A FLEXIBLE REAL-TIME MOTOR CONTROL SYSTEM FOR ROBOT JOINTS

Li Liu and Andrew B. Wright

Applied Science Department
University of Arkansas at Little Rock
Little Rock, AR 72204

Abstract--A flexible, real-time motor control system consisting of a DC motor and an incremental optical encoder is proposed. The system allows complex real-time control algorithms to be implemented with hardware control excepting PC-based software simulation. A motion control processor is used to bridge the Target PC and the robot joint through I/O devices. Simulink/Real Time Workshop (RTW) are the software interface loaded in the Workstation PC which generates the control codes and stand-alone program for various control algorithms.

I. INTRODUCTION

A robot is a set of linkages connected by joints that are governed by specific control algorithms. The task of control algorithms is to regulate the actuator to control the joint parameters. Although the recent research shows the artificial muscle-based joint has great potentials for the contributions of biologically inspired robots (Jacobs, 1999), the motor-encoder based joint coupling still is the most popular and classic configuration in robotics. At present, numerous control algorithms have been developed for robot manipulators according to this kind of joint configuration (Baines and Mills, 1998; Niemeyer and Slotine, 1991; Kelly and Ricardo, 1996). Many of them, however, are proposed based on software simulation because developing and implementing these sophisticated control strategies are difficult and time consuming, especially implementing them into the real-time hardware control. A user-friendly programming environment is essential for the development of the controller. The robot control system must have enough flexibility to meet the change of the algorithms or the modifications of physical structures. Furthermore, the system should meet the real-time requirements of various algorithms. Generally, to implement a real-time robot control, a 10ms-20ms sampling time is basically required. According to the traditional prototyping method, system and design engineers have to be both control specialists and computer specialists. They need to spend a lot of time on high level design, analysis, testing, and tuning as well as programming the algorithms into the hardware. This paper presents an innovative joint control system that automatically integrates design, testing, hardware-in-the-loop (HIL) simulation using Simulink/RTW, which graphically develop and generate the codes effectively. The basic development process for the robot joint controller using the system proposed here is shown in Figure1.

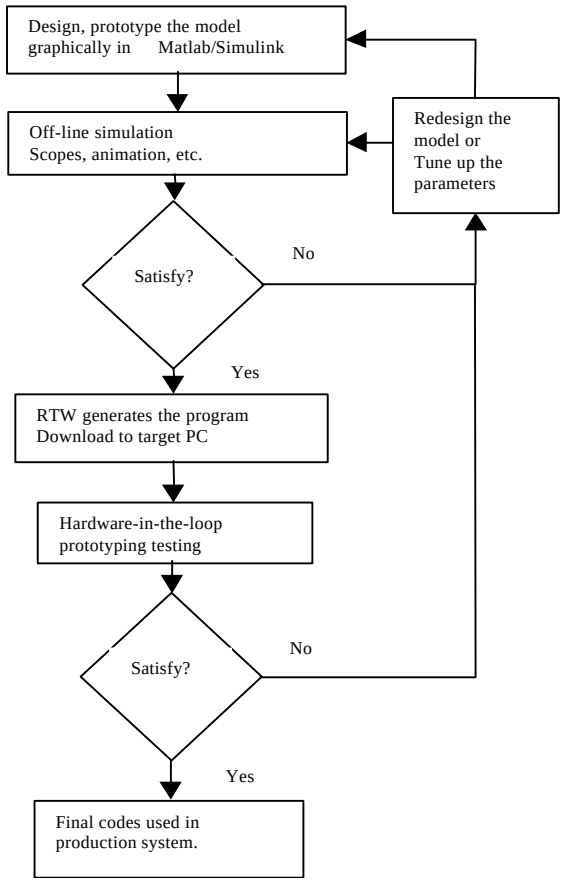

Figure1. The interactive design-prototype-test development process for the robot joint control.

