

FPGA FOR LIFE
By Cristian Oniga

Submitted for the Digilent Digital Design Contest
May 10, 2008

Instructor: Associate Professor Ing. Ștefan Oniga Ph.D
Teaching Assistant Ing. Claudiu Lung
North University of Baia Mare

Contents

1. Introduction	3
2. Design	4
Picoblaze Microcontroller – Kcpsm3	8
UART component	10
Port selector module	11
Radio station FT857d	12
CAT protocol	13
TIBBO DS100 Serial Device Server	14
Winbond's ISD2560 Chip Corder	16
Smart Voice Module	18
3. Project Functionality	19
Main control module	19
Second control module	23
4. Discussion	25
References	25

1. Introduction

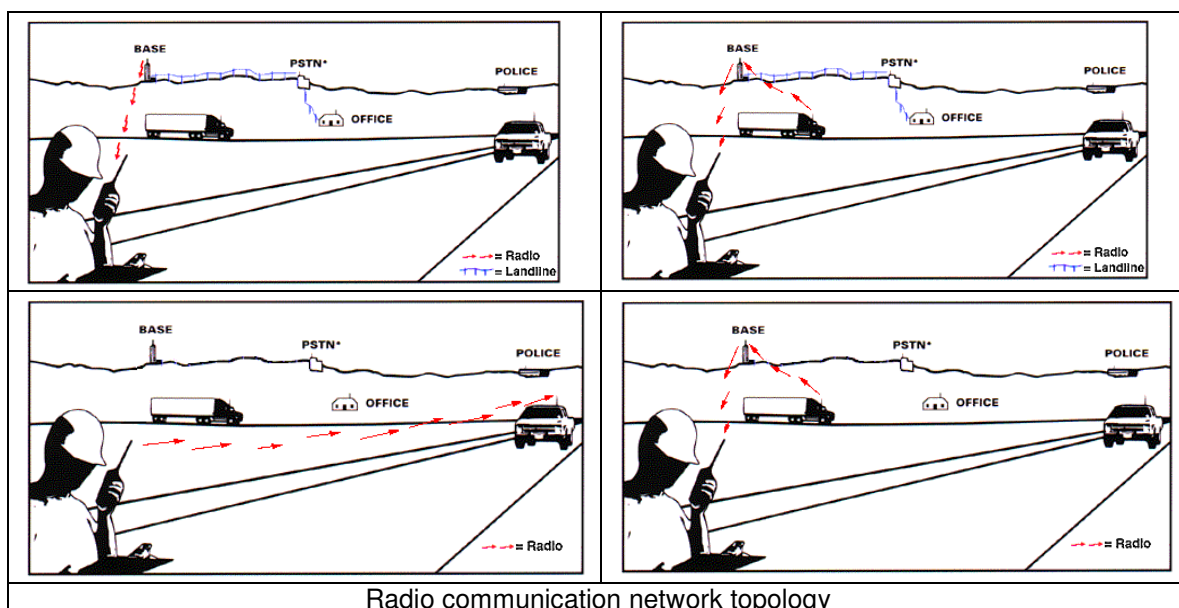
This report presents the design of an embedded system for the assistance of persons with handicaps, implemented into the Digilent Nexys Board and Nexys 2 Board

The unit is defined in VHDL, and it targets a Xilinx Spartan-3 and Spartan-3E FPGA. Some of the modules, like the ps2 interface, keyboard controller, UART controller, port_selector are generic and can be easily reused without changes in other designs. The circuit was designed on a Windows XP PC using Xilinx ISE8.2.03i CAD tool and Cadence Orcad tool.

Required tools:

- Digilent Nexys 2 (or greater) board.
- 2 Digilent PMOD modules
- PS/2 compatible keyboard
- Digilent Adept Suite
- PC running OS Windows XP Service Pack 1 (or greater)
- LCD with 4 lines and 24 columns
- 2 TIBBO serial device server modules
- Data cable for FT857d radio station
- YAESU FT857d radio station
- 2 antennas
- Power supply 13,8V/20A
- LAN Network Switch or Internet connection
- Smart voice module

The project which is presented can be used in medical applications or in live assisted applications. In this case this embedded system it can be used to offer a quick solution for old people or people with handicaps who live alone in isolated locations. In most cases the telecommunications like GSM or PSTN is not available on isolated locations, but some radio communication systems can offer possibilities to contact other persons or public institutions (hospitals, police).



2. Design

The **top levels blocks** are schematic files, as are most of the modules that appear on the top level schematic sheets. Low level modules are written in VHDL, assembly and are generously commented. Main control module is implemented onto the Nexys2 board and the second control module is implemented onto the Nexys board.

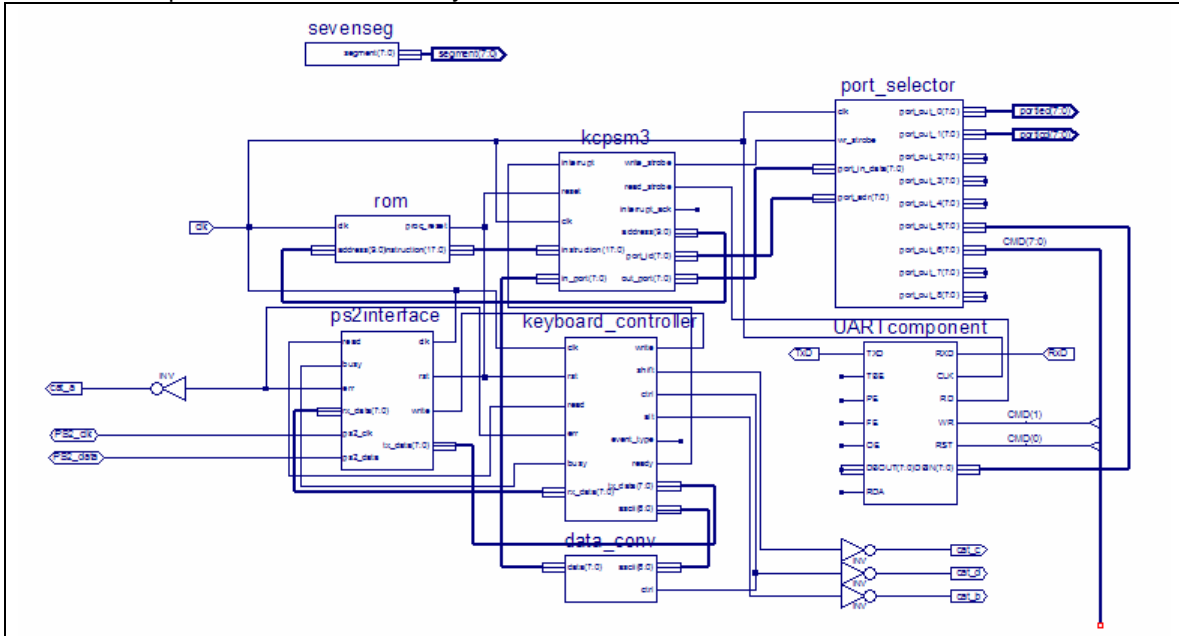


Figure 1. Main control module

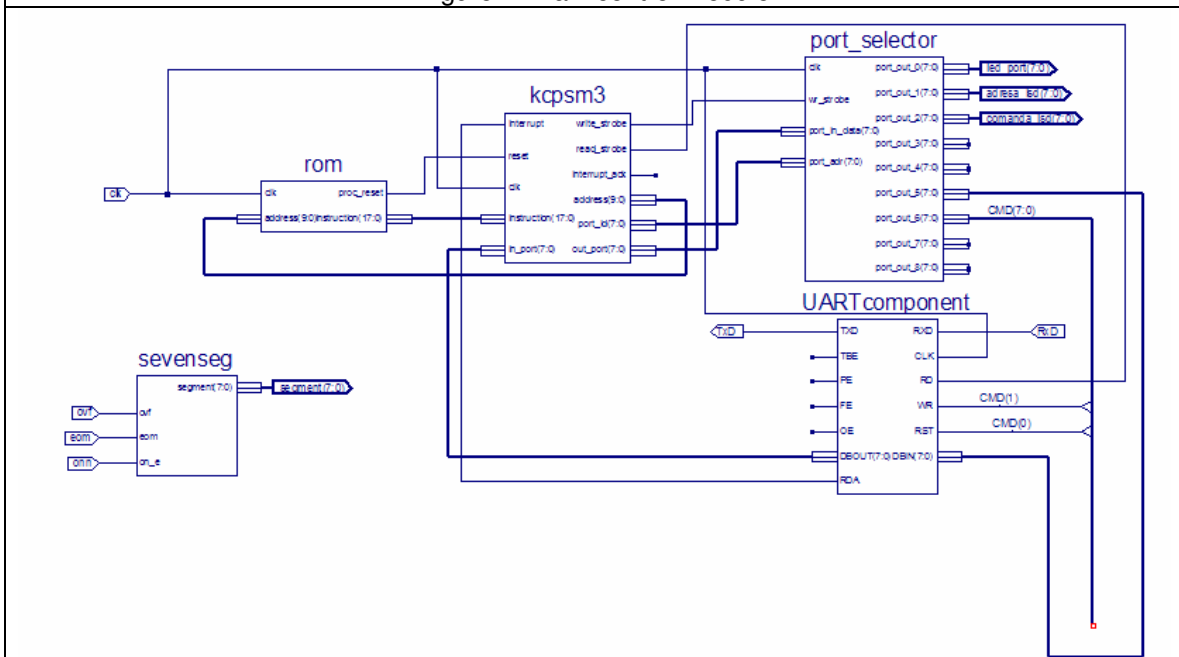


Figure 2. Second control module

Communications between them are made thru the serial connections and TCP/IP communication.

