4.1. Ziel

Der (Die) Student(in) soll eine Schicht–2 Kommunikation mit anderen Praktikumsteilnehmern auf der Basis von Feldbussen aufbauen. In diesem speziellen Fall mit den CAN–Protokoll. Er (Sie) soll dies durch die Programmierung eines Microcontrollers, der einen On–Chip–CAN–Controller enthält, realisieren.

4.2. Versuchsanordnung

Die Versuchsanordnung (Abbildung 4.1) besteht aus einem Laptop und den C164CI–Experimentalsystemen. Der Laptop enthält eine PCMCIA–CAN–Karte und ist somit ein Busmonitor, der alle CAN–Nachrichten empfängt und anzeigt.

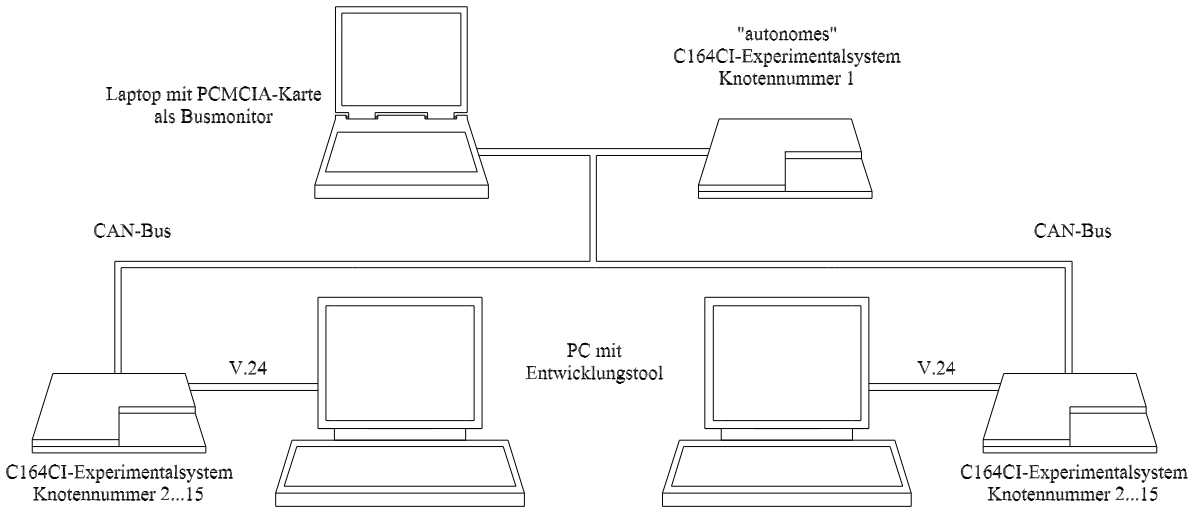

Abbildung 4.1.: Versuchsaufbau des Praktikums

Das C164CI–Experimentalsystem wurde von mir, Daniel Wolf, entwickelt, gebaut und im Kapitel 5 „Hardwarebeschreibung des C164CI–Experimentalsystem“ näher beschrieben. Die Programmierung des C164CI–Experimentalsystem, welches mit dem nanoModul–164 ausgestattet ist, erfolgt über den angeschlossenen PC mit entsprechender Software.

4.3. On–Chip–CAN–Controller

Auf dem nanoModul–164 befindet sich ein Microcontroller C164CI mit On–Chip–CAN–Protokollcontroller, der in der CAN–Spezifikation V2.0B arbeitet.

Es können 15 Nachrichtenobjekte (Messageobjekt) im Full–CAN–Modus erzeugt werden, die bis zu 8 Byte Daten enthalten. Das 15. Nachrichtenobjekt kann außerdem im Basic–CAN–Modus mit einem doppelten Empfangsbuffer betrieben werden. Beide Methoden unterstützen separate Masken für die Akzeptanzfilterung, welche den Identifier im Full–CAN–Modus aktzeptiert sowie den Identifier im Basic–CAN–Modus nicht beachtet. Beim Senden und Empfangen von Nachrichten sind 11 Bit oder 29 Bit Nachrichtenerkennungen (ID) möglich.