The user develops a model in Simulink. In robot control, this involves modeling plant dynamics. The user can use Matlab, Simulink, and toolboxes to develop algorithms and analyze the results. Then, the model is run in Simulink with off-line simulation. If the results are not satisfactory, the user can iterate the modeling /analyses process until results are acceptable.

Once have achieved the desired results, the user can generate downloadable C code and executable program using Real Time Workshop and Watcom C/C++. With the HIL simulation, the user can tune parameters and further refine the model, again rapidly iterating to achieve required results. Finally, the codes are ready for use in production systems.

II. SYSTEM CONFIGURATION

Generally, there are two methods to configure the control circuit for robot joints (Enriquez *et al*., 1995). The first approach is to use the Personal Computer (PC) to control the joints through serial or parallel ports, or through PC-bus (Mauer *et al.*, 1989). The Host PC has to take care of all the tasks such as algorithm design, control implementing, data collection and processing. This approach is completely flexible and any control algorithms can be generated and run at the Host PC. However, it suffers some disadvantages in real-time applications. The reason is the complex control algorithms and other tasks will occupy lots of PC resource so that it degrades the sample frequency that is important for real-time control. This is an extreme approach, which uses the PC to execute whole tasks, and is used in few practical applications. Another method is to use the microcontroller to control the joint directly. According to the different situation, the user can customize the hardware and circuit to meet the real-time requirements for specific control. This method may meet the real-time application but it is not easy to change the control algorithms and afford sophisticated computation due to the limitations of microcontroller. The user has to design the algorithms somewhere and then program the model into the hardware using the low-level language to verify the results. Because the controller design is an interactive design-prototype-test development process, using this method will cost much time on programming other than dedicated controller design. Importantly, It may not be flexible enough when the user wants to change the algorithms and hardware structures.

A compromise is considered in this paper. Through combining the advantages of first and second approaches, a new flexible motor control system is constructed. Figure 2 is the computer architecture of the flexible real-time motor control system. Where, the Target PC and joint processor (HCTL1100) carry out the motor control in union. A digital I/O board is put between target PC and joint processor to finish the communication. The Host PC is a workstation where the user designs and simulates the control algorithms using Matlab/Simulink and then builds them into DOS-based stand-alone program using Real Time Workshop. The files are downloaded to target PC through the Ethernet with the TCP/IP protocol.

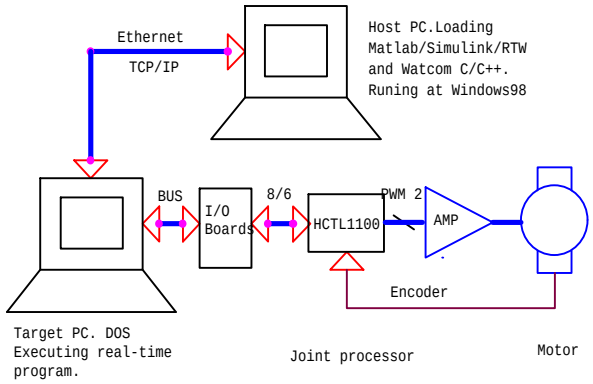

Figure 2. Hardware architecture of the flexible real-time motor control system.

III. SYSTEM HARDWARE

The Workstation runs windows98 with Matlab, Simulink, and Real Time Workshop. It is the Host where the control algorithms are designed, modeled, simulated, and the real-time programs are built. For best performances, larger memory and fast CPU are recommended for the Host PC.

A PC-AT (486 or Pentium) severs as the Target PC and runs DOS for real-time programs. It executes real-time control algorithms and sends or receives data to or from the joint processor. Since DOS is a single tasking operating system where only one process can exist at any given time, so it guarantees that the program can execute in real-time by interrupt service routine.

I/O broads provide the communication between the Target PC and external hardware. In this case, the robot joint is controlled by Target PC through HCTL1100. That means that the digital I/O boards are enough for this application. The 24 bit parallel digital I/O board, PIO12, provides three ports that can interface two joint processors at same time. Other TTL/DTL compatible digital I/O boards, like CIO-DDA06, can also be used in this system with small modifications of programming.

