A Design of Realtime Communication Based on EtherCAT in Industrial Robot Control System Based on LinuxCNC

Shi Bu-Hai[1], Wang Yong-Zhi[1], Ding Chuan[2]
1. School of Automation Science and Engineering, South China University of Technology, Guangzhou, 510641, China
Email:bhshi@scut.edu.cn
Email:wangyz_fy@163.com

2. School of Design, South China University of Technology, Guangzhou, 510641, China
Email:chding@scut.edu.cn

Abstract: In the control system of a six-axis industrial robot, a realtime and fast communication solution is needed to improve the kinematic accuracy of end effector. This paper designs a communication solution based on the CANopen over EtherCAT protocol, which supports PDO and SDO communication object in object dictionary. The robot controller is modified from LinuxCNC, and the communication solution is realized as a realtime component of HAL. This paper uses the communication solution to transfer the value of absolute encoder between PC and servo motor, and it can work stably and effectively.

Key Words: EtherCAT protocol, CoE application protocol, HAL component, LinuxCNC

1 INTRODUCTION

In the past few decades, industrial robots gradually have taken over human tasks in some manufacturing field, such as assembling automobile, welding process and so on. What's more, it will be used in more high-end manufacturing area and greatly accelerate the development of the manufacturing in the future. The end effector of a six- axis industrial robot, a kind of advanced industrial robot, can arrive at any position in three-dimensional Cartesian work space with any attitude. In the control system of a six- axis industrial robot, the motion controller and task controller are regarded as the core modules[1]. Nevertheless, in order to complete an entire control system, we need an efficient and real time communication scheme to transfer command and data between motion controller or task controller and hardware devices (IO devices, servo motor, etc.).

To improve the kinematics accuracy of an industrial robot, the motion controller should be responsible for rapidly sending command to servo motor. Meanwhile, the user's input command through control panel should arrive at peripheral equipment with low latency. The communication solution in a robotic control system should be capable of transferring periodic dense realtime data and aperiodic probabilistic data.

EtherCAT protocol has excellent performance, for example, updating 100 servo axis with 8 bytes input and output data each only costs 100us[2].The six-axis robot control system in this paper makes the best of EtherCAT protocol to realize outstanding control performance. After introducing the EtherCAT protocol and LinuxCNC, this paper will develop a concrete communication solution which works in open numerical control system LinuxCNC.

This work is supported in part by Key Laboratory of Autonomous Systems and Networked Control, College of Automation Science and Engineering, South China University of Technology, in part by Industry, Education and Research Program Foundation of Guangdong Province, China under Grant 2014B090901042.

2 EtherCAT PROTOCOL

EtherCAT is a kind of real-time industrial Ethernet raised by Germanic BECKHOFF company in 2003 and is supported and maintained by ETG (EtherCAT Technology Group), which has been applied in many automation solution because of its high message transmission rate, low time delay and flexible topology. As a kind of real time Ethernet, EtherCAT is realized by modifying the data link layer which is the 2nd layer in TCP/IP protocol, and develops a new master-slave mode to control media access instead of the traditional CSMA/CD operation[3].

2.1 Component of EtherCAT Bus System

An EtherCAT bus system consists of master stations and slave stations, and works in full-duplex data transfers mode, as shown in figure 1. In this way, master station sends EtherCAT frame to slave stations. When the frame arrives at some station, such as the 2nd slave station in figure 1, it extracts data that belong to it from the frame,inserts those data to the specified section and sends the frame to the next linked slave station. If the current slave station is the last one, then it will send the frame back to the master station.

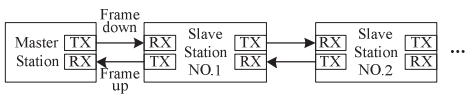

Figure 1 EtherCAT frame flow schematic

The EtherCAT master usually uses universal Network Interface Card to communicate with the bus, but there are native EtherCAT device driver and generic EtherCAT

device driver can be chosen for the NIC. EtherCAT master station with Native EtherCAT-capable Ethernet driver receives and sends frame in interrupt-less operation, so master station allows a high performance and can work in realtime linux kernel extend by RTAI package.

Slave station realizes the functions of sending and receiving frame and executing commands under the help of special ESC (EtherCAT Slave Controller). Each ESC has four ports available to bus each, and the order that the frame transfers by in internal ESC is a logic ring[4]. If some port doesn't connect to bus, the ESC will automatically close the port and reach to next port.

EtherCAT supports flexible topology based on SEC multiple ports. There is an EtherCAT topology shown as figure 2, MS, SS0, and P.0 respectively are the short name of master station, first slave station and first port in ESC. If the master station sends a frame, the order of transporting frame is from SS0, through SS1~SS10, finally to SS11. After the last slave station receives the frame, the frame will be returned to master under the order, which begins at SS11, through SS8, SS7, SS2, finally arrives at master through first slave SS0. Designed proper topology, EtherCAT can realize redundancy and support hot swapping[5].

