této kapitoly bude popis jednotlivých algoritmů, které jsou v programu implementovány a jsou používány ke zjišťování dodržování nařízení č. 561/2006.

5.1 Grafické rozhraní

V této kapitole budou postupně popsány jednotlivé uživatelské formuláře (okna) sloužící k zobrazování informací uživateli a komunikaci s ním.

5.1.1 Hlavní formulář

Hlavní okno se otevře po spuštění souboru DITA.exe, který se nachází na přiloženém CD. Pokud jde o první spuštění programu, zobrazí se toto okno uprostřed obrazovky. V případě, že byl program již dříve spuštěn a uživatel změnil umístění hlavního okna na jiné místo obrazovky, dojde při následném spuštění k zobrazení na témže místě, jako bylo při posledním ukončení programu. Hlavní okno programu DITA* je názorně zobrazeno na následujícím obrázku.



Čítač	Datum	Ujeto (km)	Řízení	Práce	Přestávka	Pohotovost
1	4.2.2009	440	04:29:00	01:24:00	07:51:59	05:01:00
2	5.2.2009	533	04:25:00	00:47:00	05:26:00	07:30:00
3	6.2.2009	175	02:22:00	01:12:00	00:14:00	04:13:00
4	11.2.2009	35	00:40:00	00:04:00	00:00:00	00:00:00
5	12.2.2009	363	05:22:00	00:31:00	01:03:00	02:10:00
6	17.2.2009	167	02:42:00	00:39:00	05:22:59	00:00:00
7	18.2.2009	400	08:46:00	03:14:00	07:32:00	00:02:00
8	19.2.2009	342	08:01:00	04:03:00	01:22:00	00:05:00
9	20.2.2009	162	04:02:00	01:11:00	04:24:00	00:00:00
10	23.2.2009	422	08:08:00	03:31:00	08:12:59	00:00:00
11	24.2.2009	0	00:04:00	00:16:00	23:39:59	00:00:00
12	25.2.2009	149	04:51:00	04:28:00	14:40:59	00:00:00
13	26.2.2009	271	06:02:00	02:14:00	15:43:59	00:00:00
14	27.2.2009	34	00:36:00	00:03:00	05:13:00	00:00:00
15	3.3.2009	62	01:35:00	00:50:00	05:10:59	00:00:00
16	4.3.2009	266	06:33:00	05:22:00	12:04:59	00:00:00
17	5.3.2009	156	04:48:00	05:20:00	13:51:59	00:00:00
18	6.3.2009	324	07:05:00	01:46:00	15:08:59	00:00:00
19	7.3.2009	0	00:03:00	00:00:00	23:56:59	00:00:00
20	8.3.2009	6	00:12:00	00:00:00	23:47:59	00:00:00
21	9.3.2009	545	08:55:00	01:13:00	13:51:59	00:00:00
22	10.3.2009	288	06:18:00	02:11:00	15:30:59	00:00:00

Čas změny (UTC)	Délka	Ak	Aktivita	Řidič/posádka	Karta vložena	Slot
4.2.2009 5:06:00	01:11:00		Pohotovost	Posádka	Ano	2. řidič
4.2.2009 6:17:00	00:08:00		Přestávka	Posádka	Ano	2. řidič
4.2.2009 6:25:00	02:09:00		Pohotovost	Posádka	Ano	2. řidič
4.2.2009 8:34:00	00:06:00		Přestávka	Posádka	Ano	2. řidič
4.2.2009 8:40:00	00:53:00		Pohotovost	Posádka	Ano	2. řidič
4.2.2009 9:33:00	00:00:00		Pohotovost	Neznámý	Ne	2. řidič
4.2.2009 9:34:00	00:18:00		Práce	Řidič	Ano	Řidič
4.2.2009 9:52:00	00:02:00		Práce	Posádka	Ano	Řidič
4.2.2009 9:54:00	00:24:00		Řízení	Posádka	Ano	Řidič
4.2.2009 10:18:00	00:02:00		Práce	Posádka	Ano	Řidič
4.2.2009 10:20:00	00:28:00		Řízení	Posádka	Ano	Řidič
4.2.2009 10:48:00	00:21:00		Přestávka	Posádka	Ano	Řidič
4.2.2009 11:09:00	00:32:00		Řízení	Posádka	Ano	Řidič
4.2.2009 11:41:00	00:10:00		Práce	Posádka	Ano	Řidič
4.2.2009 11:51:00	00:41:00		Řízení	Posádka	Ano	Řidič
4.2.2009 12:32:00	00:09:00		Práce	Posádka	Ano	Řidič
4.2.2009 12:41:00	00:14:00		Řízení	Posádka	Ano	Řidič
4.2.2009 12:55:00	00:03:00		Práce	Posádka	Ano	Řidič
4.2.2009 12:58:00	00:07:00		Řízení	Posádka	Ano	Řidič
4.2.2009 13:05:00	00:05:00		Práce	Posádka	Ano	Řidič
4.2.2009 13:10:00	00:12:00		Řízení	Posádka	Ano	Řidič
4.2.2009 13:22:00	00:04:00		Práce	Posádka	Ano	Řidič

Obr 27 – Hlavní okno

* Všechny časy používané v programu DITA jsou uvedeny v UTC

Okno hlavního formuláře obsahuje ve své horní části uživatelské menu, kde je možné zvolit následující nabídky a funkce:

1. *Soubor*: jedná se o základní nabídku, která nechybí snad v žádném programu a umožňuje následující volby:
   - *Načti z karty* – slouží pro načítání dat z karty řidiče. Pokud není při výběru této položky připojena žádná čtečka karet, program zobrazí výzvu k připojení čtečky čipových karet anebo k ukončení načítání z karty. Po připojení čtečky karet a vsunutí karty dojde k ověření zda-li se jedná opravdu o procesorovou čipovou kartu řidiče. V případě, že by nebyla vložena karta řidiče, dojde opěk k chybovému hlášení s žádostí o vložení správné karty. Pokud je vše v pořádku zobrazí se okno s průběhem načítání, které bude popisováno později. Po úspěšném dokončení načítání dat z karty dojde k jejich zobrazení v jednotlivých záložkách hlavního okna.
   - *Načti ze souboru* – podobně jako předchozí nabídka i tato slouží k načítání dat, ale tentokrát ze souboru, který je uložen na paměťovém mediu připojeném k počítači. Po výběru této položky se zobrazí dialogové okno sloužící k zadání cesty k požadovanému souboru. Soubory podporované programem DITA mají koncovku *.drc*, která je vytvořena jako zkratka z anglického pojmenování karty řidiče nebo-li *Driver Card*. Soubory s jinou koncovkou resp. soubory obsahující jinou datovou strukturu nejsou podporovány. V případě, že by došlo k pokusu načíst nepodporovaný soubor, dojde k chybovému hlášení. Po úspěšném načtení souboru obsahující data z karty řidiče dojde k jejich automatickému zobrazení v záložkách hlavního okna.
   - *Ulož* – tato položka slouží k ukládání načtených dat z karty řidiče, nebo dat upravených uživatelem. Může jít o data, která byla synchronizována s kartou, případně souborem, nebo data, která uživatel upravil (např. vložil informace o tom jakou činnost vykonával řidič mimo svou pracovní dobu). Po výběru položky Ulož, dojde k zobrazení dialogového okna, ve kterém je možné určit cestu, kam chce uživatel data uložit a dále pak je možné tato data uložit pod specifickým názvem. Po zadání názvu dojde k uložení dat do souboru s příponou *.drc*.
   - *Konec* – slouží k ukončení programu. Pokud by v průběhu práce s programem došlo k načtení/úpravě dat, která nejsou uložena, program ještě před svým ukončením zobrazí výzvu zda-li si uživatel přeje neuložená data uložit.
2. *Zobrazení*: slouží k přepnutí způsobu zobrazení aktivit řidiče v záložce „Aktivita řidiče“. Položky této nabídky jsou trochu specifické a liší se od ostatních tím, že je možné vždy vybrat pouze jednu. Pokud se vybere druhá, dojde k přepnutí a první se nepoužije. Podobně jako zobrazení hlavního formuláře na obrazovce po spuštění programu tak i nastavení zobrazení se použije takové, jaké bylo při minulém ukončení programu. Nabídka umožňuje dva způsoby zobrazení:
   - *Úplné* – při zvolení této možnosti jsou zobrazeny kompletní informace o aktivitách řidiče. Jedná se o zobrazení všech dní, které jsou načteny z karty řidiče a dále i ručně doplněných.
   - *Základní* – oproti předchozímu zobrazení se liší tím, že dochází k zobrazování pouze těch dní, které byly načteny z karty řidiče. Pokud uživatel doplní i ostatní dny nebudou ve výpisu aktivit zobrazeny.
3. *Synchronizace dat*: slouží ke spojování dvou různých dat načtených ze stejné karty řidiče, která byla pořízena v jiný časový okamžik. Jelikož se na kartu řidiče vejde

minimálně 28 celých dní a ze zákona musí být data uchovávána minimálně jeden rok, je potřeba tato data archivovat. Pokud by se ukládalo každé stažení dat samostatně, mohlo by docházet ke zbytečným ukládáním již jednou uložených dat a z tohoto důvodu je v programu použita synchronizace dat. Synchronizovat je možné dvěma způsoby:

- *S kartou* – před použitím musí být již načtena data ze souboru
- *Se souborem* – před použitím musí být načtena data z karty

Způsob synchronizace pomocí obou položek je v podstatě stejný. Nejprve dojde k načtení druhé části dat a poté k ověření shodnosti základních identifikačních údajů. Pokud by identifikační údaje nebyly shodné, program by zobrazil chybu a synchronizace by se neprovedla. V opačném případě dojde k sesynchonizování a výsledná data se zobrazí na hlavním formuláři.

4. *Zobrazení průběhu*: slouží k vizuálnímu zobrazení aktuálně načtených aktivit řidiče. Po výběru této položky se zobrazí okno pro zobrazení dat, které bude popsáno v kapitole 5.1.4
5. *Dodržování přestávek*: tato položka menu slouží k ověřování dodržování přestávek v řízení a doby jízdy podle nařízení č. 561/2006. Zde je možné si vybrat ze dvou možných variant za jaké časové období se bude kontrola nařízení počítat:
   - *Všechna data* – kontrola nařízení probíhá na všech datech, které jsou aktuálně načteny.
   - *Od zvoleného data* – zde je možné, aby si uživatel určil, od kterého časového okamžiku chce provést kontrolu normy.

   U obou těchto variant se vždy zobrazí výsledek v okně kontroly nařízení č. 561/2006, které je popsáno v kapitole 5.3.
6. *Ruční doplnění*: slouží uživateli pro ruční zadávání takových časových úseků, které řidič vozidla trávil mimo pracovní dobu. Zde je možné ručně zadat data pomocí dvou položek:
   - *Zadat ručně* – tato položka umožňuje vybrat uživateli dva časové úseky, mezi kterými se doplní zvolená činnost. Po vybrání se zobrazí okno pro ruční zadání dat, které bude podrobněji popsáno později.
   - *Kontrola spojitosti* – slouží ke kontrole spojitosti dat. Podobněji viz kapitola 5.1.7.

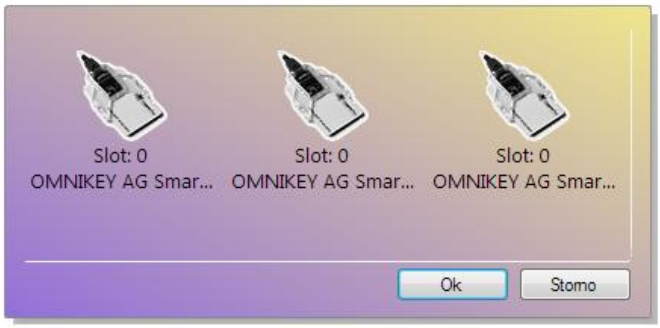
Kromě právě popsaného menu obsahuje hlavní okno prostor, který slouží pro zobrazování jednotlivých informací načtených z karty řidiče uživateli. Tato informační část zabírá skoro celou plochu hlavního okna a jednotlivé zobrazované informace jsou rozděleny do nezávislých záložek. Těchto záložek je na formuláři celkem osm:

1. *Základní informace*: na této záložce jsou zobrazeny základní informace o držiteli karty a o kartě samotné. Prostor je rozdělen na dvě části – v horní jsou zobrazovány informace o řidiči (držiteli karty): jméno, příjmení, datum narození a rodný jazyk. Dále jsou v této části informace o vydání karty řidiče: číslo karty, vydávající stát a orgán, datum vydání, začátek a konec platnosti. Jako další jsou zde informace o řidičském oprávnění: číslo řidičského průkazu a místo vydání. Poslední je datum určující poslední stažení dat z karty (pokud takové existuje). V dolní části této záložky jsou informace o kartě řidiče: typ karty, verze struktury dat na kartě a počty jednotlivých struktur, které je na kartu možné uložit: počet událostí, závad, vozidel, míst a jako poslední je zde maximální počet bajtů pro uložení jednotlivých činností řidiče.

