Fakulta elektrotechnická
Katedra aplikované elektroniky a telekomunikací

Bakalářská práce

Ovládací aplikace pro zařízení na vysílání zpráv na sběrnici CAN

Autor práce: Petr Hrubý
Vedoucí práce: Ing. Michal Kubík, Ph.D.

Plzeň 2012

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2011/2012**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Petr HRUBÝ**
Osobní číslo:	**E08B0330P**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronika a telekomunikace**
Název tématu:	**Ovládací aplikace pro zařízení na vysílání zpráv na sběrnici CAN**
Zadávající katedra:	**Katedra aplikované elektroniky a telekomunikací**

Zásady pro vypracování:

1. Seznamte se s funkcemi a možnostmi konfigurace zařízení CANtron pro vysílání zpráv na sběrnici CAN.
2. Zvolte vhodné vývojové prostředí na tvorbu aplikací pro systém MS Widows.
3. Navrhněte strukturu konfiguračních dat a implementujte ji společně s vhodnou logikou konfigurace zařízení do ovládací aplikace.

Rozsah grafických prací: **podle doporučení vedoucího**

Rozsah pracovní zprávy: **20 - 30 stran**

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

Seznam odborné literatury:

Student si vhodnou literaturu vyhledá v dostupných pramenech podle doporučení vedoucího práce.

Vedoucí bakalářské práce: **Ing. Michal Kubík**
Katedra aplikované elektroniky a telekomunikací

Konzultant bakalářské práce: **Ing. Michal Kubík**
Katedra aplikované elektroniky a telekomunikací

Datum zadání bakalářské práce: **18. října 2010**

Termín odevzdání bakalářské práce: **3. června 2012**

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

L.S.

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 17. října 2011

Abstrakt

Důvodem vzniku této bakalářské práce bylo vytvoření ovládací aplikace pro zařízení CANtron, jehož účelem je vysílání zpráv na sběrnici CAN. Toto zařízení spolu s ovládací aplikací pro OS Windows slouží účelům testovacím a simulačním. Pomocí kombinace obou nástrojů lze provádět pohotovou analýzu činnosti libovolné řídící jednotky.

Mezi nejdůležitější aspekty, které by měla ovládací aplikace splňovat, byla zařazena bezproblémová komunikace se zařízením, jehož parametry bude možno editovat za běhu programu. Uživatel bude mít například možnost nastavit přenosovou rychlost zařízení, či možnost modifikace obsahu zpráv, což je nejdůležitější funkcí vytvořené ovládací aplikace.

V rámci přípravy na samotné psaní zdrojového kódu a návrh uživatelského rozhraní se musely vyřešit otázky související s výběrem vhodného vývojového prostředí a programovacího jazyka, který by nejlépe posloužil daným požadavkům. Z tohoto důvodu byla pozornost zaměřena na jazyk Visual Basic, který se nakonec stal stěžejním nástrojem vedoucím ke splnění daných cílů zadání. K nim by se však nikdy nedospělo bez použití samotného funkčního vzorku zařízení, který posloužil jako nepostradatelný pomocník při ladění ovládacího programu.

Ačkoli se v době vytvoření ovládací aplikace jednalo o aktuální verzi vývojového prostředí, jenž zajišťovalo nejmodernější vzhled uživatelského prostředí, je jen otázkou času, kdy bude potřeba tuto aplikaci přizpůsobit budoucím požadavkům ať již z hlediska vizuálního, tak i funkčního.

Klíčová slova

CANtron, VCP, CRC, COM port, Transceiver

Abstract

Hrubý, Petr. *Control Application for a CAN Transmitter* [*Ovládací aplikace pro zařízení na vysílání zpráv na sběrnici CAN*]. Pilsen, 2012. Bachelor thesis (in Czech). University of West Bohemia. Faculty of Electrical Engineering. Department of Applied Electronics and Telecommunications. Supervisor: Michal Kubík

---

The aim of this Bachelor thesis was to design a control application for a device called CANtron. The purpose of this device is to transmit messages via a CAN bus. In this case, the CANtron is represented by a functional sample. CANtron together with the control application are very important components for testing and simulations in the domain of control units. The combination of these tools is used to perform an instantaneous performance analysis of a unit.

In order to make the control application fully operational, it should meet two key requirements. Its communication with the device should be trouble-free and the user should have an opportunity to edit its parametres during the running of the control application. For example, it will be possible to set the bit rate of the device and to modify the contents of messages. This feature is the most important function of the designed control programme.

Within the preparation for source code writing and user interface designing, it was necessary to choose a suitable developing environment and a programming language. The programming language selected as the most suitable one was Visual Basic, which became the principal tool to fulfil the set goals. However, the goals would be unrealizable without the functional device sample itself. This sample played an essential part during debugging of the control application. When creating the control application, an up-to-date developing environment was used. However, it is only a question of time when it will be necessary to adapt this application to meet the latest visual and functional requirements.

Keywords

CANtron, VCP, CRC, COM port, Transceiver

Prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci, zpracovanou na závěr studia na Fakultě elektrotechnické Západočeské univerzity v Plzni.

Prohlašuji, že jsem svou závěrečnou práci vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 270 trestního zákona č. 40/2009 Sb.

Také prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

V Plzni dne 8. června 2012

Petr Hrubý

.....

Podpis

Poděkování

Děkuji celé mé rodině a přítelkyni za veškerou podporu a trpělivost. Dále bych chtěl poděkovat mému vedoucímu panu Ing. Michalu Kubíkovi, Ph.D. za čas, vynaloženou energii a pomoc při konzultacích, neboť byl zároveň i mým konzultantem této závěrečné práce.

Obsah

Seznam obrázků

Seznam tabulek

Seznam symbolů a zkratek

ASCII	American Standard Code for Information Interchange. Americký standardní kód pro výměnu informací.
CAN	Controller Area Network. Datová komunikační síť CAN.
CRC	Cyclic Redundancy Check. Cyklický redundantní součet.
EEPROM	Electrically Erasable Programmable Read Only Memory. Elektricky přemazatelná programovatelná ROM.
FW	Firmware. Mikroprogramové vybavení.
HW	Hardware. Technické vybavení.
ID	Identifier. Identifikátor (identifikační číslo).
MSB	Most Significant Bit. Bit s nejvyšší váhou.
LSB	Least Significant Bit. Bit s nejnižší váhou.
LED	Light-Emitting Diode. Dioda emitující světlo.
PC	Personal Computer. Osobní počítač.
SW	Software. Programové vybavení.
UART	Universal Asynchronous Receiver and Transmitter. Asynchronní sériové rozhraní UART.
USB	Universal Serial Bus. Univerzální sériová sběrnice.
VB	Visual Basic.
VCP	Virtual COM Port. Virtuální (zdánlivý) sériový port.

1

Úvod

Důvodem vzniku této bakalářské práce byl požadavek na vytvoření vhodné aplikace pro OS Windows, jež by byla schopna komunikace a ovládání funkčního vzorku zařízení na vysílání zpráv na sběrnici CAN, který byl po dobu jejího návrhu k dispozici. Slovním spojením vhodná aplikace je v tomto případě myšlena skutečnost, kdy bude výsledný vzhled a funkčnost onoho ovládacího prostředí z hlediska uživatele intuitivní a příjemné, z hlediska zařízení CANtron vybavené všemi potřebnými a nezbytnými "komunikačními dovednostmi".

Vytvoření ovládacího nástroje je nezbytnou záležitostí, neboť je tato kombinace zařízení a aplikace zamýšlena pro účely simulací a testování.

Pro tvorbu uživatelského rozhraní a psaní zdrojového kódu bylo zvoleno vývojové prostředí Visual Studio od firmy Microsoft a programovací jazyk Visual Basic, jenž je tímto prostředím saportován (podporován). Celá práce byla stvořena ve verzi Visual Studio 2010 pomocí jazyka Visual Basic 2010.

Dle mně dostupných informací byla tato problematika přede mnou řešena kolegou z fakulty, jehož zásluhou byl vyroben funkční vzorek zařízení CANtron i naprogramován FW pro toto zařízení.

2

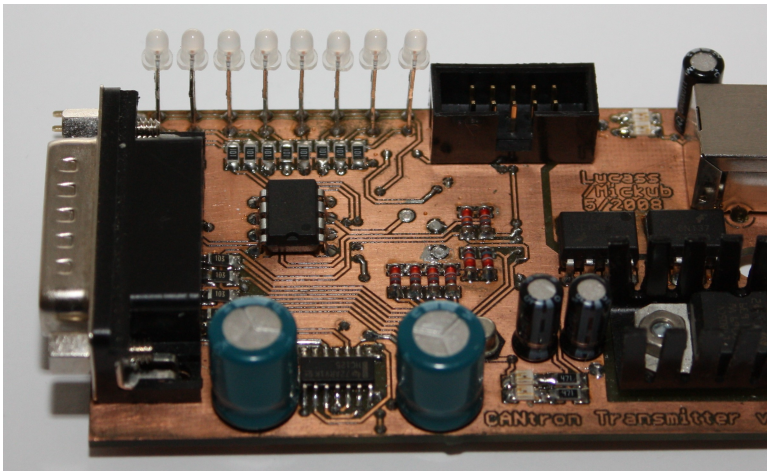
CANtron - vysílač na sběrnici CAN

2.1 Charakteristika funkčního vzorku

CANtron je zařízení, jenž umožňuje vysílání zpráv na CAN bus (sběrnici CAN). Funkční vzorek tohoto zařízení byl vyvinut pro účely simulací a testování v oblasti řídících jednotek.

Před použitím zařízení (v režimu offline) se pomocí PC nakonfigurují (nastaví) jednotlivé vysílané zprávy, vybere se druh budiče sběrnice CAN (High Speed budič MCP2551 s vysokou odolností proti šumu a power-on resetem, nebo Low Speed budič TJA1054 od firmy Philips) a nakonfiguruje se komunikační (přenosová) rychlost, již je možno vybírat z intervalu od 100 kbit/s do 1 Mbit/s. Konektivitu s PC zajišťuje interface (rozhraní) USB, jenž je uvnitř funkčního vzorku zařízení opticky odděleno.