Während des Praktikums werden 11 Bit Identifier verwendet. Die Funktionalität des CAN–Controllers wird über eine Vielzahl von Registern [10] eingestellt (siehe Anhang ab Seite 71):

- **Control/Status Register ($\mathbf{EF00_H}$)**
- **Interrupt Register ($\mathbf{EF02_H}$)**
- **Bit Timing Register ($\mathbf{EF04_H}$)**
- **Global Mask Short Register ($\mathbf{EF06_H}$)**
- **Upper Global Mask Long Register ($\mathbf{EF08_H}$)**
- **Lower Global Mask Long Register ($\mathbf{EF0A_H}$)**
- **Upper Mask of Last Message Register($\mathbf{EF0C_H}$)**
- **Lower Mask of Last Message Register($\mathbf{EF0E_H}$)**
- **Message Control Register ($\mathbf{EFn0_H}$)**
- **Upper Arbitration Register ($\mathbf{EFn2_H}$)**

- **Lower Arbitration Register ($\mathbf{EFn4}_H$)**
- **Message Configuration Register ($\mathbf{EFn6}_H$)**
- **Data Byte 0 ($\mathbf{EFn7}_H$)**
- **Data Byte 1 ($\mathbf{EFn8}_H$)**
- **Data Byte 2 ($\mathbf{EFn9}_H$)**
- **Data Byte 3 ($\mathbf{EFnA}_H$)**
- **Data Byte 4 ($\mathbf{EFnB}_H$)**
- **Data Byte 5 ($\mathbf{EFnC}_H$)**
- **Data Byte 6 ($\mathbf{EFnD}_H$)**
- **Data Byte 7 ($\mathbf{EFnE}_H$)**

4.4. „autonomes“ C164CI–Experimentalsystem

Das „autonome“ C164CI–Experimentalsystem ist eine eigenständig arbeitende Einheit, im Verbund der Versuchsanordnung. Dabei besitzt es denselben technischen Aufbau wie die anderen C164CI–Experimentalsysteme.

Durch ein Programm, welches sich im Flash (EEPROM) des nanoModul–164 befindet, werden verschiedene Softwarefunktionen bereit gestellt. Diese Funktionen sollen im Praktikum verwendet werden.

Die Abfrage der Funktionen läuft über den CAN–Bus ab, wobei alle Funktionen durch einen entsprechenden 11 Bit Identifier abgerufen werden können bzw. automatisch gesendet werden. Auf Seite 82 sind die einzelnen Identifier und die dazu gehörigen Funktionen aufgelistet. Das Senden der Nachrichten kann mit folgenden Baudraten erfolgen.

50 kBaud :	BTR = 0x7AC9
100 kBaud :	**BTR = 0x34C9 für das Praktikum**
125 kBaud :	BTR = 0x7AC3
250 kBaud :	BTR = 0x7AC1
500 kBaud :	BTR = 0x7AC0
1 MBaud :	BTR = 0x25C0

4.5. Praktikumssoftware

Als Praktikumssoftware steht das Tool **TASKING Embedded Development Environment (EDE)** zur Verfügung, welches eine 4 kByte Quellcode begrenzte Demoversion ist und für die Programmierung von Microcontroller entwickelt wurde. Die Entwicklungssoftware enthält C–Compiler, Assembler, Linker/Locater und eine Cross View Pro Debugger.