The traditional robot joint control method is to use the target PC to control the joint directly. This configuration gains flexibility but increases the processing time, which is very important for real-time control. As mentioned before, a general-purpose

motion control IC HCTL1100 is added between the Target PC and joint serving as the joint processor. The HCTL1100 is a high performance motion control IC, which frees the target PC for other tasks by performing the time intensive functions of digital motion control. The chip receives control command from the Target PC and independently process a given task through an 8-bit bi-directional multiplexed address. Also, the joint information, like position, velocity and acceleration can be fed back to the Target PC for updating the control parameters. The user can then program the joint processor for one of the four control modes: Position control; Proportional velocity control; Trapezoidal profile control; Integral velocity control. These four built-in control modes correspond to four control strategies, which can satisfy the major control requirements for robots. For example, at the Trapezoid profile mode, the user specifies only the desired final position, acceleration and maximum velocity then the joint processor will compute the necessary profile to generate the motor command.

However, one of appealing specifications is that the HCTL1100 can be set to Initialization /Idle mode and then the joint can be controlled by control algorithms generated by the Target PC. In this case, the joint processor is just like a bank of registers and the Target PC can send the command to motor command register directly. This gives the user another approach to control the joint as well as use the control modes of HCTL1100. Even when the user uses Initialization /Idle mode, the actual position registers can be always read to get the current encoder count for the Target PC. This compensates the shortcoming of the joint processor with its only four control modes. Theoretically, any flexible control algorithms can be implemented in real-time at this configuration.

IV. SOFTWARE IMPLEMENTATION

To actualize flexible real-time motor controller, software integration is necessary. Simulink is easy to use for generating the joint trajectory and modeling the control algorithms due to its graphical interface. Real Time Workshop creates the stand-alone executable program for the model. However, the I/O device driver must be developed before using RTW so that the external hardware can be treated as a Simulink block. When the user simulates the model with the I/O device, the output module will behave like a terminator block. After getting the satisfied simulation, the user can use the RTW to compile and link the Simulink model with the device driver, which contains the code that handles joint processor, timing, interrupt, data logging, and background tasks. Here, the Watcom C/C++ is used for developing the high efficient 32-bit DOS-executable program. Figure 3 illustrates the software control process of a real-time motor controller.

The device driver is written in C language in the form of an S-function. S-functions are sections of codes, written in a specific format in C or Matlab programming language. The standard S-function samples can be found in Matlab package. Here, three principle functions involved in I/O device drivers will be discussed. They are modified specifically for the application at hand.

1)mdlInitializeConditions: This function is called only once at the logging of each simulation. It is used to initiate the device and software-reset the device. Also, the user can set the sample time, initial position, etc. for the joint processor.

2) mdlOutput: This function is called at each step of simulation. There are five arguments in this function.
Y: the output vector.
X; the state vector.
U: the input vector.
S: the Simstruct for this block.
Tid: The task ID.