Během používání (v režimu online) je možno obsah zpráv překonfigurovat. K tomuto účelu slouží 8 digitálních (číslicových) vstupů zařízení, jejichž stav L/H (Low/High) je signalizován pomocí dvoubarevných LED diod přímo na desce funkčního vzorku zařízení.



Obr. 2.1: Fotografie funkčního vzorku zařízení CANtron

K napájení zařízení je možno užít stejnosměrné napětí od 8 do 16 V. Napájení, stejně jako napěťová rezistance (odolnost) osmi digitálních vstupů, které jsou odolné do ss napětí

20 V, je uzpůsobeno pro užití v automobilovém průmyslu.

V rámci funkčního vzorku zařízení CANtron byl použit 8-bitový jednočipový mikrokontrolér (mikropočítač) AT90CAN128-16AU z rodiny AVR firmy Atmel s řadičem sběrnice CAN.

2.2 Komunikační protokol pro komunikaci s PC

Funkční vzorek zařízení CANtron komunikuje prostřednictvím rozhraní USB s řídícím počítačem, které je uvedeno do režimu VCP (Virtual COM Port), což je virtuální sériové rozhraní UART. Uživatel řídícího počítače (PC) zahajuje komunikaci s CANtronem, chová se tedy jako "velitel" neboli "master". Zařízení čeká na požadavky počítače, po jejich výskytu odpovídá zpět "masterovi", chová se tedy jako "podřízený" neboli "slave". Vzájemná komunikace probíhá v obou směrech postupným přenášením definovaných paketů (datových balíčků).

Paket může mít proměnnou délku v závislosti na druhu požadavku, který určuje množství přenášených dat. Pro vzájemnou komunikaci řídícího počítače a zařízení CANtron jsou nadefinovány rozličné commandy (příkazy), jež reprezentují dané požadavky "mastera". Je rovněž definován formát přenášených datových paketů.

Pro jednodušší komunikaci byl implementován (realizován) koncept, jenž zavádí do přenosu datových paketů aplikaci ASCII znaků. To znamená, že jsou jednotlivé balíčky zastoupeny sekvencí (sledem) znaků kódovaných pomocí ASCII standardu. Tato koncepce má svá pozitiva i negativa. Před vytvořením ovládací aplikace byl pro komunikaci se zařízením využíván jednoduchý terminál, což lze zařadit mezi výhody daného konceptu. Na druhou stranu lze také najít i negativní stránku věci. Za jejího nejvýznamnějšího zástupce by šla považovat redundantnost, neboli nadbytečnost informací při přenosu datových paketů. Každý datový bajt (Byte) je zde reprezentován dvojicí ASCII symbolů, jež představují hexadecimální (šestnáctkovou) hodnotu daného datového bajtu.

2.2.0.1 Formát požadavků

Zvolený formát datových paketů určuje pro označení začátku paketu ASCII znak s dekadickým (desítkovým) kódem 27, označovaný jako esc. V hexadecimální soustavě má tento znak hodnotu 1B. Tento "startovní" znak se nazývá inicializační znak příjmu. Je možno jej vyslat vícekrát. Pro označení konce datového balíčku byl zvolen ASCII znak s dekadickým kódem 10. V šestnáctkové soustavě reprezentuje tento znak hodnota 0A. Tento "finální" znak má význam odřádkování (angl. line feed, lf). Mezi "startovní" a "finální" znak se vkládá příkaz (angl. command), který je zastoupen 1 znakem, tedy 1 bajtem. Následují data libovolné délky podle typu požadavku, obecně zapsáno 0 - n bajtů.

Jako příklad si lze uvést paket s příkazem požadavku na odezvu, který v jazyce C vypadá následujícím způsobem: `'\27' 'q' '\10'`, což je totéž jako `'\27 'q' '\n'`. Před

zasláním následujícího požadavku je nutné vyčkat, až dorazí odpověď na předchozí požadavek, jinak zařízení není s to na něj reagovat.

Obecný formát požadavků a odpovědí vypadá následovně:

`<esc><command><data><crc><lf>`

Pro vážné nasazení systému bylo implementováno zabezpečení komunikace pomocí CRC kódu, který je reprezentován 2 znaky, tedy pomocí 2 bajtů.

2.2.0.2 Druhy požadavků

Všechny následující požadavky jsou uvedeny bez inicializačního znaku, CRC a znaku odřádkování.

- Paket požadavku na odezvu má následující strukturu: `'q'`

  Echo (zde ve významu odpověď) od CANtronu má stejný tvar.

- Paket požadavku pro SW reset má následující tvar: `'Q'`

  Echo je stejné jako požadavek.

- Paket pro získání verze HW (hardware) a FW (firmware) implementovaného v zařízení vypadá následovně: `'v'`

  Echo je rozšířené o znaky reprezentující obě verze, každá z verzí je zakódována dvěma znaky: `'v'<hw_ver><sw_ver>`

- Paket pro výběr druhu budiče a nastavení přenosové rychlosti: `'S'<speed><baudrate>`

  Echo je následující: `'s'<speed><baudrate>`

  - `<baudrate>` je vyjádřeno jedním znakem od 0 do 5, kde 0 znamená rychlost 100 kbit/s a 5 znamená 1 Mbit/s.
  - `<speed>` je naopak vyjádřeno znakem L, nebo H (tedy jedním bajtem) znamenající low-speed CAN a high-speed CAN.

- Paket pro speciální "D" příkazy má tvar: `'D'<key>`

  Echo se liší ve znaku příkazu: `'d'<key>`

  - `<key>` má tvar "STORCFG", který se používá pro uložení nastavení přenosové rychlosti a druhu budiče CAN do EEPROM paměti v mikrokontroléru. Tento klíč je tedy tvořen sedmi znaky, neboli sedmi bajty. Tento příkaz trvá velmi dlouho, přibližně 6 sekund, odpověď je odeslána po jeho dokončení. Dalším tvarem je "U90128", což znamená 6 znaků (6 bajtů). Tento klíč se užívá pro upgrade (aktualizaci) FW, spustí BOOTLOADER mód.

- Paket pro konfiguraci CAN zpráv vypadá následovně:

  `'M'<msg_no><active><length><msg_id><period>{<can_msg>}`

  Paket je uveden bez znaků pro začátek a konec paketu. Echo má stejný tvar jako požadavek. Význam jednotlivých komponent paketu je následující:

  - `<msg_no>` reprezentuje číslo zprávy od 0 do 7, pro jeho zakódování je použit jeden znak.
  - `<active>` vypovídá o možnosti vysílání dané zprávy, kóduje se opět jedním znakem, a to buď znakem '1' nebo '0'.
  - `<length>` zastupuje délku dané zprávy, neboli kolik datových bajtů zpráva obsahuje, kóduje se také jedním znakem od '1' do '8'.
  - `<msg_id>` vyjadřuje identifikátor zprávy, pro jeho zakódování je užito čtveřice znaků v rozsahu 0000 do 0FFF vyjádřeno v hexadecimální soustavě.
  - `<period>` znamená vysílací perioda zprávy vyjádřena v milisekundách, pro její zakódování se používá opět čtveřice znaků (bajtů) v intervalu od 0010 do 5000.
  - Posledním segmentem konfiguračního paketu je `{<can_msg>}`, což vyjadřuje posloupnost všech osmi bajtů dané zprávy přes všechny vstupy zařízení, kde je každý jednotlivý bajt zakódován dvěma znaky, jež reprezentují jeho hexadecimální hodnotu, tedy v rozsahu 00 až FF.

- Paket pro načtení konfigurace CAN zpráv je následující: `'m'<msg_no>`

  Význam obou komponent je zřejmý z předešlého textu. Echo od zařízení CANtron vypadá následovně:

  `'m'<msg_no><active><length><msg_id><period>{<can_msg>}`

Popis	Příkaz	Data	Délka	Odpověď
Požadavek odpovědi	'q'		0	'q'
Inicializace	'Q'		0	'Q'
Čtení verze HW a FW	'v'		0	'v'
Nastavení CAN transceiveru	'S'	L Sp	2	's'
Čtení nastavení CAN transceiveru	's'		0	's'
Nastavení CAN zpráv	'M'		140	'm'
Čtení nastavení CAN zpráv	'm'	MsgNo	1	'm'
Diagnostika – nastavení	'D'	Kx dd dd	6	'd'

Tab. 2.1: Komunikace od PC k zařízení CANtron

Popis	Příkaz	Data	Délka	Odpověď
Odezva na požadavek odpovědi	’q’		0	
Inicializace	’Q’		0	
Čtení verze HW a FW	’v’	HW FW	4	
Odeslání nastavení CAN transceiveru	’s’	L Sp	2	
Čtení nastavení CAN zpráv	’m’		140	

Tab. 2.2: Komunikace od zařízení CANtron k PC

Data MSB … LSB	Počet zn.	Popis a přepočet	Rozsah uložených hodnot a interpretace
HW FW	4	Verze HW Verze FW	0.0– F.F 0.0– F.F
L Sp	2	Úrovně ’H/L’ Komunikační rychlost ’0’ = 100 [kbps]	’H’ – High (MCP2551) ’L’ - Low (TJA1054) ’0’ - ’5’ 100 - 1000 [kbps]

Tab. 2.3: Význam přenášených datových bloků

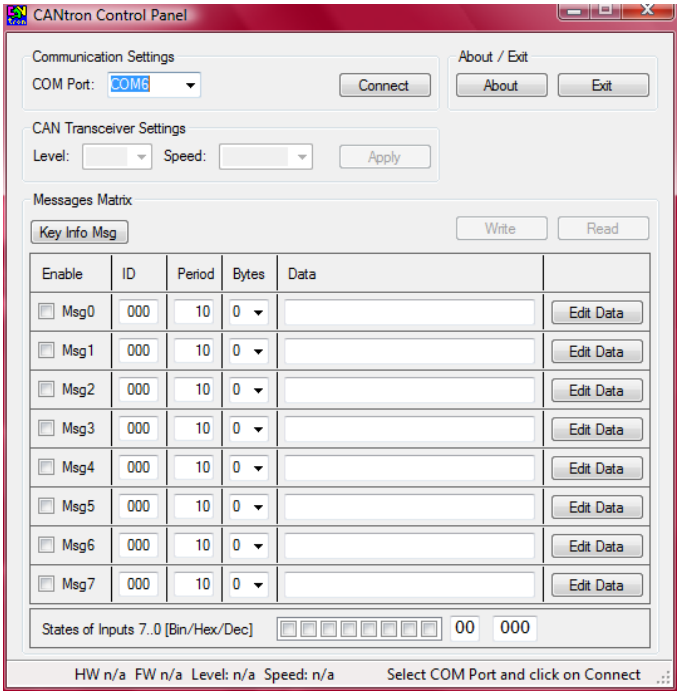
3

Ovládací aplikace pro PC

3.1 Charakteristika vzhledu a funkcí ovládací aplikace

3.1.1 Communication Settings

Při spuštění aplikace dochází k získání jmen aktuálních sériových portů počítače a k jejich nahrání a seřazení do ovládací prvku pole se seznamem (angl. ComboBox), jenž je elementem skupinového rámečku s názvem Communication Settings (nastavení komunikace). Tento skupinový rámeček je umístěn v levé horní části vstupního okna aplikace.