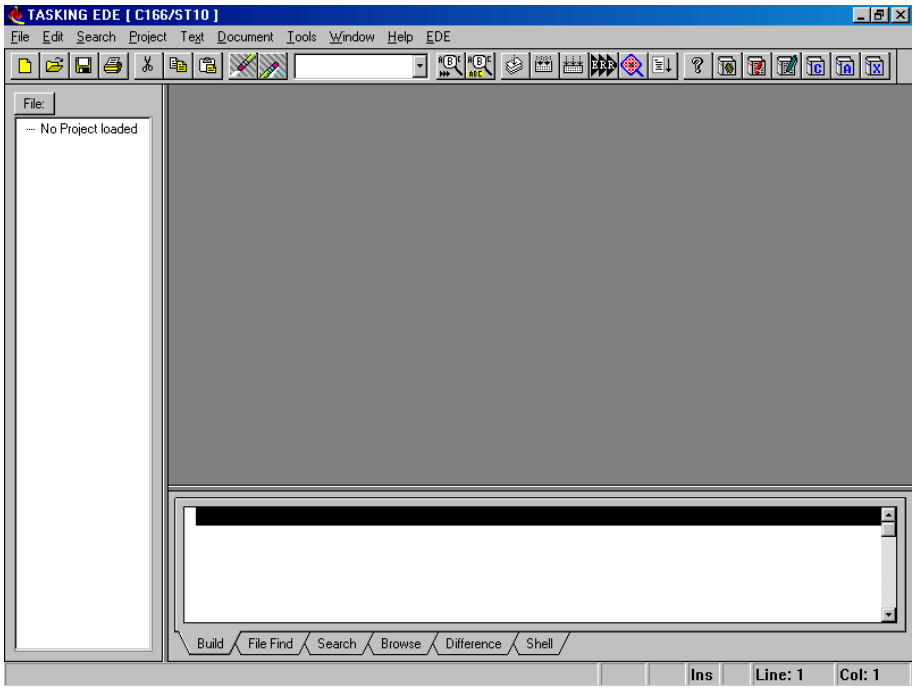

Abbildung 4.2.: Tasking Entwicklungstool

Durch das Starten der Datei „EDE.EXE“ bzw. dem Anklicken des Icons auf der Windowsoberfläche wird das Tools geladen. Das EDE ist eine menügesteuerte Arbeitsoberfläche (Abbildung 4.2), von der aus alle Einstellung (Schnittstelle, Baudrate, Microcontroller) vorgenommen werden und die Erstellung des Programmes erfolgt.

Nach dem erfolgreichen Compilieren und Erstellen eines entsprechenden Makefiles (Menü: Project/Build – *.abs) wird die Cross View Pro Oberfläche (Abbildung 4.3) geladen. Zuvor versucht der PC eine Verbindung (RS–232) zum nanoModul–164 aufzubauen, in dem er den Bootstrap Loader anspricht und somit das Modul veranlasst, das aktuelle Programm zu laden.

Die Cross View Pro Oberfläche ist ein Debugger, der alle Eigenschaften besitzt, die zur Abarbeitung und zum Test eines Programmes nötig sind, wie Einzelschrittbetrieb, setzen von Breakpoints, Anschauen der Register und Speicheradressen und vieles mehr.

