

# Final report

The Tiny Home Automation System

Pavel Kučera

Huddersfield  
16MAY2001

**Motto:**

17. Utere quaesitis modice, cum sumptus abundat,  
Labitur exiguo, quod partum est tempore longo.

*Disticha Catonis, Liber II*

## 1. Introduction

There are good descriptions of the Tiny Home Automation System (THAS) in [1] and [2]. Reader must be familiarised with these reports first, otherwise many parts of this report will not be comprehensible.

What this work does consist of? Simple question – difficult answer, since many things were created or improved, but final effect does not look like it. Notably because author's work during summer time period in laboratory of School of Engineering was separated into 3 main parts:

- First part consists of starting the THAS in operation (the most difficult aspect of the work – mostly because of documentation).
- Second part consists of electronic design of some parts of the Interface and some software design for infrared controller.
- Last part consists of design software (COMINT) for PC.

## 2. Start the THAS

Good documentation is 70% of success (or more). Unfortunately usual problem of many developers (include me) is disregarding this aspect of engineering work. When we first time tried to start THAS the main problem was in documentation, mainly documentation that describes electronic part of the system. There was no electronic scheme in documentation of our precursors! There was only several particular part of system in [1]. And this was not really enough. So author's first activity was in drawing scheme by return. After it many things were clearer and also it is necessary to say thanks to Ing. Cach, his support mainly at the beginning of work was considerable. Following things are very important if you are going to start THAS.

### 2.1 CAN network

Current state of the project is oriented to X10 system, so probably following information is not important for you. But if you start THAS with CAN network be sure that:

All nodes that are in configuration stream in Interface are connected. Otherwise invalid unit will be ignored and in Monitor Manger [3] will not this segment work.

Very important is to power ON Universal Unit first and the Interface later! And power ON means not only processor development board but also I/O board where EEPROM with configuration of universal unit is set! See [2] and [1] for details. This behaviour

of the system is very unsuitable and half-baked. Many hours we spent to observing what is rotten in the state of Denmark and only happy accident detects this „mistake”.

CAN network is properly terminated. UU2 has terminators on board so it is good idea to connect this board to the end position of CAN network. Also is important to ensure that network is terminated on BOTH sides of net – especially if you use network installed in laboratory. Many times we discover that terminator is missing, although we yesterday put it there. For this purpose it is very good idea to use oscilloscope to determine if signals on network are correct – using oscilloscope is also easy to detect basic communication problems like broadcast message is not produced or Interface is not working.

Ensure that both UU has unique CAN IDs and theirs program are adequate. At the beginning we discovered that one UU has not EPROM with program but with monitor. Next five days we tried to find correct version of UU program on computer and translate it and transfer to EPROM – but this is another problem.

Also be sure that if you are using Monitor/Manager you use correct connector of CAN Card in PC. This Card has ONLY ONE CAN driver and this driver is connected to TWO CAN connectors (parallel connection) on this PC card – see technician description of card.

Finally be very, very patiently and sometimes it is better go to the pub than spent ten of hours making the same thing without any result – believe me I know very well what I am talking about.

## **2.2 Development tool**

Our precursors decided develop all system in C under C51 compiler. Short description of this KEIL development software you can find at [2] and at [4] you can find everything what you want. With this development tool I have many experiences, mainly displeasing. Mainly debugger is very unstable and often the communication driver breaks down without any reason. Later I found out (working with this communication driver during development of COMINT) that this is not problem of the development tool, but problem of Windows. Also very often it was impossible to start compiler – it was necessary many times restart system. These problems are not if you are using Windows 95 or later version so if you can (and I could not) try to continue in development on such computer.

Start THAS was painful work and after such experience I hope that our descents will have better position. May you be happy!

### 3. Electronic Design

Because does not exist any schematic diagram of THAS before than I decided to do it, many part of this chapter will be confused. But this chapter is very important because describes what is changed in comparable with original version [1]. Also this chapter describes what is not valid in [1] and what part of program in [1] cannot work with this new electronic design. Following information is referring to the Interface.

#### 3.1 Rewiring the I/O connection

Our first problem was that not enough free I/O pins of C505 were presented with this purpose is the I/O structure of C505 changed like you can see in figure 1.

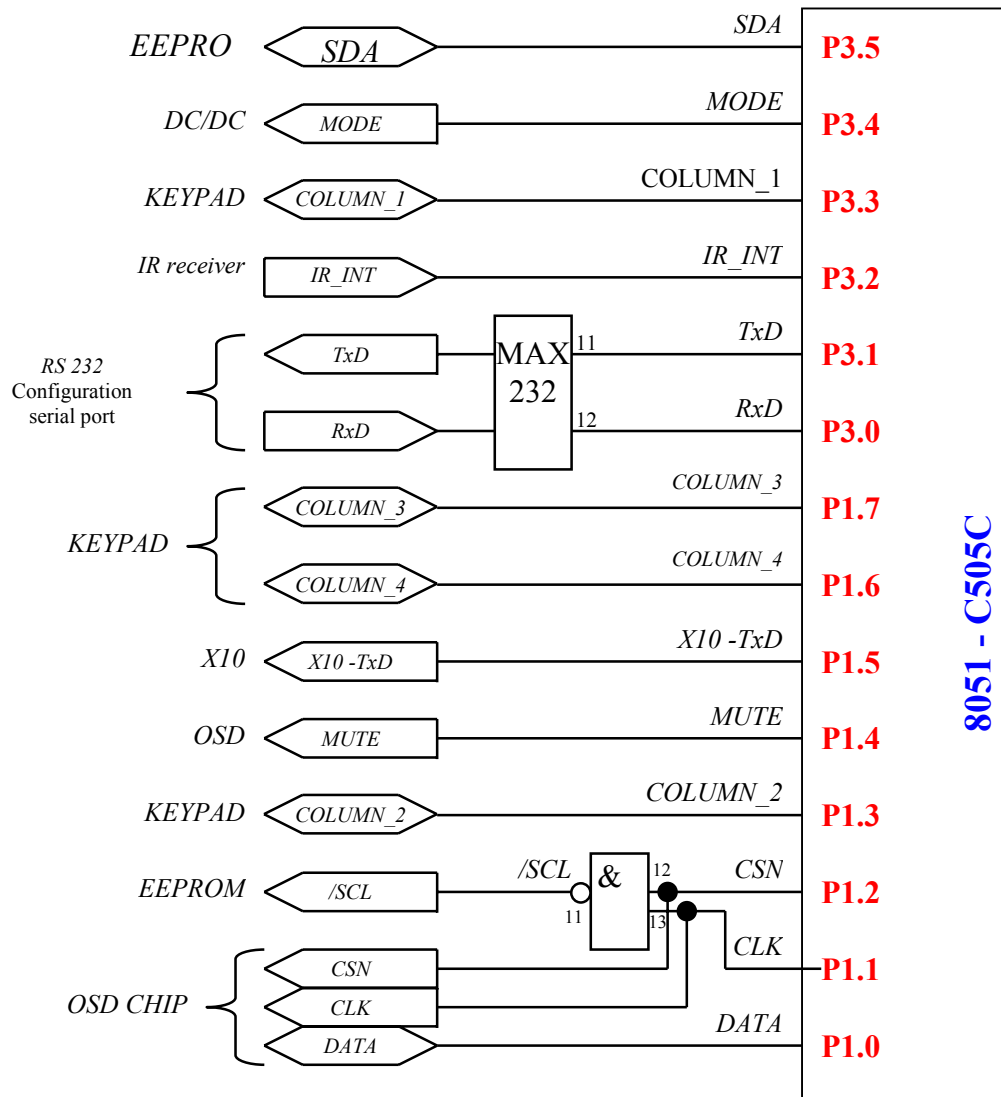


Figure 1 – final wiring of C505C I/O ports

**Legend:**

<i>SDA</i>	- Serial data (I/O data for I <sup>2</sup> C Bus of EEPROM)
<i>MODE</i>	- Output pin is used for selecting appropriate screen mode – see chapter about DC/DC converter
<i>COLUMN_1</i>	- I/O pin for reading information from keyboard see chapter about keyboard decoder
<i>IR_INT</i>	- External interrupt pin is used for reading a serial code of the infrared receiver
<i>TxD</i>	- Output pin is used for RS232 communication with COMINT or development tool (C51 debugger)
<i>RxD</i>	- Input pin is used for RS232 communication with COMINT or development tool (C51 debugger)
<i>COLUMN_3</i>	- I/O pin is used for control keyboard decoder – see chapter about keyboard decoder
<i>COLUMN_4</i>	- I/O pin is used for control keyboard decoder – see chapter about keyboard decoder
<i>X10-TxD</i>	- Output pin is used for communication with X10 interface
<i>MUTE</i>	- Input pin is used for detecting stable television signal of OSD chip, see chapter about OSD
<i>COLUMN_2</i>	- I/O pin for reading information from keyboard see chapter about keyboard decoder
<i>/SCL</i>	- Inverted output pin is used for synchronisation of data transfer between EEPROM and the Interface
<i>CSN</i>	- Inverted chip select output pin is used during data transfer between OSD chip and the Interface
<i>CLK</i>	- Output pin is used for synchronisation of data transfer between OSD chip and the Interface
<i>DATA</i>	- I/O pin is used for data bus during data transfer between OSD chip and the Interface

Such connection structure is absolutely incompatible with the software that is described in [1]. Thus it is important take into account new version of the software that was developed together with the new electronic design.

### **3.2 EEPROM wiring**

Originally the EEPROM of the Interface was connected to the C505C by 2 its pins (P3.5 and P3.4). Function of the EEPROM pin SCL (serial clock) is derived from function CSN of the OSD chip (P1.2). Because at the same moment cannot be both devices operationally. Thus pin CSN is used for gating the CLK signal (P1.1) such a way that either OSD chip CSN is activated using NAND gate (log. 0 on pin P1.2) or EEPROM clock are provided from CLK signal originally reserved for OSD chip. It means that one pin was saved and this pin was later used for reading information from infrared receiver. Such a way it is possible to save next 2 pins of the I/O of the interface.