Obr. 3.1: Vzhled vstupního okna při startu aplikace

Do tohoto rámečku patří kromě pole se seznamem také tlačítko (angl. Button) s vepsaným textem Connect (připojit). Posledním elementem patřícím do skupinového rámečku

Communication Settings je popisek (angl. Label) s textem COM Port, který značí účel a význam ovládacího prvku umístěného v těsné blízkosti, v našem případě se jedná o již zmíněný a popsaný prvek pole se seznamem. Z tohoto nahraného a seřazeného seznamu jmen aktuálních sériových portů je vybrána (zvýrazněna) první položka ze seznamu, která se zobrazuje jako výchozí hodnota pole se seznamem.

3.1.2 About/Exit

Napravo od skupinového rámečku Communication Settings se zobrazí další ovládací prvek stejného typu, ovšem nazvaný About/Exit.

Tento rámeček zahrnuje pouze 2 tlačítka, která jsou funkční již po startu programu. Levé tlačítko s vepsaným textem About slouží k otevření dialogového okna (dialogu) About, jenž podává uživateli informace o aplikaci. Dialog About bude popsán dále v této kapitole.

Pravé tlačítko Exit slouží k ukončení aplikace. Při jeho stisknutí je vyvoláno malé dialogové okno, které je někdy označováno jako okno se zprávou. Toto okno obsahuje text pro potvrzení ukončení aplikace.

3.1.3 CAN Transceiver Settings

Třetím skupinovým rámečkem v rámci vstupního okna aplikace je CAN Transceiver Settings (nastavení budiče sběrnice CAN). Tento rámeček je umístěn ihned pod rámečkem Connection Settings.

CAN Transceiver Settings zahrnuje 2 popisky, 2 pole se seznamem a jedno tlačítko. Prvním popiskem je Level (úroveň), jenž poukazuje na účel výběru položky z prvního pole se seznamem zleva. Oním účelem je vybrat druh budiče sběrnice CAN. Možnosti jsou pouze 2, a to High-speed budič, nebo Low-speed budič. Proto se v tomto poli mohou zobrazit 2 položky Low a High. Po spuštění aplikace je však toto pole se seznamem deaktivováno, stejně jako druhé pole se seznamem v rámci skupinového rámečku CAN Transceiver Settings.

Poslání tohoto druhého pole je zřejmé z textu dalšího popisku. Je jím výběr (nastavení) přenosové rychlosti zařízení CANtron. Nabízí se celkem 5 položek, v hodnotách 100 kbit/s, 125 kbit/s, 200 kbit/s, 250 kbit/s, 500 kbit/s a 1 Mbit/s. Posledním elementem této skupiny ovládacích prvků je tlačítko Apply (aplikuj). Toto tlačítko zajistí správné nastavení parametrů dle volby uživatele. Stejně jako obě pole se seznam je i toto tlačítko neaktivní pe zahájení programu.

3.1.4 Message Matrix

Dominantou celého vstupního okna aplikace je čtvrtý a poslední skupinový rámeček Messages Matrix (matice zpráv).

Jeho obsah je značně složitější nežli v předcházejících případech. Při samém horním okraji tohoto rámečku jsou umístěna 3 tlačítka. Prvním tlačítko Key Info Msg slouží k nahrání (přednastavení) prvních dvou zpráv, tedy zprávy 0 a 1. Další 2 tlačítka jsou po náběhu programu prozatímně neaktivní, zaktivují se až po připojení se k zařízení CANtron.

Zbytek skupinového rámečku zabírá samotná matice parametrů zpráv, na jehož spodním okraji je umístěn ovládací prvek sloužící uživateli k manipulaci se stavy osmi digitálních vstupů zařízení CANtron. Celá matice parametrů zpráv je rozdělena do 9 řádků a 6 sloupců. Vrchní řádek poukazuje na význam jednotlivých sloupců.

V prvém sloupci si uživatel volí (ne)aktivitu zpráv, jinak řečeno, zda-li se budou jednotlivé zprávy vysílat, či nikoliv. Zaškrtnuté políčko značí aktivitu (vysílací "schopnost"), nezaškrtnuté značí neaktivitu (vysílací "neschopnost") dané zprávy.

Ve druhém sloupci si uživatel vybírá hodnotu identifikátoru zprávy ID, jež je programem omezena na rozmezí 0 až 7FF hexadecimálně. Z této informace vyplývá maximální délka identifikátoru, která činí 11 bitů.

Ve třetím sloupečku si uživatel zvolí vysílací periodu zprávy. Zadaná hodnota reprezentuje hodnotu periody v milisekundách. Tato hodnota je opět programem limitována na interval 10 až 5000 ms.

V následujícím sloupečku si uživatel vybírá pomocí polí se seznamem počet datových bajtů jednotlivých vysílaných zpráv. Počet datových bajtů je opět ohraničen. Ze seznamu si lze vybrat 0 až 8 datových bajtů.

Pátý sloupec je sloupcem osmi textových polí (angl. TextBox) příslušející osmi editovatelným zprávám. V těchto textových polích se zobrazuje hexadecimální reprezentace stavu nakonfigurovaných datových bajtů jednotlivých zpráv. Tato reprezentace je měnitelná nejen změnou konfigurace samotných datových bajtů dané zprávy, ale i uživatelem nastaveného ovládacího prvku se skupinou zaškrtávacích políček na samém spodním okraji rámečku Message Matrix.

Tento ovládací prvek obsahuje celkem 8 zaškrtávacích políček, stejně jako je 8 digitálních vstupů. Každý vstup má své políčko. Je-li políčko zaškrtnuto, aktivuje se vstup příslušející konkrétnímu políčku. Při nezaškrtnutí je tomu právě naopak. Lze individuálně nastavovat datové bajty dané zprávy pro všech 8 vstupů. Při manipulaci se skupinou políček, zastupující digitální vstupy zařízení, se v jednotlivých textových polích, zobrazující hexadecimální reprezentaci nakonfigurovaných zpráv, ukazuje exkluzivní součet datových bajtů přes všechny uživatelem zvolené vstupy pro každou zprávu zvlášť.

V posledním sloupci matice zpráv, jenž je umístěn nejvíce vpravo, je situováno 8 tlačítek stejného významu pro všech 8 nastavitelných zpráv. Oním významem je zobrazení dalšího (přídavného) okna pro editaci zpráv.

V tomto přídavném okně je uživateli umožněna konfigurace datových bajtů zprávy pro jednotlivé digitální vstupy zařízení. Tato konfigurace není jediným rysem editačního okna. Ale o tom dále v této kapitole.

Poslední komponentou skupinového rámečku Message Matrix je již zmíněná skupina osmi zaškrtávacích políček. Jejich účel jsem již objasnil, proto jen pro úplnost dodám, že je tato skupina doplněna dvojicí textových polí, které zobrazují dekadicky a hexadecimálně stav skupiny definované (editované) uživatelem, kdy zaškrtnuté políčko odpovídá logické hodnotě 1, a nezaškrtnuté políčko logické hodnotě 0.

3.1.5 StatusStrip

Poslední komponentou startovního okna aplikace je stavový řádek (ve VB StatusStrip) umístěný na spodním okraji okna a zobrazující stav a děje probíhající při běhu programu.

Tato stavová lišta v sobě obsahuje celkem 6 popisků umístěných těsně vedle sebe. Všechny popisky mají pevně zadané dimenze, jejich velikost se tedy během programu nemění.

Text zobrazitelný v obou krajních popiscích je zarovnán na střed ohraničené oblasti popisku, jež není během práce s aplikací viditelná. Ostatní popisky mají text zarovnán k levému okraji pomyslné hranice.

Po startu aplikace neobsahuje první popisek text, je tedy prázdný. Ve druhém popisku se zobrazí "HW n/a", což značí nedostupnost verze HW, kterou má tento popisek za úkol zobrazit (n/a znamená Not Available, což v češtině znamená nedostupné).

K podobné situaci dochází u následujících popisků s verzí FW, výběru budiče a hodnotou přenosové rychlosti.

Text posledního popisku nabádá uživatele, aby si ze seznamu vybral číslo sériového portu a kliknul na tlačítko Connect pro spojení se zařízením.

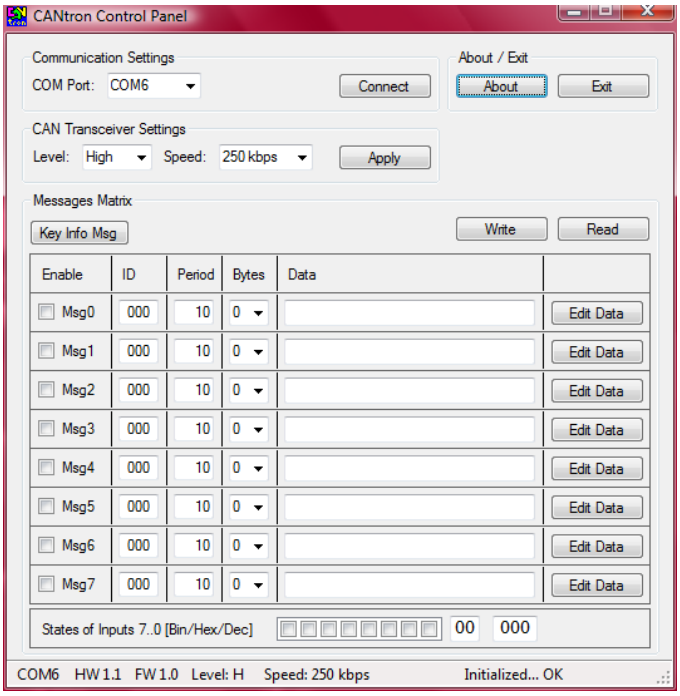
3.1.6 Připojení k zařízení CANtron

Z ovládacího prvku pole se seznamem v rámci skupinového rámečku Connection Settings si uživatel vybere číslo sériového portu, jenž odpovídá takovému, ke kterému je připojeno zařízení CANtron. Po výběru příslušného COM portu stačí již kliknout na vedlejší tlačítko Connect. Celá situace je patrná na obrázku 3.2.

