Annals of DAAAM for 2011 & Proceedings of the 22nd International DAAAM Symposium, Volume 22, No. 1, ISSN 1726-9679
ISBN 978-3-901509-83-4, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria, EU, 2011
Make Harmony between Technology and Nature, and Your Mind will Fly Free as a Bird
Annals & Proceedings of DAAAM International 2011

PULSE WIDTH MODULATION MIDI APPLICATION

SLOVAK, D[alibor]

Abstract: *Result of my aim is managing software. Application controls via MIDI and USB protocols various devices. At the output is pulse width modulation. Modulation value is given to value of the third byte in MIDI message. This application uses this value for external timer and for output managing voltage. Voltage is main value for transformation interface and terminal electrical devices. Paper describes software standards, development and testing environments and terminal testing hardware*

Key words: *pulse width modulation, musical instrument digital interface, universal serial bus*

1. INTRODUCTION

Research target was to design simple, time effective software application for managing various devices on the stage. For first test was used MIDI communication to devices without discreet signal regulation. There was created Universal serial bus pulse width modulation interface as a testing device. Article describes applied software norms and technologies for development. Article describes customizations of whole application too (Slovák, 2008).

2. SOFTWARE NORMS

2.1 Universal Serial Bus

The basis for this specification of this software is a general standard USB 2.0, followed by the standard for USB Audio device with special norm for USB MIDI standard. This is now widely used because MIDI transmission of information via the USB is more efficient than transmission of information using standard MIDI DIN jacks. DIN occupies too much space in the PC case, so the external sound cards are used most often with laptops. Placement DIN connectors on notebooks is not possible because of size. The USB connects USB devices with the USB host. The USB physical interconnect is realized via star topology. A hub is at the center of each star topology. Each wire segment is a point-to-point connection between the host and a hub or function, or a hub connected to another hub or function. Fiveth non-root hubs can be supported in a communication path between the host and any device. A compound device occupies two levels, therefore it cannot be enabled if attached at seventh level. Only functions can be enabled at seventh level. The USB physical interconnect is a tiered star topology. A hub is at the center of each star. Each wire segment is a point-to-point connection between the host and a hub or function, or a hub connected to another hub or function. The USB connection is defined by the so-called Endpoints, which are in any USB devices, its functionality, there may be to 15. *Endpoint zero* - the default endpoint is the default pipe, which connects USB hub and device, when the USB device identified in the hardware configuration. They demand signal IRP (I / O Request Packet), which are performed in serial order. If an error occurs in the implementation of an IRP, the other IRP are removed from the queue and is then to correct errors and reload the IRP requirements of the USB device. The communication pipes are of two kinds - Messages and Streams and each endpoint has its own descriptor. It is a structure that describes the components of the USB device and its identification in the system. Classification and description of the descriptors is given below.

1. Device Descriptor

The items correspond to the standard CDC device class and describes device. Descriptor consists information about USB class of device.

2. Configuration Descriptor

Like the device descriptor and additional information about device identification.

3. Standard AC Interface Descriptor

Audio Control interface has no its endpoint. Default endpoint zero is used for communication. Class-specific Audio Control requests are sent out via default channel. It does not provide any endpoints for settings USB device interrupt.

4. Class-specific AC Interface Descriptor

It is always connected with Standard (header) descriptor, which contains basic information about audio interfaces. It contains all pointers needed to describe a group of audio interfaces in conjunction with a given audio device.

5. Standard MIDI Streaming Interface Descriptor

Standard MIDI Streaming Interface Descriptor characterizes the device as such. There is specified by the internal structure of the Universal Serial Bus Musical Instruments Digital Interface device, and further detailed description is contained in descriptors, which are part of the configuration structure.

6. Class-specific MIDI Streaming Interface Header Descriptor

It provides more information relating to the internal structure of the device.

7. MIDI IN Jack Descriptor

Describes MIDI IN jacks. There is logical level identification for musical instruments digital interface communications This descriptor identifies channels of the communication.

8. MIDI OUT Jack Descriptor

It describes the MIDI OUT jacks, as well as MIDI IN descriptor. Its structure is added to other items that are necessary for accurate specification of the corresponding External links, respectively Embedded MIDI IN descriptor. These additional items specified with each pin of the MIDI OUT connector, and his status for data transmission.

9. Element Descriptor

Extends structure MIDI OUT Jack descriptor.

10. Standard MIDI Streaming Bulk Data Endpoint Descriptor

The content of this descriptor is consistent with a standard endpoint descriptor as described in chapter 9.6.4 USB specification.

11. Class-Specific MS Bulk Data Endpoint Descriptor

The bNumEmbMIDIJack field contains the number Embedded MIDI Jacks associated with this descriptor. There is an input endpoint, then embedded jack should goes to the MIDI OUT. If this is the final endpoint, it should input the Embedded MIDI IN jack. BaAssocJacks field contains the ID of the embedded jacks.

12. Standard MS Transfer Bulk Data Endpoint Descriptor
BEndpointAdress field designates by the help of D7 parameter. Descriptor controls data communication between host and device (www.usb.org).

3 DEVELOPMENT ENVIRONMENTS AND USED FIRMWARE, SOFTWARE AND TESTING HARDWARE

3.1 MPLAB Integrated Development Environment

To edit and develop the source code was used development environment, which Microchip adds and also offers as a shareware to application development on their boards. Reason using this environment was facilitation development of a case study, which is a separate hardware platform PWM MIDI application, in which was implemented above-mentioned software (www.microchip.com).