Table 1. Top Level I/O pins

Main control module		
Pin Name	Pin Direction	Description
CLK	Input	Global clock line, from board oscillator. Frequency: 50MHz
RST	Input	Reset input from button 0 of the board.
RxD	Input	Receive serial data
TxD	Output	Transmit serial data
CAT_A	Output	Control pin for 7segments display
CAT_B	Output	Control pin for 7segments display
CAT_C	Output	Control pin for 7segments display
CAT_D	Output	Control pin for 7segments display
PS2_clk	Output	Clock pin for PS2 device
PS2_data	Bidirectional	Data to/from PS2 device
SEGMENT<7:0>	Output	Data to 7 segments display
Portled<7:0>	Output	Data to onboard LED for additional information
Portlcd	Output	Data&Command for LCD
Second control module		
Pin Name	Pin Direction	Description
CLK	Input	Global clock line, from board oscillator. Frequency: 50MHz
RxD	Input	Receive serial data
TxD	Output	Transmit serial data
Led_port<7:0>	Output	Data to onboard LED for additional information
Adresa_isd<7:0>	Output	Address pin for ISD2560
Comanda_isd	Output	Command pin for ISD2560

Table 2. Modules on the top level schematic of main control module

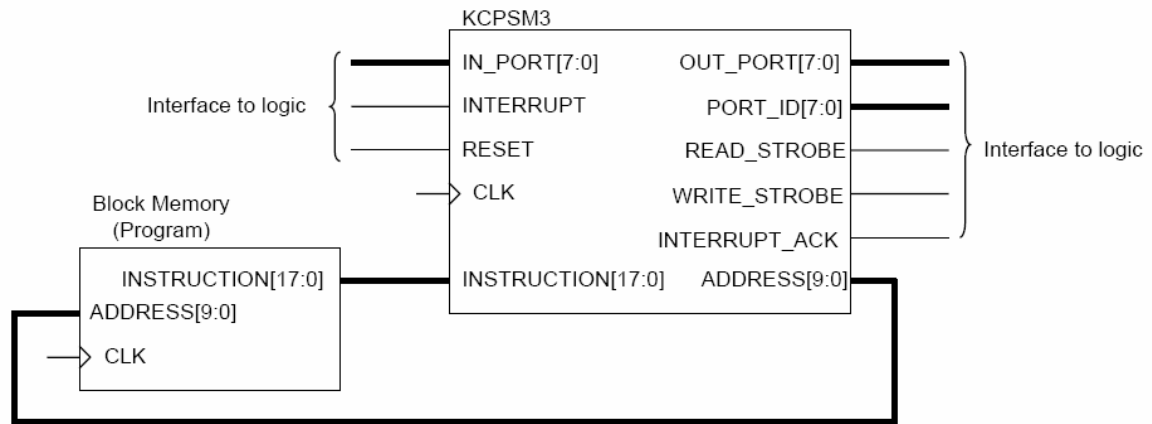
Module Name	Description
Kcpsm3	<p>This module is the core of Picoblaze microcontroller. KCPSM3 is a very simple 8 bit microcontroller primarily for the Spartan-3 devices but also suitable for use in Virtex-II and Virtex-IIPRO devices. Although it could be used for processing data, it is most likely to be employed in applications requiring a complex, but non-time critical state machine. Hence it the name of "(K)constant Coded State Machine".</p> <p>One of the most exciting features of the KCPSM3 is that it is totally embedded into the device and requires no external support. The very fact that ANY logic can be connected to the module inside the Spartan-3 or Virtex-II device means that any additional features can be added to provide ultimate flexibility. It is not so much what is inside the KCPSM-3 module that makes it useful, but the environment in which it lives.</p> <p>KCPSM-3 is an very simple processor architecture and anyone familiar with PSM, KCPSM, KSPSM2 will recognize that this is just the latest in a close family of 8-bit programmable state machines.</p>
ROM block module	<p>The principle method by which KCPSM3 program ROM will be used is in a VHDL design flow. The KCPSM3 assembler will generate a VHDL file in which a block RAM and its initial contents are defined (see assembler notes for more detail). This VHDL can be used for implementation and simulation of the processor. It has been developed and tested using XST for implementation and ModelSim for simulation.</p>

keyboard_control_module	<p>Generates requests made by the user using the keyboard and forwards them to the next module</p> <p>The keyboard controller receives bytes from the ps2interface which in turn, receives them from the keyboard device. Data is received on the rx_data input port, and is validated by the read signal. "read" is active for one clock period when new byte available on rx_data. Data is sent to the ps2interface on the tx_data output port and validated by the write output signal. "write" should be active for one clock period when tx_data contains the command or data to be sent to the keyboard. ps2interface wraps the byte in a 11 bits packet that is sent through the ps/2 port using the ps/2 protocol. Similarly, when the keyboard sends data, the ps2interface receives 11 bits for every byte, extracts the byte from the ps/2 frame, puts it on rx_data and activates read for one clock period. If an error occurs when sending or receiving a frame from the keyboard, the err input goes high for one clock period. When this occurs, the controller enters reset state.</p>
ps2interface	<p>This module implements a generic bidirectional ps/2 interface. It can be used with any ps/2 compatible device. It offers its clients a convenient way to exchange data with the device. The interface transparently wraps the byte to be sent into a ps/2 frame, generates parity for byte and sends the frame one bit at a time to the device. Similarly, when receiving data from the ps2 device, the interface receives the frame, checks for parity, and extract the useful data and forwards it to the client. If an error occurs during receiving or sending a byte, the client is informed by settings the err output line high. This way, the client can resend the data or can issue a resend command to the device.</p>
UART component	<p>A Universal Asynchronous Receiver Transmitter (UART) component is used to convert serial data to parallel data, and parallel data to serial data. A UART is extremely useful when dealing with serial input and output between a gate array and another component. The serial data transferred into the UART is placed on an output bus after the UART converts it into parallel information.</p>
Port_selector	<p>This block, written in VDL, allows the selection of any 8-bit port of the 256 of ports that Picoblaze microcontroller can access depending on the address generated. With this block we put in value the microcontroller's ability to access a large number of ports on 8 bits. The timing of data transfer is done with WRITE_STROBE signal on the ascending front of clock signal.</p>
Data_conv	<p>VHDL written module add to 6 bits ASCII format data the CTRL state bit received from the keyboard_controller block</p>
Sevensseg	<p>This module written in VHDL allow to display auxiliary information about system state, error state</p>

Table 3. Modules on the top level schematic of second control module

Module Name	Description
Kcpsm3	<p>This module is the core of Picoblaze microcontroller. KCPSM3 is a very simple 8 bit microcontroller primarily for the Spartan-3 devices but also suitable for use in Virtex-II and Virtex-IIPRO devices. Although it could be used for processing data, it is most likely to be employed in applications requiring a complex, but non-time critical state machine. Hence it the name of "(K)constant Coded State Machine".</p> <p>One of the most exciting features of the KCPSM3 is that it is totally embedded into the device and requires no external support. The very fact that ANY logic can be connected to the module inside the Spartan-3 or Virtex-II device means that any additional features can be added to provide ultimate flexibility. It is not so much what is inside the KCPSM-3 module that makes it useful, but the environment in which it lives.</p> <p>KCPSM-3 is an very simple processor architecture and anyone familiar with PSM, KCPSM, KSPSM2 will recognize that this is just the latest in a close family of 8-bit programmable state machines.</p>
ROM block module	<p>The principle method by which KCPSM3 program ROM will be used is in a VHDL design flow. The KCPSM3 assembler will generate a VHDL file in which a block RAM and its initial contents are defined (see assembler notes for more detail). This VHDL can be used for implementation and simulation of the processor. It has been developed and tested using XST for implementation and ModelSim for simulation.</p>
UART component	<p>A Universal Asynchronous Receiver Transmitter (UART) component is used to convert serial data to parallel data, and parallel data to serial data. A UART is extremely useful when dealing with serial input and output between a gate array and another component. The serial data transferred into the UART is placed on an output bus after the UART converts it into parallel information.</p>
Port_selector	<p>This block, written in VDL, allows the selection of any 8-bit port of the 256 of ports that Picoblaze microcontroller can access depending on the address generated. With this block we put in value the microcontroller's ability to access a large number of ports on 8 bits. The timing of data transfer is done with WRITE_STROBE signal on the ascending front of clock signal.</p>