2. *Aktivita řidiče*: slouží pro zobrazování jednotlivých aktivit řidiče v rámci jednotlivých dnů. I tato záložka je rozdělená na dvě části, ale tentokrát svisle. Levá část zobrazuje informace o jednotlivých dnech, ve kterých řidič vykonával nějakou činnost a to ve formě tabulky. V této tabulce jsou zobrazeny: čítač, datum, počet ujetých kilometrů v tomto dni a délky jednotlivých činností vykonané v aktuálním dni (řízení, práce, přestávka, pohotovost). Na pravé straně této záložky jsou zobrazeny jednotlivé aktivity příslušného vybraného dne z levé tabulky. Tato data jsou také zobrazována ve formě tabulky, která obsahuje tyto sloupce: čas změny, délka trvání této změny, aktivita ve formě obrázku, aktivita – slovní popis, dále pak zda řidič jel s posádkou či sám, zda byla při činnosti zasunuta karta do tachografu a poslední sloupec informuje o tom, v jakém slotu byla karta vložena. Pravá část této záložky vždy zobrazuje aktuálně vybraný den z tabulky levé. Takže pokud si uživatel vybere jiný den, automaticky se v pravé tabulce zobrazí data související s tímto dnem. Jednotlivé sloupce obou tabulek je možné řadit vzestupně/sestupně kliknutím na záhlaví příslušného sloupce. Řádky levé tabulky jsou barevně odlišeny podle aktivity řidiče na tomto řádku zobrazené.
3. *Užívání vozidla*: zde jsou zobrazena data spojena s užíváním jednotlivých vozidel, které řidič během své činnosti používal. Data jsou zobrazována v přehledné tabulce, která obsahuje tyto sloupce: čítač, registrační značka vozidla, se kterým řidič vykonával svoji činnost, stát, kde je registrováno vozidlo, začátek a konec užívání vozidla a jako poslední je zde počáteční a konečný stav tachometru pořízený v začátku a konci užívání.
4. *Poslední kontrola*: jedná se o výpis jednotlivých položek spojených s poslední kontrolou provedenou na celku vozidla. Vypisované informace jsou tyto: typ kontroly jaká byla na vozidle provedena, datum a čas provedení kontroly, typ, číslo a stát karty která prováděla kontrolu, registrační značka a stát, ve kterém je vozidlo registrováno a jako poslední začátek a konec stahování dat z karty. Pokud nebyla žádná kontrola provedena, dojde k zobrazení „Nebyla zjištěna žádná kontrola“, které se zobrazí v typu kontroly.
5. *Události a Závady*: tyto dvě záložky jsou v podstatě stejné, ale v každé se zobrazují jiné informace. V první jmenované jsou vypisovány zjištěné události a v druhé zjištěné závady na celku vozidla. Události i závady jsou vypisovány rovněž do tabulek obsahující tyto sloupce: typ události/závady, začátek a konec události/závady, registrační značka, na kterém událost/závada vznikla a stát, ve kterém je vozidlo registrováno. Pokud by nebyla zjištěna událost/závada vypíše se informace o tom, že nebyla žádná událost/závada nalezena.
6. *Místa práce*: na této záložce se zobrazují informace související s jednotlivým vložením/vyjmutím karty z digitálního tachografu a dále pak informací, ve kterém regionu a zemi se řidič zrovna nacházel. Informace jsou opět vypisovány do tabulky mající tyto sloupce: datum a čas vstupu, typ vstupu (zde je nejčastěji informace o vložení/vyjmutí karty), registrační značka vozidla, stav tachometru v době vstupu, stát, ve kterém se řidič aktuálně nacházel a region v daném státu. Bohužel pro ČR nejsou definované žádné regiony a proto v případě, že se řidič nachází v ČR, není region dostupný.
7. *Specifické podmínky*: v této záložce jsou zobrazovány jednotlivé specifické podmínky přepravy, které proběhly a byly zaznamenány na kartě řidiče. Jedná se především o přepravu vlakem/lodí apod. Jednotlivé podmínky jsou zde také vypisovány v podobě tabulky obsahující pouze datum a čas, kdy byla započata přeprava za specifických podmínek a dále pak typ vlastní podmínky.

5.1.2 Okno Výběr čtečky karet

Jestliže se na počítači používá více než jedna čtečka čipových karet zároveň, slouží toto okno k výběru té čtečky čipových karet, kterou chce uživatel použít pro načtení dat z karty řidiče. Pokud by k počítači při spuštění načítání dat z karty byla připojena pouze jedna čtečka, toto okno se nezobrazí. V opačném případě dojde k zobrazení jednotlivých připojených čteček včetně jejich slotu a názvu. Uživatel má poté dvě možnosti výběru: vybere si příslušnou čtečku a potvrdí výběr tlačítkem „Ok“ anebo přímo při výběru dvojklikem levým tlačítkem myši na příslušnou čtečku. V obou případech dojde k zavření tohoto okna a dále program pokračuje v načítání dat z vybrané čtečky čipových karet.



Obr 28 – Okno Výběr čtečky karet

5.1.3 Okno Načítání dat

Toto okno se zobrazí pouze tehdy, když si uživatel zvolí z menu nabídku *Soubor – Načti z karty*. Po zobrazení tohoto okna se na pozadí (ve vlastním vlákně) provádí načítání a dekódování jednotlivých souborů. Podle průběhu vlastního načítání a dekódování dochází ke grafickému zobrazování průběhu tohoto načítání, které se zobrazuje na ukazateli průběhu zpracování umístěného uprostřed tohoto okna. Během načítání dat z karty řidiče je možné proces načítání přerušit a to stisknutím tlačítka „Ukončit načítání“. Po ukončení načítání běžným způsobem (tj. bezchybném načtení všech souborů nacházející se na kartě) dojde k zobrazení těchto dat na hlavním formuláři. Pokud by byl proces načítání přerušen uživatelem, nebo by došlo k nějaké chybě, na hlavním formuláři by zůstala zobrazená data, která byla načtena před spuštěním načítání z karty.

Obr 29 – Okno Načítání dat

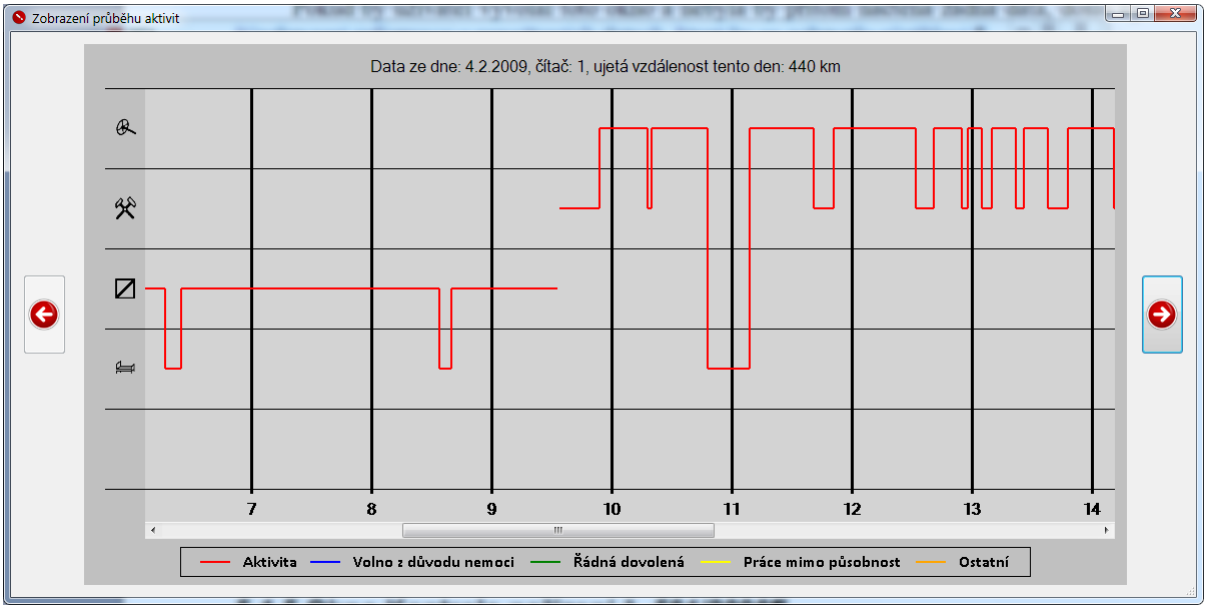
5.1.4 Okno Zobrazení průběhu aktivit

Toto okno je možné vyvolat dvěma způsoby. První je výběr z nabídky menu hlavního okna „Zobraz průběh“ a druhá možnost je dvojklikem levým tlačítkem myši na řádek levé tabulky aktivit na záložce „Aktivita řidiče“. Jak již název okna napovídá, slouží ke grafickému zobrazení průběhu jednotlivých aktivit řidiče v daný den. Pokud došlo k zobrazení okna z nabídky menu, zobrazí se implicitně první den, který je načten a zobrazen v levé tabulce aktivit řidiče. Informace o aktuálně zobrazeném dni jsou vždy popsány v záhlaví zobrazované oblasti. Zde je vypsán datum, čítač a celková ujetá vzdálenost. Pod tímto

záhlavím se nachází velká šedá oblast, která vyplňuje většinu zobrazeného okna. V této šedé oblasti se graficky zobrazuje průběh jednotlivých aktivit řidiče v čase. Časová osa zde vždy znázorňuje celých 24 hodin. Dále je zde pět vodorovných oblastí, které od shora označují: řízení, práce, pohotovost, přestávka a poslední oblast vyznačuje ručně vložená data. V této oblasti jsou pak dále zobrazeny i svislé čáry označující jednotlivé celé hodiny v aktuálním dni. Jednotlivé aktivity dne se vykreslují souvislou barevnou čarou mezi tyto oblasti a vytváří tak spojitý, případně nespojitý graf. Spojitý graf se vykresluje tam, kde existují data, která byla načtena z karty řidiče anebo byla ručně doplněna uživatelem. Nespojitost, nebo-li místo grafu, které není propojeno, vzniká tím, že v tabulce aktivit neexistuje žádný záznam. Taková nespojitost je způsobena převážně vyjmutím karty z tachografu. Ve spodní části vykreslované oblasti se nachází legenda, která znázorňuje, co která barva grafu znamená.

Na tomto okně se ještě nacházejí dvě tlačítka (jedno v pravé a druhé v levé části okna). Tato tlačítka slouží k zobrazení následujícího/předcházejícího dne v tabulce aktivit. Pokud by takový den v tabulce neexistoval, tlačítko zšedne a nelze ho použít. Při každém stisknutí kteréhokoli tlačítka se vždy překreslí zobrazovaná oblast aktuálními daty. Pokud první změna aktivit nenastane v několika prvních hodinách, automaticky se nastaví zobrazení dat těsně před první aktivitu, která proběhla tento den. Na procházení jednotlivých dnů také závisí to, jak je nastaveno zobrazení v hlavním okně. Pokud je v hlavním okně nastaveno zobrazení *úplné*, i zde se zobrazují jednotlivé záznamy, tak jak jdou po sobě a to i včetně dní, které nemusí obsahovat žádné aktivity. Pokud je však nastaveno zobrazení na *základní*, dochází k zobrazování pouze aktivit, které byly načteny z karty řidiče.

Pokud by uživatel vyvolal toto okno a nebyla by přitom načtena žádná data, došlo by k zobrazení informace o nenačtených datech, které by se zobrazilo v záhlaví.