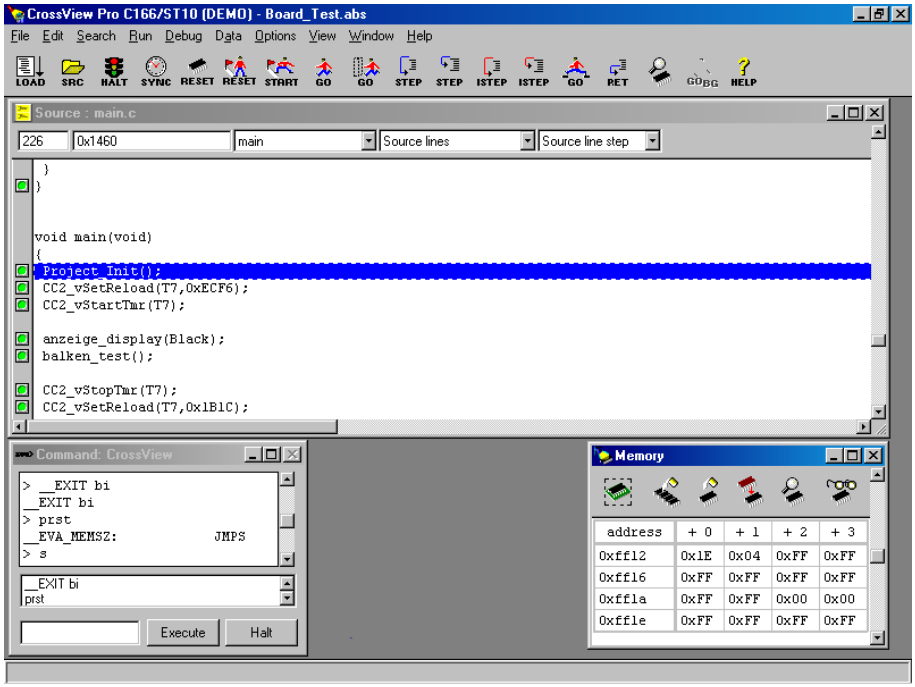

Abbildung 4.3.: Cross View Pro Debugger

4.6. Aufgabenstellung und Vorbereitung

Die Hauptaufgabe besteht darin, den On–Chip–CAN–Controller des nanoModul–164 zu programmieren, d.h. Definieren der einzelnen Messageobjekte (Senden oder Empfangen, Datenlänge festlegen, Interrupts auslösen uvm.) und somit eine Kommunikation zwischen den einzelnen C164CI–Experimentalsystemen auf dem Schicht–2–Niveau zu realisieren.

Als Vorbereitung auf das Praktikum sollte man sich mit dem Tasking–Tool, dem Microcontroller C164CI und dem Aufbau der C164CI–Experimentalsystem vertraut machen. Dafür erhält man die entsprechende Dateien beim Betreuer oder lädt sie sich von der Fachgebietshomepage herunter (Demoversion, User Manual des C164CI und C164CI–Experimentalsystem).

Die folgende Aufgaben sollen während des Praktikums realisiert werden:

1. Es soll ein 8 Bit Wert im Gerät gehalten und auf der Anzeige als Hex–Wert angezeigt werden.
2. Der 8 Bit Wert soll inkrementiert („+“) und dekrementiert („-“) werden.
3. Kommunikation mit anderen Teilnehmern

   3.1 Definieren eines Messageobjektes mit dem aktuellen Hex–Wert (Datenbyte 0).

      3.1.1 Auf Anfrage eines anderen Teilnehmers (durch Remote Frame) soll das Messageobjekt mit dem aktuellen Hex–Wert übertragen (Request) werden.

   3.2 Definieren eines Messageobjektes für das Ermitteln des Hex–Wertes von einem anderen Teilnehmer (Datenbyte 0).

      3.2.1 Senden des Messageobjektes (als Remote Frame), um den Hex–Werte vom anderen Teilnehmer anzufordern.

3.2.2 Die Anforderung des Hex–Wertes soll zyklisch (Schleife oder Timer) und auf Tastendruck erfolgen.

3.2.3 Der Hex–Wert soll auf der 7–Segmentanzeige dargestellt werden.

4. Kommunikation mit dem „autonomen“ C164CI–Experimentalsystem

4.1 Definieren eines Messageobjektes für das Empfangen der aktuellen Uhrzeit.

4.1.1 Die aktuelle Uhrzeit soll auf der 7–Segmentanzeige dargestellt werden.

Anmerkung: Das Telegramm mit der aktuellen Uhrzeit hat 6 Datenbytes. Bei einer Uhrzeit von z.B. 17:45:23 steht im ersten Byte eine 0x01, im zweiten 0x07, im dritten 0x04 usw...
Achtung: Der Identifier für dieses Telegramm wird von Jahr zu Jahr verändert.

Die Festlegung (Abbildung 4.4) der Schiebeschalter (S1...S16) und Taster (SW3...SW10) wurde laut der Aufgabenstellung vorgenommen.

Wie bei der Versuchsanordnung (Abbildung 4.1) auf Seite 31 ersichtlich ist, besitzt jeder Teilnehmer eine feste Knotennummer (1...15), die sich im 11 Bit Identifier widerspiegelt. Weiterhin erfolgt eine Unterteilung des Identifiers in Nachrichtentyp, der wiederum durch den Nachrichtenfeincode unterteilt ist. Damit ist es möglich, die einzelnen Nachrichten der Teilnehmer (Tabelle 4.1) zu unterscheiden und Konflikten aus dem Weg zu gehen. Die Abbildung 4.5 soll dies noch einmal verdeutlichen.

Das Programm soll in C mit der Entwicklungssoftware EDE erstellt werden. Der Test wird mit dem Cross View Pro Debugger vorgenommen.

