XIX ННТК с международно участие „АДП-2010”

COMMUNICATION SYSTEMS OF THE EDUCATIONAL ROBOTS ROBCO

N. Chivarov, S. Shivarov, P. Kopacek and N. Shivarov

Abstract: *The design of the currently described system includes a main module – Controller – and the rest of the modules are subordinate to it. The user interacts with the system via User Interface, which is connected to the Controller. The User Interface is implemented by software written in C# that accepts user commands, passes them to the Controller via a serial port of the PC, and gets feedback of the system status and other important data to display back to the user. The Controller implements basic system operation logic. It has two interfaces. The first is a serial interface to the PC or other terminal of the operator, and the other one is internal for the system and is named System Bus. The System Bus carries all commands and data to and from subordinate modules and is essential for the modular system design.*

Keywords: Main Controller, User Interface, Serial Interface, System Bus

1. Introduction

Modular design is a very important concept that allows for both flexibility and scalability of the system, as well as easy overall planning, construction, support and maintenance. This concept is especially useful when the design (mechanical, electrical and software) can easily be divided into smaller units operating relatively independent from one another.

Such a design enables the separation of the main logic of operation from detailed control (driving) of end actuators (motors, etc.). In other words, it allows for introduction of an abstraction layer between the main system's intelligence and specific needs of different mechanical parts that directly implement motion, sound/light interfaces, etc. This makes the system design even more versatile and enables module hot plug-in (during operation) and self-configuration.

2. Concept

The design of the currently described system includes a main module – Controller – and the rest of the modules are subordinate to it. The user interacts with the system via User Interface, which is connected to the Controller. The latter implements the main logic of operation and intelligence. It passes commands to the subordinate modules and retrieves data or feedback from them via the main System Bus.

Figure 1 illustrates basic design ideas of the system.

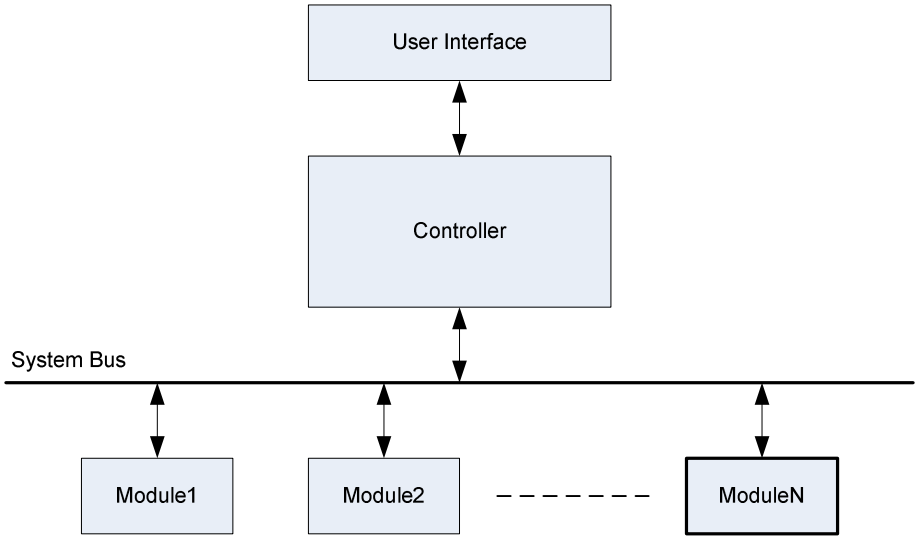

Fig.1
Basic modular design

3. General Description

The User Interface is implemented by software written in C# that accepts user commands, passes them to the Controller via a serial port of the PC, and gets feedback of the system status and other important data to display back to the user.

The Controller implements basic system operation logic. It has two interfaces. The first is a serial interface to the PC or other terminal of the operator, and the other one is internal for the system and is named System Bus. The System Bus carries all commands and data to and from subordinate modules and is essential for the modular system design.

Modules *1* to *N* represent basic building blocks of the system. They are intelligent drivers (physical and firmware) of the Robot's actuators – stepper motors that implement motion. The modules also implement the means of feedback – states of end switches, steps counting, data from encoders, etc. Each intelligent module has an address that uniquely identifies it over the System Bus, as well as description of the type of the module – whether it controls stepper or DC motors, encoder presence and type, and other specific data.

One Module can control up to 2 stepper or DC brush motors and drive respectively up to 2 mechanical joints of the Robot Arm. It also can read data from their encoders (if present).