### 3.3 Keyboard decoder

Keyboard electronic design was changed with the plan to reduce number of chips on development board. Originally four chips were used [1] for reading the state of 12 keys keyboard. New design more simplify electronic design – only 6 diodes is necessary- and new algorithm of reading was written. In figure 2 is showed new decoder of keyboard with 6 diodes and 4 I/O pins. This decoder decreases number of necessary active components to minimum and is as well functional like its precursor was.

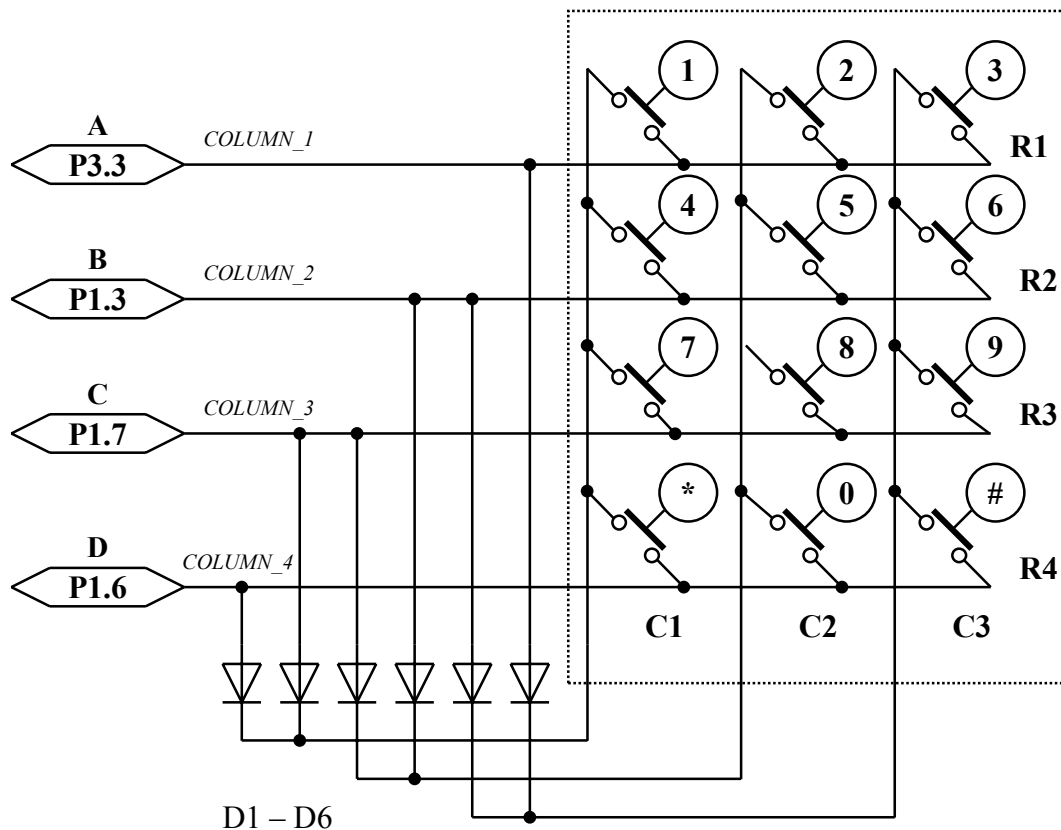


Fig 2 – Schematic diagram of the keyboard decoder

This new keyboard decoder requires new software in the Interface. Simple algorithm is described and implemented in C language in Interface see fig. 3. This circuit in fact simulates diode logic and its function is simple and I think that is not necessary to describe it. After all, the algorithm in fig. 3 is more than expressive.

At this moment attentive reader could put the question: „Why 6 diodes is used. 4 of them are not enough?“ like putting me my fellow a time ago. Really 6 diodes is a minimum. The answer to this question is simply but first you must analyse this circuit and find out it. Otherwise you must believe me.

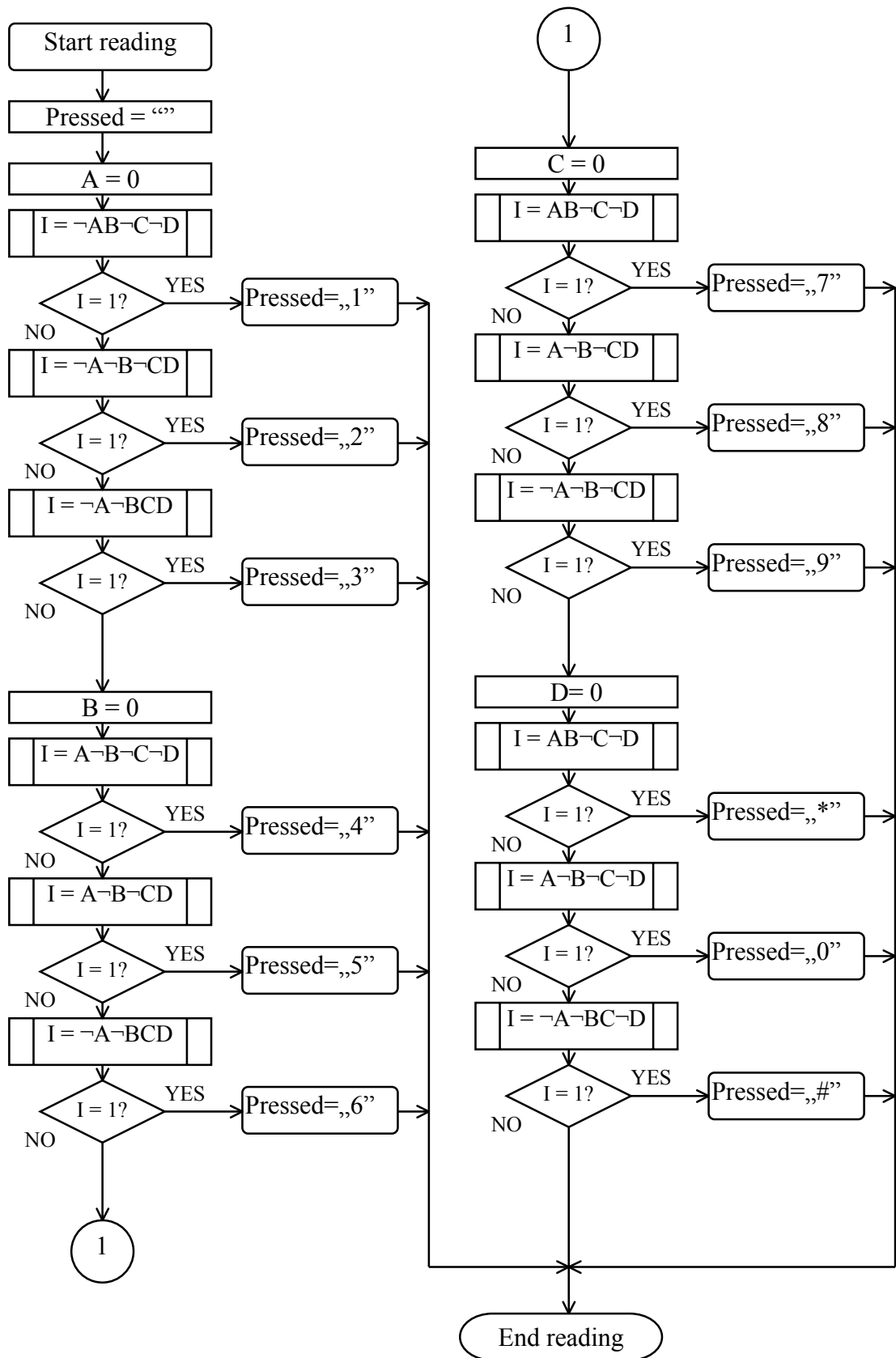


Figure 3 – Algorithm of decoding keyboard

### 3.4 OSD chip – DC/DC converter

Detailed schematic diagram of STV5730A you can find in final report of my fellow, who redrawn my sketch of circuit into electronic form (I hope). Fortunately this electric scheme is practically the same as SGS-Thomson datasheet – on page 6. Following variances have not important influence:

- Input video signal into pin 2 (video in – SCART pin 20) pass through integrated circuit that consists of resistance  $75\Omega$  and capacity  $2,2\ \mu\text{F}$ . In fact the resistance only decrease input spectrum of incoming signal degrading it. Up to this day I don't know why is this resistance there and in accordance with my opinion it is nonsense.
- Also interesting change consists in power supply circuit of the chip. In original documentation is device supplied through band-pass filter what is comprehensible while this filter is missing in fact. The reason may be that was difficult build this filter (because consists  $10\mu\text{H}$  coil). But in future, when final version of circuit will be prepared I strictly advice to incorporate this filter into circuit. Not all what is operational in laboratory is operational in another setting.
- The rest of changes like voltage dividers before amply transistors are comprehensible.

Another part of work is oriented to the possibility to change TV mode by C50C chip. The OSD chip can operate in two different modes: mixed mode and full-page mode. For detail description see [5]. Main problem in mixed mode is stabile television signal. Because the text is superimposed with this signal, horizontal and vertical synchronisation pulses must be present inside its. And main problem in full-page mode is appropriate setting of TV.

We solved this problem using signal MUTE from OSD chip and in virtue of this signal can Interface decide if it is possible to show text in mixed mode (stabile television signal) or in full-page mode (AV mode of TV is required). While detecting of signal MUTE is simple matter, changing TV from antenna signal to RGB signal is difficult because standard SCART connector (pin 8) requires for this action voltage  $9,5\ \text{V}$  minimally. It is good example of using a DC/DC converter in accordance with my opinion. But not simple device that is able produce as many as  $500\text{mA}$  but simple circuit that produces only several mA but is cheaper.