3.2 Firmware – modules and source files

Basic and indispensable files are:
main.c - Function main() includes infinite program central loop while(1). In this loop are valet through procedures USBTASKS(void) and void ProcessIO(void) all requisite tasks in programmatic succession that the is given instructions source code.
typedefs.h - In this file are defined individual data type.
interrupt.c and .h - This module contains an implementation of high and low priority interrupts.
usb.h – This file contains needed header file to the whole application program source code.
usbdefs_std_dsc.h – Content of file is description of descriptors arrangement as well as definition values for input and check out endpoints. This file includes all variables for full structures of endpoints.
usb_compile_time_validation.h– Verification sizes endpoint descriptor according to standard of USB. Then reach size descriptors can be either 8, 16, 32 or 64 bytes.
usbcfg.h – By the help this file is effected endpoint device configuration. There are intended value and starting set for endpointzero, endpoint assignment for configuration descriptor and values for interface descriptors and their endpoints.
usbdsc.h and .c - This modules includes information about USB descriptors. There are included definition structures of configuration and globalize descriptors for visibility and others modules.
usbmap.h and .c - This module presents USB memory manager. Allocation of USB endpoint and their buffer descriptors proceeds dynamically - at compile time - with usage some parameters defined in usbcfg.h.
There are not necessary any changes in programming of standard devices. But without a deeper study is not possible to customize standard devices for various usage. There is described USB PWM interface and its special programming in second paper. The most important changes are created in *user.c and user.h* source file. It is described in next paragraph.
user.c and h. - Fundamental module for user needs are two source text files user.c and user.h, that module contains custom functions setting and macro. There is user does pertinent modification necessary for given functionality of application (www.usb.org).

3.3 Testing terminal devices

There were used lights and water fountains for testing. See figure 1. Glass bath consists two water fountain. The first is cycle and second is central. Glass bath consists three lights channels too. There is pissible to see results of managing via pulse widt modulation. Red lights shines more then green light. Musical instruments digital interface message third byte value is higher than value for green light. There is similar for fountains. Central fountain value is higher than value for circle fountain. Managing voltage value is continuous due to 127 possibly level value from musical instruments digital interface message.

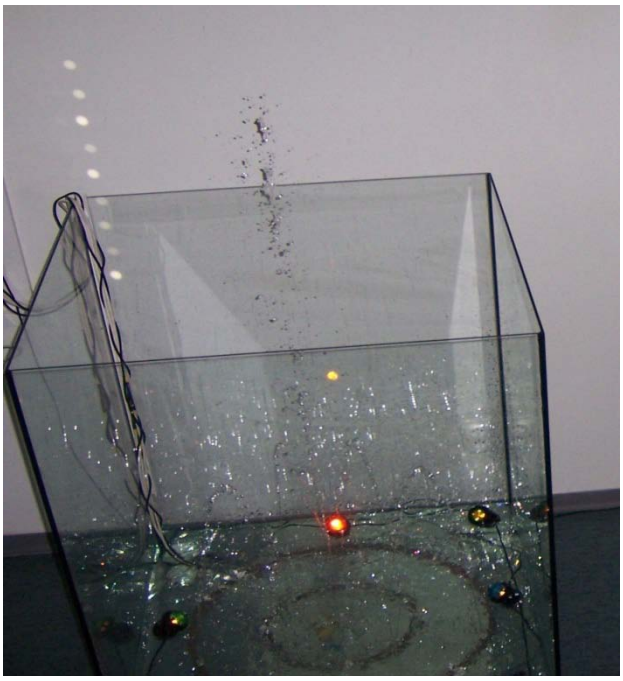

Fig. 1 Testing water fountain and lights

5 CONCLUSION

The proposal was to create a financially optimized, the cheapest possible, programmable robust software application without time delay. The basic advantage of the software application is that of the manufacturing cost. This cost is very low. Aim proposal was create moneywise optimized, programmatic robust software application for purposes specified in introduction of this paper. Among basic benefits software then belongs to its system catholicity. Next advantage is easy implementation to the lights applications. There is accessible USB connector and eventually jumper commutator for easy application reprogramming in the event of changes in lights configuration. An important benefit is the ability to exploit any effect devices, regardless of it, which protocol is used for control various exist devices.

6. ACKNOWLEDGEMENTS

This work was performed with financial support of research project NPVII-2C06007, by the Ministry of Education of the Czech Republic. This work was supported by the grant Internal Grant Agency of Tomas Bata University under the project No. IGA SV30111148020.

7. REFERENCES

PIC18F2455/2550/4455/4550 Data Sheet.pdf, *Available online:*http://www.microchip.com, *Accessed*: 2004-04-01

MPLAB C18 C Compiler User's guide.pdf *Available online:*http://www.microchip.com, *Accessed*: 2011-09-30

Slovák, D. *Protocol MIDI and stage equipment:Diploma Thesis Tomas Bata University*, Faculty of Applied Informatics, Department of Applied Infomatics, 2008, 64 p. Thesis supervisor prof. Ing. Vladimír Vašek, CSc

Universal Serial Bus specification.pdf *Available from:*http://www.usb.org, *Accessed*: 2011-09-30

Universal Serial Bus Device Class Definition for Audio Devices.pdf, *Available online:*http://www.usb.org> *Accessed*: 2011-01-01