Die Zuordnung der Schiebeschalter und Taster zu den Datenbusleitungen sollte im Abschnitt 5.5 nachgelesen werden (Schiebeschalter: Bit 0 (S1, S9) und nach rechts steigend, Taster: Bit 0 (SW3), Bit 4 (SW7) und nach rechts steigend).

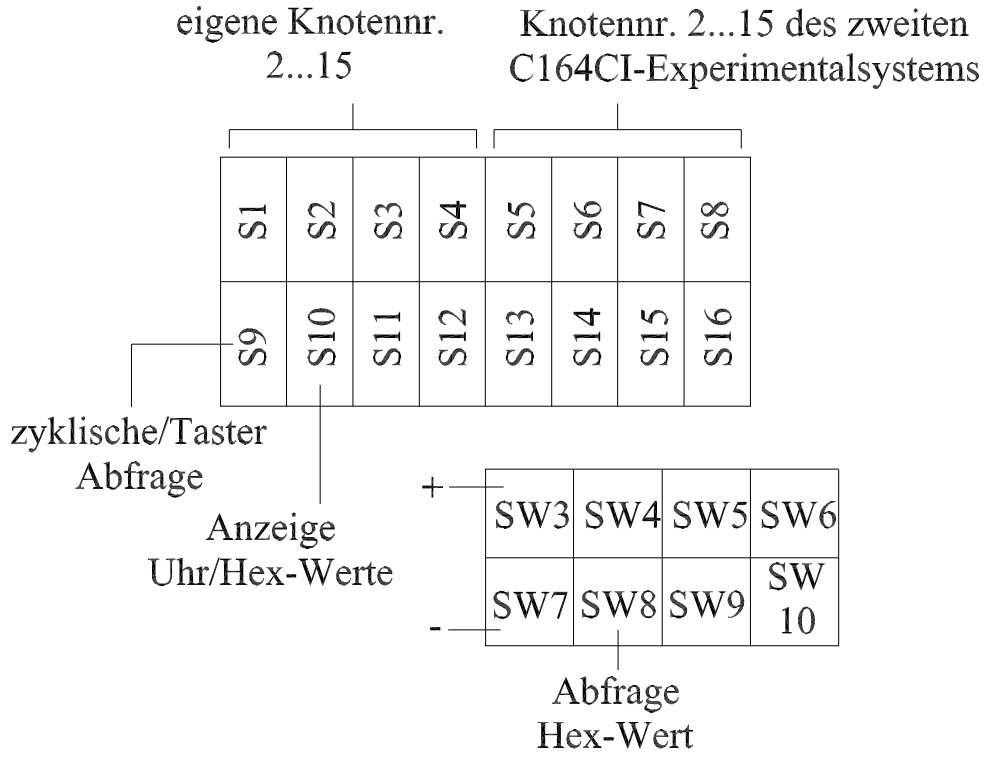

Abbildung 4.4.: Belegung der Schalter und Taster beim Praktikum

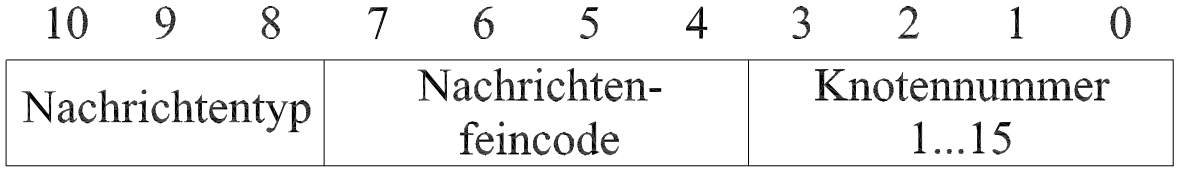

Abbildung 4.5.: Aufbau des 11 Bit Identifier beim Praktikum

Nachrichten		**Nachricht**
Typ	**Feincode**	
0	0	zyklisches Senden der Uhrzeit vom „autonomen“ C164CI–Experimentalsystem
2	1	Anfordern des zweiten Hex–Wertes vom anderen C164CI–Experimentalsystem

Tabelle 4.1.: Aufteilung des 11 Bit Identifiers beim Praktikum