In figure 4 is a schematic diagram of such converter including analogue switch. This circuit was designed exactly for our purpose, it is multiplied 2x and maximally output current is  $5\ \text{mA}$ . Restriction of output current is accordance with multiplied coefficient.

System consists of astable flip-flop circuit (components A, B, C, R1, and C1), DC/DC converter (components T1, T2, D1, D2, R2, R3, C2, C3 and C4) and analogue switch (T3, T4, R4, R5, R6 and R7).

Function of astable flip-flop circuit is transparent and it is not necessary to describe it. Pulses from this circuit are used like control signal for following circuit DC/DC converter.

This converter change  $+5\text{V}$  to approximately  $9,5\ \text{V}$ . Function of this circuit is based on two complementary transistors T1 and T2. These transistors during several

periods of clock signal from flip-flop circuit gradually charge the capacitors C2 and C3 through diodes D1 and D2. Thus the cathode of diode D2 has approximately 9.5 V.

Output of this converter is fetched to analogue switch that is controlled by output pin P3.4 of C505 microcontroller. Basic component is transistor T4 that is able to switch multiplied voltage from the converter using standard TTL voltage levels. This TTL signal is fetched to the transistor T3 and this transistor changes the voltage level 0-5V to 0-9.5V. This voltage range is suitable for T4 because when this voltage range is on its Gate pin, output voltage range is 1.5 - 9.5 V what is sufficient.

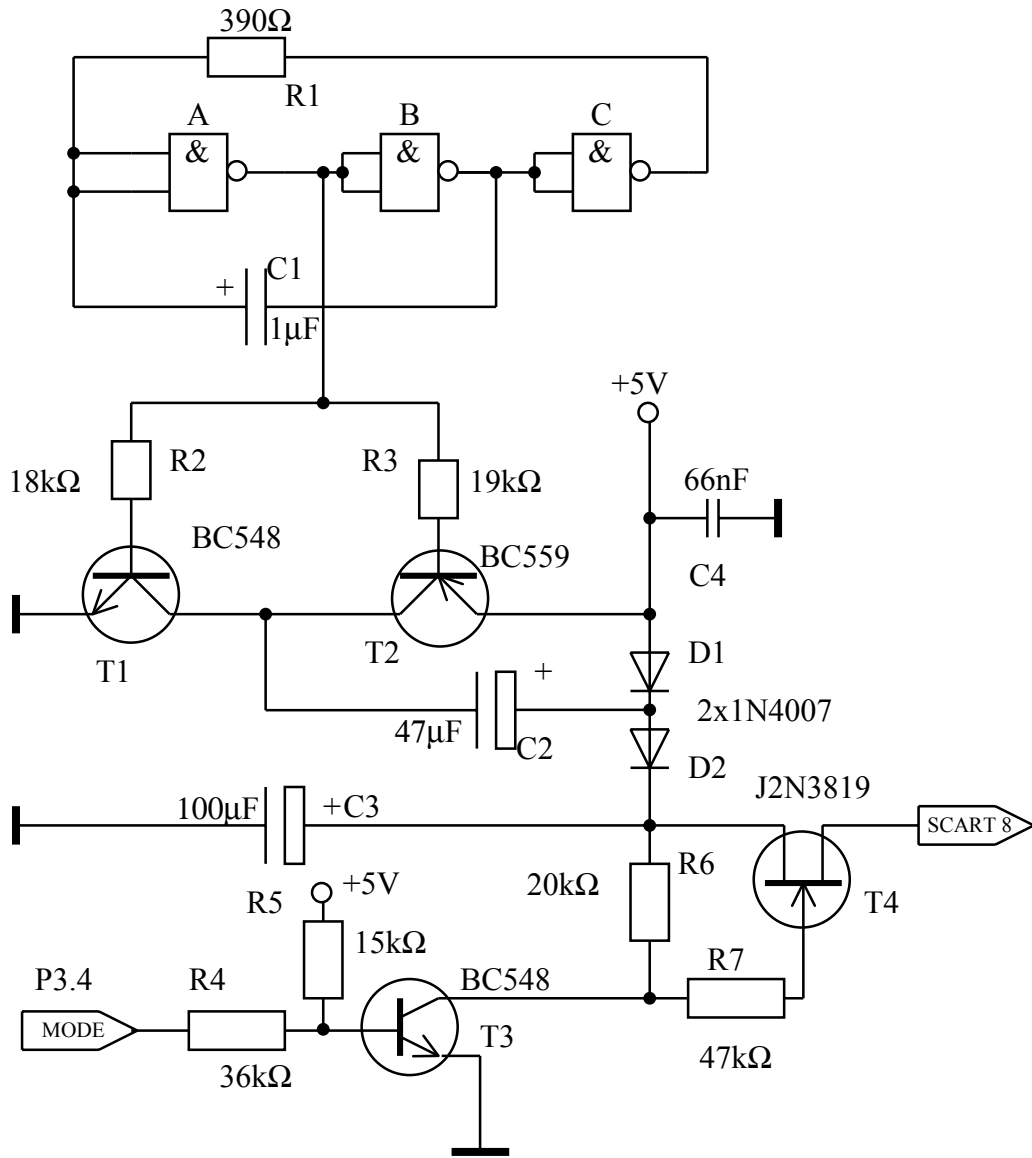


Figure 4 – DC/DC converter and analogue switch

### 3.5 Infra Red receiver

Infra red receiver is a new element in THAS. Its schematic diagram is very simple, just IR receiver, one diode and one capacitor is used – figure 5.

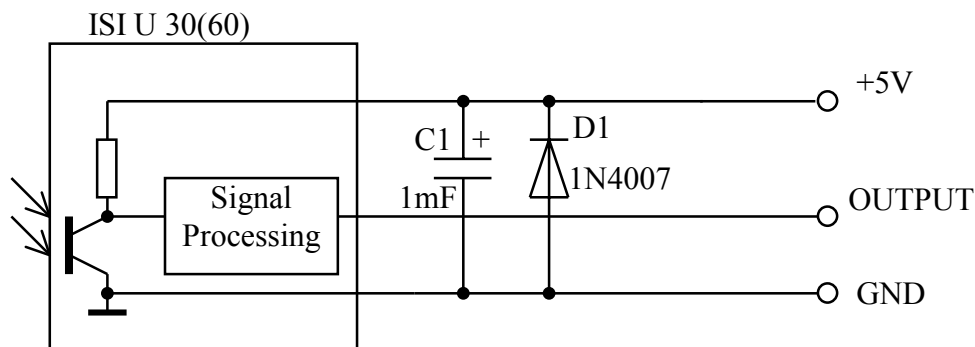


Figure 5 – IR receiver schematic diagram

Because during development many wires are on the table and oftenly it is very easy to find one end of wire that is not connected it is very practical to protect sensitive components (like IR receiver) against unsuitable voltage levels. This is purpose od diode D1. Purpose of capacity C1 is that whole circuit is detached from power suply and this device (IS1U) is sensitive to voltage drop so this capacity serves like local power suply.

How you can see hardware part of IR receiver is simple, not so software. For receiving information from infrared controller was used RC5 code because of its spread. Short description of RC5 code follows:

The RC5 code set was developed by Phillips and allows 2048 commands to be transmitted divided into 32 addressable groups of 64 commands each. The transmitted code consists of a 14bit data of the following structure.

2 run-in bits to adjust the AGC level in the receiver IC

1 toggle bit

5 system address bits (table 1)

6 command bits (table 2)

These have been assigned as follows:

Table - 1

<i>System address</i>	<i>Equipment</i>
0	TV set
2	Teletext
5	Video recorder
7	Experimental
16	Preamplifier
18	receiver/tuner
17	tape/cassette recorder
19	experimental

Table - 2

<i>Command code</i>	<i>Function</i>	<i>Command code</i>	<i>Function</i>
0-9	0-9	24	treble +
12	standby	25	treble -
13	mute	26	balance right
14	presets	27	balance left
16	Volume +	48	pause
17	Volume -	50	fast reverse
18	brightness +	52	fast forward
19	brightness -	53	play
20	colour saturation +	54	stop
21	colour saturation -	55	Record
22	bass +	63	system select
23	bass -		

Not all codes are used in our project.

The code is transmitted in biphase format (differential Manchester code).

In this system, logic 1 is transmitted as a half bit time without signal, followed by a half bit time with signal. Logic 0 has exactly the opposite structure. See fig. 6.

Each half bit consists of 32 shorter pulses. Each transmitted bit has a length of 1.778 ms, the shorter pulses have a pulse width of 6.9444  $\mu$ s on time and 20.8332  $\mu$ s off time. A complete data word has a length of 24.889 ms, and is always transmitted completely. If the key is held pressed the code is repeated in intervals of 64 bit times (i.e. 113.778 ms).

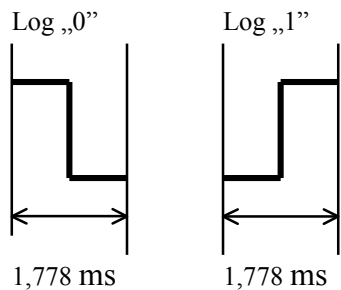


Figure 6 – timing of pulses

What we must do is to connect output of circuit in figure 5 to the C505 microcontroller. In figure 1 you can see that it is pin P3.2. This pin enables external interrupts, what is necessary condition for prosperous receiving and decoding RC5 code. Algorithm receiving and decoding this code is in figure 7 and 8.