Picoblaze microcontroller - Kcpsm3

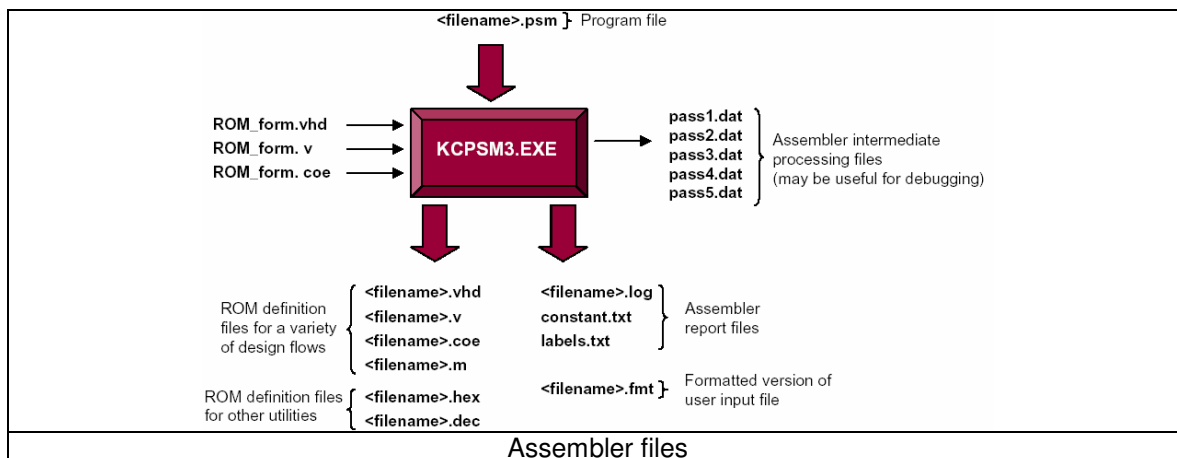


Spartan-3 has adopted the 18Kbit Block RAM elements previously seen in the Virtex-II devices. This enables KCSPM to support programs up to 1024 locations which overcomes the most commonly encountered limit of KCPSM with Spartan-II(E).

Program Size

KCPSM3 supports a program up to a length of 1024 instructions utilizing one block memory. Requirements for larger program space are typically addressed by using multiple KCPSM3 processors, each with an associated block memory to distribute the various system tasks. Programs requiring significantly more memory are normally the domain of a full data processor such as MicroBlaze with its C-language programming support.

The KCPSM3 assembler actually reads four input files and generates 15 output files.



16 General Purpose Registers.

There are 16 general purpose registers of 8-bits specified as 's0' through to 'sF' which may be renamed in the assembler code. All operations are completely flexible about the use of registers with no registers reserved for special tasks or having any priority over any other register. There is no accumulator as any register can be adopted for this task.

ALU

The Arithmetic Logic Unit (ALU) provides many simple operations expected in an 8-bit processing unit. All operations are performed using an operand provided from any register (sX). The result is returned to the same register. For operations requiring a second operand, a second register can be specified (sY) or a constant 8-bit value (kk) can be supplied. The ability to specify any constant value with no additional penalty to program size or performance enhances the simple instruction set i.e. the ability to 'ADD 1' is the same as a dedicated INCREMENT operation. Addition (ADD) and Subtraction (SUB) have the option to include the carry flag as an input (ADDCY and SUBCY) for the support of arithmetic operations requiring more than 8-bits. LOAD, AND, OR and XOR bit-wise operators provide ability to manipulate and test values. Comprehensive SHIFT and ROTATE group. COMPARE and TEST instructions enable register contents to be tested without altering their contents and determine PARITY.

Flags and Program Flow Control

The results of ALU operations determine the status of the ZERO and CARRY flags. The ZERO flag is set whenever the ALU result has all bits reset (0016). The CARRY flag is set when there is an overflow from an arithmetic operation. It is also used to capture the bit moved out of a register during shift and rotate instructions. During a TEST instruction, the carry flag is used to indicate if the 8-bit temporary result has ODD PARITY. This status of the flags can be used to determine the execution sequence of the program using conditional and non-conditional program flow control instructions. JUMP commands are used to specify absolute addresses (aaa) within the program space. CALL and RETURN commands provide sub-routine facilities for commonly used sections of code. A CALL is made to an absolute address (aaa) and an internal program counter stack preserves the associated address required by the RETURN instruction. The stack supports up to 31 nested subroutine levels.

Reset

The RESET input forces the processor back into the initial state. The program will execute from address '000' and interrupts will be disabled. The status flags and CALL/RETURN stack will also be reset. Note that register contents are not affected.

Input/Output

KCPSM3 effectively has 256 input ports and 256 output ports. The port being accessed is indicated by an 8-bit address value provided on the 'PORT_ID'. The port address can be specified in the program as an absolute value (pp), or may be indirectly specified as the contents of any of the 16 registers (sY). During an 'INPUT' operation the value provided at the input port is transferred into any of the 16 registers. An input operation is indicated by a pulse being output on the READ_STROBE. It is not always necessary to use this signal in the input interface logic, but it can be useful to indicate that data has been acquired by the processor. During an 'OUTPUT', the contents of any of the 16 registers are transferred to the output port. An output operation is indicated by a pulse being output on the WRITE_STROBE. This strobe signal will be used by the interface logic to ensure that only valid data is passed to external systems. Typically, WRITE_STROBE will be used as clock enable or write enable (see 'READ and WRITE STROBES').

Interrupt

The processor provides a single INTERRUPT input signal. Simple logic can be used to combine multiple signals if required. Interrupts are disabled (masked) by default, and are then enabled and disabled under program control. An active interrupt forces KCPSM3 to initiate a 'CALL 3FF' (a subroutine call to the last program memory location) from where the user can define a suitable jump vector to an Interrupt Service Routine (ISR). At this time, a pulse is

generated on the INTERRUPT_ACK output, the ZERO and CARRY flags are automatically preserved and any further interrupts are disabled. The 'RETURNI' instruction ensures that the end of an ISR restores the status of the flags and specifies if future interrupts will be enabled or disabled.

Scratch Pad Memory

This is an internal 64 byte general purpose memory. The contents of any of the 16 registers can be written to any of the 64 locations using a STORE instruction. The complementary FETCH instruction allows the contents of any of the 64 memory locations to be written to any of the 16 registers. This allows a much greater number of variables to be held within the boundary of the processor and tends to reserve all of the I/O space for real inputs and output signals. The 6-bit address to specify a scratch pad memory location can be specified in the program as an absolute value (ss), or may be indirectly specified as the contents of any of the 16 registers (sY). Only the lower 6-bits of the register are used, so care must be taken not to exceed the 00 - 3F₁₆ range of the available memory.

UART component

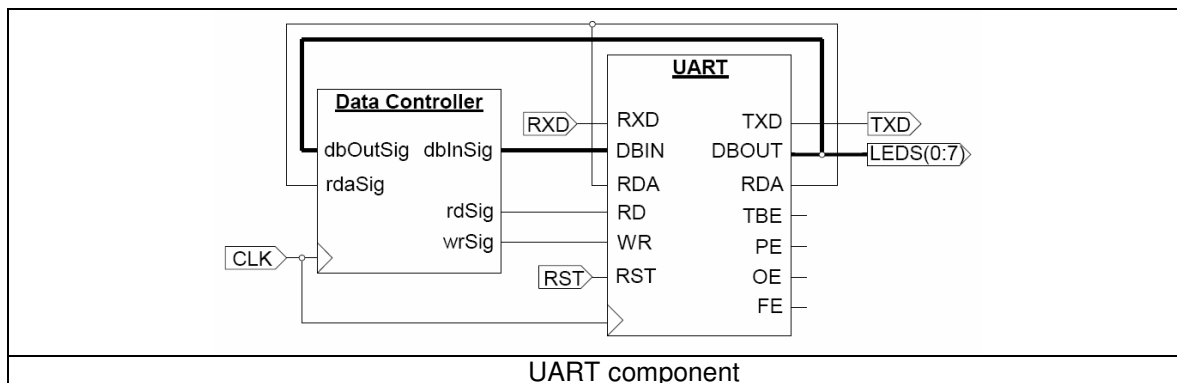
A Universal Asynchronous Receiver Transmitter (UART) component is used to convert serial data to parallel data, and parallel data to serial data. A UART is extremely useful when dealing with serial input and output between a gate array and another component. The serial data transferred into the UART is placed on an output bus after the UART converts it into parallel information. This bus can then be used as input to other logic in the gate array. The resulting data can then be sent back out serially by using the UART again.