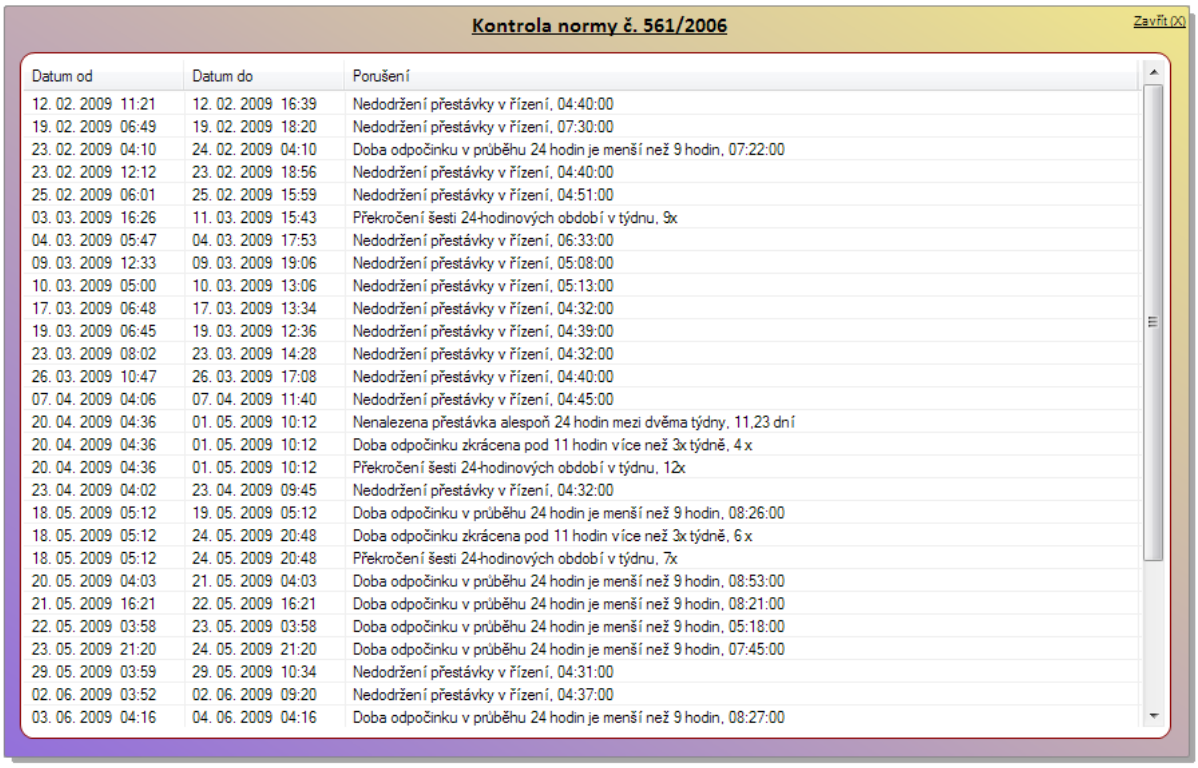


Obr 30 – Okno Zobrazení průběhu aktivit

5.1.5 Okno Kontrola nařízení č. 561/2006

Toto okno slouží k zobrazení informací, zda řidič porušil dodržování předepsaných dob řízení a odpočinku podle nařízení č. 561/2006. Výpočet se provádí podle algoritmů popsaných v kapitole 5.3. Každému takovému porušení odpovídá jeden řádek tabulky, která je v tomto okně zobrazena. V tabulce jsou vypsány jednotlivá porušení včetně počátečního a konečného data a času určujícího, v jakém časovém období porušení trvalo. Dále je zde slovní popis daného porušení včetně případné doplňující informace např.: délka trvání, nebo kolikrát bylo porušení zjištěno během určité doby.

Vyvolání tohoto formuláře je možné pouze z položky „Dodržování přestávek“ v menu hlavního okna. Pokud se v této položce vybere možnost „Všechna data“ dojde v programu ke kontrole všech aktivit řidiče. Pokud se však vybere „Od zvoleného data“ dojde po zadání příslušného data ke kontrole od takto zadaného data.

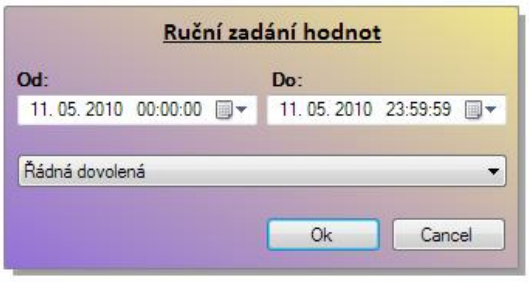


Datum od	Datum do	Porušení
12. 02. 2009 11:21	12. 02. 2009 16:39	Nedodržení přestávky v řízení, 04:40:00
19. 02. 2009 06:49	19. 02. 2009 18:20	Nedodržení přestávky v řízení, 07:30:00
23. 02. 2009 04:10	24. 02. 2009 04:10	Doba odpočinku v průběhu 24 hodin je menší než 9 hodin, 07:22:00
23. 02. 2009 12:12	23. 02. 2009 18:56	Nedodržení přestávky v řízení, 04:40:00
25. 02. 2009 06:01	25. 02. 2009 15:59	Nedodržení přestávky v řízení, 04:51:00
03. 03. 2009 16:26	11. 03. 2009 15:43	Překročení šesti 24-hodinových období v týdnu, 9x
04. 03. 2009 05:47	04. 03. 2009 17:53	Nedodržení přestávky v řízení, 06:33:00
09. 03. 2009 12:33	09. 03. 2009 19:06	Nedodržení přestávky v řízení, 05:08:00
10. 03. 2009 05:00	10. 03. 2009 13:06	Nedodržení přestávky v řízení, 05:13:00
17. 03. 2009 06:48	17. 03. 2009 13:34	Nedodržení přestávky v řízení, 04:32:00
19. 03. 2009 06:45	19. 03. 2009 12:36	Nedodržení přestávky v řízení, 04:39:00
23. 03. 2009 08:02	23. 03. 2009 14:28	Nedodržení přestávky v řízení, 04:32:00
26. 03. 2009 10:47	26. 03. 2009 17:08	Nedodržení přestávky v řízení, 04:40:00
07. 04. 2009 04:06	07. 04. 2009 11:40	Nedodržení přestávky v řízení, 04:45:00
20. 04. 2009 04:36	01. 05. 2009 10:12	Nenalezena přestávka alespoň 24 hodin mezi dvěma týdny, 11,23 dní
20. 04. 2009 04:36	01. 05. 2009 10:12	Doba odpočinku zkrácena pod 11 hodin více než 3x týdně, 4 x
20. 04. 2009 04:36	01. 05. 2009 10:12	Překročení šesti 24-hodinových období v týdnu, 12x
23. 04. 2009 04:02	23. 04. 2009 09:45	Nedodržení přestávky v řízení, 04:32:00
18. 05. 2009 05:12	19. 05. 2009 05:12	Doba odpočinku v průběhu 24 hodin je menší než 9 hodin, 08:26:00
18. 05. 2009 05:12	24. 05. 2009 20:48	Doba odpočinku zkrácena pod 11 hodin více než 3x týdně, 6 x
18. 05. 2009 05:12	24. 05. 2009 20:48	Překročení šesti 24-hodinových období v týdnu, 7x
20. 05. 2009 04:03	21. 05. 2009 04:03	Doba odpočinku v průběhu 24 hodin je menší než 9 hodin, 08:53:00
21. 05. 2009 16:21	22. 05. 2009 16:21	Doba odpočinku v průběhu 24 hodin je menší než 9 hodin, 08:21:00
22. 05. 2009 03:58	23. 05. 2009 03:58	Doba odpočinku v průběhu 24 hodin je menší než 9 hodin, 05:18:00
23. 05. 2009 21:20	24. 05. 2009 21:20	Doba odpočinku v průběhu 24 hodin je menší než 9 hodin, 07:45:00
29. 05. 2009 03:59	29. 05. 2009 10:34	Nedodržení přestávky v řízení, 04:31:00
02. 06. 2009 03:52	02. 06. 2009 09:20	Nedodržení přestávky v řízení, 04:37:00
03. 06. 2009 04:16	04. 06. 2009 04:16	Doba odpočinku v průběhu 24 hodin je menší než 9 hodin, 08:27:00

Obr 31 – Okno Kontrola normy č. 561/2006

5.1.6 Okno Ruční zadání hodnot

Jedná se o jednoduché dialogové okno, které umožňuje uživateli zadat celkem tři položky. Prvními dvěma jsou výchozí a koncové datum a čas a třetím je volená aktivita. Na základě zadaných datumů a časů se program pokusí nalézt takové aktivity, které vytvářejí nespojitý průběh a vložením nových aktivit se zvolenou činností docílit toho, aby výsledná data byla spojitá. Při potvrzení zadání tlačítkem „Ok“ musí být zadané všechny tři položky a zároveň datum a čas „Od“ nesmí být větší než datum „Do“. Pokud by k tomu došlo, program by vypsal chybové hlášení. Po úspěšném zadání vstupních dat a úspěšném doplnění aktivit do jednotlivých dnů je možné si tyto změny zobrazit v hlavním okně v záložce „Aktivita řidiče“.



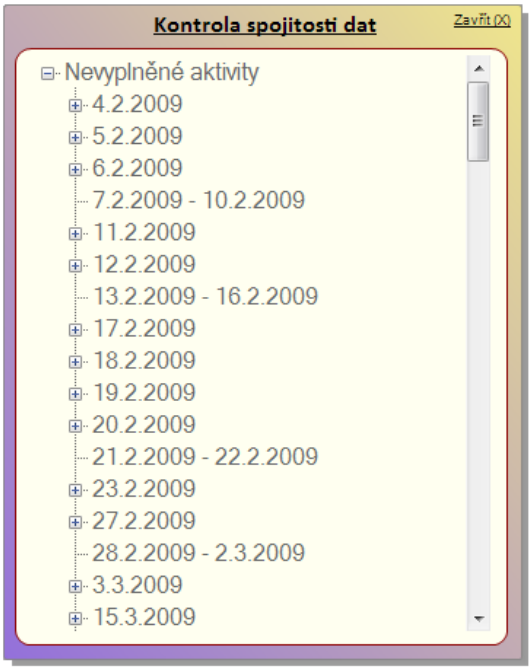
Obr 32 – Okno Ruční zadání hodnot

5.1.7 Okno Kontrola spojitosti dat

Toto okno slouží ke kontrole spojitosti aktuálně načtených dat. Nespojitost se v datech vyskytuje tam, kde došlo k vyjmutí karty z digitálního tachografu. Tyto nespojitosti se zobrazují ve formě hierarchické stromové struktury. První úroveň této struktury odpovídá jednomu nebo popř. více dnům (zde záleží zda-li jsou nespojitosti uprostřed dne, nebo nespojitost trvá několik dní). Pokud je vlevo od takovéhoto záznamu zobrazen symbol „+“, znamená to, že tento den obsahuje alespoň jednu nespojitost, kterou je možno zobrazit kliknutím právě na toto „plus“. Pokud však u této první úrovně žádné „plus“ není, znamená to, že nespojitost trvá celý tento den případně několik dní.

U takto zobrazené neprázdné stromové struktury nespojitých aktivit je možné jednotlivé nespojitosti během dne případně celé dny, nebo dokonce i celý strom odstranit. K odstranění slouží kontextové menu, které je možné vyvolat kliknutím pravým tlačítkem myši na jakoukoliv část tohoto stromu. Výběrem položky menu je možné odstranit jednotlivé nespojitosti vložením aktivit o zadané činnosti. Činnosti, které je zde možno zvolit, jsou stejné jako v okně pro ruční zadávání dat (tj. Volno z důvodu nemoci, Řádná dovolená, Práce mimo působnost a Ostatní).

Pokud by uživatel chtěl doplnit všechny nespojité aktivity stejnou činností, stačí vybrat kořenový prvek stromu „Nevyplněné aktivity“ a zvolit si z menu požadovanou činnost. Po výběru se všechny jednotlivé podstromy nespojitostí doplní požadovanou činností. V případě, že v aktuálních datech neexistuje žádná nespojitost, dojde při vyvolání tohoto okna k výpisu „Nejsou nalezeny žádné nespojité aktivity“.

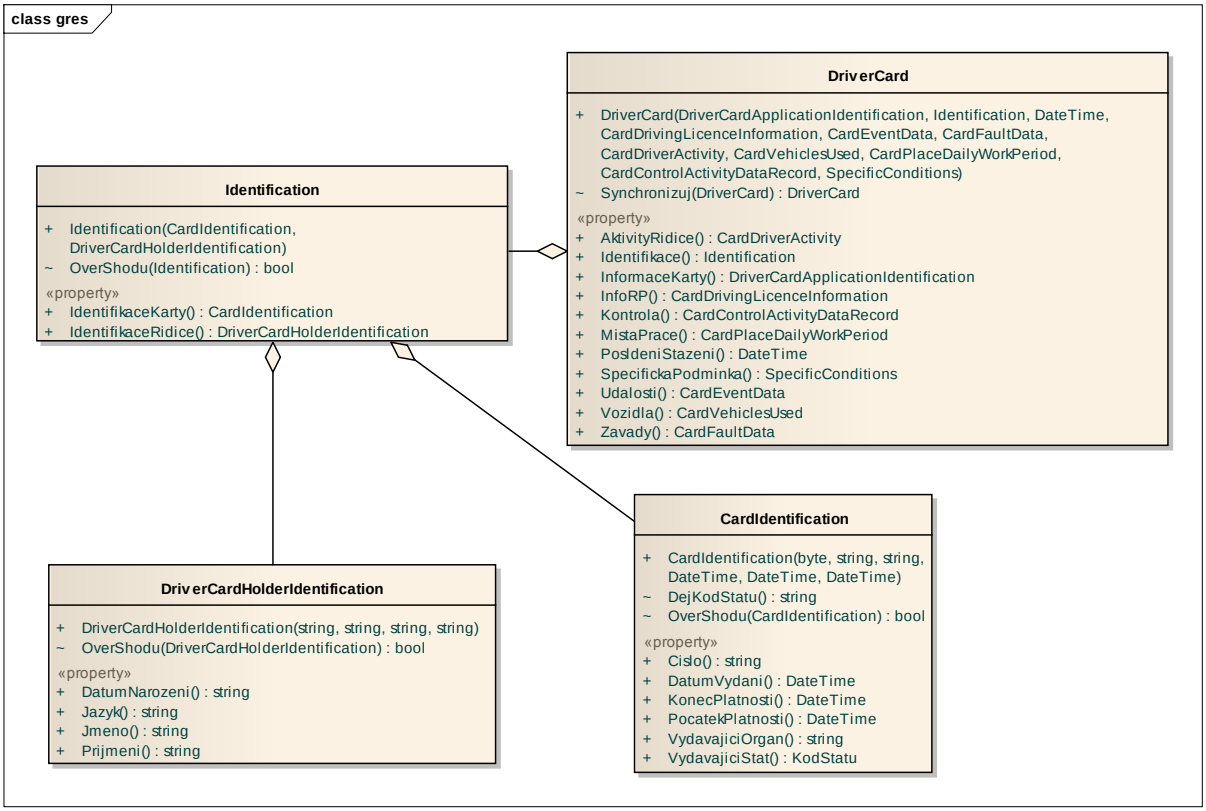


Obr 33 – Okno Kontrola spojitosti dat

5.2 Použité třídy

V této kapitole budou popsány základní třídy využívané v programu a to zejména třídy pro uchování načtené struktury dat z karty řidiče a dále pak třídy, pomocí nichž lze data z čipové karty získat.