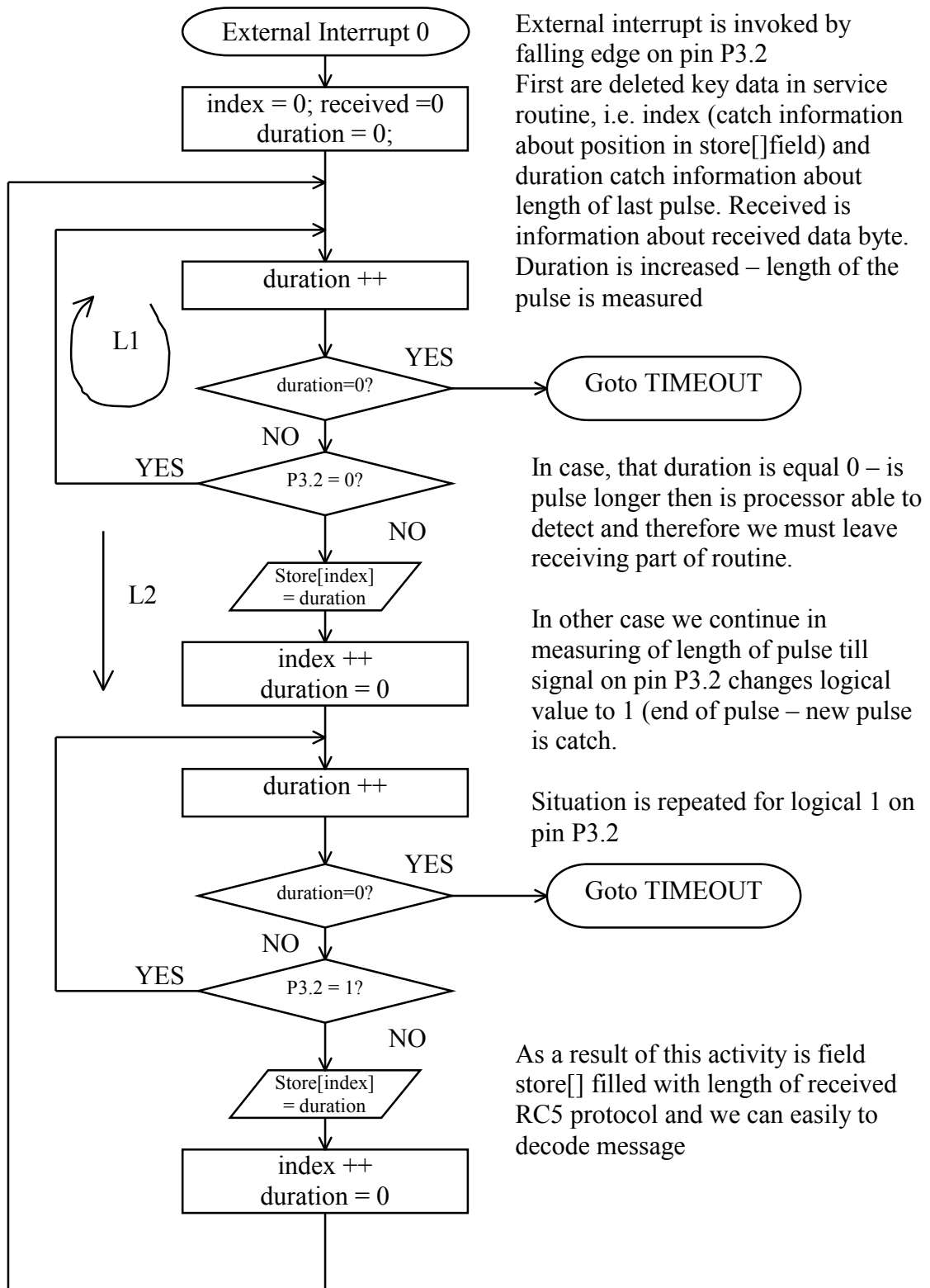


Figure 7 – Receiving part of algorithm

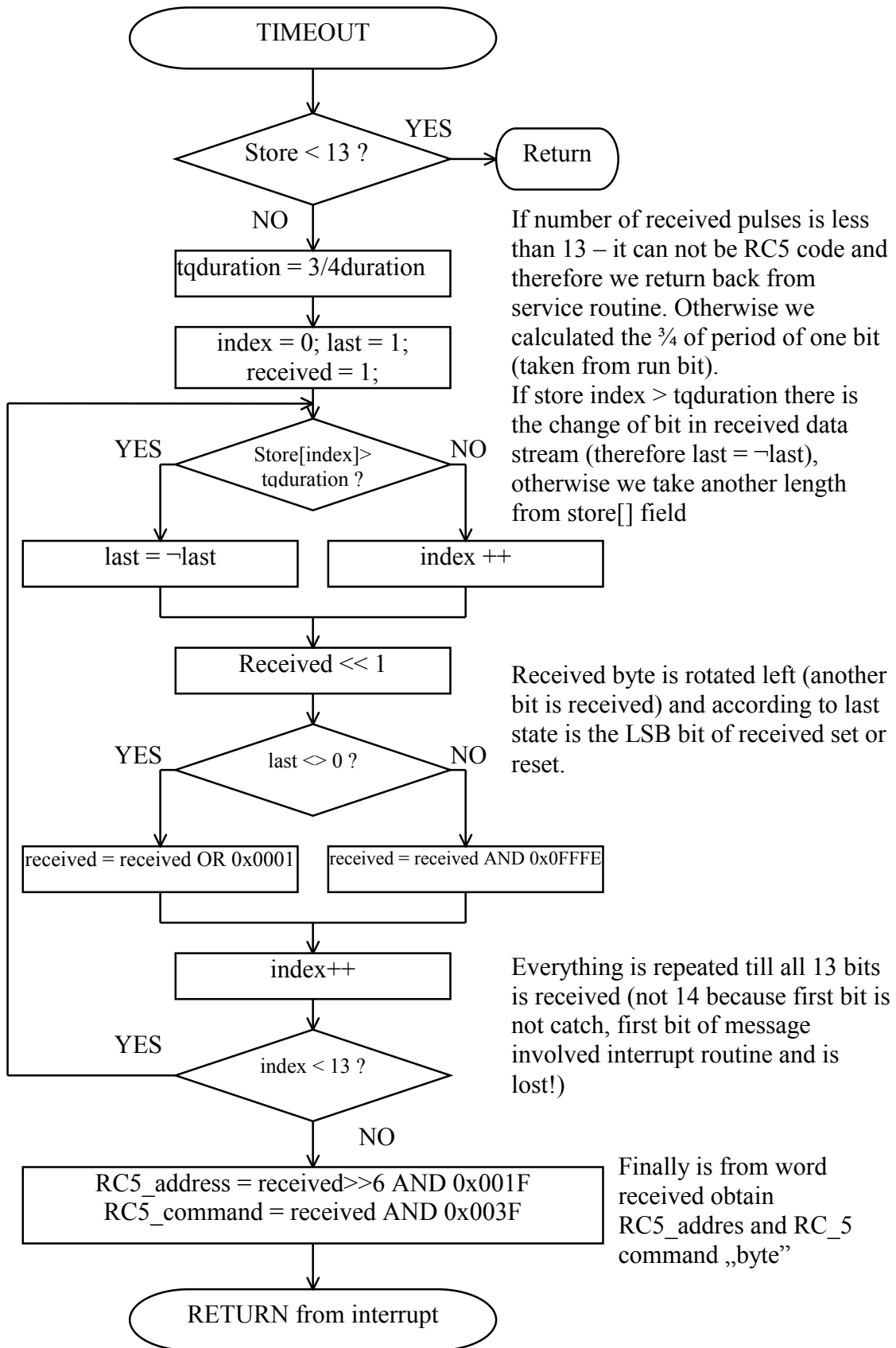


Figure 7 – Decoding part of algorithm

This algorithm is independent on length of pulse because decoding of pulse is calculated relatively to the received length. For instance, minimal resolution of receiving one pulse if clock frequency is 16Mhz:  $t_{\min} = 13 \cdot 12 / 16 \text{Mhz} = 9,75\mu\text{s}$ .

13 means number of tact of instruction in loop L1 (see Fig. 7)

12 is duration of machine cycle (8051 instruction)

It means that if we receiving pulse that has period 1,778ms its logical „1” part has 889 $\mu\text{s}$  (or logical „0” has 889 $\mu\text{s}$ ) and loop variable „duration” achieve  $889 / 9,75 = 91$ . If any inaccuracy is appear, we are able to measure pulse that has maximum length  $t_{\max} = 2,496 \text{ ms}$  ( $256 \cdot 9,75\mu\text{s}$ ). Pulse that is length means end of message.

We also need some time to change the measurement of logical „0” duration to logical „1” duration. This time is from L2:  $t_{\text{trans}} = 29 \cdot 12 / 16 \text{Mhz} = 21,75\mu\text{s}$ . It means that 21,75 $\mu\text{s}$  of the following pulse are lost. But because length of half pulse is 889 $\mu\text{s}$  this time is only 2.5% what is acceptable error of measurement.

In decoding part of algorithm is calculated  $\frac{3}{4}$  of duration the half pulse. This time interval *tqduration* is the basic interval that we use for decision if inside the 1,778ms interval was the pulse changed. This is the strategy of the independent measurement of pulse length. We can suppose, that if at the beginning of message was certain duration of 0 to 1 or 1 to 0 transitions, these duration will be produced also at the rest of message. This premise is important and how later used IR controller showed us, its lengths of pulses of RC5 code were more than non-standard, nevertheless microcontroller receives this code correctly.

Also must be mentioned one matter. Service routine was written in C. Statement *goto* was used for quickly dereliction from the receiving part of routine. Before then reader starts marvel at it (like my fellow did) I must explain one thing, if it is not obvious by now. Timing is in external service routine **VERY** critical. How you can see from paragraphs above all times are exactly calculated and one instruction more or less means destroying whole receiving algorithm. Statement *goto* is not often used in C language, but it is this statement that has MINIMAL duration of execution after translation into assembler! Any different structure (like *while*, *do* or *return* need minimal 2 times more time for execution).

### 3.6 Rest of electronic design

The rest of electric components like MAX323 converter for X10 protocol or connection of data transfer between OSD chip and microcontroller were not changed.

I hope that completely documentation of electric scheme reader can find in final year project of my fellow F.Lahner.