Features include:

- Parallel to serial and serial to parallel data conversion
- Uses a user-defined odd or even parity bit (default is odd parity)
- Easily changeable baud rate (default at 9600)

Functional Description:

The UART component includes two main functions: parallel to serial conversion, and serial to parallel conversion. The receiving portion of the UART handles the serial to parallel conversion, while the transmitting portion of the UART handles the parallel to serial conversion.



Port_selector module

This block allows the selection of any 8-bit port of the 256 of ports that Picoblaze microcontroller can access depending on the address generated. With this block we put in value the microcontroller's ability to access a large number of ports on 8 bits. The timing of data transfer is done with WRITE_STROBE signal on the ascending front of the clock signal.

```

entity port_selector is
    Port (      clk      :      in std_logic;
            port_in_data : in  STD_LOGIC_VECTOR (7 downto 0);
            wr_strobe    : in  STD_LOGIC;
            port_adr     : in  STD_LOGIC_VECTOR (7 downto 0);
            port_out_0   : out  STD_LOGIC_VECTOR (7 downto 0);
            port_out_1   : out  STD_LOGIC_VECTOR (7 downto 0);
            port_out_2   : out  STD_LOGIC_VECTOR (7 downto 0);
            port_out_3   : out  STD_LOGIC_VECTOR (7 downto 0);
            port_out_4   : out  STD_LOGIC_VECTOR (7 downto 0);
            port_out_5   : out  STD_LOGIC_VECTOR (7 downto 0);
            port_out_6   : out  STD_LOGIC_VECTOR (7 downto 0);
            port_out_7   : out  STD_LOGIC_VECTOR (7 downto 0);
            port_out_8   : out  STD_LOGIC_VECTOR (7 downto 0));
end port_selector;

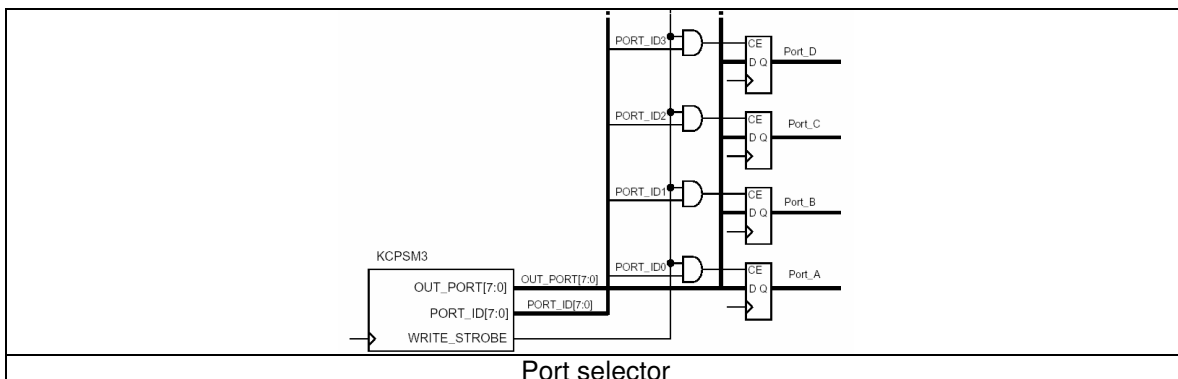
architecture Behavioral of port_selector is
begin
    process (clk, wr_strobe, port_adr)
    begin
        if wr_strobe = '1' and rising_edge (clk) then
            case port_adr is
                when "00000000" => port_out_0 <= port_in_data;
                when "00000001" => port_out_1 <= port_in_data;
                when "00000010" => port_out_2 <= port_in_data;
                when "00000011" => port_out_3 <= port_in_data;
                when "00000100" => port_out_4 <= port_in_data;
                when "00000101" => port_out_5 <= port_in_data;
                when "00000110" => port_out_6 <= port_in_data;
                when "00000111" => port_out_7 <= port_in_data;
                when others => port_out_8 <= port_in_data;
            end case;
        end if;
    end process;
end Behavioral;

```

Port_selector module - VHDL code

This module can be reused and increase the number of ports up to 256, very easy by adding new ports and new case combination stings.

Another possibilities to access more ports are resented in next figure, but it's more easy to use the module presented earlier.



For 8 or less simple output ports try to assign 'one-hot' addresses and then make sure that your design only decodes the appropriate PORT_ID signal. This greatly reduces the logic for address decoding which is advantageous for lower cost and performance. It also reduces the loading on the PORT_ID bus which is often critical to overall system performance.

Radio station YAESU FT857d

The FT-857D is rugged, innovative multi band, multimode mobile/portable transceiver for the amateur radio MF/HF/VHF/UHF bands. Providing coverage of the 160-10 meter bands (include the 60 m band: USA version) plus the 6 m, 2 m, and 70 cm bands, the FT-857D includes operation on the SSB, CW, AM, FM and Digital modes, yielding the most comprehensive performance package available for mobile and field operation.



FT 857D Radio Station

Engineered for high performance, the FT-857D provides 100 watts power output on the 160 through 6 meter bands, 50 watts output on 2 meters, and 20 watts output on 70 centimeters.

The multi-function Liquid-Crystal Display includes attractive backlighting. The display includes bar-graph indication of power output, ALC voltage, SWR, modulation level, and/or signal strength. Also included are a number of operating status icons, as well as the function displays for three operating function keys ([A], [B], and [C]).

Among the advanced features of the FT-857D are many incorporated only in large base station transceivers. These include Dual VFOs; Split-Frequency operation; Digital Signal Processing (Bandpass filtering, Noise Reduction, Auto-Notch, and Microphone Equalizer); IF Shift; Clarifier ("R.I.T."); IF Noise Blanker; AGC Fast/Slow/Auto/Off selection; RF Gain and Squelch control; IPO (Intercept Point Optimization) and receiver front-end Attenuator; AM Aircraft reception; AM and FM broadcast reception; U.S. weather Band reception; VOX; Built-in Electronic Keyer with Memories and a Beacon mode; Adjustable CW Pitch; Automatic FM Repeater Shift (ARS); Built-in CTCSS Encoder/Decoders; ARTS™ (Auto Range Transponder System); Smart Search™ Automatic Memory Loading System; Spectrum Scope; 200 Memories plus Home Channels and Band-limiting Memories; Alpha-Numeric Labeling of Memories; Automatic Power-Off (APO) and Time-Out Timer (TOT) functions; Computer interface capability; and Cloning capability.

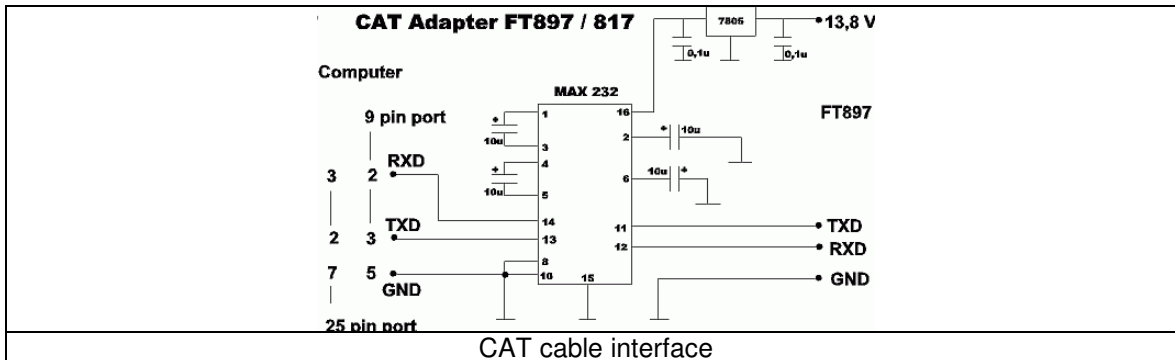
The FT-857D's CAT system allows the transceiver to be controlled by a personal computer.