Spustí se část programu příslušející tomuto tlačítku, jež zajistí navázání spojení se zařízením.

Dojde k první změně vzhledu vstupního okna aplikace. Do popisku ve stavovém řádku se vepíše text "Trying to open port COMx...", kde x je číslo vybraného sériového portu ze seznamu. Tato hláška informuje uživatele, že se aplikace pokouší otevřít sériový port COMx. Pokud se otevření zdaří, dopíše se k této hlášce potvrzovací "OK". V opačném případě se vepíše chybová hláška a segment programu příslušející tlačítku Connect končí. Pokud pokus o otevření portu skončí první zmiňovanou možností, napíše se do prvního popisku ve stavovém řádku název vybraného a úspěšně otevřeného portu.

V následujícím kroku se text úplně pravého popisku změní na hlášku "Trying to connect...", v této chvíli se aplikace pokouší navázat spojení se zařízením. Navazování je



Obr. 3.2: Vzhled vstupního okna po výběru příslušného sériového portu a připojení k zařízení CANtron

zahájeno vysláním požadavku na odpověď. V případě, že zařízení odpoví řídícímu počítači, je navázání komunikace úspěšné, a do příslušné části stavového řádku se dopíše potvrzovací "OK". V opačném případě se místo "OK" původní text vymaže a vypíše se "Click again to retry or change port". Tato hláška nabádá uživatele k tomu, aby se pokusil o navázání spojení opětovným stisknutím tlačítka Connect, nebo změnou čísla sériového portu, a poté stiskem tlačítka Connect. Mimoto se do prvního popisku stavové lišty "nahraje" slovo "FAILED", jenž zřetelně poukazuje na neúspěšnou akci. Toto slovo se obecně vypisuje při každé neúspěšné akci.

Pokud dojde k úspěšnému navázání spojení, aktivují se tlačítka Apply, Write i Read. V případě neúspěšného navázání zůstanou tato tlačítka i nadále deaktivována.

V případě úspěchu dochází nadále k procesu inicializace, kdy se do posledního popisku napíše text "Initialized...". Pokud se inicializace podaří, doplní se tento text opět potvrzovacím "OK".

V posledních dvou krocích, které příslušejí programu patřícímu tlačítku Connect, je získání verze HW i FW a nastavení transceiveru CAN.

Čísla verzí se vepisují do druhého a třetího popisku stavového řádku zleva.

Načtené nastavení CAN transceiveru se vepisuje nejen do stavového řádku na pozici čtvrtého a pátého popisku zleva, ale i do obou polí se seznamem v rámci skupinového rámečku CAN Transceiver Settings, jak je patrné z obrázku, kde se jako číslo verze HW objevilo 1.1, jako číslo verze FW 1.0.

Dále byla získána hodnota přenosové rychlosti 250 kbit/s (angl. kbps) a hodnota úrovně High.

Tímto je celý proces pokusu o připojení k zařízení CANtron u konce a nyní pokročím k popisu aplikace nastavení CAN transceiveru.

3.1.7 Aplikování konfigurace CAN transceiveru

V případě úspěšného navázání spojení se zařízením a procesu inicializace dochází k aktivaci tlačítka Apply, jak jsem zmínil dříve. K tomu je navíc nutné, aby si mohl uživatel nakonfigurovat (navolit) parametry CAN transceiveru pomocí ovládacích prvků k tomuto účelu určených. Proto se paralelně k aktivaci tlačítka Apply zpřístupní i pole se seznamem pro výběr hodnot přenosové rychlosti a úrovně. Chce-li se uživatel nastavit parametry CAN transceiveru, je mu tato možnost zpřístupněna právě pomocí těchto ovládacích prvků.

Rozbalením prvního pole se seznamem si uživatel zvolí ze dvou budičů. Na výběr je High-speed budič MCP2551 a Low-speed budič TJA1054.

Rozbalením druhého pole se seznamem si může uživatel zvolit ze 6 předvolených hodnot přenosových rychlostí (100, 125, 200, 250, 500, 1000 kbps).

Pro nakonfigurování CAN transceiveru slouží tlačítko Apply. Po jeho stisknutí dochází ke spuštění části programu, která patří tomuto tlačítku.

Program zajistí vepsání textu "Setting CAN parametres..." do posledního popisku stavového řádku. Tato hláška dává uživateli informaci o tom, že dochází k nastavování transceiveru CAN dle předvolených hodnot v rámci CAN Transceiver Settings. Současně se vyresetuje (vymaže) jakýkoli text prvého popisku lišty stavu.

Pokud dojde k zavření sériového portu před stiskem tlačítka Apply, je kód příslušející tomuto tlačítku schopen tuto událost odhalit. V tom případě se do posledního popisku zprava vepíše hláška "Not Connected" a do prvního popisku obecná hláška nezdaru "FAILED". Provádění programu se ukončí.

Pokud si uživatel navolí hodnoty parametrů, jež jsou již aplikovány, je o této skutečnosti uživatel informován pomocí hlášky "Already applied!" a i v tomto případě je další provádění programu předčasně ukončeno.

V situaci, kdy bude aspoň jedno pole se seznamem nevyplněno, jinými slovy bude aspoň jedno textové pole ovládacího prvku pole se seznamem prázdné, je provádění programu opětovně předčasně ukončeno. O tom, že je tato eventualita nesprávná a není aplikací podporována, je uživatel zřetelně informován pomocí vepsaného textu posledního popisku ve znění "Bad parametres" (špatné údaje), a opět obecnou hláškou nezdaru "FAILED". Špatným parametr se rozumí také sice vyplněné textové pole, ale neodpovící předvoleným hodnotám výstupní úrovně a přenosové rychlosti. To znamená, že jakýkoli text lišící se od hodnot High a Low pro výstupní úroveň, stejně jako jakýkoliv text lišící se od hodnot 100 kbps, 125 kbps, 200 kbps, 250 kbps, 500 kbps, 1 Mbps, je považován za špatný parametr.

O úspěšném nakonfigurování CAN transceiveru je uživatel informován pomocí potvrzovacího OK, jenž se připíše za původní text při startu programu po stisknutí tlačítka

Apply. Mimoto se změní hodnoty přenosové rychlosti a výstupní úrovně v popiscích stavového řádku i textových polích patřících ovládacím prvkům pole se seznamem v rámci skupinového rámečku CAN Transceiver Settings.

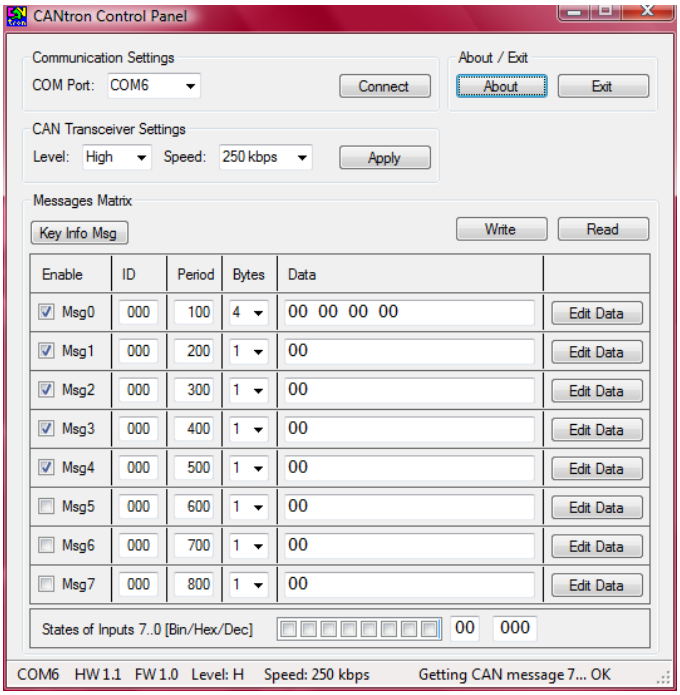
3.1.8 Čtení nastavení CAN zpráv

K načtení a zobrazení konfigurace zpráv CAN slouží tlačítko Read, jenž je součástí skupinového rámečku Message Matrix.

Funkce tohoto tlačítka má charakter cykličnosti (opakovatelnosti) a sekvenčnosti (sledu). Zprávy jsou načítání jedna po druhé. K tomu stačí jeden stisk tlačítka. Začíná se se zprávou číslo 0, po načtení zprávy 7 je opět možné číst od zprávy 0. Počet "přechodů" od zprávy 7 ke zprávě 0 je neomezený, zprávy lze načítat neomezeně.

Po stisku tlačítka Read se v popisku nejvíce vpravo ve stavovém řádku postupně zobrazuje text "Getting CAN message x...", kde x značí číslo zprávy, jež se právě získává od CANtronu.

I v tomto případě je zdrojový kód příslušející tomuto tlačítku schopen detekovat zavření sériového portu před stisknutím tlačítka Read. A i v tomto případě se vypíší chybové hlášky, které jsou shodné s hláškami pro tlačítko Apply. V případě úspěšného přečtení libovolné zprávy CAN je původní nedokončený text doplněn potvrzovacím "OK", jak můžete vidět na obrázku 3.3.



Obr. 3.3: Vzhled vstupního okna aplikace po úspěšném načtení sedmi CAN zpráv

Všechny získané parametry zprávy jsou explicitně zobrazeny v příslušných ovládacích prvcích logicky patřících stejnému číslu zprávy, tedy v prvcích, jež jsou všechny v jednom řádku matice zpráv hlavního okna aplikace. Charakteristiku jednotlivých komponent

řádku matice zpráv jsem uvedl již dříve, nemělo by proto smysl zabývat se jím i zde.