## 4. COMINT.exe

Rest of this report is interested in the program for PC that is called COMINT (COMMunication with INTerface). Main aim of this program is to create a configuration of the Interface's EEPROM in PC and then transfer it into the Interface using serial port RS232. Also this program allows saving older configuration files on hard disk or work with more than one configuration stream at the same moment.

### 4.1 Changes in configuration structure

It was not our aim to create serious changes in data structure of the configuration that was created and implemented of our precursor. Mainly because Monitor/manager is finished and we suppose back compatibility of new features with this PC software. But some new features were implemented because configuration stream can be saved on hard disk. Following table 3 resumes final structure of the Interface configuration:

Table - 3

<i>loaction</i>	<i>data</i>	<i>comment</i>
0 . . . 1755	CONFIGURATION	<i>This section contains configuration structure like was described in [1] without any change</i>
1756 . . . 1935	COMMENT	<i>Comment is short description of current configuration structure. It can allow 180 characters maximally (including non-visible characters). Comment is zero terminated string.</i>
1936 . . . . 2025	FILENAME	<i>Filename describes the path and filename of a file on hard disk of PC. This is useful information if the user is not sure what file on disk is downloaded in the Interface. This string is also zero terminated and maximum size is 90 characters.</i>
2026 . . 2035	DATE OF CREATION	<i>Date of creation of the configuration in PC or interface. 10 zero terminated character is maximum.</i>
2036 . . 2045	DATE OF LAST MODIFICATION	<i>Date of last modification of the configuration in PC or Interface. 10 zero terminated character is maximum.</i>
2046 2047	CHECKSUM	<i>16 bit checksum is calculated from 0 to 2045 data bytes</i>

The position of the added information of the configuration (1756) is selected that way to maximal configuration of screens and buttons (4 screen and 9 button inside each) does not exceed this position. It also allows downloading configuration streams through CAN neither any conflict is occurred.

Fields *date of creation* and *date of last modification* have following structure:  
DDMMYYYY\0 where DD is day 00 to 31  
MMM is month JAN, FEB ... NOV, DEC  
YYYY is year 0000 to 9999  
\0 is null terminating character

*Checksum* is important item of configuration stream. Because this stream is transferred by serial link and some interference may be occurred then this simple 16bit checksum can protect transfer against them.

## 4.1 Introduce into COMINT

If the Comint is executed the user can see window like in fig. 8. Basic actions are described by callouts in figure and when user shows by mouse pointer to the appropriate toolbar icon at the bottom of the window short description of tool is showed. Create new file (also File:New or Ctrl+N) command creates new configuration file. This configuration includes no screen and no buttons and configuration parameters of the Interface are set to their default values (see description of interface screen). Open file (also File:Open or Ctrl+O) command opens configuration file from disk, using open dialog window. If this configuration file (\*.cfg) is not valid user is refer on it and opening is cancelled. Another tool Setup communication parameters (also Comm:Setup or Ctrl+T) allows user to change basic communication parameters of the serial port (see description of communication). If no document is opened (what is ordinary after start the program) only Upload command (Comm:Upload also Ctrl+O) is allowed.

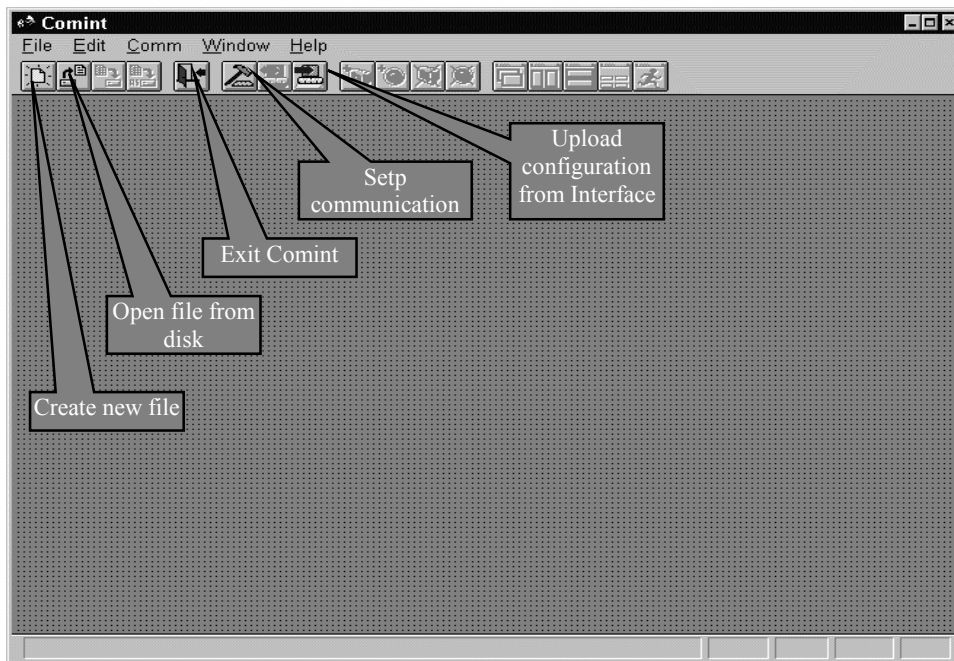


Figure 8 – basic window after start of comint

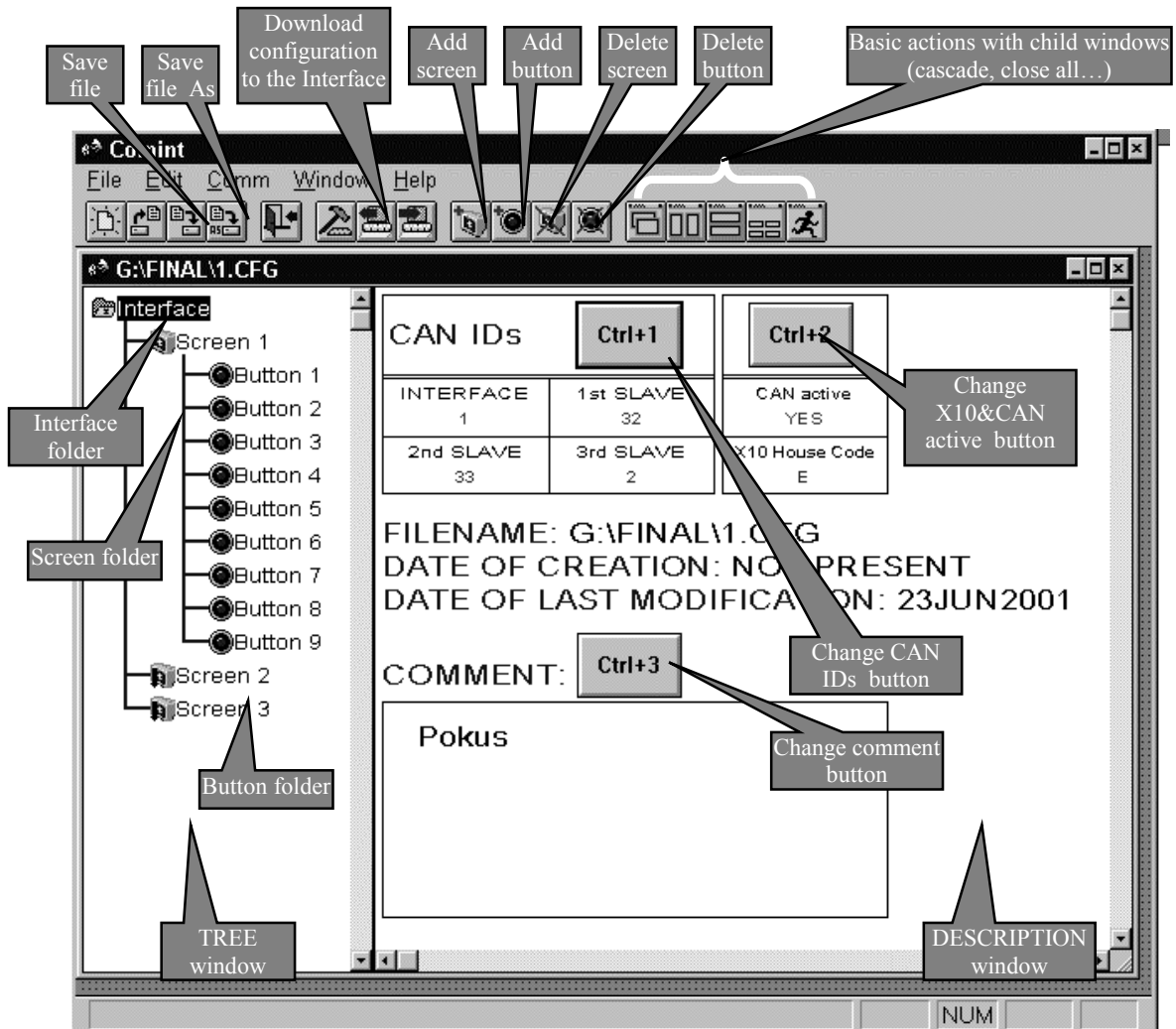


Figure 8 – Comint with opened configuration file

After opening existing file or uploading configuration from the Interface it is new window opened (new child window) and the rest of commands are allowed – figure 8. Now user can save file (File:Save or Ctrl+S) or save file under new name (File:Save as). Because there is a configuration file, we can download it to the Interface by command (Comm:Download or Ctr+D). Following 4 buttons on toolbox allow user to add or to delete elements from configuration (screen and buttons). Last 5 buttons are for handling with opened window, from the left it is Cascade open windows (Window:Cascade), Tile open windows vertical (Window:Tile vertical), Tile open windows horizontal (Window:Tile horizontal), Arrange icons (Window:Arrange icons) and Close all open windows (Window:Close all).

Every configuration window is separated into two windows. In figure 8 it is Tree window and Description window. Tree window serves for easy access to the elements (screens and buttons) in configuration. It is possible use mouse to click to the visible part of the tree structure and according to this action is element selected (inverted) and in Description window are showed its parameter(s). Also it is possible use keyboard cursor (up arrow, down arrow and enter) to access to the tree structure. If instead simple click by mouse to the element of tree double click is applied the current element is unpacked like directory structure.