This allows multiple control operations to be fully automated as a single mouse click, or it allows a third-party software package (such as contest logging software) to communicate with the FT-857D without (redundant) operator intervention.

Vertex standard does not CAT system operating software, due to the wide variety of personal computers, operating systems, and applications in use today.

The information presented in this section will allow the programmer to understand the command structure and opcodes used in the FT-857D's CAT system.

The optional CAT interface cable CT-62 is a connection cable for the FT-857D and a computer. The CT-62 has a built-in level converter, allowing direct connection from the rear panel CAT/LINEAR jack to the serial port of the computer, without the need for an external RS-232C level converter box.



CAT Data Protocol

All commands sent from computer to the transceiver consist of five-byte blocks, with up to 200ms between each byte. The last byte each block is the instruction opcode, while the first four bytes of each block are arguments (either parameters for that instruction, or dummy values required to pad the block out to five bytes). Each byte consists of 1 start bit, 8 data bits, no parity bit, and two stop bits.

There are 17 instruction opcodes for the FT-857D, in this project are used only four of them, listed in the chart on the next page. Many of these opcodes are on/off toggle commands for the same action (e.g. "PTT On" and "PTT Off"). Most of those commands require some parameter or parameters to be set. Irrespective of the number of parameters present, every Command Block sent consists of five bytes.

Accordingly, any CAT control program must construct the five-byte block by selecting the appropriate instruction opcode, organizing the parameters as needed, and providing unused "dummy" argument bytes to pad the block to its five-byte length (the dummy bytes can contain any value). The resulting five bytes are then sent, opcode last, from the computer to the FT-857D CPU via the computer's serial port and the transceiver's CAT/LINEAR jack

All CAT data values are hexadecimal

Opcode Command Chart

Command Title	Parameter				Opcode	Notes
PTT ON/OFF	x	x	x	x	CMD	CMD=00:LOCK ON CMD=80:LOCK OFF
Set Frequency	P1	P2	P3	P4	01	P1~P4:Frequency Digits 01,42,34,56,[01]=14.23456 MHz
Operating Mode	P1	x	x	x	07	P1=00:LSB, P1=01:USB, P1=02:CW, P1=03:CWR, P1=04:AM, P1=08:FM, P1=88: FM-N, P1=0A:DIG, P1=0C:PKT
Repeater Offset Frequency	P1	x	x	x	09	P1=09:"-" SHIFT P1=49:"+" SHIFT P1=89:SIMPLEX

TIBBO DS100 Serial Device Server

The DS100 is a first-generation Tibbo **Serial Device Server** that externally connects any existing serial device to an Ethernet network.



TIBBO serial Device Server

The DS100 is supplied in two versions: DS100R with RS232 serial port and DS100B with universal RS232/422/485 serial port.

The DS100 Serial Device Server is internally based on the EM100 Ethernet Module.



TIBBO EM 100 Ethernet module

The EM100 is Tibbo's original **10BaseT serial-to-Ethernet module** for onboard installation. The EM100 is highly integrated and requires only one external component- RJ45 Ethernet connector. The EM100 has the lowest power consumption of all Ethernet modules offered by Tibbo. Economical pricing makes the module an ideal choice for low-cost applications.

Hardware Features

- Compact design, dimensions only 89x51x30mm.
- Power requirements: 12V DC, app. 80mA.
- One 10BaseT Ethernet port.
- DS100R: one RS232 serial port (TX, RX, RTS, CTS).
- DS100B: one RS232/422/485 port* (RX, TX, RTS, CTS, DTR, DSR**).
- Four LEDs display status information.
- Internal firmware is upgradeable through the serial port or network.

Firmware Features

- Some serial port lines can be used as remotely controlled I/O.
- 510-byte routing buffers (x2, one in each direction).
- Onboard EEPROM stores numerous user-definable functioning parameters (Settings).
- Setup through the serial port or network.

- Over-the-network IP address configuration is also supported.
- Specialized "on-the-fly" commands for serial port control.
- "Modem" commands to control network connections; status commands for remote monitoring, etc.
- Direct control of ADSL modems.
- Supported network protocols include UDP, TCP, ARP, ICMP (PING), DHCP, PPPoE, and LCP.

From the hardware standpoint, the DS100 can be viewed as a universal platform suitable for running a variety of network and serial communications-related applications. It is the application firmware, not the hardware that gives the DS100 most of its functionality. The firmware is currently in its 3rd generation.

The application firmware of the DS100 can be upgraded through the device's serial port or Ethernet port. Serial upgrades are facilitated by a so-called Monitor- a fixed "service" firmware inside the DS100. The Monitor itself cannot be upgraded. Network upgrades rely on the NetLoader firmware component that, like the application firmware itself, can be upgraded through the serial port of the DS100 (using the Monitor). The DS100 is supplied with the application firmware and the NetLoader already pre-loaded.

Configuration of TIBBO module is realized with Device Server Toolkit (DST)



Connection Wizard is useful in creating data links involving Tibbo Device Servers and Virtual Serial Ports



DS Manager is used to locate, setup, manage, monitor, and upgrade Tibbo Device Servers.

New DS Manager adds powerful management features such as Device Server status monitoring, network upgrades, improved connectivity and support for old firmware versions.



Virtual Serial Port Driver (VSPD) turns serial ports of Tibbo Device Servers into remote COM ports of your PC. VSP Manager is used to create and manage Virtual Serial Ports (VSPs).

New VSPD introduces additional routing and connection modes, support for reverse connections, connection timeouts, and MAC->IP mapping.



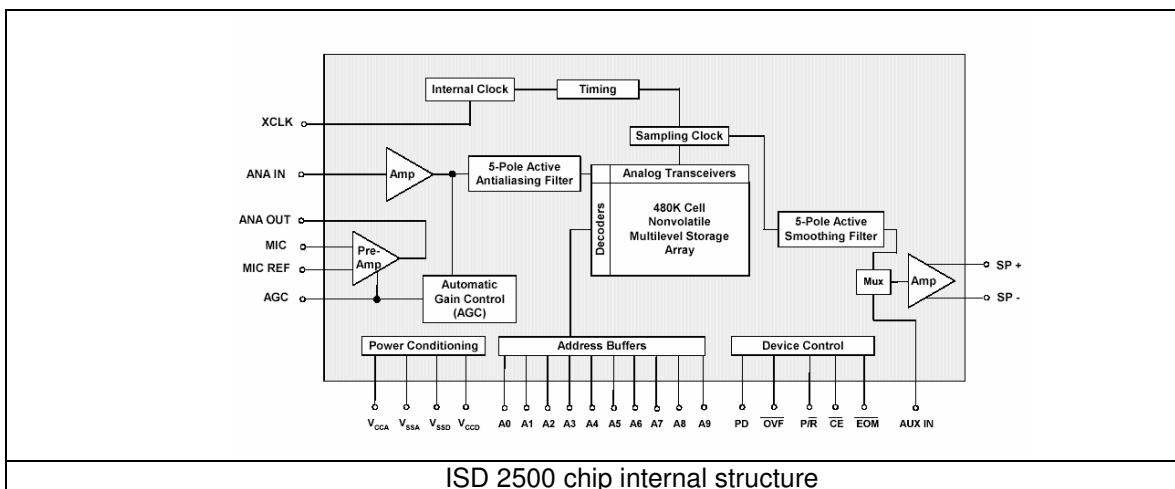
Port Monitor logs the activity of Virtual Serial Ports and remains an indispensable tool in troubleshooting communications problems in systems involving Virtual Serial Ports.

Winbond's ISD2560 ChipCorder

Winbond's ISD2500 ChipCorder® Series provide high-quality, single-chip, Record/Playback solutions for 60- to 120-second messaging applications. The CMOS devices include an on-chip oscillator, microphone preamplifier, automatic gain control, antialiasing filter, smoothing filter, speaker amplifier, and high density multi-level storage array. In addition, the ISD2500 is microcontroller compatible, allowing complex messaging and addressing to be achieved. Recordings are stored into on-chip nonvolatile memory cells, providing zero-power message storage. This unique, single-chip solution is made possible through Winbond's patented multilevel storage technology. Voice and audio signals are stored directly into memory in their natural form, providing high-quality, solid-state voice reproduction.