Parametry získané zprávy jsou zobrazeny nejen v tomto hlavním okně aplikace, ale také v přídavném okně, o kterém jsem se dosud moc nezmínil. Toto přídavné okno se jmenuje CANtron Edit Window, neboli editační okno zařízení CANtron. Výhodou editačního okna je přehledná vizuální reprezentace nakonfigurovaných datových bajtů dané zprávy patřící ke všem 8 digitálním vstupům. O tento prvek je hlavní okno ochuzeno, avšak částečně nahrazeno díky textovým polím ve sloupci Data v rámci matice zpráv, která zobrazují exkluzivní součet datových bajtů přes všechny vstupy.

Posledním rysem patřícím k manipulaci s tlačítkem Read je zobrazování krátké textové nápovědy zobrazitelné u tohoto ovládacího prvku při setrvání kurzoru myši v oblasti tlačítka. Tato nápověda, jež je zajišťována pomocí ovládacího prvku ToolTip, podává uživateli informaci o významu tohoto tlačítka, jímž je získání parametrů všech zpráv CAN po pouhém jednom stisknutí. ToolTip je v mé aplikaci hojně využívaným ovládacím prvkem, i když jsem se o něm dosud nezmínil. Jeho účelem je ve velmi jednoduché formě uživatele informovat o účelu ovladače, u kterého se textová nápověda zobrazuje.

3.1.9 Zápis nastavených CAN zpráv

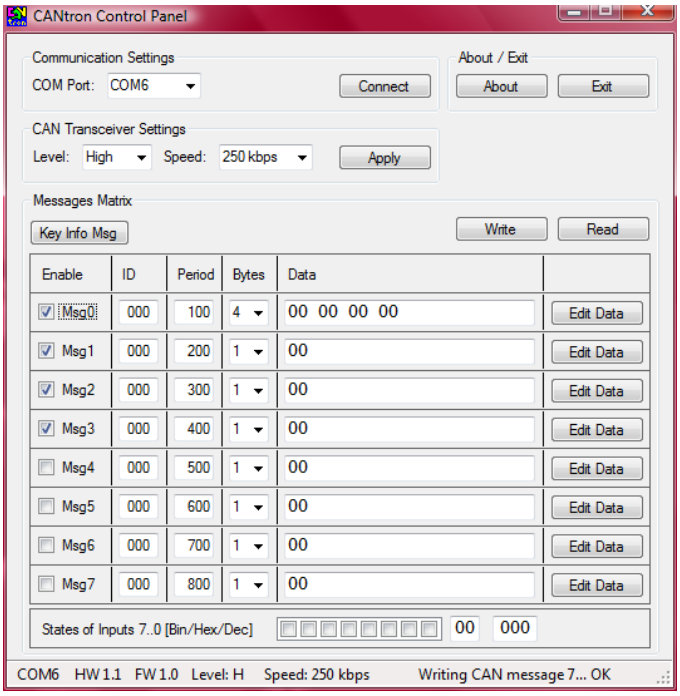
Zápis CAN zpráv je velice analogický čtení CAN zpráv, postupuje se však v opačném smyslu. Uživatel si zvolí číslo zprávy, kterou chce editovat, a klikne na příslušné tlačítko Edit Data, jenž je umístěno v řádku, které je vyhrazeno pro uživatelem vybranou zprávu. Následkem toho se otevře již zmiňované přídavné okno CANtron Edit Window, ve kterém je uživateli nabídnuta vizuální reprezentace datových bajtů příslušných editované zprávě pro každý z osmi digitálních vstupů zařízení CANtron odděleně. Podrobnější popis tohoto editačního okna zpráv dále v této kapitole.

Poté co je uživatel spokojen s konfigurací zprávy pro jednotlivé vstupy a s nastavením ostatních parametrů, jako je vysílací perioda a identifikátor zprávy, stačí již jen zmáčknout tlačítko Write, které je umístěno v rámci skupinového rámečku Message Matrix nad samotnou maticí zpráv, což můžete vidět na obrázku 3.4.

Toto tlačítko má obdobnou funkci jako tlačítko Read, jenž je situováno vedle tlačítka Write. I zde je použita cykličnost a sekvenčnost používání. K zapsání všech zpráv CAN stačí jediné stisknutí tohoto ovladače.

O tom, jaké má tlačítko Write použití, je uživatel informován prostřednictvím stručné textové nápovědy, kterou zajišťuje ovládací prvek ToolTip, jenž ji zobrazuje u tlačítka Write po ustálení kurzoru myši nad ovladačem.

Po stisknutí tlačítka Write je do posledního popisku stavového řádku postupně vkládán text "Writing CAN message x...", kde x koresponduje s číslem zprávy, jež je momentálně zapisována. I v tomto případě je tlačítko Write natolik "chytré", neboť "umí" detekovat situaci, při níž je sériový port uzavřen před stisknutím tlačítka. V tom případě je do "informačního" popisku stavového řádku vepsána hláška "Not connected!", což je obdobná funkce jako v případě tlačítka Read a Apply.



Obr. 3.4: Vzhled vstupního okna aplikace po úspěšném zápisu všech osmi zpráv CAN

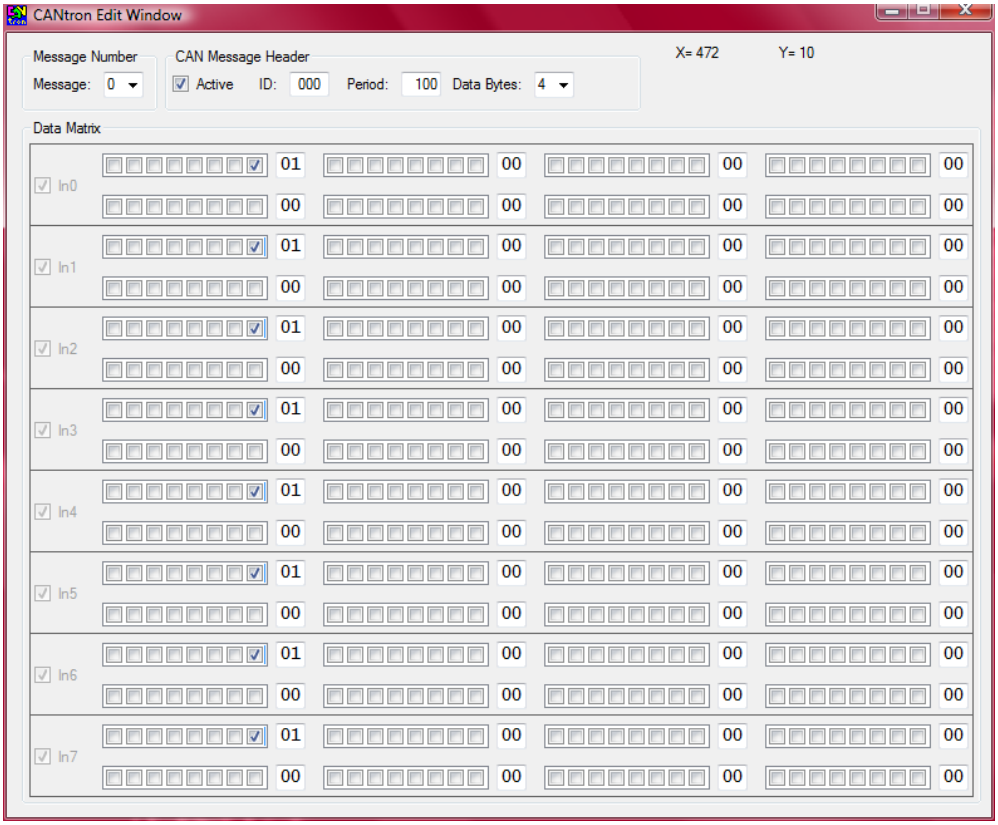
V případě úspěchu, kdy se editovaná zpráva zapsala do paměti zařízení a byla potvrzena vyslaná data, se do posledního popisku připíše potvrzovací "OK". Následkem toho se změní "orientace" procedury události tohoto tlačítka na následující číslo zprávy v pořadí. V tom případě se aktualizuje textová informace o momentálním zapisování údajů zprávy v posledním popisku řádku stavu.

3.1.10 Editační okno CANtron Edit Window

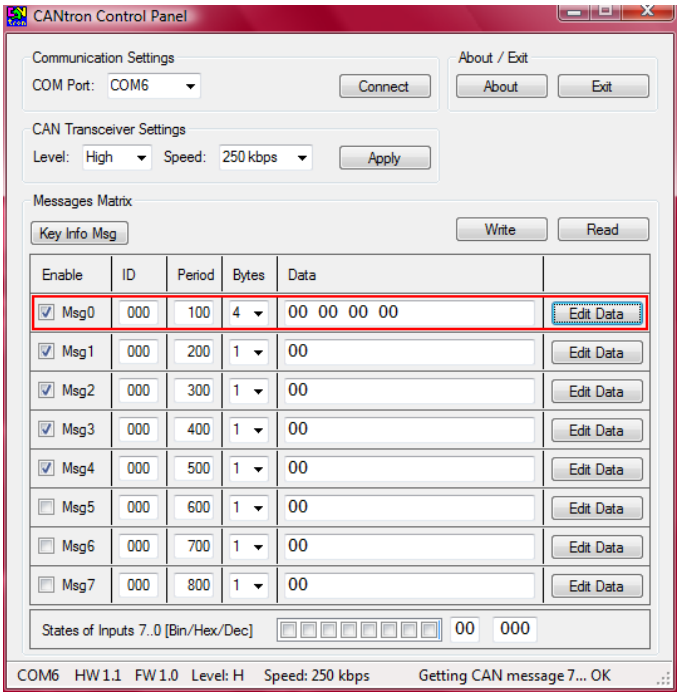
Nyní se dostávám k zevrubnému popisu vzhledu a zejména funkcí přídavného editačního okna. Jak takové editační okno vypadá, se můžete na vlastní oči přesvědčit na obrázku 3.5.