5.2.1 Třídy ukládající strukturu dat z karty



Obr 34 – Diagram tříd 1

5.2.1.1 DriverCard

Třída DriverCard je základní třída uchovávající celou strukturu dat, která se stahuje z karty řidiče. Třída obsahuje jednotlivé soubory respektive datové struktury ze souborů z karty. Všechny tyto struktury jsou uloženy ve speciálních třídách, které budou podrobně rozebrány v následujícím textu. DriverCard obsahuje kromě konstruktoru, který inicializuje jednotlivé datové složky (v tomto případě jednotlivé vlastnosti třídy) pouze jednu metodu:

- *Synchronizuj* – zde nejprve dojde k ověření základních identifikačních údajů a pokud jsou tyto údaje shodné jako již načtená data z karty (případně ze souboru) dojde k postupnému volání jednotlivých synchronizačních metod daných tříd

5.2.1.2 DriverCardHolderIdentification

Třída zapouzdřuje základní informace o držiteli karty. Mezi tyto informace patří: jméno a příjmení, dále pak datum narození a jazyk, kterým obvykle držitel karty hovoří. Stejně jako u předchozí třídy i tato obsahuje krom konstruktoru pouze jednu metodu:

- *OverShodu* – k volání této metody dochází z hlavní třídy Identification a její funkcí je ověřit shodnost obsažených dat s daty, se kterými se mají sesynchronizovat

5.2.1.3 CardIdentification

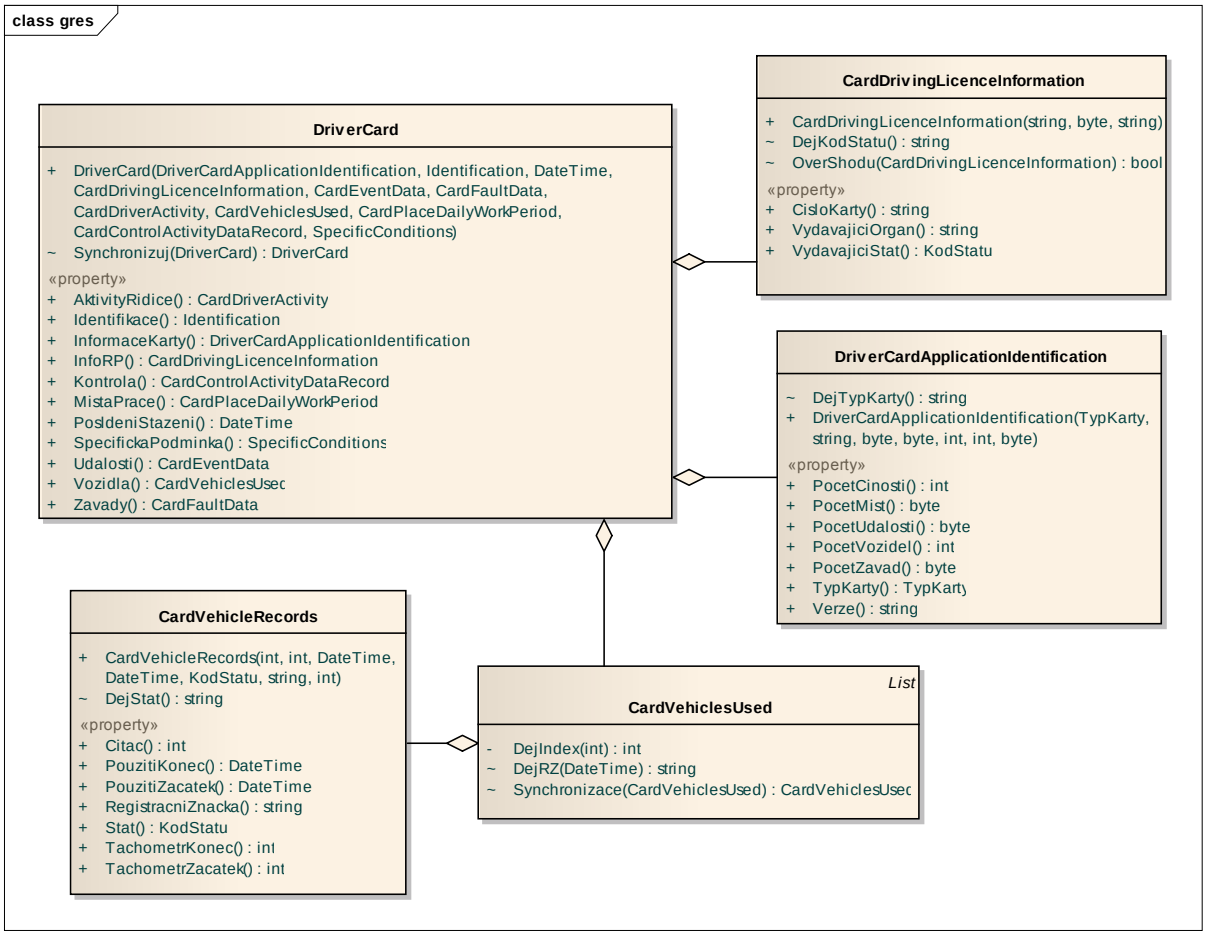
V této třídě jsou zapouzdřena data tykající se vydání karty. V jednotlivých vlastnostech se tak mohou uložit data o čísle karty, datu vydání, konci platnosti, počátku platnosti, vydávajícím orgánu a státu. Krom těchto vlastností obsahuje třída ještě dvě metody:

- *OverShodu* – stejně jako u předchozí třídy slouží k ověření, které je potřebné pro synchronizaci dat
- *DejKodStatu* – ze zadaného kódu státu vrací celý jeho název

5.2.1.4 Identification

Zde jsou zapouzdřeny dvě výše zmíněné třídy a dohromady tak tvoří samostatně oddělenou část identifikující držitele karty i informace o vydání této karty. Třída obsahuje pouze jednu metodu:

- *OverShodu* – metoda postupně volá metody OverShodu svých podtříd. Na základě vrácených informací od těchto metod určuje zda-li se bude v synchronizaci pokračovat, nebo skončí chybovým hlášením o neshodnosti synchronizovaných dat



Obr 35 – Diagram tříd 2

5.2.1.5 CardDrivingLicenceInformation

Třída obsahuje tyto informace o řidičském oprávnění: číslo řidičského průkazu, vydávající stát a orgán v daném státu. Metody jsou shodné jako u CardIdentification a mají i stejnou funkčnost.

5.2.1.6 DriverCardApplicationIdentification

V této třídě jsou zapouzdřeny informace o samotné kartě a její struktuře a dále pak jedna metoda:

- *TypKarty* – určuje, o jakou čipovou kartu se jedná (karta řidiče, karta dílny, …)
- *Verze* – jaká je na kartě použita verze datových struktur
- *Počet…* – celkem je v této třídě uloženo pět informací o maximálních počtech jednotlivých struktur, které karta může obsahovat (počet činností řidiče, počet navštívených míst, počet událostí a závad od každého druhu a počet vozidel)
- Metoda *DejTypKarty* – ze zadaného kódu určí, o jaký typ karty se jedná

5.2.1.7 CardVehicleRecords

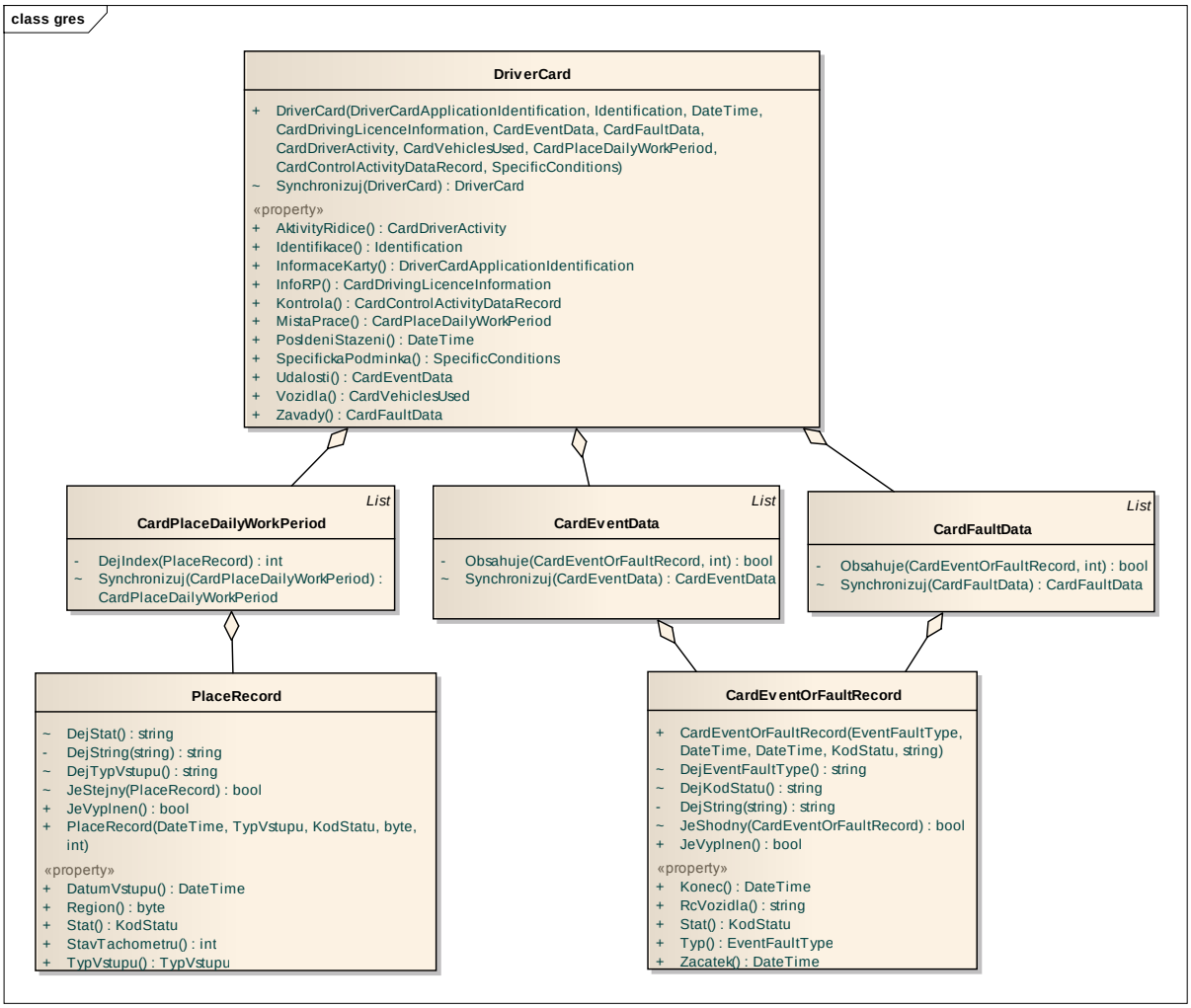
V této třídě jsou uchovávány informace o jednotlivých vozidlech, které řidič používal ke své činnosti. Jedná se o tyto informace:

- *Citac* – jednoznačně určuje číslo záznamu
- *RegistracniZnacka* – určuje registrační značku vozidla, se kterým řidič vykonával svou práci
- *Stat* – je název státu, ve kterém je vozidlo registrováno
- *PouzitiZacatek* – je datum a čas, kdy do tachografu vozidla vložil řidič svou kartu
- *PouzitiKonec* – je datum a čas, kdy svou kartu řidič vyjmul z tachografu vozidla
- *TachometrZacatek* – určuje počáteční stav tachometru při vložení karty
- *TachometrKonec* – určuje konečný stav tachometru při vyjmutí karty