FEATURES

- Easy-to-use single-chip, voice record/playback solution
- High-quality, natural voice/audio reproduction
- Single-chip with duration of 60, 75, 90, or 120 seconds.
- Manual switch or microcontroller compatible
- Playback can be edge- or level-activated
- Directly cascadable for longer durations
- Automatic power-down (push-button mode) Standby current 1 μ A (typical)
- Zero-power message storage Eliminates battery backup circuits
- Fully addressable to handle multiple messages
- 100-year message retention (typical)
- 100,000 record cycles (typical)
- On-chip clock source
- Programmer support for play-only applications
- Single +5 volt power supply
- Available in die form, PDIP, SOIC and TSOP packaging
- Temperature = die (0 °C to +50 °C) and package (0 °C to +70 °C)



The Winbond's ISD2500 series includes devices offered at 4.0, 5.3, 6.4, and 8.0 kHz sampling frequencies, allowing the user a choice of speech quality options. Increasing the duration within a product series decreases the sampling frequency and bandwidth, which affects the sound quality.

The speech samples are stored directly into the on-chip nonvolatile memory without any digitization and compression associated like other solutions. Direct analog storage provides a

very true, natural sounding reproduction of voice, music, tones, and sound effects not available with most solid state digital solutions.

To meet various system requirements, the ISD2560/75/90/120 products offer single-chip solutions at 60, 75, 90, and 120 seconds. Parts may also be cascaded together for longer durations.

One of the benefits of Winbond's ChipCorder® technology is the use of on-chip nonvolatile memory, providing zero-power message storage. The message is retained for up to 100 years typically without power. In addition, the device can be re-recorded typically over 100,000 times.

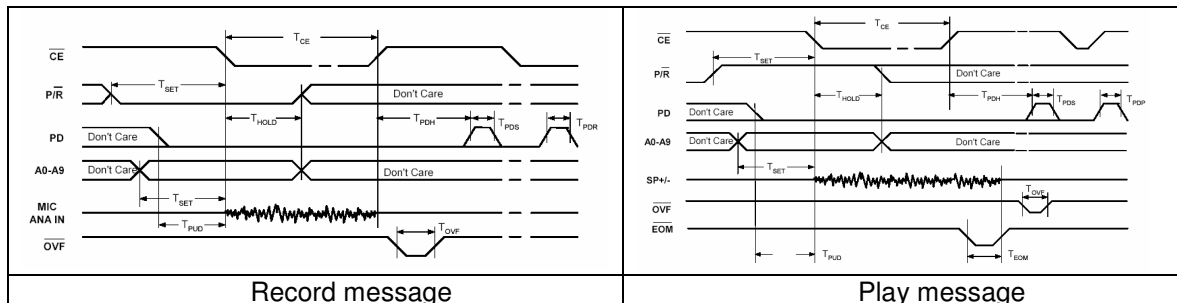
In addition to its simplicity and ease of use, the ISD2500 series includes all the interfaces necessary for microcontroller-driven applications. The address and control lines can be interfaced to a microcontroller and manipulated to perform a variety of tasks, including message assembly, message concatenation, predefined fixed message segmentation, and message management.

OPERATIONAL MODES

The ISD2500 series is designed with several built-in Operational Modes that provide maximum functionality with minimum external components. These modes are described in details as below. The Operational Modes are accessed via the address pins and mapped beyond the normal message address range. When the two Most Significant Bits (MSB), A8 and A9, are HIGH, the remaining address signals are interpreted as mode bits and not as address bits. Therefore, Operational Modes and direct addressing are not compatible and cannot be used simultaneously.

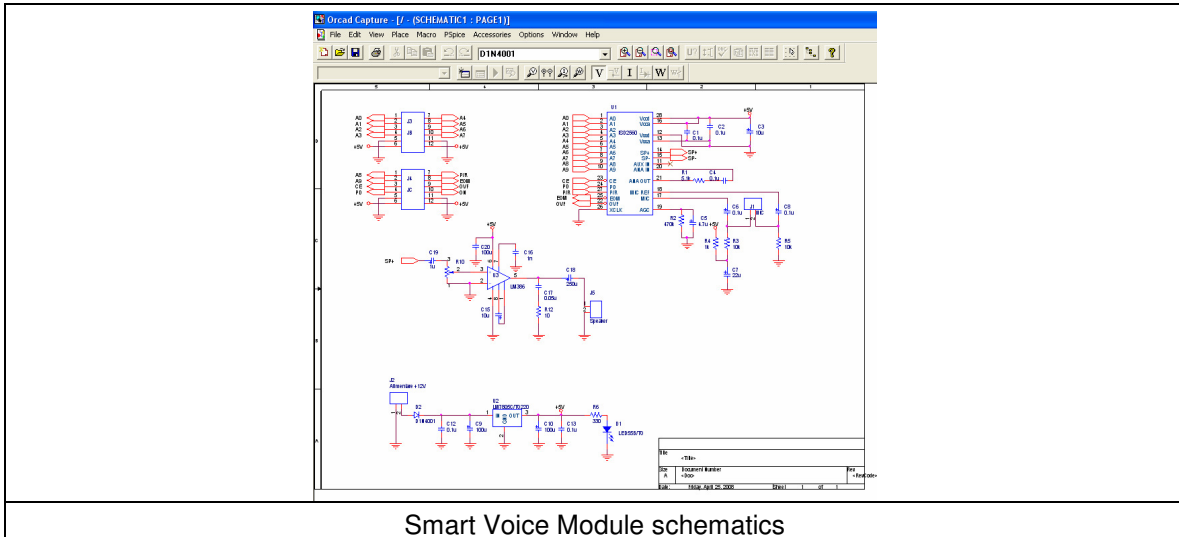
There are two important considerations for using Operational Modes. First, all operations begin initially at address 0 of its memory. Later operations can begin at other address locations, depending on the Operational Mode(s) chosen. In addition, the address pointer is reset to 0 when the device is changed from record to playback, playback to record (except M6 mode), or when a Power-Down cycle is executed.

Second, Operational Modes are executed when CE goes LOW. This Operational Mode remains in effect until the next LOW-going CE signal, at which point the current mode(s) are sampled and executed.

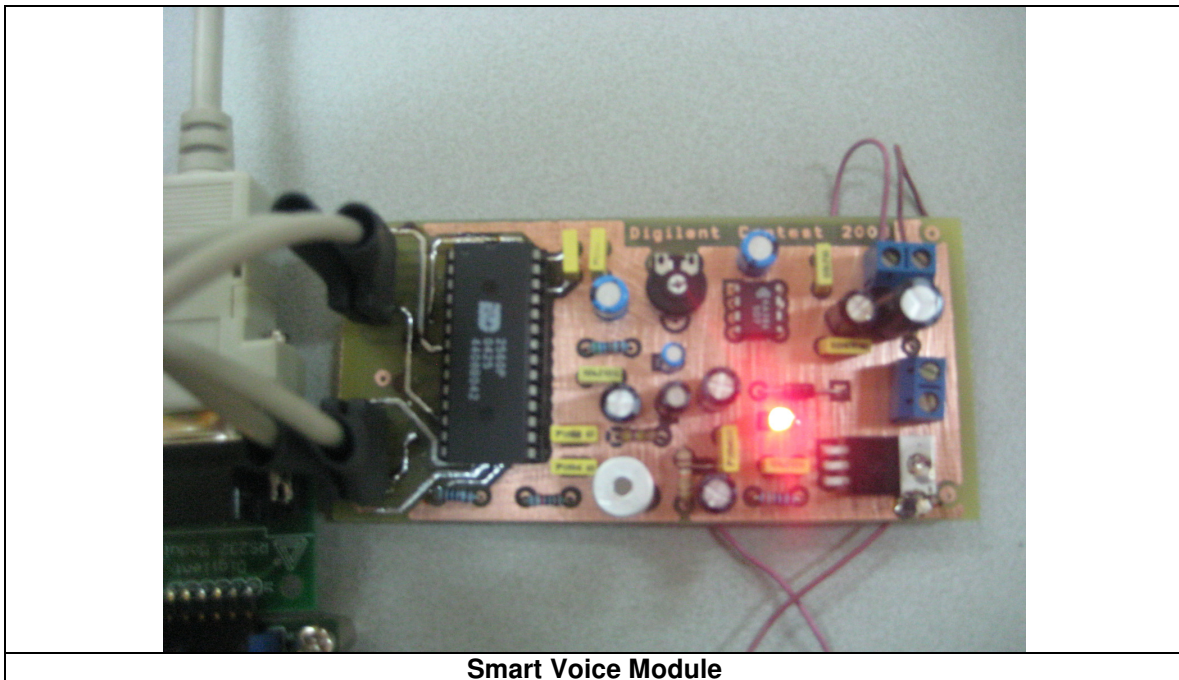


Smart voice module

This module is based on the ISD2560 chipcode witch allow to transmit some predefined messages. These messages are useful to indicate the nature of the panic alert. This project will be used on the life assistance of an person with handicaps and this feature is very useful.