Jak jsem již poznamenal dříve, spouští se toto okno následkem stisknutí libovolného tlačítka Edit Data ze startovního okna aplikace (CANtron Control Panel). V podstatě je svým způsobem jedno, jaké z 8 nabízených "editačních" tlačítek uživatel zvolí. Vždy se zobrazí totéž okno nabízející možnost konfigurace zpráv CAN. Rozdíl je pouze v tom, že je toto přídavné okno vždy inicializováno na editaci té zprávy, ke které přísluší tlačítko Edit Data, jenž způsobilo jeho otevření. V praxi to znamená, že pokud uživatel klepnul na tlačítko Edit Data z řádky "patřící" zprávě s číslem 0, spustí se CANtron Edit Window s obsahem inicializovaným pro okamžitou editaci datových bajtů patřící právě zprávě s číslem 0.

Po stisknutí tlačítka Edit Data se však stane ještě jedna "maličkost". Ohraničí se řádek, do kterého patří použité tlačítko Edit Data, červeným rámečkem, jak můžete vidět na obrázku 3.6.



Obr. 3.5: Přídavné editační okno CANtron Edit Window, které je ve stavu editace zprávy 0 s nastavenými prvními bity (v log. 1) prvních bajtů pro všechny vstupy zařízení



Obr. 3.6: Orámování řádku po stisknutí tlačítka Edit Data patřícímu zprávě 0 (Msg0)

Tímto je uživateli zajištěna výrazná a nepřehlédnutelná vizuální pomůcka, která podává informaci o tom, jaká zpráva je momentálně konfigurována. Červený rámeček se objevuje při otevření editačního okna a ztrácí se při jeho zavření.

K rychlému a intuitivnímu přepínání mezi zprávami, které chce uživatel editovat, slouží ovládací prvek pole se seznamem, které je elementem skupinového rámečku Message Number (číslo zprávy), jenž je situováno v levém horním rohu editačního okna CANtron Edit Window. V textovém poli je zobrazováno číslo aktuálně editované zprávy.

Vedle Message Number je umístěno další skupinový rámeček CAN Message Header (hlavička zprávy CAN), jenž v sobě obsahuje jedno zaškrtávací políčko Active, které má stejný účel jako ve vstupním okně aplikace v rámci matice zpráv, tedy umožnění vysílání dané zprávy.

Dalšími prvky jsou 2 textová pole zobrazující identifikátor (ID) a vysílací periodu (Period) aktuálně editované zprávy. Posledním elementem této skupiny ovladačů je pole se seznamem, pomocí něhož zadává uživatel počet datových bajtů zprávy.

Přídavné editační okno nabízí uživateli jednu "zajímavost" v podobě souřadnic x a y, které ukazují polohu tohoto okna v rámci pracovní plochy monitoru v počtu obrazových bodů (pixelů) počítáno od levého horního rohu obrazovky k levému hornímu rohu okna Na obrázku 3.5. je zřetelně vidět, že je editační okno vzdáleno od rohu obrazovky o 472 pixelů ve směru osy x (v horizontálním směru) a o 10 pixelů ve směru osy y (ve vertikálním směru).

Dominantním prvkem celého editačního okna je bezesporu obsáhlý skupinový rámeček Data Matrix (matice dat). Celá tato matice je rozdělena na 8 řádků, jako je tomu u matice zpráv ve startovním okně aplikace.

Na rozdíl od matice zpráv, kde řádka zastupovala jednu ze zpráv, zde řádka reprezentuje jeden z 8 digitálních vstupů zařízení. Vstupy jsou číslovány obdobně jako zprávy od 0 do 7. Každému vstupu přísluší osm skupin zaškrtávacích políček po osmi zástupcích tohoto ovládacího prvku, kde každá ze skupin zastupuje jeden datový bajt zprávy. Jednotlivá políčka reprezentují bity. První skupina políček shora patřící do řádku pro vstup 0 (In0) odpovídá bajtu 0, druhá skupina shora z téže řádky odpovídá bajtu 1 atd., až poslední skupina zleva ve druhém řádku, který ještě patří vstupu 0, odpovídá bajtu s číslem 7. Takto jsou očíslovány všechny skupiny políček pro všechny vstupy.

Označené políčko libovolné skupiny libovolného vstupu charakterizuje daný bit zprávy jako by byl v log. 1. Bity jakékoliv skupiny políček jsou seřaženy zleva od MSB po LSB, který je úplně napravo.

Každé textové pole v této matici dat vyjadřuje svoji zobrazovanou hodnotou hexadecimální ekvivalent k binárnímu vyjádření každého z bajtů.

Jakmile je uživatel spokojen s konfigurací dat, je vhodný čas pro jeho zapsání pomocí tlačítka Write.

3.1.11 Okno About

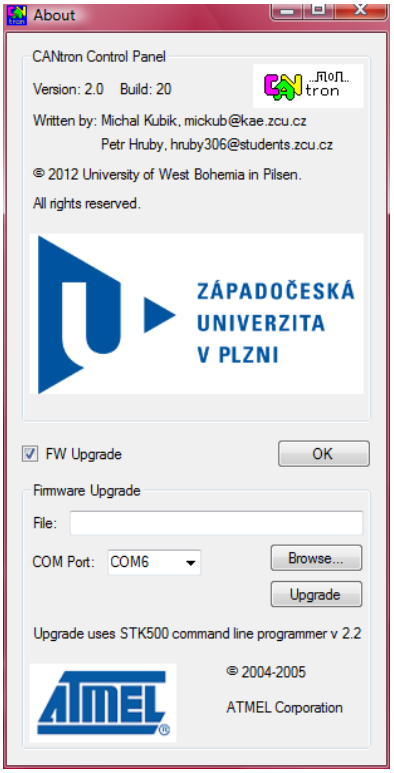
Posledním oknem, o němž jsem se dosud moc nezmínil, je okno About, neboli okno podávající základní informace o aplikaci. Screenshot tohoto okna, které má své nezastupitelné místo u každé aplikace, můžete vidět na následujícím obrázku.

Obr. 3.7: Okno About při nezaškrtnutém políčku FW Upgrade

V okně About je uvedena verze aplikace, její autoři, vlastník autorského práva, dále logo Západočeské univerzity v Plzni a logo zařízení CANtron.

Okno About v jeho základní nerozšířené podobě obsahuje už jen tlačítko OK, které slouží k zavření dialogového okna (dialogu) About, a zaškrtávací políčko, jehož účelem je rozšíření dialogu o část sloužící k upgradu (vylepšení, aktualizace) FW.

V rámci tohoto doplňku základní dimenze (rozměru) okna About jsou situovány ovládací prvky sloužící uživateli k zahájení procesu aktualizace. Okno About v jeho rozšířeném formátu můžete vidět na obr. 3.8.



Obr. 3.8: Okno About při zaškrtnutém políčku FW Upgrade, které způsobí jeho rozšíření o komponentu procesu aktualizace

4

Zdrojový kód ovládací aplikace

V této kapitole se zaměřím na popis struktury zdrojového textu, jenž tvoří samotné "jádro" aplikace. Bude zde rozebrána provázanost jednotlivých tříd (angl. classes), která je klíčem ke správnému chování programu v různorodých situacích. Samozřejmě bude zpočátku dána přednost popisu samotných tříd, neboť jsou to právě ony, jež jsou "duší" každého objektově orientovaného programu.

Protože je program velmi obsáhlý, zaměřím se především na to nejpodstatnější a nejdůležitější, zevrubný popis celého kódu by byl velice bohatý. Vyhnu se charakteristice procedur událostí vstupního okna aplikace, kterým jsem věnoval veliký prostor v minulé kapitole.

4.1 Třída CANmessage

Základní třídou, jež je šablonou popisující parametry každé z 8 zpráv CAN, je právě třída CANmessage (angl. CANmessage class).

Obsahuje několik vlastností (angl. properties), které v podstatě reprezentují parametry každé zprávy. Patří mezi ně MessageNumber (číslo zprávy), TransmitEnable (aktivita, možnost vysílání), ID (11-bitový identifikátor zprávy), Period (vysílací perioda zprávy uváděná v milisekundách limitovaná od hodnoty 10 ms do hodnoty 5000 ms), BytesNum (udává počet datových bajtů zprávy limitované od 0 do 8), strDataField (vlastnost pouze ke čtení určující funkčnost datového pole v rámci skupinového rámečku Message Matrix startovního okna aplikace), DataByte (vlastnost určující obsah libovolného datového bajtu libovolné zprávy), Editing (vlastnost dávající najevo momentální editaci libovolné zprávy uživatelem pomocí přídavného editačního okna), LinkedInputs (pomocí této vlastnosti se zjistí stav jakéhokoli vstupu užitím parametru idx) a Inputstates (podává informaci o stavech všech 8 vstupů).

Dalšími elementy třídy CANmessage jsou deklarace 2 konstruktorů, jenž slouží k inicializaci instancí "modelovaných" na základě obsahu této třídy.

Jako první je uveden "prázdný" konstruktor nevyžadující žádné parametry, v jehož těle je uveden kód, který způsobí "vytvoření" tzv. prázdné instance třídy, jejíž vlastnosti

mají nulové počáteční hodnoty.

Vedle tohoto "prázdného" konstruktoru je uveden i konstruktor s parametry, pomocí něhož lze vytvořit instanci třídy CANmessage s inicializovanými vlastnostmi na základě hodnot parametrů tohoto konstruktoru. Konstruktor s parametry "vytváří" instanci třídy z přímých hodnot uvedených na místě parametrů.

Mimoto jsou v těle třídy CANmessage uvedeny privátní (soukromé) funkce pro konverzi (převod) celočíselné proměnné na řetězcovou proměnnou, vyjadřující hexadecimální reprezentaci dané celočíselné proměnné.

4.2 Třída CANmessageControl

Další vygenerovanou třídou je CANmessageControl, jež je šablonou objektu pro zacházení s CAN zprávami, který má svoji vizuální reprezentaci. Tento objekt je součástí vstupního okna aplikace, ve kterém jich je zastoupeno celkem 8, pro každou zprávu jeden. Jeho vzhled a funkce jsem podrobně popsal v minulé kapitole. Zde se zaměřím na strukturu jeho "předobrazu" v podobě třídy CANmessageControl.