## 4.1 Interface Window

If user selects by mouse or arrows the highest item of tree structure – Interface (this item is always visible and can not be deleted from the project) the Interface window is displayed in description window – figure 8. There are three basic configuration parameters:

1. CAN IDs – if user click to button Ctrl+1 or press this combination of keys on the keyboard then window in figure 9 is appeared. In configuration of THAS 4



Figure 9 – Change CAN IDs dialog window

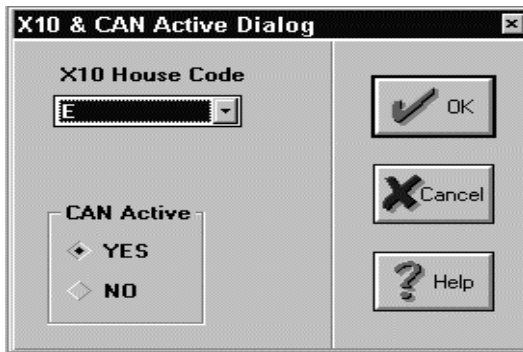


Figure 10 – Change X10&CAN active dialog window

CAN devices may by occurred. Theirs CAN ID's must be unique and program do not allow to user to set two same ID. If the user want not make any changes then Cancel is pressed. If user press help button short description of valid ID range is displayed.

2. X10 & CAN active - if user clicks to button Ctrl+2 or press this combination of keys on the keyboard then window in figure 10 is appeared. User can simply select house code (for X10 devices) and to choose if Interface will send broadcast message on CAN network (CAN Active)

2. Change comment – if user click to button Ctrl+2 or press this combination of keys on the keyboard then window in figure 11 is appeared.

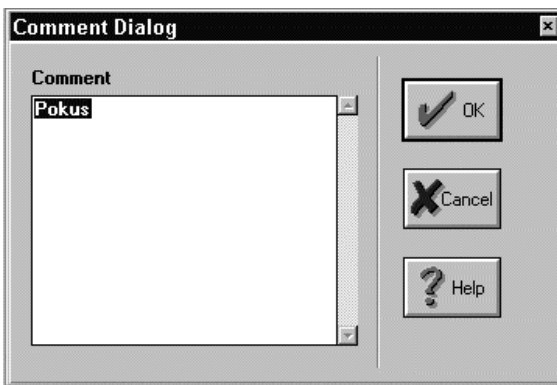


Figure 11 – Change comment dialog window

In this window user can type the comment of the project. Restriction is 7 raw and 165 characters (not 180 because of non-visible characters). New line is entered like Ctrl+Enter – only Enter causes closing window.

These parameters are always presented in the project and it is possible anytime during editing to press combination (Ctrl+1,2 and 3) and to invoke appropriate dialog window and make necessary changes.

## 4.2 Screen Window

If user selects by mouse or arrows any of the screen items in tree structure the Screen window is displayed in description window – figure 12. There is only one configuration parameter: Label of the window. Label of the window is a name describing this screen and button inside its, maximum number of characters is 28.

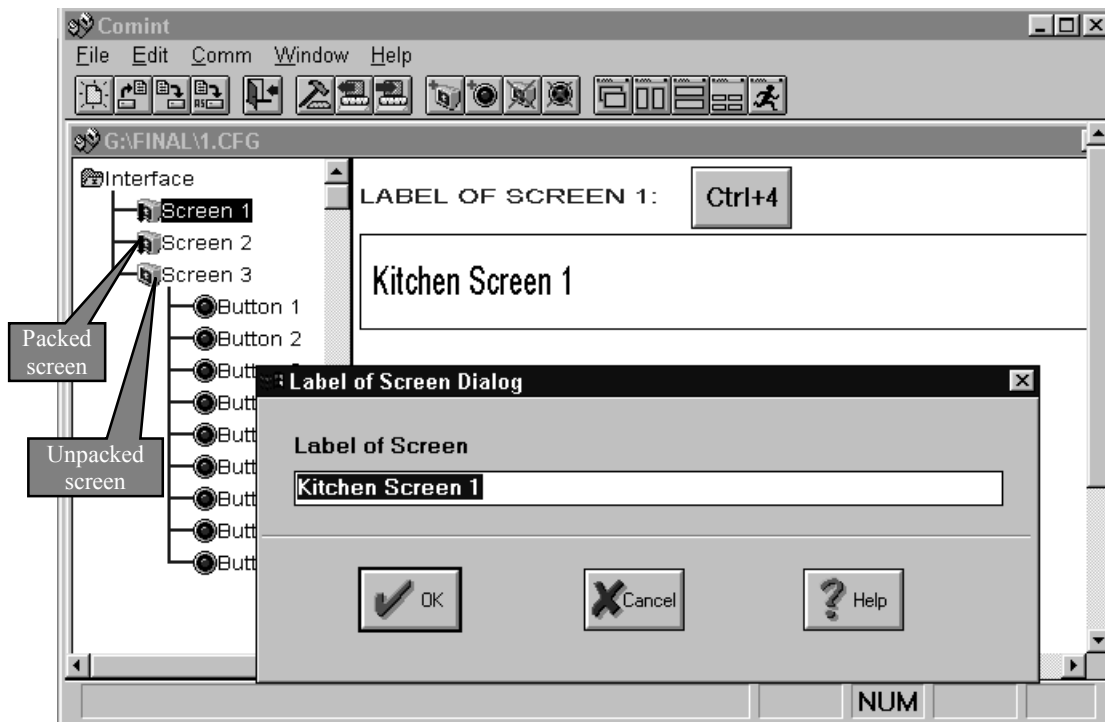


Figure 12 – Project layout when Screen item is selected and Label of screen dialog is invoked

Clicking to button Ctrl+4 or pressing this combination of keys invoke dialog window where user can edit the label of current screen.

Now we must discuss some features of configuration structure. Maximum number of windows in the project is 4. Inside one window 9 buttons are allowed. These buttons are numbered not after their order in window but after user's requisition. If screen does not contain any button or screen is unpacked then only symbol of screen is showed in the tree structure otherwise is this screen symbol modified with small arrows – see callouts in figure 12.

### 4.3 Button Window

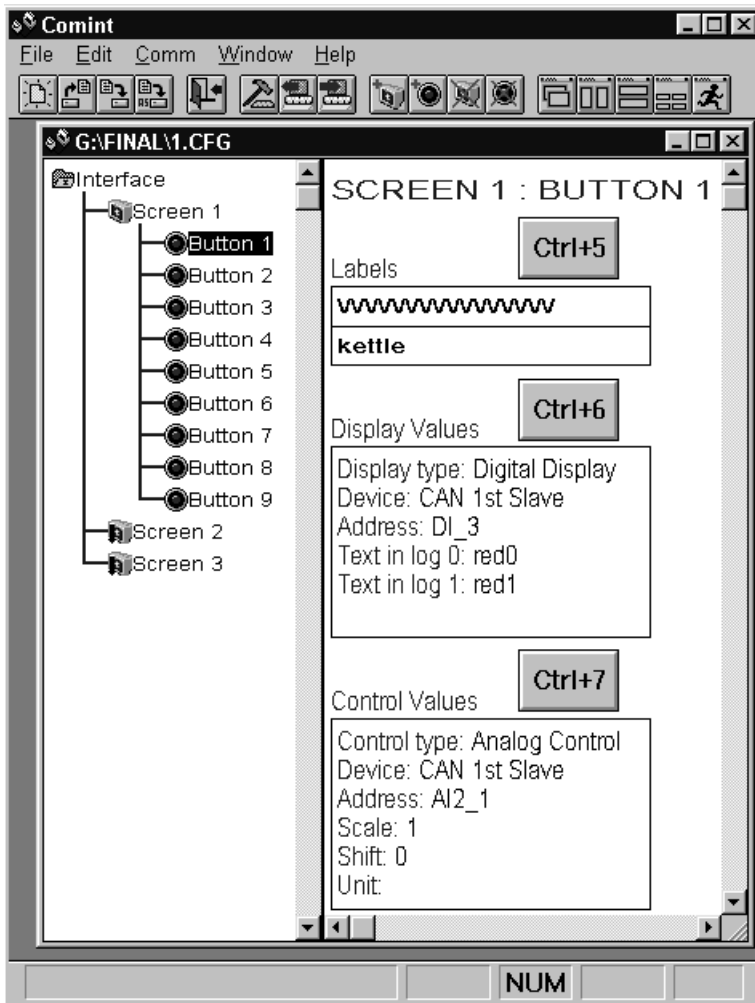


Figure 13 – Project when button item is selected

If user select by mouse or arrows any of the button items in tree structure the Button window is displayed in description window – figure 13. There are three configuration parameters:

#### 1. Labels Button Ctrl+5

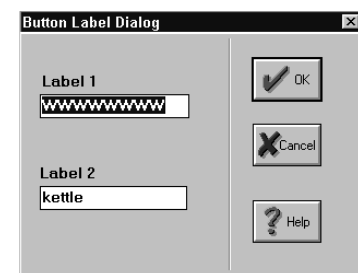


Figure 14 – Label of Buttons dialog window

Each label has 8 characters maximum and Label 2 is not in new version of Interface's program supported.