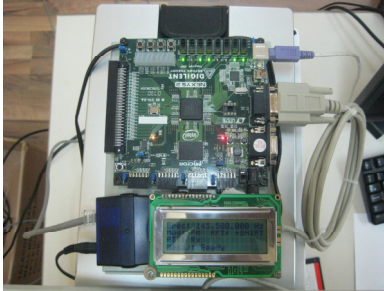

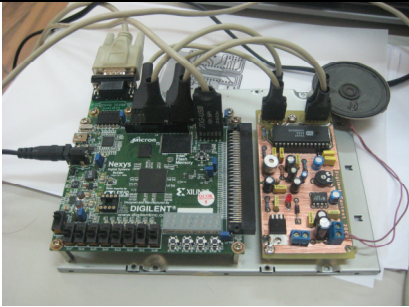



This module is designed for full compatibility with Nexys 2 board. It can provide the power supply for the Nexys2 board or it can use the power provided by the Nexys board if that is 5V.



3. Project functionality

The main module is next to the assisted person and allow them to transmit some prerecord messages. It's well known the fact that the RF can affect the electrical equipment, such cardio stimulator or other medical equipment attached to the assisted person. For this reason the transmitter is moved so far as possible to this equipment, but a long distance between the transmitter and the main module can lead to loss of connection. The technical solution consists in a LAN network to connect both modules. Connections between them are made by TIBBO modules. Main module send data serial and TIBBO module sent them TCP/IP over Ethernet to the second control module witch filter information and program radio station or generates commands for SMART VOICE MODULE.

	
Main control module	LAN communication
	
Second control module and Smart Voice Module	Radio station

Main control module

The functionality of this module is give by the picoblaze microcontroller. Program is stored on ROM block, generated from the .PSM file.

First are set some constants and renamed some internal registers to understand much better the program, easy upgrade and service.

```
CONSTANT LCD_E, 01
CONSTANT LCD_RW, 02      ; Read=1 Write=0          RW - bit1
CONSTANT LCD_RS, 04      ; Instruction=0 Data=1      RS - bit2
CONSTANT LCD_drive, 08   ; Master enable (active High) - bit3
CONSTANT LCD_DB4, 10     ; 4-bit                Data DB4 - bit4
CONSTANT LCD_DB5, 20     ; interface           Data DB5 - bit5
CONSTANT LCD_DB6, 40     ;                     Data DB6 - bit6
CONSTANT LCD_DB7, 80
```

LCD signals definition

Next are displayed some important parameters of the radio communication connection.

LOAD s5, 10	;Poziționare cursor linia 1 poziția 0
CALL LCD_cursor	
CALL disp_frecventa	;Afișare mesaj 'Frec:'
LOAD s5, 12	;Poziționare cursor linia 1 poziția 18
CALL LCD_cursor2	
CALL disp_hz	; Afișare mesaj 'Hz'
LOAD s5, 20	; Poziționare cursor linia 2 poziția 0
CALL LCD_cursor	
CALL disp_mod	; Afișare mesaj 'Mod:'
LOAD s5, 29	; Poziționare cursor linia 2 poziția 9
CALL LCD_cursor	
CALL disp_rpt	; Afișare mesaj 'RPT:'
LOAD s5, 40	; Poziționare cursor linia 3 poziția 0
CALL LCD_cursor	
CALL disp_ptt	; Afișare mesaj 'PTT:'
LOAD s5, 80	; Poziționare cursor linia 4 poziția 0
CALL LCD_cursor	
CALL disp_mesaj	; Afișare mesaj 'Mesaj:'
Parameters for radio station	

In this part of the program are initialized this parameters for start-up reasons. The radio station must start on the known frequency, mode and shift parameters.

NAMEREG sf, sute_mhz	; registrul pentru setarea frecvenței
NAMEREG se, mhz	; registrul pentru setarea frecvenței
NAMEREG sd, khz	; registrul pentru setarea frecvenței
NAMEREG sc, hz	; registrul pentru setarea frecvenței
NAMEREG sb, mode	; registrul pentru setarea modulației
NAMEREG sa, rpt	; registrul pentru setarea tipului comunicației
NAMEREG s9, ptt	; registrul pentru comanda emisiei-recepției
LOAD sute_mhz, 14	;inițializarea frecvenței
LOAD mhz, 56	;inițializarea frecvenței
LOAD khz, 12	;inițializarea frecvenței
LOAD hz, 50	;inițializarea frecvenței
LOAD mode, 08	;inițializarea tipului modulației
LOAD rpt, 09	;inițializarea tipului comunicației
LOAD ptt, 88	;inițializarea în stare de recepție
Parameters value initialization	

The frequency parameters are sent to the radio station over the serial port and displayed on the LCD.

set_freq:	LOAD s7, sute_mhz ; încărcare registru comunicație serială	
	CALL serial ; transmitere Byte 1 (sute-zeci MHz)	
	LOAD s7, mhz ; încărcare registru comunicație serială	
	CALL serial ; transmitere Byte 2 (MHz-sute KHz)	
	LOAD s7, khz ; încărcare registru comunicație serială	
	CALL serial ; transmitere Byte 3(zeci KHz - KHz)	
	LOAD s7, hz ; încărcare registru comunicație serială	
	CALL serial ; transmitere Byte 4(sute-zeci Hz)	
	LOAD s7, 01 ; încărcare registru comunicație serială	
	CALL serial ; transmitere Byte 5(comanda setare frec)	
	RETURN	
Frequency set		
disp_fr:	OR s5, 30	CALL LCD_write_data
LOAD s5, sute_mhz	CALL LCD_write_data	LOAD s5, character_stop
AND s5, f0	LOAD s5, character_stop	CALL LCD_write_data
RR s5	CALL LCD_write_data	LOAD s5, hz
RR s5	LOAD s5, mhz	AND s5, f0
RR s5	AND s5, 0f	RR s5
RR s5	OR s5, 30	RR s5
OR s5, 30	CALL LCD_write_data	RR s5

CALL LCD_write_data LOAD s5, sute_mhz AND s5, 0f OR s5, 30 CALL LCD_write_data LOAD s5, mhz AND s5, f0 RR s5 RR s5 RR s5 RR s5	LOAD s5, khz AND s5, f0 RR s5 RR s5 RR s5 RR s5 OR s5, 30 CALL LCD_write_data LOAD s5, khz AND s5, 0f OR s5, 30	RR s5 OR s5, 30 CALL LCD_write_data LOAD s5, hz AND s5, 0f OR s5, 30 CALL LCD_write_data LOAD s5, character_0 CALL LCD_write_data RETURN
Frequency display		

The modulation type is set and displayed with this subroutine.

<pre> set_mod: LOAD s7, mode ; încărcare registru comunicație serială CALL serial ; transmitere Byte 1 (tip modulație) LOAD s7, 00 ; încărcare registru comunicație serială CALL serial ; transmitere Byte 2 (dummy data) LOAD s7, 00 ; încărcare registru comunicație serială CALL serial ; transmitere Byte 3 (dummy data) LOAD s7, 00 ; încărcare registru comunicație serială CALL serial ; transmitere Byte 4 (dummy data) LOAD s7, 07 ; încărcare registru comunicație serială CALL serial ; transmitere Byte 5 (comandă setare mod) RETURN </pre>		
Type of modulation set		
<pre> disp_modulație: COMPARE mode, 00 JUMP z, disp_lsb COMPARE mode, 01 JUMP z, disp_usb COMPARE mode, 02 JUMP z, disp_cw COMPARE mode, 03 JUMP z, disp_cwr COMPARE mode, 04 JUMP z, disp_am COMPARE mode, 08 disp_cw: LOAD s5, character_C CALL LCD_write_data LOAD s5, character_W CALL LCD_write_data RETURN disp_cwr: LOAD s5, character_C CALL LCD_write_data LOAD s5, character_W CALL LCD_write_data LOAD s5, character_R CALL LCD_write_data RETURN disp_dig: LOAD s5, character_D CALL LCD_write_data LOAD s5, character_i CALL LCD_write_data LOAD s5, character_g CALL LCD_write_data RETURN </pre>	<pre> JUMP z, disp_fm COMPARE mode, 88 JUMP z, disp_fm_n COMPARE mode, 0A JUMP z, disp_dig COMPARE mode, 0c JUMP z, disp_pkt RETURN disp_lsb: LOAD s5, character_L CALL LCD_write_data LOAD s5, character_S CALL LCD_write_data disp_fm: LOAD s5, character_F CALL LCD_write_data LOAD s5, character_M CALL LCD_write_data RETURN disp_fm_n: LOAD s5, character_F CALL LCD_write_data LOAD s5, character_M CALL LCD_write_data LOAD s5, character_n CALL LCD_write_data RETURN </pre>	<pre> CALL LCD_write_data LOAD s5, character_B CALL LCD_write_data RETURN disp_usb: LOAD s5, character_U CALL LCD_write_data LOAD s5, character_S CALL LCD_write_data LOAD s5, character_B CALL LCD_write_data RETURN disp_am: LOAD s5, character_A CALL LCD_write_data LOAD s5, character_M CALL LCD_write_data RETURN disp_pkt: LOAD s5, character_P CALL LCD_write_data LOAD s5, character_k CALL LCD_write_data LOAD s5, character_t CALL LCD_write_data RETURN </pre>
Type of modulation display		

This routine sets and display the shift frequency parameter.