Ve zdrojovém kódu je obsažena deklarace celkem tří vlastností. Jsou jimi MsgIndex (index, neboli číslo zprávy CAN), MsgTransmitEnable (vlastnost vyjadřující schopnost vysílání libovolné zprávy CAN) a MsgEdit (vlastnost vyjadřující situaci, kdy jsou datové bajty libovolné zprávy aktuálně editovány).

V rámci zdrojového textu příslušejícímu této třídě jsou mimo vlastností deklarovány rovněž některé procedury událostí související s objektem z této třídy "vytvořeném".

Jako první obsluhou události je `CANmesssageControl_Paint`, jež zajišťuje vykreslení svislých separátorů (oddělovačů) v podobě černých tenkých čar, které zajišťují vizuální separaci jednotlivých ovládacích prvků v rámci vizuální reprezentace objektu CANmessageControl. Další "schopností" této procedury je vykreslení červeného rámečku, který je zobrazen v době úpravy datových bajtů libovolné zprávy.

Vedle událostní procedury `CANmessageControl_Paint` je deklarována i obslužná procedura `ButtonEdit_Click`. Zdrojový kód této procedury je spuštěn stisknutím tlačítka Edit Data, který je elementem objektu CANmessageControl. Úkolem této procedury je spouštění a uzavírání přídavného editačního okna CANtron Edit Window.

Důležitou procedurou je rovněž `CANmessageControl_Load`, jež se spouští při nahrávání objektu CANmessageControl při startu aplikace. Jejím posláním je nadefinování parametrů pomocného ovládací prvku ErrorProvider, jehož úkolem je podat uživateli informaci o případné chybě při editaci údajů jakékoliv zprávy CAN. Příkladem budiž překročení pomyslné hranice při zadávání hodnoty vysílací periody, která je limitována zdola hodnotou 10 a shora hodnotou 5000. Pokud uživatel zadá hodnotu mimo tento rozsah, objeví se symbol chyby s krátkou textovou zprávou objasňující její příčinu.

4.3 Třída CANmsgHeaderControl

Objekt této třídy má rovněž svoji vizuální reprezentaci. Je jí hlavička souboru osmi objektů CANmessageControl v rámci skupinového rámečku Message Matrix hlavního okna aplikace.

Tělo třídy CANmsgHeaderControl je velmi jednoduché, obsahuje pouze jedinou obsluhu události `CANmsgHeaderControl_Paint`, jejíž úkolem je vykreslení celkem 5 tenkých černých separátorů oddělující jednotlivé popisky.

4.4 Třída CANmsgFooterControl

I tato třída je velmi primitivní. Obsahuje rovněž pouze jedinou proceduru události, která má název `InputStates_ByteInputBox_DataByteChanged` a spouští se při manipulaci se zaškrtávacími políčky, jež se prvky skupiny vyskytující se na spodním okraji okna aplikace pod rámečkem Message Matrix. Zajišťuje zobrazení dekadického ekvivalentu binárnímu vyjádření, jenž je reprezentován skupinou editovatelných zaškrtávacích políček, ve vedlejším textovém poli. Významnou "schopností" této jediné procedury je však zajištění zobrazitelnosti stavu datových bajtů každé ze zpráv v textových polích ve sloupci Data ve vstupním okně.

4.5 Třída ByteInputBox

Objekt třídy ByteInputBox je rovněž viditelný při běhu aplikace. Skládá se pouze ze dvou ovládacích prvků. Jsou jimi skupina 8 zaškrtávacích políček a textové pole. Tento objekt je zastoupen jak ve startovním okně aplikace, tak zejména v přídavném editačním okně.

Zdrojový text je o něco složitější nežli u dvou předešlých tříd. Obsahuje pár vlastností, několik privátních funkcí i procedury událostí.

První deklarovanou vlastností je DataByte (vlastnost popisující stav libovolného datového bajtu). Druhou a zároveň poslední vlastností je strDataByte, která charakterizuje jakýkoli datový bajt pomocí dekadického vyjádření.

Mezi privátní funkce patří funkce pro konverzi celočíselné osmibitové proměnné na řetězcovou proměnnou vyjadřující hexadecimální reprezentaci hodnoty této celočíselné proměnné.

Další funkce slouží pro generaci hodnoty celočíselné, 8-bitové, v této soukromé funkci definované, proměnné, jež je "obrazem" stavu zaškrtnutí skupiny 8 zaškrtávacích políček objektu ByteInputBox.

Přínosem poslední funkce této třídy je nastavení stavu zaškrtnutí skupiny osmi políček na základě hodnoty celočíselné 8-bitové proměnné, která je jako jediný parametr této soukromé funkci předávána.

Následující komponenty třídy ByteInputBox mají charakter procedur událostí. Jsou zde celkem tři.

Tou první je procedura, jež se spouští při uživatelem editovaném obsahu textového pole objektu ByteInputBox, `TextBox1_TextChanged`. Tato procedura způsobí změnu stavu zaškrtnutí osmice políček příslušející k textovému poli, který proceduru vyvolal na základě uživatelem zadané hodnoty v hexadecimální soustavě. Mimoto dovede uživatele informovat o špatném formátu zadávané hodnoty pomocí změny barvy pozadí ovládacího prvku textové pole a krátké textové zprávy, jež je zajišťována kontrolním elementem ToolTip.

Druhou procedurou je `CheckedListBox1_SelectedValueChanged`, jež zajišťuje změnu hodnoty textového pole reprezentující stav zaškrtnutí skupiny políček hexadecimálním formátem, právě při konfiguraci tohoto stavu skupiny uživatelem.

Třetí a zároveň poslední událostní procedurou, která se spouští při nahrávání objektu ByteInputBox do okna aplikace, ve kterém má svoji předdefinovanou pozici, je `ByteInputBox_Load`. Její obsah je primitivní a v podstatě zajišťuje pouze inicializaci ovládací prvku ToolTip, jenž je svázán s textovým polem v rámci daného objektu.

4.6 Třída EditForm

Také tato třída má svoji vizuální reprezentaci. Je jí přídavné okno k úpravám parametrů CAN zpráv CANtron Edit Window, které se zobrazí po stisknutí libovolného tlačítka Edit Data v rámci vstupního okna aplikace.

Zdrojový kód této třídy je celkem jednoduchý. Obsahuje několik událostních procedur.

Jako první by se hodilo jmenovat proceduru `EditForm_Load`. Její pracovní náplní je inicializace parametrů ovládacího elementu ErrorProvider, jenž je spřažen s textovými poli určenými pro zadávání hodnot ID a periody.

Opakem budiž procedura `EditForm_FormClosing`, jejíž posláním je odstranění tenkého červeného rámečku okolo objektu CANmessageControl, který indikuje aktuální editaci parametrů libovolné zprávy.

"Zajímavou" obsluhou události je `EditForm_Move`, jež má na starosti informování uživatele o aktuální pozici editačního okna v rámci pracovní plochy monitoru. O této "vychytávce" jsem se zmiňoval už v minulé kapitole.

Poslední obsluhou události, o které se zde zmíním v souvislosti se třídou EditForm, je `ComboBox_SelectMsgNum_SelectedIndexChanged`. Její spuštění zapříčiní ohraničení správného objektu CANmessagControl červeným rámečkem ve startovním okně aplikace. Adjektivem "správného" je zde míněno číslo daného objektu korespondující s číslem zprávy, jež byla vybrána pomocí ovládací prvku pole se seznamem (ComboBox) umístěném v editačním okně. Mimoto je odstraněn rámeček původně konfigurované zprávy. Avšak nejdůležitější rolí této obsluhy je aktualizace obsahu CANtron Edit Window v závislosti na vybraném číslu zprávy z ComboBoxu. Aktualizuje se nejen ID, perioda, ale i nastavení všech datových bajtů pro všech 8 digitálních vstupů zařízení.

4.7 Třída EditInputLinkedDataControl

Je to poslední třída, kterou se budu v této kapitole zabývat. Stejně jako ostatní třídy, vyjma CANmessage, má i tato svoji vizuální stránku v podobě uživatelského objektu. Objektů EditInputLinkedDataControl je celkově 8, každý složen z 8 objektů ByteInputBox. Navíc jsou všechny součástí jediného formuláře v podobě CANtron Edit Window. Zdrojový text této třídy je relativně málo obsáhlý, neboť obsahuje jedinou vlastnost a 3 stručné událostní obsluhy.

Onou jedinou vlastností je MsgIndex, jež stejně jako v případě CANmessageControl vyjadřuje číslo zprávy CAN.

První ze 3 procedur událostí je `ByteInputBox_DataByteChanged`. Obsluha se zavolá při konfiguraci libovolného objektu ByteInputBox uživatelem. Jejím "posláním" je správně modifikovat vlastnost DataByte třídy CANmessage, konkrétně instance této třídy. Instancí CANmessage je rovněž 8, neboť je možno editovat celkem 8 zpráv CAN. Je nutno pozměnit vlastnost DataByte takového čísla instance, které koresponduje s číslem právě editované zprávy.

Druhou událostní procedurou budiž `InputLinked_CheckBox_CheckedChanged`. Jejím jediným úkolem je zajistit překonfigurování vlastnosti LinkedInputs odpovídající instance třídy CANmessage v závislosti na stavu zaškrtnutí políčka `InputLinked_CheckBox`.

Poslední ze 3 obsluh "pouze" nastavuje vlastnost Tag všech objektů ByteInputBox při nahrávání EditInputLinkedDataControl a jmenuje se `EditInputLinkedDataControl_Load`. Jelikož je v rámci tohoto objektu dohromady 8 ByteInputBoxů, má vlastnost Tag 8 různých hodnot, aby nedocházelo k jejich záměně. Mimoto způsobí generaci textu Inx, kde je x ve významu čísla zprávy od 0 do 7.

4.8 Konektivita mezi třídami

Nejzásadnější roli v celém "systému" tříd této aplikace hraje CANmessage. Jedná se o základní třídu, o kterou se "opírají" téměř všechny ostatní. Zejména její vlastnosti jsou "protkány" veškerým zdrojovým kódem programu.