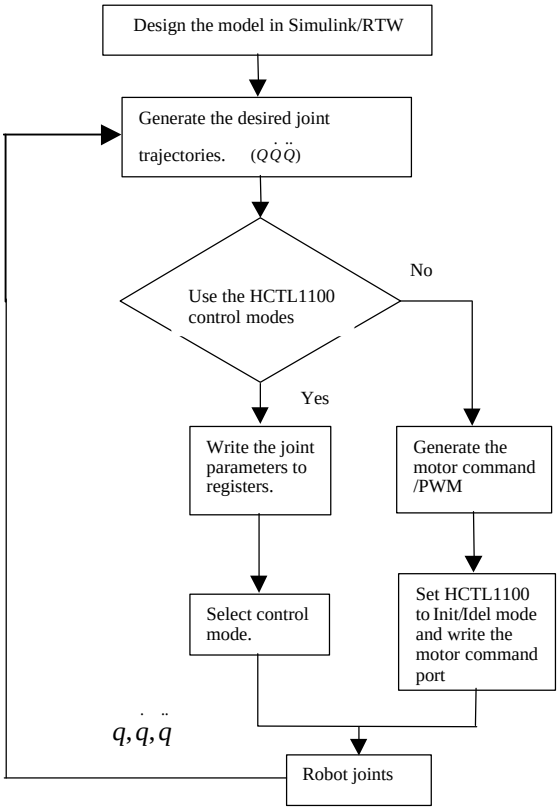

Figure 3. Software flowchart of controlling the robot joints using flexible real-time motor controller.

The user can put each input (U) to the specific function, which represents the control algorithm or control mode. Output vector Y can be used to check the output results of I/O device block.

3)mdlTerminate: This function is called once at the end of the simulation. It is used to reconfigure all

ports of digital I/O board as input ports as a precaution and set back the HCTL1100 to the Initialization /Idle mode.

The joint processor, HCTL1100, also needs programming so that the corresponding control algorithms can be integrated into the device driver. Using ANSI C to develop the program for HCTL1100 guarantees that it can be seamlessly incorporated into the device driver, which is developed in S-functions. Programming of the joint processor is straightforward. A head file (hctl.h) includes all macro definitions that are easy for the joint processor to access registers. Two basic read and write functions are called by all macros. For example, use following definition to get and set the motor command register.

```
#define get_motor_command    read_hctl(0x08)
#define set_motor_command(value) write_hctl(0x08,value)
```

Another, the four control modes of HCTL1100 are written into four functions so that it is easy to call them whenever the control strategy need them. The simple position control function for position control mode is shown in figure4.

```
void position_control(int Zero,int Pole,int
Gain,int Time, float position)

{
...
/*program digital filter with zero, pole and
gain*/
 set_filter_zero (Zero);
 set_filter_pole(Pole);
 set_filter_gain(Gain);
 set_sample_time(Time);
 /* Set Position Mode */
 write_hctl (0x05,0x03);
/* Command Desired Position */
set_com_position ((position)^0xFFFFFF+1);
/*2's complement */
}
```

Figure 4. Code sections for programming HCTL1100 to position control mode. This function can be called by mdlOutput.

V. CONCLUSION

In this paper, an innovative flexible real-time motor control system for robot joints is proposed. It extends the controller design from software simulation to real-time experiment. Various control algorithms generated by the Host PC can be downloaded, run, and validated on the Target PC. The joint processor can afford the tasks of data collection and computation as well as support built-in four motion control modes. The robot joint can be controlled by Target PC, joint processor, or both depending on different control schemes. It shortens development cycles and reduces costs by implementing hardware-in-the-loop simulations for robot joints. Compared to the large and costly real-time distribution system, this economic motor control device is very useful in academic research. This fast design-prototype-test tool has been used in UALR for working robot research.

REFERENCE

Baines, P.J., J.K. Mills (1998) Feedback linearized joint torque control of a geared, DC motor driven industrial robot. *International Journal of Robotics Research*. 17, 169-92.

Enriquez, A.L., A. H Eltimsahy and M. M. Jamali. (1995). Flexible control for robot manipulators. *IEEE Micro*. 15, 55-60.

Jacobs, R. (1999). Biologically inspired legged robots for space operations. *Final report NIAC Phase1*. 6-8.

Kelly, R., R. Carelli (1996). A class of nonlinear PD-type controllers for robot manipulators. *Journal of Robotics System*. 13, 793-802.

Mauer, G.F., J. Skaggs and R.M. Turner (1989). A Flexible PC_Based Robot Controller. *Proceedings Robots 13 Conference. Society. Of Manufacturing Engineer*. 23-31.

Niemeyer, G., Jean-Jacques E. Slotine (1991). Performance in adaptive manipulator Control. *The International Journal of Robotics Research*. 10, 149-161.