5.2.1.8 CardVehiclesUsed

Třída je odvozená od generického seznamu používaného v jazyce C#. Jako prvky tohoto seznamu jsou použity instance třídy CardVehicleRecords. Jednotlivé metody, které jsou v této třídě implementovány, se poté provádí nad generickým seznamem. Metody jsou následující:

- *DejIndex* – podle zadaného čítače se prohledává seznam a dojde-li k nalezení shodného čítače, vrátí se index, na kterém se tento prvek nachází
- *DejRZ* – metoda se snaží najít v seznamu registrační značku vozidla
- *Synchronizace* – v této metodě se sesynchronizují data načtená z karty a ze souboru do jednoho výsledného seznamu používaných vozidel



Obr 36 – Diagram tříd 3

5.2.1.9 PlaceRecord

Tato třída uchovává záznam o místu vykonávané práce řidiče. Umožňuje uchovat následující informace:

- *TypVstupu* – určuje, o jakou činnost s kartou se jedná (většinou zasunutí a vyjmutí karty)
- *DatumVstupu* – datum a čas, který se vztahuje k typu vstupu
- *StavTachometru* – stav tachometru vztahující se k datu a typu vstupu
- *Stat* – určuje, ve kterém státě se řidič nachází
- *Region* – specifikuje region v daném státě

Dále jsou v této třídě následující metody:

- *DejStat* – podle kódu státu vrací celý název státu
- *DejTypVstupu* – podle kódu vstupu vrací informace, o jaký vstup se jedna
- *JeStejnyPlaceRecord* – porovnaní dvou PlaceRecordu zda-li jsou totožné
- *JeVyplnen* – slouží při načítání (dekódování) dat z karty, aby nedošlo k vložení prázdných záznamů

5.2.1.10 CardPlaceDailyWorkPeriod

Podobně jako CardVehiclesUsed je i tato třída odvozena od generického seznamu. Jako prvky seznamu jsou instance třídy PlaceRecord. Třída obsahuje dvě metody pracující s tímto seznamem:

- *DejIndex* – metoda zjišťuje, zda-li se v seznamu vyskytuje zadaný prvek a pokud ano vrací jeho index
- *Synchronizuj* – slouží k synchronizaci dvou tříd CardPlaceDailyWorkPeriod (resp. dvou jejich seznamů). K tomu se využívá metoda JeStejnyPlaceRecord, která zabraňuje vložení dvou stejných záznamů do výsledného seznamu

5.2.1.11 CardEventOrFaultRecord

Tato třída zapouzdřuje základní informace o událostech nebo závadách. Jak události tak i závady obsahují stejná základní data a proto jsou ukládány pomocí této jedné třídy. Vlastností třídy jsou tyto:

- *Typ* – určuje o jaký typ události nebo závady se jedná
- *Zacatek* – jednoznačně určuje datum a čas od kdy byla závada nebo událost zjištěna
- *Konec* – jednoznačně určuje datum a čas od kdy byla závada nebo událost odstraněna (ukončena)
- *Stat* – je kód státu, ve kterém se událost/závada vyskytla
- *RCVozidla* – je registrační značka vozidla, na kterém se událost/závada vyskytla

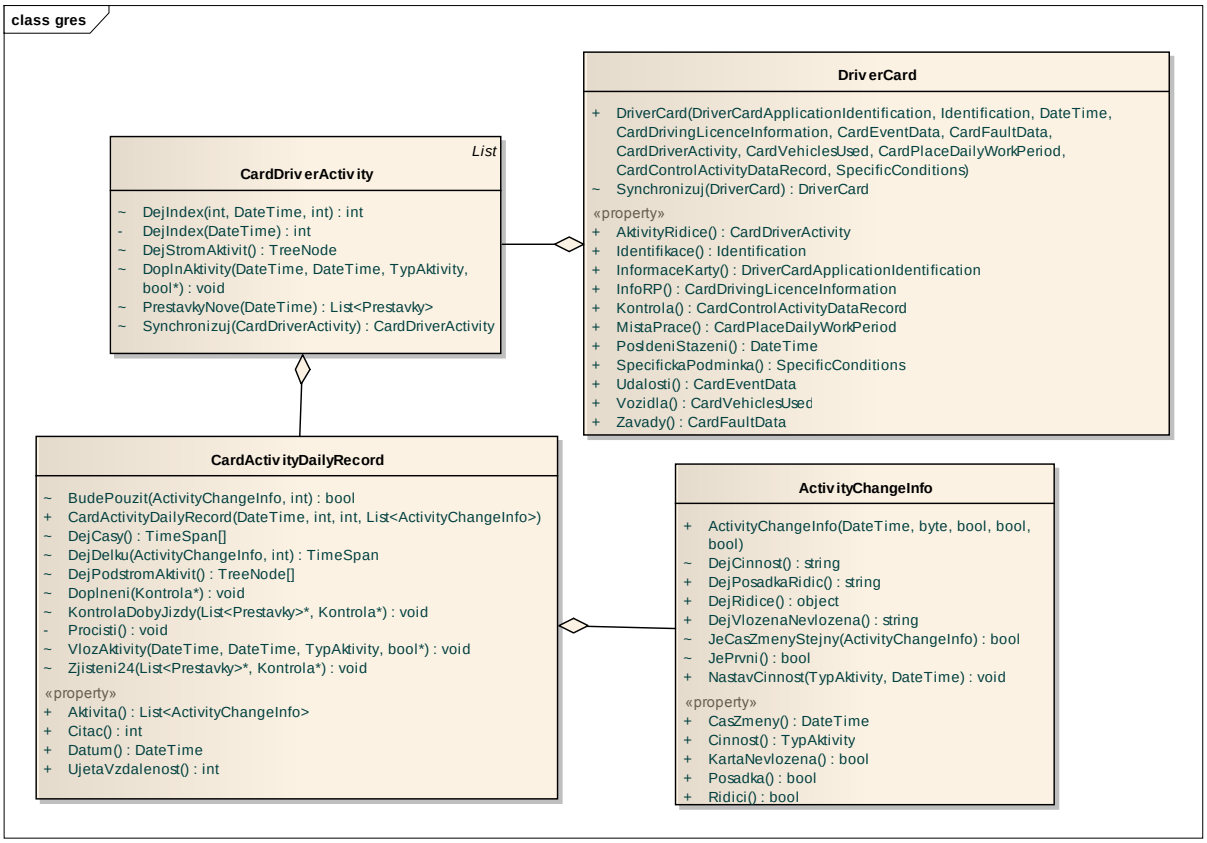
Metody, které jsou v třídě implementovány:

- *DejEventFaultTyp* a *DejKodStatu* – první metoda vrací na základě kódu událost/závadu a druhá celý název státu
- *JeShodny* a *JeVyplněn* – tyto metody mají stejnou činnost jako metody JeStejnyPlaceRecord a JeVyplnen u třídy PlaceRecord

5.2.1.12 CardEventData a CardFaultData

Tyto dvě třídy jsou téměř totožné. Obě jsou odvozené od generického seznamu. Třída CardEventData v sobě uchovává seznam událostí a CardFaultData seznam závad. Třídy mají také shodné metody se stejnou funčností:

- *Obsahuje* – metoda slouží k porovnání dvou událostí/závad. Využívá se opět při synchronizaci, aby se vyloučila možnost vytvoření výsledného seznamu, který by obsahoval dva stejné záznamy
- *Synchronizuj* – metoda opět slouží k synchronizaci jednotlivých seznamů událostí/závad



Obr 37 – Diagram tříd 4

5.2.1.13 ActivityChangeInfo

Třída ActivityChangeInfo v sobě umožňuje uchovat jednu činnost řidiče, která je zaznamenána na kartě. Každá tato činnost obsahuje informace:

- *CasZmeny* – určuje čas změny, kdy došlo k zaznamenání aktivity (činnosti) řidiče
- *Cinnost* – v sobě nese kód činnosti, která nastala v nějakém čase (práce, řízení …)
- *KartaNevlozena* – pokud karta při činnosti nebyla vložena v tachografu, je zde uložena 1 a pokud karta byla vložena, je zde uložena 0
- *Posadka* – zde je uložena informace o tom zda řídí samotný řidič nebo posádka
- *Ridic* – v této datové složce se uchovává informace o tom, ve kterém slotu byla karta zasunuta v době činnosti

Třída ActivityChangeInfo obsahuje i následující metody:

- *DejCinnost* – na základě kódu určující činnost řidiče vrací metoda slovní popis, o jakou činnost se jedná
- *DejPosadkaRidic* – na základě vlastnosti Posadka a KartaNevlozena vrací buď Řidič/Posádka nebo Známý/Neznámý
- *DejRidice* – metoda vrací slovní popis, ve kterém slotu se karta nacházela při zápisu činnosti na kartu
- *DejVlozenaNevlozena* – vrací popis Ano/Ne podle toho jestli byla karta vložena nebo nikoliv
- *JeCasZmenyStejny* – porovnává dvě aktivity a zjišťuje zda-li mají stejný čas změny

- *JePrvni* – metoda ověřuje zda-li je aktivita první (tachografem vložený) záznam, který má čas změny 00:00:00 a karta nebyla vložena. Na základě tohoto zjištění se poté zobrazí nebo nezobrazí tento záznam
- *NastavCinnost* – metoda sloužící k nastavení kódu činnost a času změny při ručním vkládání dat

5.2.1.14 CardActivityDailyRecord

Tato třída v sobě implementuje tyto vlastnosti:

- *Aktivita* – je generický seznam obsahující jako své prvky instance třídy ActivityChangeInfo, který v sobě uchovává jednotlivé aktivity za jeden den
- *Citac* – jednoznačně určuje, kolik záznamů karta již zaznamenala od začátku jejího používání
- *Datum* – je datum dne, pro který je uložen seznam aktivit
- *UjetaVzdalenost* – určuje ujetou vzdálenost za konkrétní den

Metody:

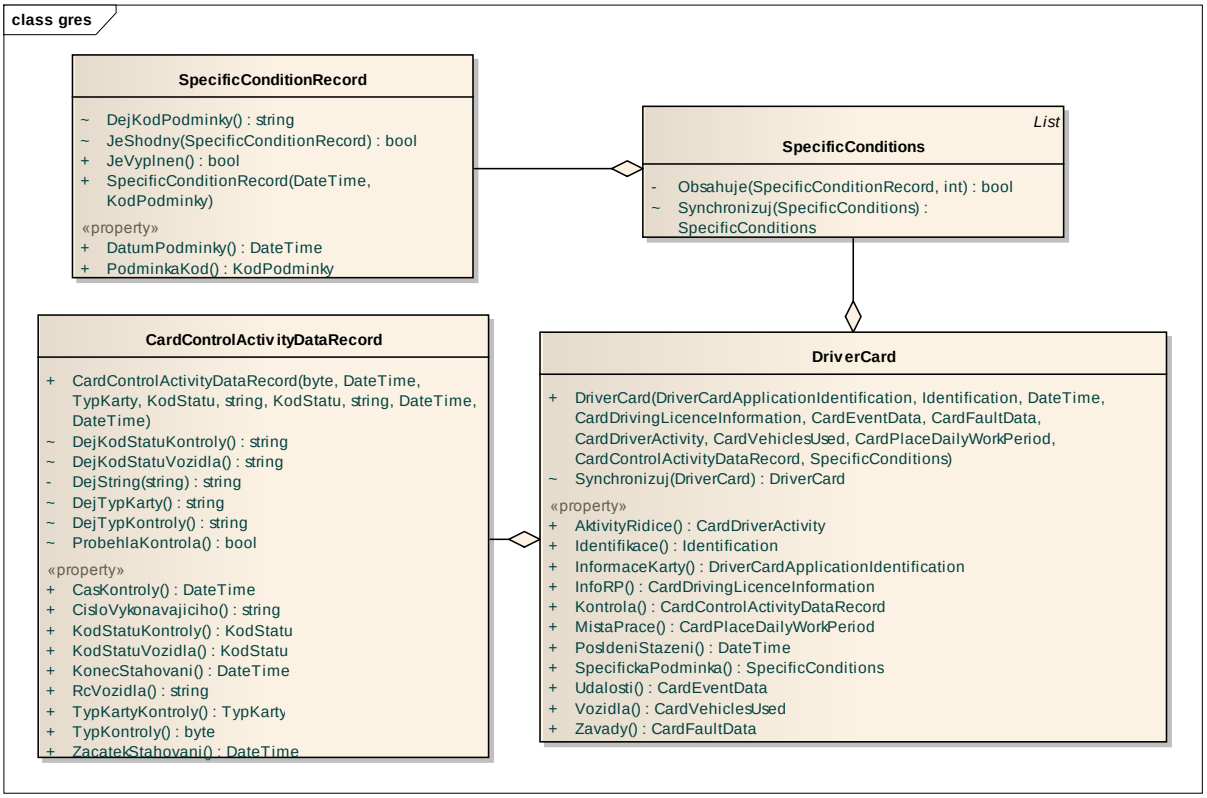
- *BudePouzit* – metoda zjišťuje zda bude aktuální aktivita v seznamu použita při výpisu
- *DejDelku* – metoda vrací délku aktuální aktivity
- *VlozAktivity* – při ručním doplňování dat se pomocí této metody doplní do seznamu aktivity se zvolenou činností a to v požadovaném časovém rozmezí
- *Procisti* – jedná se o pomocnou metodu, která v případě pozdější změny ručně zadaných dat pročistí seznam aktivit, aby neobsahoval předchozí vložené aktivity
- *DejCasy* – se využívá pro zjištění dob jednotlivých činností v aktuálním dni. Tyto doby se používají pro informativní výpis jednotlivých dnů
- *DejPodstromAktivit* – zde se sestavuje podstrom nevyplněných (nezadaných) aktivit, které tvoří nespojitost v aktuálním dni
- *KontrolaDobyJizdy* – zjišťuje, zda nedošlo k porušení doby řízení delší než 4,5 hodiny bez přestávky
- *Zjisteni24* – tato metoda rozděluje načtené aktivity do jednotlivých 24hodinových období, které se později používají na určení jednotlivých týdnů
- *Doplneni* – k nalezeným 24hodinových obdobím doplňuje tato metoda celkovou dobu řízení a zda během tohoto řízení byla nalezena přestávka alespoň tři hodiny