Modules are capable of performing simple tasks by themselves, such as moving joints to a given absolute position or relative distance from its current one. This allows for the arm to perform complex motions that include some or all of its joints without charging additional processing load of Controller, no matter how many mechanical joints (or degrees of freedom) the mechanical arm has.

Below a detailed description of all system components follows.

4. Intelligent Modules

Figure 2 displays the basic block design of an Intelligent Module, controlling one stepper motor and reading feedback from the attached absolute encoder.

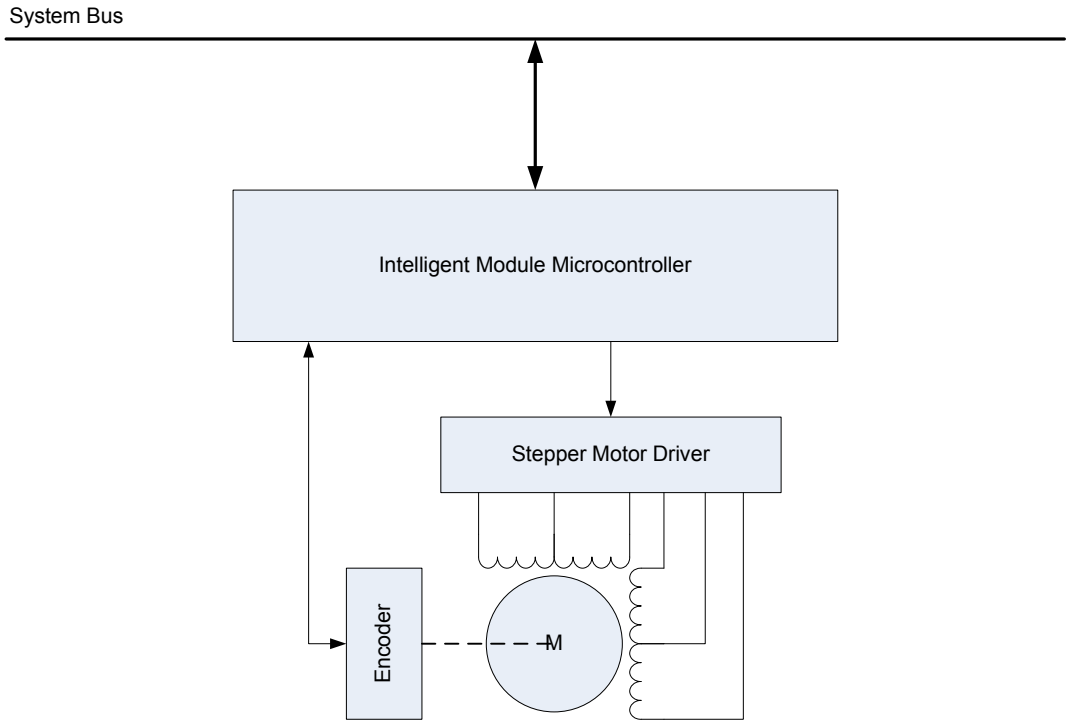

Fig. 2
Basic Intelligent
Module design

The core of an Intelligent Module is a microcontroller from Atmel's AVR family. The microcontroller provides necessary signals to control the Stepper Motor Driver, as well as reads the data for the absolute position from the absolute Encoder. It is also connected to the System Bus via its TWI (Two-Wire Interface).

5. RS-232 Communication Interface

RS-232 communication has the following settings:

- **Cable:** DB9 or DB25 (depends on PC side, usually DB9) Female to DB9 Male Straight Serial Cable (only Rx, Tx and GND pins are used);
- **Baud rate:** 9600
- **Data bits:** 8
- **Parity:** None
- **Stop bits:** 1
- **Flow control:** None
- **Note:** The interface is compatible with common USB-to-RS232 (COM) port adapters. A connection to modern desktop PCs and laptops (without COM ports) requires such an adapter.

6. System Bus

The System Bus is a simple TWI (Two-Wire Interface) bus. TWI allows for connection of maximum 127 devices (modules) which is more than enough and also this number can be extended by firmware.

The System Bus consists of two signals – Clock and Data. The Clock signal is generated by the Master (the Controller) but can be extended by the slave (Intelligent Module) in order to implement flow control mechanism. The Data signal is used to transfer data to and from the Intelligent Modules.

The common packet format of the messages transferred through the System Bus is shown on Fig. 3.