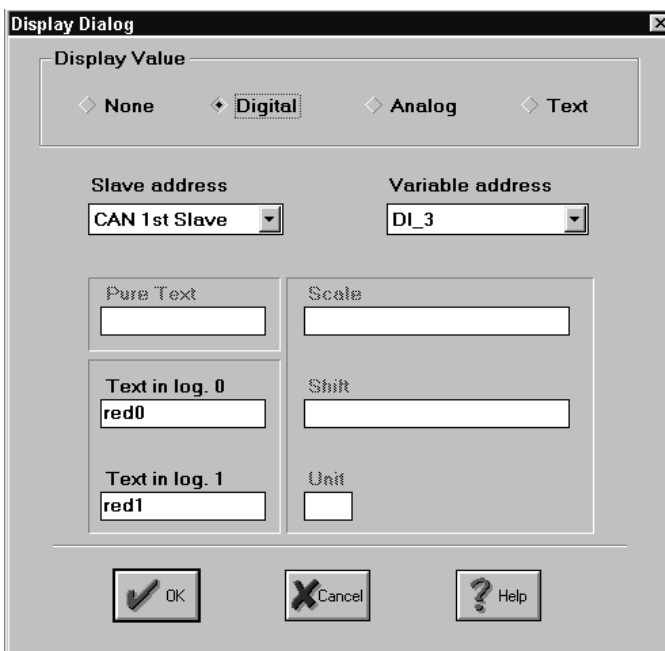


Figure 15 – Display dialog window

#### 2. Display Values button Ctrl+6

Example of such window is in figure 15. Each button can display:

- None
- Digital value
- Analogue value
- Text description

User can simply selecting appropriate radio button change the display state of button and then enter required parameters into non-greyled fields. Slave address is taken from configuration of interface (Ctrl+1 or Ctrl+2). Variable address is in according to the [3].

### 3. Control Values button Ctrl+7

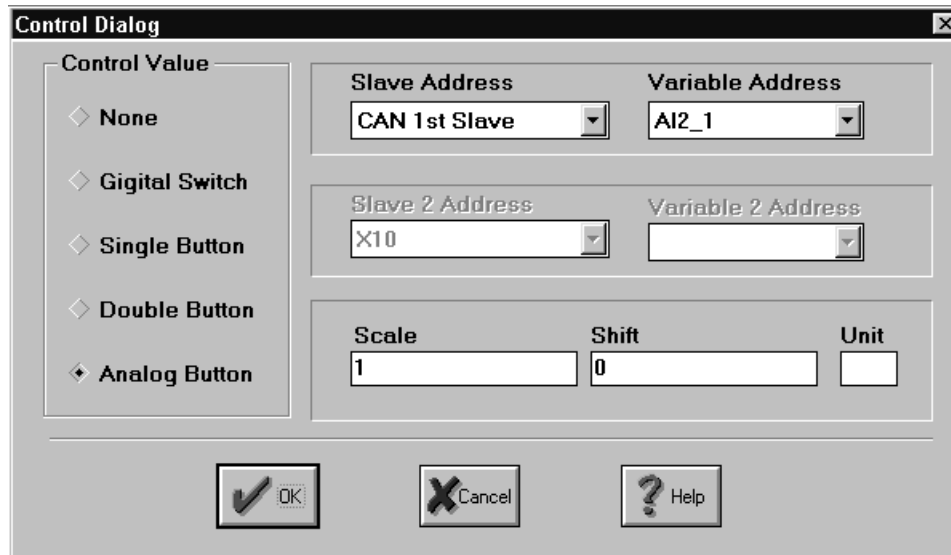


Figure 16 – Control values dialog window

Invoking this dialog window user can change the control value of the button. Button can control:

- None (button has no output action)
- Digital switch
- Single button
- Double button
- Analog button

Descriptions of these possibilities are in [1].

Last two dialog windows allows to user enter analogue values in fields *Scale* and *Shift*. These values can be entered either in real form (e.g. 874.145) or semi-logarithmic form (e.g. 8.7e145E+2).

Also it is important to say that not all combination that program allows are possible in Interface. The restrictions, mainly for X10 devices, were not in Interface changed so for instance if user selected that button is analogue and its address is X10 then after downloading such configuration into Interface this have not any effect! For restrictions see [1]. It raises question why are these features implemented in Comint. The answer is that such a way this program is prepared to work with newer version of the Interface. And I hope that these features will be into the Interface implemented. Sometime.

## 4.4 Add screen

Add screen into project is simple job. There are three possibilities how to do it.

1. Select from main menu Edit: Add screen
2. Press Ctrl+E
3. Press right button of a mouse while mouse pointer is in tree window and then select from popup menu Add screen.

All these possibilities are equal and it is only matter of user which one is the best for him. If the number of screen exceeds adding next window is not allowed.

## 4.5 Delete screen

To delete screen first must be screen selected in tree structure. If it is there are 2 possibilities how to do it:

1. Select from main menu Edit: Delete screen
2. Press right button of a mouse while mouse pointer is in the tree window on position of the screen which user wants to delete and then select from popup menu Delete screen.

Before deleting must user to confirm this action otherwise this command will not executed. Deleting screen we also delete all screen's buttons!

## 4.6 Add button

To add button user must first selects screen in tree structure. Like for adding window, there are three possibilities how to do it:

1. Select from main menu Edit: Add button
2. Press Ctrl+B
3. Press right button of a mouse while mouse pointer is in tree window at the position of the screen and then select from popup menu Add button.

If add button into screen simple dialog window give the user possibility of position of this button in the screen. If add button is the last button inside the screen this dialog window is not invoked.

## 4.5 Delete button

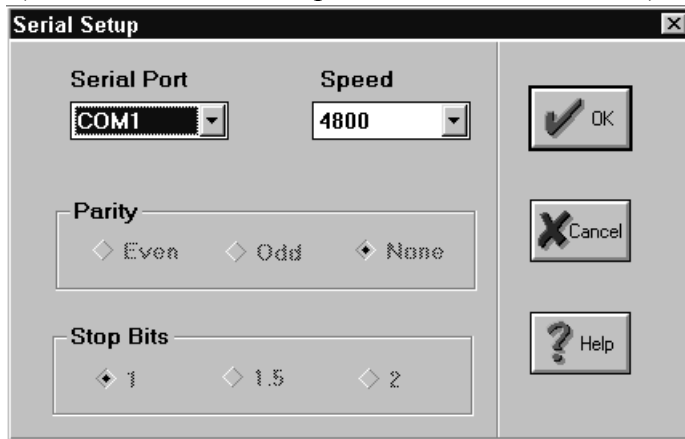
To delete button user must first to select button in tree structure. After it there are two possibilities how to delete button:

1. Select from main menu Edit: Delete button
2. Press right button of a mouse while mouse pointer is in the tree window on position of the button which user wants to delete and then select from popup menu Delete button.

Before deleting must user to confirm this action otherwise this command will not executed.

## 4.5 Downloading/Uploading the configuration

It is nothing easy then these two actions. If user wants to upload configuration from interface to the PC first must connect PC with Interface using standard serial cable (Txd->Rxd, RxD->Txd, Gnd->Gnd). After it sets communication parameters (Ctrl+T or Comm:Setup or click to toolbox icon). It invokes dialog window (see fig.



17) where user can set only one parameter – serial port on PC machine where is the Interface connected. Rest of the parameters is blocked because software in the Interface is not prepared for it. It means that only one speed: 4800 baud, none parity and 1 stop bit is used for communication via RS232.

Figure 17 – Dialog window of communication parameters

Then user simple press Ctrl+U or Comm:U

upload and interface downloads configuration from the Interface. If any error is during transmission occurred then dialog window warns the user. Otherwise program opens new window with the uploaded configuration stream.

Downloading starts by pressing Ctrl+D or Comm:D

ownload but first must be any window with the configuration opened (otherwise are item in system menu and icon in toolbox disabled).

During uploading configuration or opening existing file from disk the message „Invalid checksum” appears. Then can user decide if continue and work with this „corrupt” configuration or not.

## 5. COMINT – source code

Program Comint was written in Borland<sup>®</sup> C++ Version 4.51. Version is in this case very important because source code is INCOMPATIBLE with earlier version (e.g. 4.4 or less). Also I must explain why I decide write this program in pure C++ and why I did not use Visual version (builder) that is very famous and favoured. In fact there are three reasons.

1. I start to study Windows OS at home (in classical C absolutely disgusting job) and here was an excellent opportunity to start to study this OS using object-oriented tool (C++). And for this purpose above-mentioned compiler is nonpareil (that is fact). Visual version of OOP hides before user many things and aspects of programming under Windows. Whereas classical C++ ties programmer to know something more and this was what I mentioned – know something more from England.

2. Eventual exe file is shorter and more quickly if it is compiled in non-Visual version of compiler (it is also fact). Comint.exe has only 190 kB!
3. There was this version installed on computer where I was the Comint evolving.

Following files are strictly necessary if you try to compile Comint (tab 4):

Tab - 4

<i>filename</i>	<i>description</i>
Child.cpp	implements all operation with child windows
Client.cpp	implements all operation over child window
Comint.cpp	implements entry point to window application
Dialogs.cpp	implements dialog windows
Dialogs.cpp	header file for appropriate *.cpp file
Child.h	header file for appropriate *.cpp file
Client.h	header file for appropriate *.cpp file
Comint.h	header file for appropriate *.cpp file
Comint.def	definition file for linker
Comint.rc	resource file for development tool
Comint.rh	header file for resource file
Comint.ide	ini file for development tool

Following files are necessary if you want to achieve same look of the Comint:

Comint.ico    About.bmp    Addbutt.bmp    Addscr.bmp    Arrange.bmp    Bb0.bmp  
 Bb1.bmp    Bi0.bmp    Bi1.bmp    Bs0.bmp    Bs1.bmp    Close.bmp  
 Delbutt.bmp    Delscr.bmp    Download.bmp    Exit.bmp    New.bmp    Open.bmp  
 Save.bmp    Saveas.bmp    Setup.bmp    Tile.bmp    Tileh.bmp    Upload.bmp  
 Comint.ico

To run the program following dlls are necessary:

BCW450rtl.dll;bcwobj.dll; bids45.dll;BWCC.dll;ctl3d.dll;OWL250.dll

## 5.1 Class Hierarchy

Most difficult aspect of OOP is how to describe the problem using OOP techniques. In our case following class hierarchy was selected – fig. 17. Classes with red colour are Borland's OWL class, blue classes our Comint's program classes.