<pre> set_rpt: LOAD s7, rpt ; încărcare registru comunicație serială CALL serial ; transmitere Byte 1 (mod) LOAD s7, rpt ; încărcare registru comunicație serială CALL serial ; transmitere Byte 1 (dummy data) LOAD s7, rpt ; încărcare registru comunicație serială CALL serial ; transmitere Byte 1 (dummy data) LOAD s7, rpt ; încărcare registru comunicație serială CALL serial ; transmitere Byte 1 (dummy data) LOAD s7, 09 ; încărcare registru comunicație serială CALL serial ; transmitere Byte 1 (comanda setare mod) RETURN </pre>	
Shift frequency set	
<pre> disp_rpt_status: COMPARE rpt, 09 JUMP z, disp_minus COMPARE rpt, 49 JUMP z, disp_plus COMPARE rpt, 89 JUMP z, disp_simplex RETURN disp_minus: LOAD s5, character_space CALL LCD_write_data LOAD s5, character_minus CALL LCD_write_data LOAD s5, character_S CALL LCD_write_data LOAD s5, character_H CALL LCD_write_data LOAD s5, character_I CALL LCD_write_data LOAD s5, character_F CALL LCD_write_data LOAD s5, character_T CALL LCD_write_data RETURN disp_plus: LOAD s5, character_space CALL LCD_write_data LOAD s5, character_plus CALL LCD_write_data </pre>	<pre> LOAD s5, character_S CALL LCD_write_data LOAD s5, character_H CALL LCD_write_data LOAD s5, character_I CALL LCD_write_data LOAD s5, character_F CALL LCD_write_data LOAD s5, character_T CALL LCD_write_data RETURN disp_simplex: LOAD s5, character_S CALL LCD_write_data LOAD s5, character_i CALL LCD_write_data LOAD s5, character_m CALL LCD_write_data LOAD s5, character_p CALL LCD_write_data LOAD s5, character_l CALL LCD_write_data LOAD s5, character_e CALL LCD_write_data LOAD s5, character_x CALL LCD_write_data RETURN </pre>
Shift frequency display	

After the initialization session is finalized the user can reprogram the functionality of the system using some keyboards combinations, witch are presented in next table.

CTRL + F	Set frequency sequence
Key 0	145.600.00 Hz - repeater channel 0
CTRL + key 0	145.612.50 Hz - repeater channel 0x
Key 1	145.625.00 HZ - repeater channel 1
CTRL + key 1	145.637.50 Hz - repeater channel 1x
Key 2	145.500.00 Hz - mobile communication channel
Key ESC	Exit from set frequency sequence
CTRL+ M	Set modulation sequence
Key F	FM mode – frequency modulation
CTRL + Key F	FMn mode – narrow frequency modulation
Key U	USB mode – Up Side Band modulation
Key L	LSB mode – Low Side Band modulation
Key C	CW mode – telegraphy (Morse Code transmission)
CTRL + key C	CWR mode – CW Reverse mode
Key A	AM mode – Amplitude Modulation
Key D	DIG mode – FSK modulation

Key P	PKT – AFSK modulation
Key ESC	Exit from set frequency sequence
CTRL + S	Set Shift frequency sequence
Key N	Normal shift – simplex
Key P	Plus shift
Key M	Minus Shift
Key ESC	Exit from set shift frequency sequence
CTRL + R	Receive mode – standby
CTRL + T	Transmit mode – on air
CTRL + 0	Initial state of message section
CTRL + 1	Transmit message 1 – call ambulance
CTRL + 2	Transmit message 2 – call police
CTRL + 3	Transmit message 3 – call fire (not working in this release)
CTRL + 4	Transmit message 4 – call doctor (not working in this release)

Data from keyboard_controller are converted by adding to ascii code format the CTRL state bit. This operation is made in data_conv module, in VHDL.

```
entity data_conv is
    Port ( ascii : in  STD_LOGIC_VECTOR (6 downto 0);
          ctrl  : in  STD_LOGIC;
          data  : out STD_LOGIC_VECTOR (7 downto 0));
end data_conv;

architecture Behavioral of data_conv is

begin

    data <= ascii & ctrl ;
end Behavioral;
```

Data_conv module VHDL code

All of this commands are sent over LAN network to the second control module witch programmed radio station or start message transmit operation.

This module is based on the second picoblaze microcontroller running another program stored on ROM.

```
CONSTANT led_port, 00
CONSTANT adresa_isd, 01
CONSTANT comanda_isd, 02
CONSTANT serial_port, 05
CONSTANT cmd_port, 06
CONSTANT delay_lus_CONSTANT, 0B
ENABLE INTERRUPT
CALL mem_0
```

Port name and other constants definition

Functionality of this program is based on interrupt input signal provided by the serial interface. When new dates are received from the main control module and are available, UART component generates a signal that thresholds the interrupt pin on picoblaze microcontroller.

This dates are analysis and if are commands for radio station they are sent to the radio station, else they are used for start one of the message recorded on Smart Voice Module.

```
rx:
    INPUT s7, 00
    COMPARE s7, FF
    JUMP z, mem_0
    COMPARE s7, F1
```

<pre> JUMP z, mem_1 COMPARE s7, F2 JUMP z, mem_2 COMPARE s7, F3 JUMP z, mem_3 COMPARE s7, F4 JUMP z, mem_4 CALL serial RETURN </pre>
Data analysis subroutine

According to the result of analysis picoblaze microcontroller sets the parameter for playing the right message.

<pre> mem_0: output s7, led_port LOAD se, 1C OUTPUT se, comanda_isd RETURN </pre>	<pre> mem_1: output s7, led_port LOAD sf, 00 OUTPUT sf, adresa_isd LOAD se, 1C OUTPUT se, comanda_isd CALL delay_20ms LOAD se, 10 OUTPUT se, comanda_isd CALL delay_5s LOAD se, 1C OUTPUT se, comanda_isd LOAD s7, 00 OUTPUT s7, led_port RETURN </pre>	<pre> mem_2: output s7, led_port LOAD sf, FF OUTPUT sf, adresa_isd LOAD se, 1C OUTPUT se, comanda_isd CALL delay_20ms LOAD se, 10 OUTPUT se, comanda_isd CALL delay_5s LOAD se, 1C OUTPUT se, comanda_isd LOAD s7, 00 OUTPUT s7, led_port RETURN </pre>
Set initial message	Play message no 1	Play message no 1

4. Conclusions

The design was created for the Diligent Nexys board, with a Spartan-3 200K FPGA, speed grade -4 and ft256 package (xc3s200-4ft256) and Nexys 2, with Spartan 3E FPGA 500k (XC3S500). It was not tested on other boards, but it should run fine on any Spartan-3, Spartan 3E or Virtex device.

For further reference for this project, please consult the comments in the source files. They are generously commented and hopefully insightful.

The project is fully functionally and it can be used for on real situation for assistance of persons with handicaps.

The application presented and performed, is a component within a wider project iși which proposes the creation of a dedicated system implemented with intelligent programmable logical circuits that will apply in assistance to persons with disabilities.

Further development of the project involves: implementation of an algorithm for the command module extension VDEC dedicated to the acquisition of images. will implement an algorithm for transmission of data relating to the functions of vital patient on a regular basis so you can keep a record on the evolution of the health status of the patient and preventing an imbalance.

References

<http://www.xilinx.com>

<http://www.fpga4fun.com>

<http://www.yaesu.com/>

<http://radio.ubm.ro/download/download.html>

Circuite logice programabile – author: ing. Mic Daniel,

Associated Professor Ing. Oniga Ștefan PhD



**Disabilities, Assistive Technology
and Online Environments**