Fig. 3
Common Message format

The *ID/RW* field contains the address of the receiver module (7 bits) and Read/Write flag, indicating whether a Read or Write operation is to be performed. Since the Controller is always the TWI Master (it initiates all transfers on the System Bus), the Message does not contain return address (of sender ID). Instead, when some data is needed to be transferred from a Module to the Controller, the latter initiates a TWI Read sequence, addressing the device and sending a Read bit in ID/RW field. In this case, the Payload is actually transferred from the addressed Module to the Controller. If data is to be sent from the Controller to a specified Module, the RW flag is set to Write and the Payload direction is sent to the Module.

A special *ID = 0* is used as a General Call Address. When a message packet is sent from the Controller to that address, all Modules are addressed and receive the packet. This is especially useful when sending some emergency calls such as *stop all* command. Note that General Call is not applicable when reading from Modules as only one device can transmit data on the System Bus at a time. This means that using *ID = 0* is meaningful only with conjunction with Write flag (*RW = 0*).

The Payload contains the message itself. Its format is shown on Fig. 4.

Fig, 4
Payload – Command format

The first field shows the total length (in bytes) of the Payload (including the *LEN* field itself). The second field – *CMD* - is the code of the command to be performed. The remaining fields $PAR_1 \div PAR_N$ – provide necessary parameters for the command.

An example C source code for transmitting a command from the Controller to a Module is shown below. It is clearly seen that Controller looks for a packet confirmation code from the Module after sending the packet, and this code is equal to the inverted byte code of the CMD.

```
//*****************************************************************
// Joint stop; arguments - all, 0, 1, 2, 3, 4…
//*****************************************************************
void joint_stop(char*arg)
{
  char par[CLI_MAX_CMDLEN];
  BYTE buff[STEPPER_MAX_PACKET_LEN], addr, i;

  if(!strcmp(CT_Arg(par, 1), "all"))
    addr = 0;
  else
    addr = MODULES_START_ADDR + (atoi(CT_Arg(par, 1)) >> 1);

  i = 0;
  buff[i++] = 0;  // Later here will be put the total length
  buff[i++] = CMD_STOP;
  buff[i++] = atoi(CT_Arg(par, 1)) & 0x01;
  if(!strcmp(CT_Arg(par, 2), "-b"))
    buff[i++] = true; // brake
  if(!strcmp(CT_Arg(par, 2), "-r"))
    buff[i++] = false; // release
  buff[0] = i;
  TWI_SendBytes(addr, buff, i);
  OS_Delay(200);
  if(addr)
  {
    buff[0] = buff[1];
    TWI_ReadBytes(addr, buff, 1);
    if(buff[0] == (BYTE)~buff[1])
      printf("ok");
    else
      printf("err");
  }
}
```

There are multiple commands that allow for the Controller to start, stop and control the movement of the motors and mechanisms, attached to Modules. In order for the user to be able to make use of all those useful tools, a special CLI (Command-Line Interface) environment is created.

Literature:

1. http://en.wikipedia.org/wiki/RS-232
2. http://en.wikipedia.org/wiki/Digital_signal.
3. N. Chivarov, N. Shivarov and P. Kopacek; Educational Articulated Robot - ROBKO PHOENIX, Robotics and Mechatronics 2008, September 17-21, Varna, Bulgaria; p.207-p.211, ISSN1310-3946.
4. N. Chivarov, N., Shivarov, N. and Kopacek, P.; Mechatronics Educational Robots ROBKO Phoenix; "International Journal Automation Austria - IJAA" 2009, Volume 17, Issue 2, ISSN 1562 – 2703, pp. 68-73.
5. N. Chivarov, P. Kopacek and N. Shivarov; Modular Control System For The Family Of Educational Robots “ROBCO”; Robotics and Mechatronics 2009, October 7-9, Varna, Bulgaria; p23 – p. 27, ISSN1310-3946.

Authors Data:

Nayden Chivarov, Dr. techn., Central Laboratory of Mechatronics and Instrumentation of the Bulgarian Academy of Sciences, Bloc II Acad. Bonchev str., Sofia 1113, Bulgaria, e-mail: nshivarov@code.bg

Stefan Shivarov. Mag. Eng., Code Assitance Ltd., 20, 20th of April str., Sofia 1606, Bulgaria, e-mail: shivarov@code.bg

Peter Kopacek, Prof. Dr. Mult., Intelligent Handling and Robotics, Vienna University of Technology, Favoritenstr. 9-11, 1040 Vienna, Austria, e-mail: kopacek@ihrt.tuwien.ac.at

Nedko Shivarov, Assoc. Prof., Central Laboratory of Mechatronics and Instrumentation of the Bulgarian Academy of Sciences, Bloc II Acad. Bonchev str., Sofia 1113, Bulgaria, e-mail: nedko@code.bg