5.2.1.15 CardDriverActivity

CardDriverActivity je třída, která je opět odvozená od generického seznamu, jejíž prvky jsou jednotlivé instance třídy CardActivityDailyRecord, neboli jednotlivé denní záznamy aktivit řidiče. V této třídě jsou zapouzdřeny následující metody:

- *DejIndex* – porovnává zadaný prvek s prvky seznamu a pokud jsou shodné, vrací index, na kterém se prvek nachází v tomto seznamu
- *Synchronizuj* – slouží pro synchronizaci dvou seznamů denních aktivit (jednoho načteného z karty a druhého ze souboru). Pokud oba neobsahují totožná data, vytvoří se nový seznam, do kterého se sloučí prvky z obou seznamů tak, aby byly správně seřazeny podle času

- *DoplnAktivity* – při ručním doplňování dat jsou vybrány dva časové body, mezi kterými se tato metoda snaží doplnit do prázdných (nespojitých) míst jednotlivé aktivity
- *Prestavky* – v této metodě dochází k procházení seznamu aktivit a na základě jejich vyhodnocení se zjišťují chyby v dodržování přestávek
- *DejStromAktivit* – na základě volání metody DejPodstromAktivit vytváří strom nespojitých dat



Obr 38 – Diagram tříd 5

5.2.1.16 CardControlActivityDataRecord

Pomocí této třídy se v programu uchovává poslední kontrola, která byla provedena na vozidle. Třída obsahuje jednotlivé vlastnosti, které uchovávají následující informace: datum a čas, kdy byla kontrola provedena, číslo karty toho, kdo kontrolu vykonal, kód státu kde byla kontrola provedena, registrační značka vozidla a stát, kde je vozidlo registrováno, o jaký typ kontroly se jednalo a pokud při ní došlo k stahování dat. Metody, které tato třída obsahuje, jsou pouze takové, které vracejí jednotlivé kódy a typy související s kontrolou. V tomto textu nebudou dále rozepisovány, protože jsou víceméně totožné jako metody popisované u předchozích tříd.

5.2.1.17 SpecificConditionRecord

Je třída, která se používá pro uchování jednoho druhu specifické podmínky, která je spojena s přepravou (např. převoz lodí/vlakem). Pro uchování informací se používají pouze dvě vlastnosti a to:

- *DatumPodminky* – určuje datum a čas, kdy došlo k začátku přepravy za specifických podmínek
- *PodminkaKod* – kód, který jednoznačně určuje, o jakou specifickou podmínku se jedná

SpecificConditionRecord dále obsahuje celkem tři metody:

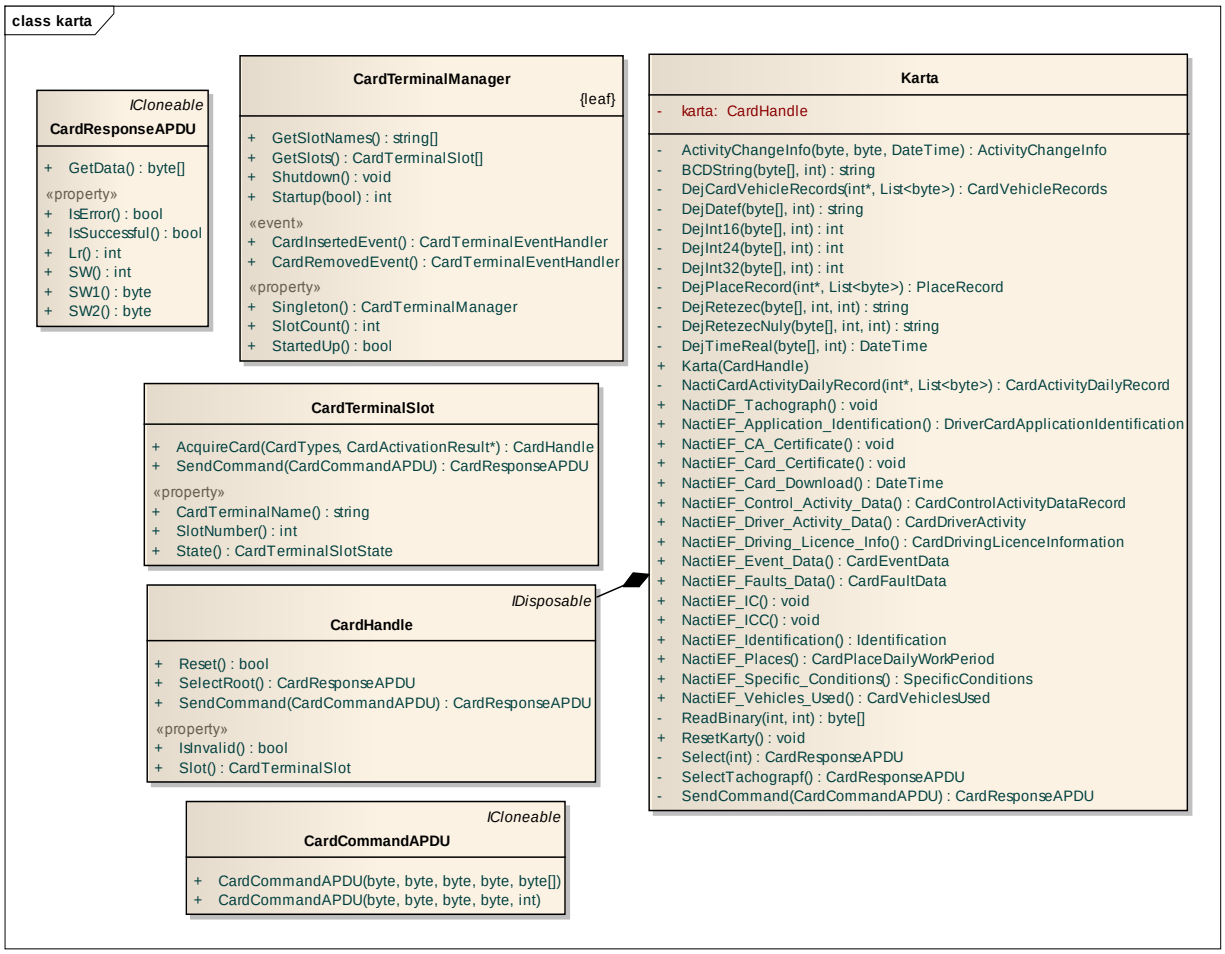
- *JeVyplnen* – kontroluje, zda je instance této třídy vyplněna. Tím se zabraňuje vložení prázdného (nevyplněného) prvku do seznamu
- *DejKodPodminky* – podle kódu podmínky vrací jeho slovní popis
- *JeShodny* – používá se při synchronizaci dat, aby se vyloučily shodné prvky v seznamu

5.2.1.18 SpecificConditions

Tvoří generický seznam SpecificConditionRecordů, které tvoří jednotlivé jeho prvky. Třída obsahuje pouze dvě metody: první se používá k synchronizaci a je podobná výše popsaným synchronizačním metodám a dále pak metodu, která vylučuje při synchronizaci shodné prvky.

5.2.2 Třídy sloužící ke komunikaci s kartou

Pro komunikaci s čtečkou čipových karet je v programu použito pět tříd z knihovny Subsembly SmartCard API (Professional), která je volně stažitelná (pro nekomerční použití) z internetových stránek www.smartcard-api.com. Tato knihovna obsahuje přístup k čtečkám čipových karet pomocí rozhraní PC/SC Workgroup API (WinSCard), které je nativně implementováno v operačním systému Windows a je také používáno v programu DITA. Dále knihovna obsahuje německé standardizované rozhraní CT-API, které má oproti předchozímu několik výhod a obsahuje některé pokročilé funkce jako např.: práce s několika sloty a rozšířené operace s PIN kódy. Další důvodem zvolení této knihovny je plné využití a podpora platformy .NET a jazyka C#.



Obr 39 – Diagram tříd 6

Na předchozím obrázku je znázorněno celkem šest tříd. Třídy zobrazeny vlevo jsou všechny součástí knihovny Subsembly.SmartCard (tedy všechny krom třídy Karta). Tyto třídy jsou celkem dosti rozsáhlé a z toho důvodu jsou zde uvedeny pouze základní metody, které jsou v programu používány.

5.2.2.1 CardTerminalManager

Tato třída je jednou ze základních a její funkcí je monitorování, konfigurování a spravování jednotlivých terminálů čtecích zařízení, které se připojují k počítači, nebo případně odpojují. Tato třída je založena na návrhovém vzoru singleton. Singleton (česky jedináček nebo unikát) představuje řešení problému, kdy v celém programu má běžet pouze jediná instance nějaké třídy (pouze 1 objekt dané třídy). V programu jsou z této třídy používány následující náležitosti:

1. Vlastnosti:
    - *Singleton* – tato vlastnost vrací instanci této třídy. Při volání vlastnosti mohou nastat dvě možnosti. První: před voláním vlastnosti ještě instance třídy neexistuje, a proto dojde k jejímu vytvoření a následnému vrácení. Druhá možnost: instance třídy již byla vytvořena, a proto dojde k vrácení této již dříve vytvořené instance
    - *SlotCount* – vrací celkový počet připojených slotů, které jsou obsaženy ve všech čtecích terminálech (čtečkách)
    - *StartedUp* – pokud již je vytvořena instance singletona je vraceno *true* v opačném případě *false*
2. Události:
    - *CardInsertedEvent* – je událost reagující na vložení čipové karty do čtecího zařízení. K této události je možné „připojit“ metodu, která se vykoná po vložení karty
    - *CardRemovedEvent* – je událost reagující na vyjmutí čipové karty ze čtecího zařízení. Stejně jako u předchozí i zde je možné spřáhnout tuto událost s určitou metodou, která se provede po vyjmutí karty
3. Metody:
    - *Startup* – slouží k vytvoření (spuštění) singletonu a zároveň vrací počet čtecích zařízení připojených k počítači
    - *Shutdown* – slouží k ukončení (vypnutí) singletonu
    - *GetSlots* – vrací pole připojených čtecích zařízení (terminálů), které jsou interpretovány třídou CardTerminalSlot
    - *GetSlotNames* – vrací pole názvů jednotlivých připojených čtecích zařízení

5.2.2.2 CardTerminalSlot

CardTerminalSlot je třída interpretující jedno čtecí zařízení, které je připojeno k počítači. V programu jsou využívány následující tři vlastnosti a dvě metody:

- Vlastnost *CardTerminalName* – je název čtecího zařízení (většinou je složen z označení výrobce a typu)
- Vlastnost *SlotNumber* – označuje číslo slotu, které má čtecí zařízení
- Vlastnost *State* – označuje stav, ve kterém se čtečka nachází (vloženo, prázdný)

- Metoda *AcquireCard* – je nejdůležitější metoda této třídy, která na základě ověřování typu karty (např.: procesorová karta) vrací instanci třídy CardHandle
- Metoda *SendCommand* – slouží k zasílání APDU příkazů a vrací odpověď na příkaz

5.2.2.3 CardHandle

Jedná se o jednu z nejdůležitějších tříd, která má na starosti všechny skutečné přístupy k čipové kartě. V rámci této třídy je možné zasílání příkazů, ověřování PIN kódů, stahování a ukládání dat z karty a na kartu. Třída obsahuje velké množství jednotlivých metod a spousta z nich je přetížená což znamená, že mají stejné názvy, ale liší se počty nebo typy vstupních parametrů. Mezi základní vlastnosti a metody, používané v rámci této práce, patří:

- Vlastnost *IsInvalid* – určuje, zda-li je přístup ke kartě platný či nikoliv
- Vlastnost *Slot* – na základě této vlastnosti lze zjistit, ke kterému terminálu (čtecímu zařízení) se CardHandle vztahuje
- Metoda *Reset* – slouží k resetování karty popisovaný v kapitole 2.2.2 Kontaktní karty
- Metoda *SelectRoot* – metoda sloužící k návratu ke kořenovému souboru karty
- Metoda *SendCommand* – slouží k odesílání příkazu APDU interpretovaných třídou CardCommandAPDU a získávání odpovědí, které jsou interpretované třídou CardResponseAPDU

5.2.2.4 CardCommandAPDU

Jak již bylo zmíněno výše, tato třída zajišťuje nebo spíše vytváří jednotlivé APDU příkazy, které jsou následně posílané na kartu. Podobně jako u předchozích tříd i tato je tvořena velkým počtem metod a vlastností. Pro jednoduchost a použití ve výsledném programu zde budou uvedeny pouze dvě: dva konstruktory, které mají stejný název, ale liší se svými parametry a používají se tak potom k jiným účelům. První z konstruktorů se používá pro výběr elementárních a adresářových souborů a druhý, který obsahuje parametr *Le*, se používá pro binární čtení dat z karty.