Spojení mezi CANmessage a ostatními třídami zprostředkovává komponenta BindingSource, jež je v mém programu pojmenována CANmessageBindingSource. Její nejdůležitější vlastnost DataSource (zdroj dat) je nastavena právě na CANmessage. Je-li například modifikován atribut CANmessage prostřednitvím jednoho objektu, je pomocí CANmessageBindingSource změněna vlastnost jiného objektu, jež je prostřednictvím této "konektorovací" komponenty "svázána" s modifikovaným atributem CANmessage.

Tímto způsobem funguje kupříkladu konfigurace ID zprávy CAN ve vstupním okně aplikace v rámci skupinového rámečku Message Matrix. Po úpravě vlastnosti Text textového pole ID uživatelem dochází na základě propojení tohoto ovládací prvku s textovým polem stejného významu v editačním okně CANtron Edit Window prostřednictvím CAN-

messageBindingSource také k modifikaci textu onoho prvku se stejným významem. Proto stačí uživateli zadat žádanou hodnotu pouze jednou, "konektorovací" komponenta se sama postará o její replikaci do dalšího elementu, který slouží ke stejnému účelu.

S objektem CANmessageBindingSource spolupracují EditForm, EditInputLinkedDataControl a CANmessageControl.

"Svázání" prvků pomocí CANmessageBindingSource existuje kromě textového pole ID rovněž u TextBoxu Period (perioda), pole se seznamem k výběru počtu datových bajtů, zaškrtávacího políčka Msgx apod.

Kromě komponenty CANmessageBindingSource existuje v mém programu další nástroj se stejným "posláním". Je jím ByteInputBoxBindingSource, která dává dohromady třídy ByteInputBox a CANmsgFooterControl. Nejenže je objekt ByteInputBox elementem objektu CANmsgFooterControl, navíc je textové pole, jenž je rovněž jeho součástí, spojeno prostřednictvím ByteInputBoxBindigSource s atributem strDataByte třídy ByteInputBox. Díky tomu je možno modifikovat stav skupiny zaškrtávacích políček pomocí editace obsahu textové pole objektu CANmsgFooterControl.

4.9 Objekt SerialPort

Komponenta SerialPort je nezbytným prvkem pro komunikaci ovládací aplikace "s vnějším světem", proto jsem ji vyhradil zvláštní podkapitolu. Uvedu zde přínos tohoto důležitého objektu v rámci ovládacího programu, jenž reprezentuje prostředek sériového portu.

Po spuštění aplikace dochází pro naše účely k inicializaci vlastnosti Baudrate (přenosová rychlost) této komponenty na hodnotu 115 200 [bps], neboť je jako výchozí hodnota nastavena 9600 [bps]. Mimoto se pomocí metody GetPortNames získá seznam platných jmen sériových portů na daném počítači. Tato jména se následujícím segmentem kódu "vloží" do obou objektů pole se seznamem pro výběr sériového portu. Jedno pole je součástí vstupního okna, druhé naopak okna About.

K další spolupráci s objektem SerialPort dochází v rámci obsluhy události, jež patří stisknutí tlačítka Connect. Na začátku se testuje stav otevření sériového portu. Pokud je již port otevřen, dochází k jeho uzavření. V případě neotevřeného portu se neprovede v dané sekci programu žádný příkaz a postupuje se dále.

Následuje nastavení vlastnosti PortName na jméno sériového portu, který si uživatel vybral z ovládací prvku ComboBox před stisknutím tlačítka Connect, a k pokusu o otevření sériového portu. U objektu SerialPort může existovat pouze jedno otevřené spojení. V případě úspěšného pokusu o otevření dochází k vypsání potvrzovacího "OK" za původní text "Trying to open port COMx..." a k uvedení jména otevřeného portu do prvního popisku stavového řádku zleva. V případě nepodařeného pokusu o otevření sériového portu dochází k napsání "FAILED" namísto jména a chybové hlášky namísto původního textu posledního popisku řádku stavu s potvrzovacím "OK".

Poté dochází k několikanásobnému volání privátní funkce SerialPortWriteCmd třídy

MainForm, jež užívá veledůležitou metodu objektu SerialPort Write(). Tato metoda zajistí zápis námi nadefinovaného datového paketu do výstupního bufferu sériového portu. Soukromá funkce SerialPortWriteCmd je hojně užívaná v celé třídě MainForm.

Jedinou událostí, jejíž obsluha je implementována ve zdrojovém kódu ovládací aplikace, je DataReceived. Procedura této události je spuštěna při příjmu dat sériovým portem. Nejprve je testováno, kolik se uložilo bajtů dat do přijímací bufferu. Poté je zavolána metoda ReadExisting objektu SerialPort, která zajistí přečtení všech bajtů ze vstupní vyrovnávací paměti, jež jsou k dispozici.

Při ukončování spuštěné ovládací aplikace je nutné otestovat pomocí vlastnosti IsOpen, zda je sériový port otevřen. Pokud tomu tak je, zavolá se metoda Close, jež zajistí jeho správné uzavření. Poté je možno aplikaci ukončit.

5

Závěr

Na základě zadání bakalářské práce byla vytvořena ovládací aplikace, jež je schopna bezproblémové komunikace se zařízením CANtron. Byly splněny hlavní požadavky, týkající se návrhu struktury konfiguračních dat do ovládací aplikace a implementace vhodné logiky nastavení zařízení na vysílání zpráv. Bylo uskutečněno přehledné zobrazení uživatelem editovatelných parametrů zpráv, sloužících pro vysílání na sběrnici CAN, prostřednictvím spolupráce několika objektů sloužících snadnému a intuitivnímu ovládání při používání aplikace.

Podařilo se realizovat nastavení parametrů CAN transceiveru, které si bude zařízení pamatovat i po ukončení činnosti programu. Díky tomu se nemusí přenosová rychlost komunikace a volba výstupního budiče sběrnice opětovně konfigurovat po dalším spuštění řídící aplikace.

Mezi pozitivní rysy vyvinutého programu patří úspěšná realizace čtení a zapisování zpráv CAN. Klíčová byla zejména transformace nakonfigurovaných parametrů zpráv do takové podoby, kterou je možno v rámci paketů posílat po lince zařízení CANtron tak, aby bylo schopno odpovědět zpět řídícímu počítači. Tato záležitost činila v průběhu vývoje ovládací aplikace mnohé problémy, proto ji kladu za důležitý mezník, který se podařilo s úspěchem implementovat. Musela se rovněž vymyslet vhodná funkce pro obousměrnou konverzi dat, neboť bylo požadavkem reprezentovat každý konfigurační bajt pomocí 2 komunikačních ASCII znaků.

V neposlední řadě byla komunikace řídícího počítače s CANtronem zabezpečena pomocí CRC kódu, jenž je uskutečněn pomocí dvou znaků zařazených mezi datovou část a ukončovací část paketu.

Mezi nedostatky lze zařadit upgrade firmware, který se spouští z okna About, neboť nebyla tato záležitost dostatečně verifikována, tudíž se nabízí možnost ji v blízké době uvést do konečné podoby. Dalším možným vylepšením by mohla být inovace celé ovládací aplikace tak, aby splňovala budoucí požadavky v oblasti vzhledu i funkcionality.

Literatura

[1] Halvorson, Michael. *Microsoft Visual Basic 2008 Krok za krokem.* Brno: Computer Press, a.s., 2008. ISBN 978-80-251-2221-1.

[2] Halvorson, Michael. *Microsoft Visual Basic 2010 Krok za krokem.* Brno: Computer Press, a.s., 2010. ISBN 978-80-251-3146-6.

[3] Petroutsos, Evangelos. *Myslíme v jazyku Visual Basic .NET - 1. díl.* Praha: Grada Publishing a.s., 2003. ISBN 80-247-0371-8.

[4] Petroutsos, Evangelos. *Myslíme v jazyku Visual Basic .NET - 2. díl.* Praha: Grada Publishing a.s., 2003. ISBN 80-247-0372-6.

[5] Fuchs, J., Barchfeld, A. *Visual Basic Velká kniha řešení.* Brno: Computer Press a.s., 2010. ISBN 978-80-251-2212-9.

[6] Roman, S., Petrusha, R., Lomax, P. *Visual Basic .NET v kostce.* Praha: Grada Publishing a.s., 2003. ISBN 80-247-0388-2.

Příloha A

Komunikace se zařízením CANtron

Ukázka vzájemného dorozumívání ovládací aplikace a funkčního vzorku zařízení, jenž byl sledován pomocí programu Free Serial Port Monitor:

Port opened by process "CANtron.vshost.exe" (PID: 5884)

Request: 30.5.2012 9:31:04.04164 (+176.4219 seconds)

1B 1B 1B 71 61 36 0A ...qa6.

Answer: 30.5.2012 9:31:04.04164 (+0.0000 seconds)

1B 71 61 36 0A .qa6.

Request: 30.5.2012 9:31:05.75964 (+1.7188 seconds)

1B 1B 1B 51 38 35 0A ...Q85.

Answer: 30.5.2012 9:31:06.77564 (+0.0156 seconds)

1B 51 38 35 0A .Q85.

Request: 30.5.2012 9:31:07.30664 (+1.5313 seconds)

1B 1B 1B 76 32 35 0A ...v25.

Answer: 30.5.2012 9:31:07.30664 (+0.0000 seconds)

1B 76 31 31 31 30 31 34 0A .v111014.

Request: 30.5.2012 9:31:07.30664 (+0.0000 seconds)

1B 1B 1B 73 31 61 0A ...s1a.

Answer: 30.5.2012 9:31:07.43164 (+0.1250 seconds)

1B 73 48 33 62 61 0A .sH3ba.

Request: 30.5.2012 9:31:11.08764 (+3.6563 seconds)

1B 1B 1B 6D 30 36 64 0A ...m06d.

Answer: 30.5.2012 9:31:11.10364 (+0.0156 seconds)

1B 6D 30 31 34 30 35 37 35 30 32 30 30 30 30 30 .m01405750200000

30 30 30 30 30 30 30 30 30 30 30 30 31 30 30 30 0000000000001000

30 30 30 30 30 30 30 30 30 30 30 30 32 30 30 30 0000000000002000

30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 0000000000000000

30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 0000000000000000

30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 0000000000000000

30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 0000000000000000

30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 0000000000000000

30 30 30 30 30 30 30 30 30 30 30 30 30 61 31 0A 0000000000000a1.