It is difficult (if not impossible) to describe whole project. Probably next 20 pages would not be enough. But the figure 17 shows more important aspects of the project.

How is evident whole project is based on Class TApplication. It is Borland class that allows our application to execute under Windows. Our class TComintApp is derived from TApplication and their most important member data is instance of class TDecoratedMDIFrame. This is also Borland OWL Class and main aim of it is to provide basic I/O operation between application and main window of the program. In fact it is during start sequence of TComintApp class created instance of TDecoratedMDIFrame class and this class need pointer to MDI client of application (TComintClient). MDI means Multiple Document Interface and it is one of the main feature of the Comint.

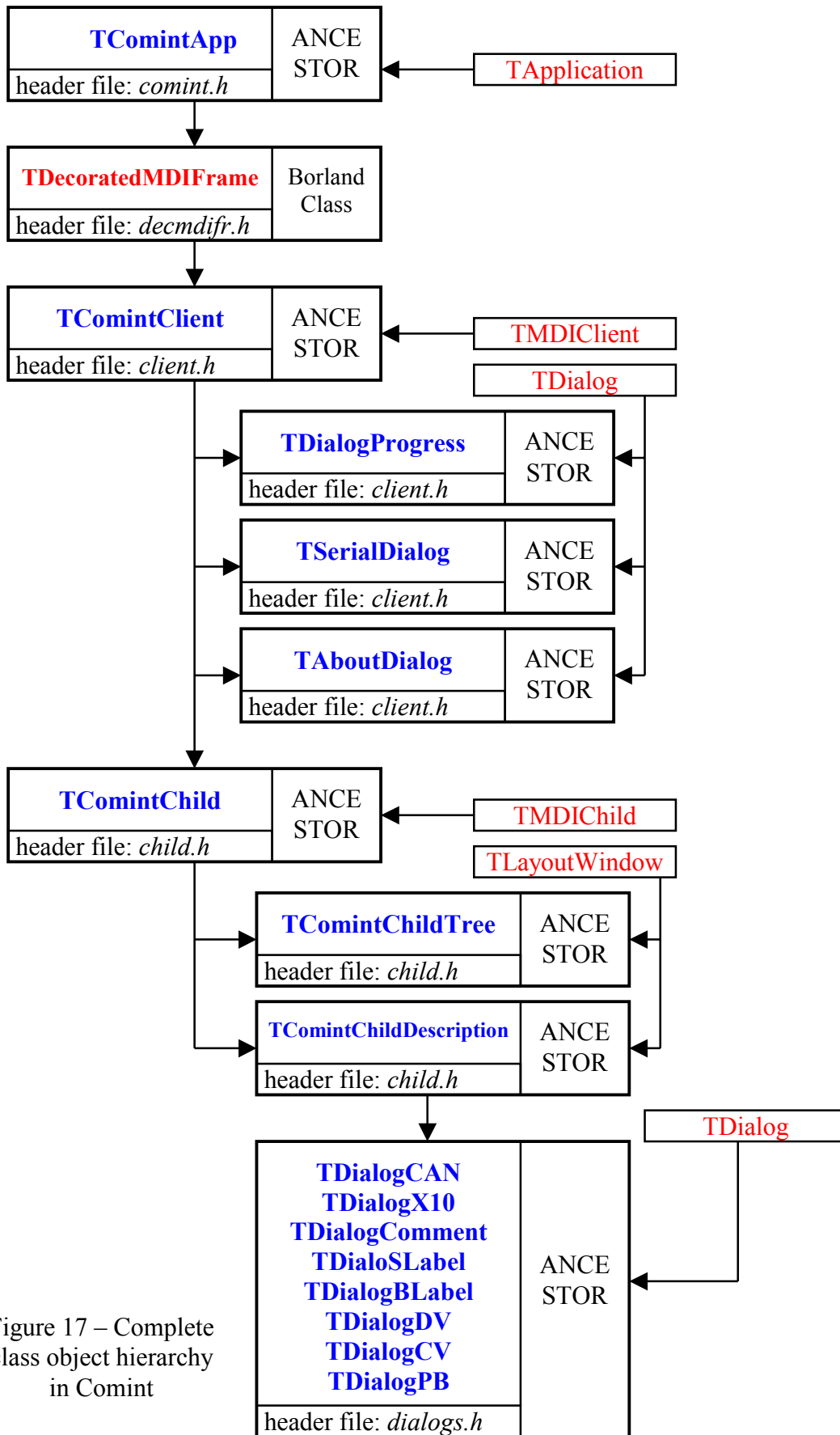


Figure 17 – Complete class object hierarchy in Comint

TComintClient is basic class for all children windows inside application. TComintClient is derived from TMDIClient (OWL class) and this class encapsulates 4 basic activities:

1. Progress window during data transfer - TDialogProgressClass
2. Serial window during setting communication parameters - TSerialDialog
3. Dialog about application - TAboutDialog
4. Prototype of child window – TComintChild

Most important class is TComintChild that ensures all operation with configuration stream. This class is derived from TMDIChild (OWL class) and has two important data members:

1. TComintChildTree – this class ensures all operation with tree structure
2. TComintChildDescription – this class ensures all operations that are available over configuration stream. Every dialog window has class that is derived from TDialog OWL class:

TDialogCAN	- changing CAN IDs
TDialogX10	- changing X10 House Code and CAN Active variable
TDialogComment	- changing comment of project
TDialoSLabel	- changing screen's label
TDialogBLabel	- changing button's label
TDialogDV	- changing display values of the button
TDialogCV	- changing control values of the button
TDialogPB	- selecting position of the button inside the screen

Serious problem during development of Comint was occurred when Window's serial driver was used. As was mentioned at the chapter 2.2 about development tool this driver is very unstable and work with it is very painful. Without any serious reason is sometimes very difficult this driver to close, or perform some basic I/O operation like is a cleaning of the communication buffer. These problems are not so evident if Windows 95 OS is used (because Windows 3.11 has very different system of multitasking) but also under Windows 95 are these errors explicit. The solution is in downloading better driver (probably not from Microsoft).

Another problem is in TMenu class. While error above is evidently problem in OS, this error is very interesting. It is not the problem to create TMenu class (OWL) and use it dynamically during execution of the program. But we work with static version of menu inside memory it is impossible to create some changes, like TMenu::EnableMenuItem() or TMenu::ModifyMenu. Even if standard API functions are called result is unsatisfactory.

Also is very interesting why it is impossible to set member data in structure TWindow::Attr.W and TWindow::Attr.H.

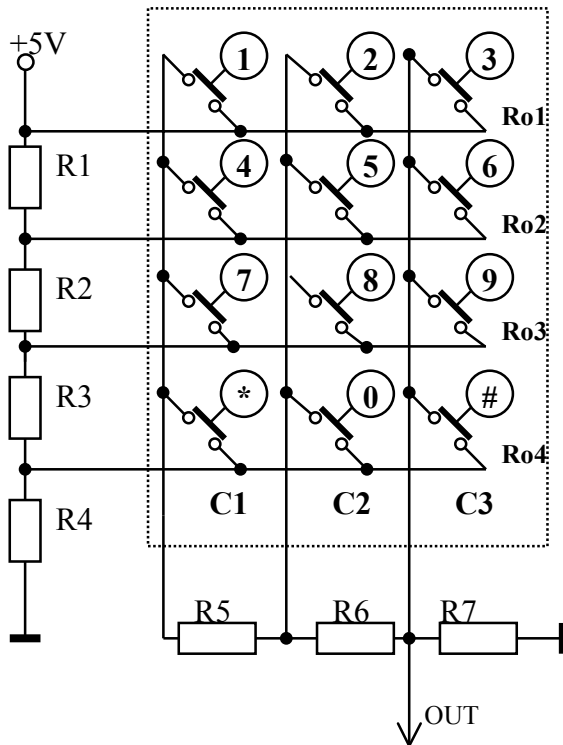
And one teaser finally. OWL application class allows overriding control over standard control elements (dialog windows, buttons ...) using Ctl3D.dll library. But if this library is enabled it is impossible to change dimension of basic window's element. Probably this problem is very similar to the problem with Attr structure in TWindow class.

Who read this part of the report, he is really very brave and I must say him thank you: Thank You! Many nights when I was drowsing many of these problems troubled me.

## 6. Conclusion

What say in fine? May be that many things could be better but the are not. For instance MDI application Comint has not simple commands Copy and Paste or it is not possible to drag element from one window and move it into another – there is no time. Also in electronic part of work is more than enough place for further developing but this is work for somebody else.

During my staying in laboratory at the University of Huddersfield I found another possibility how to connect keyboard to the microcontroller using only one pin and 7 resistors. Here it is:



How to calculate values of resistors R1 to R7 is really difficult job. Theoretically it is possible to achieve voltage step between arbitrary two key press  $U = 5 / 12 = 417\text{mV}$ .

If following values of resistors are used:

R1 = 600Ω    R2 = 600Ω  
 R3 = 2k1    R4 = 1k3  
 R5 = 8k5    R6 = 3k5  
 R6 = 9k9

The maximum voltage step is 300 mV. If 8 bit A/D converter is used (C505C) then minimal distance between keys is in decimal 15. This is really more then enough!

## References

- [1] Cach, P. „User Interface for the Tiny Home Automation System - CONSTRUCT”, Final Report, Huddersfield 2000
- [2] Zdenek Kaspar, „Design of a Home Automation System”, Description of the Universal Unit, Final year project, Huddersfield 2000
- [3] Bretislav Passinger, „Design of A Home Automation System”, Description of the Monitor/Manger, Final year project, Huddersfield 2000
- [4] Keil Software, Keil electronic GmbH: „Keil Development Software C51”, 2000
- [5] Datasheet of STV 5730, OSD for VCR, Pay-TV, Satellite Receiver, [www.st.com](http://www.st.com) (5730A-04)
- [6] Borland OWL 2.5 – Language reference user’s manual