5.2.2.5 CardResponseAPDU

Tato třída reprezentuje odpověď na zasílání APDU příkazů. Od předchozí se liší tím, že v programu není vytvářena pomocí konstruktoru, ale vždy je výsledkem metody SendCommand třídy CardHandle případně CardTerminalSlot. V rámci vytvořeného programu je používáno několik vlastností a jedna metoda.

- Metoda *GetData* – jak již název metody napovídá, účelem je získat data, která byla vrácena kartou při zaslání příkazu pro čtení dat
- *IsError* – vlastnost, která informuje zda-li odpověď skončila chybovým stavem
- *IsSuccessful* – naopak tato vlastnost informuje, zda byl odeslaný příkaz úspěšný
- *Lr* – v případě, že jsou požadována data z karty, tato vlastnost určuje jejich délku
- *SW* – je dvou bajtové hlášení o stavu zpracování. SW je složeno z SW1 a SW2. které mají každé také své vlastnosti a dohromady informují o případných chybových stavech

5.2.2.6 Karta

Podobně jako CardHandle i tato třída je jednou z nejdůležitějších. Pomocí této třídy je možné sestavovat jednotlivé APDU příkazy a vyhodnocovat odpovědi na ně. Dále pak umožňuje načítání jednotlivých souborů z karty a dekódovat jednotlivé struktury dat, které jsou v těchto souborech uloženy. Karta obsahuje pouze jednu datovou složku a tou je instance třídy

CardHandle, pomocí které je možné zasílání jednotlivých příkazů a získávání odpovědí. Dále pak třída Karta obsahuje poměrně velký počet metod, které lze rozdělit do třech okruhů podle jejich využívání a funkčnosti:

1. Metody sloužící ke komunikaci s kartou:
    - *SendCommand* – slouží k odesílání příkazů APDU voláním příslušné metody z třídy CardHandle
    - *Select* – tato metoda slouží k sestavování jednotlivých příkazů, které se používají pro zpřístupnění elementárních souborů v aktuálním adresáři na kartě. Vstupním parametrem je *ID*, které jednoznačně identifikuje zpřístupňovaný soubor
    - *SelectTachograph* – podobně jako předchozí i tato metoda slouží k vytváření APDU příkazu pro přístup k souboru na kartě, ale tentokrát je možné zpřístupnit pouze jediný soubor a to DF_Tachograph, ke kterému se přistupuje pomocí jeho názvu a ne pomocí *ID*
    - *ReadBinary* – jedná se o metodu, která vytváří jednotlivé příkazy pro čtení binárních dat z již předtím zpřístupněných souborů na kartě řidiče. Jako vstupní parametry jsou: offset, který určuje paměťové místo v transparentním souboru, odkud se budou data číst a dále pak délka, která určuje velikost (délku) načítaných dat
2. Metody sloužící k načítání jednotlivých souborů z karty:
    - *NactiEF_xxx* – jsou metody sloužící k načítání a dekódování jednotlivých souborů z karty řidiče (xxx je vždy název načítaného souboru z karty řidiče, které jsou popsány v kapitole 4.2.3). V těchto metodách se vždy nejprve zavolá metoda pro zpřístupnění souboru, dále pak jednou či vícekrát metoda pro binární čtení dat ze souboru. To, kolikrát se bude metoda pro čtení dat volat, vždy záleží na velikosti daného souboru. Soubory mající velikost do 255 bajtů je možné načíst jedním příkazem, ale větší už nikoliv a proto je potřeba metodu volat v cyklu. Po správném načtení celého souboru přichází na řadu dekódování vnitřních struktur, které se poté uloží do struktur programu popsané v kapitole 5.2.1 a tyto pak metody NactiEF_xxx vrací
    - *NactiDF_Tachograph* – metoda pouze zpřístupňuje složku Tachograph a případně reaguje na vzniklé chyby
    - *ResetKarty* – slouží k zaslání RTS signálu na kartu, díky kterému je pak možné opět přistoupit k souborům v kořenové složce
3. Metody dekódující datové struktury:
    - Většina těchto metod slouží pro dekódování takových datových struktur, které jsou na kartě uloženy ve formě jednotlivých záznamů a dohromady tvoří nějaký seznam anebo se jedná o struktury, které jsou často používány a jsou obsaženy téměř ve všech souborech. Mezi takové často používané struktury paří například uložený datum včetně času, dále pak jedno, dvou a tří bajtové celočíselné typy apod.

5.2.3 Ostatní a pomocné třídy

Jelikož je jazyk C# čistě objektově orientovaný jazyk, vše musí být realizováno pomocí tříd. Výše popsané třídy jsou tedy pouze základní, které tvoří datovou strukturu a přístup k čipovým kartám. Výsledný program samozřejmě obsahuje celou řadu dalších tříd, které se používají zejména pro jednotlivé formuláře a jejich činnosti by bylo možné popsat následujícím způsobem:

- Reakce na podněty uživatele – obsluhování jednotlivých stisknutí tlačítek na klávesnici, reakce na položky menu, tlačítek a dalších komponent, zvětšování/zmenšování jednotlivých oken apod.
- Zobrazování (vizualizace) jednotlivých dat potřebných pro informování uživatele, zobrazení chybových hlášení a případné reakce na ně
- Pomocné výpočty a provádění určitých úloh spojených se základními daty
- Umožnění zadávání vstupů od uživatele pro fungování programu

5.3 Použité algoritmy

Algoritmy, které jsou v programu implementovány, slouží ke kontrole činností řidiče podle nařízení č. 561/2006 popisované v kapitole 4.1. Z důvodu optimalizace jsou některé algoritmy vykonávány zároveň a tím dochází k úspoře délky provádění kontrol. Popisované algoritmy jsou relativně složité, a proto zde budou uvedeny vždy jen nejpodstatnější kroky, které dostatečně vysvětlí jejich princip. Všechny níže popisované algoritmy pracují s takzvaným „plovoucím" obdobím. To znamená, že například týden není pevně vymezen časovým rozmezím pondělí 00:00:00 až neděle 23:59:59.

5.3.1 Kontrola dob řízení a přestávek

Tento algoritmus slouží ke kontrole zda-li řidič dodržuje povinné přestávky v řízení a nepřekračuje povolenou dobu řízení 4,5 hodiny. Před samotným spuštěním algoritmu dojde v programu ke sloučení jednotlivých denních seznamů aktivit do jednoho, který obsahuje všechny aktivity řidiče, a teprve nad ním se vykoná tento algoritmus:

1) Postupný průchod seznamem aktivit řidiče a sčítání dob řízení
2) Detekována přestávka:
   a) Přestávka je kratší než 15 minut – pokračování krokem 1
   b) Přestávka je delší než 15 minut a dosud není zaznamenáno „nalezena 15minutová přestávka" – zaznamenání „nalezena 15minutová přestávka" a pokračování krokem 1
   c) Přestávka je delší než 30 minut – „nalezena 15minutová přestávka"?
      - Ano – přesun na krok 3
      - Ne – záznam „nalezena 15minutová přestávka" a pokračování krokem 1
   d) Přestávka je delší než 45 minut – pokračování krok 3
3) Překračuje sčítaná doba řízení 4 hodiny a 30 minut?
   a) Ano – výpis o porušení – vynulování sčítané hodnoty, pokračování krokem 1
   b) Ne – vynulování sčítané hodnoty, pokračování krokem 1

5.3.2 Sestavení 24hodinových období

Podobně jako předchozí algoritmus tak i tento je prováděn nad seznamem všech aktivit řidiče. Nejjednodušeji by šel tento algoritmus popsat takto: nejprve se nalezne první práce/řízení. K tomuto času se přičte 24 hodin a v období mezi těmito dvěma časy se pak hledá nejdelší možná přestávka/nespojitost vcelku. Po nalezení takovéto přestávky se pak na základě jejího konce určí, kde bude končit první 24hod. období a kde bude začínat následující. V ideálním případě se nejdelší přestávka vcelku nachází na konci takového 24hod. období a následující

začíná až po této přestávce. Bohužel se praxi často vyskytuje situace, kdy ještě před koncem prvního 24hod. období již začíná druhé. V takovém případě začátek nejdelší přestávky označuje konec prvního 24hod. období a konec nejdelší přestávky označuje začátek následujícího 24hod. období, ke kterému se opět přičte 24 hodin a postup se opakuje. Tento algoritmus by se dal velice zjednodušeně popsat následujícími kroky:

1) Postupný průchod aktivitami, u každé aktivity přechod na krok 2, 3 nebo 4 v závislosti na aktuální činnosti
2) Detekce práce nebo řízení: byla již dříve nalezena práce/řízení?
   a) Ano – zaznamenání „možný konec období“, pokračování krokem 1
   b) Ne – zaznamenání „začátek období“ a vypočtení „konec období“ („konec období“ je vypočten jako „začátek období“ + 24 hodin), pokračování krokem 1
3) Detekce přestávky/nespojitosti: je nastaven „začátek období“?
   a) Ano – zjištění všech po sobě jdoucích přestávek/nespojitostí. Bylo již dříve nalezeno takovéto období?
      - Ano – porovnání s dříve nalezeným. Je toto období delší?
        - Ano – zaznamenání začátku, konce a délky aktuálně nalezeného období. Pokračování krokem 1
        - Ne – začátek, konec a délka období přestávek/nespojitostí zůstane nezměněna. Pokračování krokem 1
      - Ne – uložení začátku, konce a délky aktuálně nalezeného období. Pokračování krokem 1
   b) Ne – Pokračování krokem 1
4) Detekce překročení „konec období“: bylo nalezeno nejdelší období přestávek/nespojitostí na konci 24hod. cyklu?
   a) Ano – začátek a konec 24hod. období je v rozmezí „začátek období“ a „možný konec“. Vymazání všech nastavených hodnot. Pokračování krokem 1
   b) Ne – následující 24hod. období začalo ještě před skončením aktuálního 24hod. období. (tj. první 24hod. období je dáno „začátek období“ a „začátek období přestávek/nespojitostí“). Přednastavení zaznamenaných hodnot na nové, které označují následující 24hod. období. Pokračování krokem 1

5.3.3 Sestavení týdenních období

Na základě předchozího algoritmu vznikne seznam 24hodinových období. Každý prvek tohoto seznamu si s sebou nese informace o začátku a konci tohoto období, délce řízení a zda se v tomto období vyskytuje přestávka přesahující tři hodiny. Tento algoritmus má za úkol analyzováním prvků tohoto seznamu detekovat jednotlivé týdenní období. Jednoduše řečeno se provádí: v seznamu 24hod. období se hledají takové přestávky (mezery mezi jednotlivými 24hod. obdobími), které jsou větší než 45 hodin. Pokud je rozmezí mezi těmito přestávkami menší než týden, došlo k nalezení jednoho týdne. Pokud by však tato délka byla větší, muselo by se toto dvoutýdenní období ještě rozdělit na dva týdny a to tak, že by se v tomto období hledala přestávka větší než 24 hodin, která by se ale nacházela co nejvíce ke středu takovéhoto období. Nalezením takovéto přestávky by byl jednoznačně určen konec prvního a začátek druhého týdne. Zjednodušený algoritmus je možné popsat takto:

1) Postupný průchod seznamem a zjišťování „mezery“ mezi dvěma po sobě jdoucími obdobími přesahující 45 hodin. Pokračování krokem 2
2) Přesahuje od začátku hledání po nalezení 45hodinové přestávky 14 dní?
    a) Ano – oznámení o nedodržení přestávky 45 hodin po 14 dnech – pokračování krokem 2b
    b) Ne – Přesahuje od začátku hledání po nalezení 45 hodin přestávky 7 dní?
        - Ano – průchod tímto obdobím a hledání přestávky delší než 24 hodin, která toto období rozděluje na dvě týdenní období. Pokračování krokem 1
        - Ne – nalezené období odpovídá týdennímu období. Pokračování krokem 1

5.3.4 Kontrola týdenních a dvoutýdenních období

Předchozí algoritmus rozdělil seznam 24hodinových období na jednotlivé týdenní období. Procházením a analyzováním těchto „týdnů“ se v algoritmu zjišťuje následující:

- Kontrola překročení doby jízdy maximálně 9 hodin za den, nebo 10 hodin maximálně dvakrát týdně – zde se u každého 24hod. období v týdnu kontroluje délka jízdy v tomto období a porovnává se s hodnotou 9 resp. 10 hodin. Pokud je doba jízdy delší než 10 hodin, je zaznamenáno porušení doby řízení. Pokud je doba jízdy mezi 9–10 hodin dojde k navýšení čítače o jedna. Na konci týdenního období se porovná čítač s hodnotou 2. Pokud je hodnota čítače větší, došlo k porušení doby řízení více než 9 hodin a čítač minus 2 udává, kolikrát k tomuto porušení došlo. Poté se čítač vynuluje a postup se opakuje u dalšího týdne.
- Kontrola dodržování 11hodinové doby odpočinku (lze rozdělit na 3+9 hodin), respektive 9hodinové doby odpočinku maximálně třikrát za týden – v této části algoritmu dochází k porovnávání konce jednoho 24hod. období a začátku následujícího. Tento začátek a konec určují přestávku mezi těmito obdobími. Pokud je tato přestávka kratší než 9 hodin, je detekováno porušení. Pokud je taková přestávka v rozmezí 9–11 hodin a v průběhu aktuálního 24hod. období není nalezena přestávka v řízení alespoň 3 hodiny, dojde k zvětšení čítače (jiného než v předchozí části) určující kolikrát v týdnu došlo k takovémuto porušení. Na konci aktuálního týdne dojde k porovnání hodnoty čítače, a pokud je hodnota vyšší než 3 dojde k detekování porušení nedodržení přestávky. Zde pak opět dojde k vynulování čítače a tento postup se aplikuje na následující týden.
- Kontrola maximálně šesti 24hodinových období v jednom týdnu – tato část algoritmu je celkem triviální a zde se pouze sčítá počet nalezených 24hod. období v jednom týdnu. Pokud je celkový počet těchto období v týdnu větší než 6 dojde k detekci porušení.
- Kontrola doby řízení za týden, která smí být maximálně 56 hodin – podobně jako předchozí i zde dochází ke kontrole aktuálního týdne, ale tentokráte se sčítají jednotlivé doby jízdy v jednotlivých 24hod. obdobích. Pokud celková doba jízdy v týdnu přesáhne 56 hodin je detekováno porušení.
- Kontrola doby řízení ve zkráceném týdnu, která smí být maximálně 34 hodin – tato část je stejná jako předchozí, jen s tím rozdílem že k sčítání dob řízení dochází v následujícím týdnu než u předchozí části algoritmu. Zde se potom porovnává celková doba řízení s hodnotou 34 hodin a pokud je větší, je detekováno porušení.

- Kontrola doby řízení za dva po sobě jdoucí týdny, která může být maximálně 90 hodin – tato část algoritmu je v sobě obsahuje oba výše popsané kroky. Hodnota doby řízení z prvního týdne se přičte k hodnotě doby řízení v následujícím týdnu a výsledný čas nesmí překročit 90 hodin. V opačném případě dojde opět k detekci porušení.

6 Závěr

Výsledkem této práce je vytvoření fungující aplikace, která obsahuje všechny vlastnosti a funkce, které byly stanoveny v zadání. Analýza dat z karty řidiče podle nařízení č. 561/2006 byla porovnávána s existujícím softwarovým řešením *TAGRA* společnosti *Truck Data Technology, s. r. o.* vyvinutým pro účely kontroly dat a evidence záznamů z digitálních i analogových tachografů. Výsledky, které poskytuje zde popsaná aplikace, se téměř ve všech bodech shodují s výsledky, které poskytuje program TAGRA, jež je na trhu dostupný již delší dobu a jeho výsledky by tak měly být ověřeny v praxi.

Program DITA je v této fázi vývoje vytvořen pouze pro jeden druh čipových karet, kartu řidiče. Možným pokračováním by bylo rozšíření podpory i o ostatní karty digitálního tachografu. Vznikl by tak kompletní softwarový nástroj, který by mohl najít své uplatnění u malých přepravních společností či jednotlivců využívajících digitální tachografy.

Mezi vlastní přínosy při zpracování této diplomové práce patří zejména návrh jednotlivých datových struktur umožňující načítání a ukládání dat z karty řidiče. Dále pak vytvoření všech popsaných algoritmů, které kontrolují dodržování přestávek a řízení řidičů.

Soupis bibliografických citací

[1] ROSOL, Ivo. OKsystem [online]. 2008-2010 [cit. 2010-04-16]. Technologie čipových karet. Dostupné z WWW: <http://www.oksystem.cz/o-nas/servis-pro-novinare/napsali-o-nas/2005/07-business-world>

[2] DE LUXE s.r.o. *PvcCard* [online]. 2008 [cit. 2010-04-16]. Smart karty. Dostupné z WWW: <http://www.pvccard.cz/smart-karty/>

[3] NÁVRAT, Lumír, et al. *Semesrální projekt z Číslicových počítačů II.* [online]. 2008 [cit. 2010-04-16]. Čipové karty. Dostupné z WWW: <http://homel.vsb.cz/~nav79/cipkart/index.htm>

[4] DLOUHÝ, Libor. *Frenkyland* [online]. 1998 [cit. 2010-04-16]. Přístupový identifikační systém s využitím telefonních karet. Dostupné z WWW: <http://frenkyland.silper.cz/projekt.htm>

[5] *Card House* [online]. 2009 [cit. 2010-04-16]. Vše o kartách - karta jako datový nosič. Dostupné z WWW: <http://cardhouse.cz/produkty/plastove-karty/vse-o-kartach/karta-jako-datovy-nosic/>

[6] CHAUHAN, Digvijay. *Devshed* [online]. 2004-10-11 [cit. 2010-04-16]. Smart Cards. Dostupné z WWW: <http://www.devshed.com/c/a/Practices/Smart-Cards-An-Introduction/>

[7] *Sumitdhar* [online]. 2004 [cit. 2010-04-16]. Introduction to Smart Cards . Dostupné z WWW: <http://sumitdhar.blogspot.com/2004/11/introduction-to-smart-cards.html>

[8] EVERETT, David B. *Smart Card News* [online]. 1992 [cit. 2010-04-16]. Smart Card Tutorial. Dostupné z WWW: <http://www.smartcard.co.uk/tutorials/sct-itsc.pdf>

[9] SHILLINGTON, Nicole; WAKER, Travers. *University of Cape Town* [online]. 2000 [cit. 2010-04-16]. The Design of a Smart Card Interface Device. Dostupné z WWW: <http://www.cs.uct.ac.za/Research/DNA/SOCS/paper.html>

[10] Česmad Bohemia. *TRUCK TRADE* [online]. 1. 6. 2006 [cit. 2010-04-25]. Ověřování tachografů. Dostupné z WWW: <http://www.trucktrade.cz/fx/cz/42/tachografy.html>

[11] Česmad Bohemia. *Řidičova knihovna* [online]. [s.l.] : [s.n.], 2008 [cit. 2010-04-25]. Digitální tachograf, s. . Dostupné z WWW: <http://www.ridicovaknihovna.cz/files/nahled1.pdf>

[12] *Brno* [online]. 2007 [cit. 2010-04-25]. Paměťové karty. Dostupné z WWW: <http://www2.brno.cz/index.php?nav01=2226&nav02=2231&nav03=10259&nav04=8887&nav05=8891>

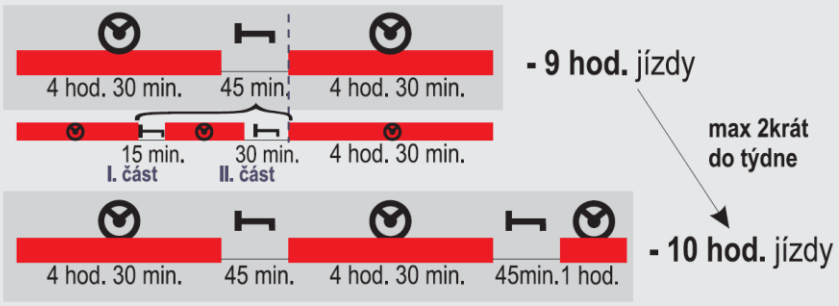
[13] Evropské unie. NAŘÍZENÍ KOMISE (ES) č. 1360/2002. In *ÚŘEDNÍ VĚSTNÍK EVROPSKÝCH SPOLEČENSTVÍ*. 2002, 07, s. 279-530. Dostupný také z WWW: <http://tachospeed.cz/pliki/Narizeni_1360_2002.pdf>

[14] Evropské unie. NAŘÍZENÍ EVROPSKÉHO PARLAMENTU A RADY (ES) č. 561/2006. In *Úřední věstník Evropské unie*. 2006, 102, s. 1-13. Dostupný také z WWW: http://tachospeed.cz/pliki/Narizeni_561_2006.pdf

Přílohy

Nařízení 561/2006

DOBA ŘÍZENÍ A PŘESTÁVEK - 1 ŘIDIČ



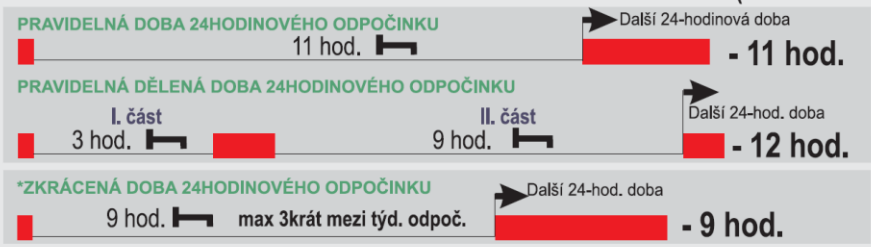
PLATÍ OD 11.04.2007

www.tachospeed.cz

- doba jízdy
- doba pohotovosti
- doba přestávky
- > 1h = odpočinek
- > 15min = přestávka

týden - období od po. 0:00 do ne. 24:00

24 HODINOVÁ DOBA ODPOČINKU - 1 ŘIDIČ (24 hod. od 1.ní aktivity)

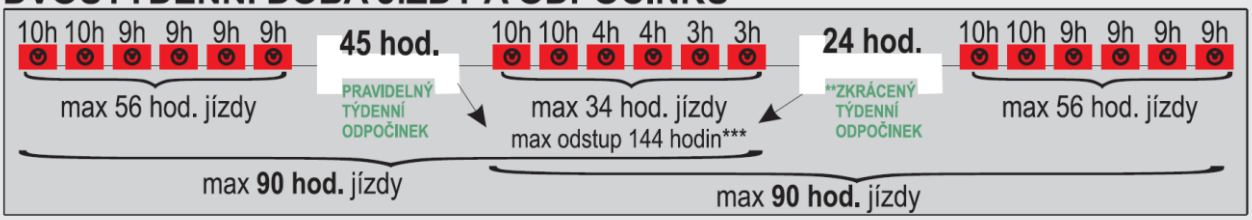


*pokud není příslušný týdenní odpočinek nebo náhrada za zkrácený týdenní odpočinek (viz ***) není nárok na možnost zkrácení 24hodinového odpočinku na méně než 11 hodin v daném dni

www.tachospeed.cz

infolab

DVOUTÝDENNÍ DOBA JÍZDY A ODPOČINKU



*** max. po 6-ti 24 hodinových obdobích musí následovat týdenní odpočinek; 2 další týdny jsou kontrolovány křížovým způsobem tzn. v čtyřech po sebe následujících týdnech 1. s 2., 2. s 3, 3. s 4.; ve dvou následujících týdnech musí být jeden z odpočinků min. 45 hodinový, v následujícím týdnu musí být zkrácen do max. 24 hodin, který musí být rekompensován vyváženým odpočinke z minimálně 9 hodinovým denním odpočinkem využitým jednorázově před koncem třetího týdne následujícím po daném týdnu

NA TRASE KE KONTROLE – KOTOUČE /VÝTISKY /KARTA ŘIDIČE + OSVĚDČENÍ O PRÁCI Z TOHOTO TÝDNE A TAKY 28 PŘEDCHÁZEJÍCÍCH DNŮ
VE FIRMĚ KE KONTROLE – KOTOUČKY /DIGITÁLNÍ SOUBORY .DDM/.ESM Z KARET A TACHOGRAFU 365 DNÍ OD MOMENTU ARCHIVACE

infolab