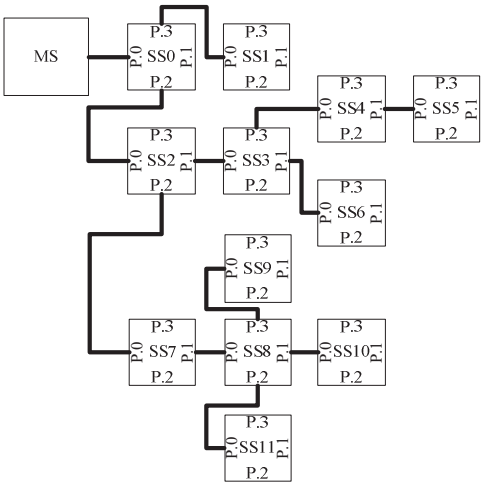

Figure 2 EtherCAT topology illustrating flexible topology

2.2 EtherCAT Frame Structure

EtherCAT frame is modified directly from DIX Ethernet V2 criterion, and the frame is shown in figure 3. The number under each section of frame means the length of corresponding section.It is a rule in this paper that numbers with parenthesis indicate the number of bytes and the numbers with brackets indicate the number of bits. EtherCAT frames have the same frame header including destination address, source address and type, but there are also some differences: when EtherCAT frame is transfered in same network segment, the MAC destination address is 0xffffffff, and type of EtherCAT frame is 0x88A4[6].

Telegram Length section means the length of EtherCAT Telegram including datagrams. Type in EtherCAT Header only one type filling with bit 1, which means the EtherCAT Telegram is communicate with slave station.

In an EtherCAT frame, it is allowed to have more than one datagram, but each datagram belongs to one specified slave station. Datagram header is the most important section of EtherCAT frame. Command of datagram header will determine addressing mode and that the datagram works in read or write way. Address section determines that which slave station should receive and process corresponding datagram. Addressing mode has device addressing and logical addressing which needs to use FMMU (Fieldbus Memory Manager Unit) in ESC.

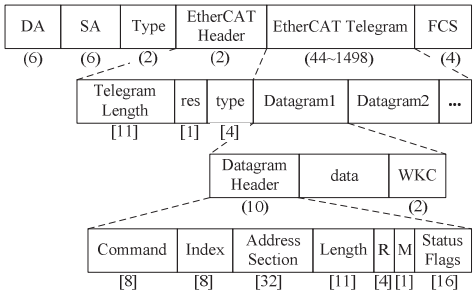

Figure 3 EtherCAT frame structure

3 ROBOT CONTROLLER

A six-axis industrial robot is a six-degrees-of-freedom system, and the computation task focuses on inverse kinematics solution, route and velocity planning, and interpolation arithmetic. The positional accuracy of end effector in Cartesian space lies on whether the rotational angle in articulate space of each servo motor is correct and arrive at motor driver in defined time.

3.1 Architecture of LinuxCNC Control System

LinuxCNC is an open numerical control system originally supported by NIST, the National Institute of Standards and Technology. LinuxCNC is an excellent open source software inherited the modular design technology from Linux, and its simple architecture is shown in figure 4[7].

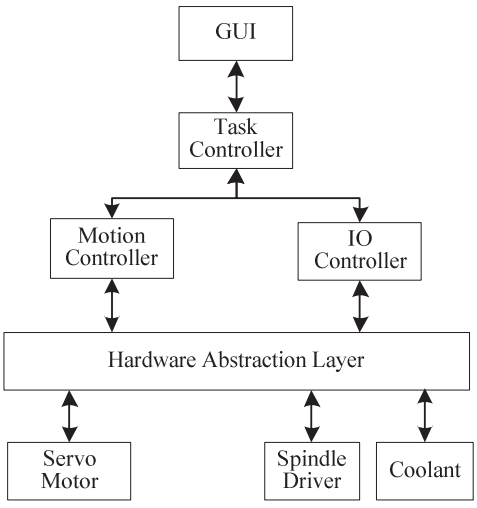

Figure 4 LinuxCNC functional architecture

GUI (Graphical User Interface) is the top level in the architecture which generates command and monitors the status of the whole system. The motion controller solves

forward kinematics and inverse kinematics solutions and plans path and velocity.The task controller schedules tasks passed from user control panel and issues corresponding command to motion controller, hardware device, etc. Motion controller is running in realtime space which is implemented by extending general Linux operating system with RTAI package, and task controller and IO controller is running in non-realtime space[8].

The above each several part is organized and implemented in HAL as component. Generic Linux kernel extendimg to a realtime kernel by RTAI package supports rtai_ interface function, and HAL in LinuxCNC supports hal_ interface function above RTAI. The EtherCAT communication module is a realtime HAL component.

3.2 Component of Hardware Abstraction Layer

HAL is based on the same principles that are used to design hardware circuits and system[9], which is the key component to realizing communication. HAL concepts used in this paper are as following.

1) Component: A HAL component executive finite and certain task, like an ADC in circuit, which consist of pins, signals, and functions. And a HAL component can be loaded and unloaded according to requirements.

2) Pin: Pins realize the input and output of component, like physical pins in ADC, which also have parameter to define direction (in or out) and size (bit, float, unsigned integer, and signed integer).

3) Signal: Signal is used to connect pins as wire does, and by the signal, a component can exchange data with linked component. There is a rule that a signal can connect only one out direction pin and some in direction pins.

4) Function: A function in component is a computing element, which defines what a component can do. If the component runs in real time, the function must be loaded to a thread. If a component runs in userspace, after the function runs one time, the function and component will exit both.

5) Thread: Behaviors of a realtime component depends on the function part of the component, but calling the function must be done by a thread. And the functions loaded to the same thread run at specific intervals and execute in order every time the thread runs. A thread can load some functions that don't belong to same component.

4 DESIGN REALTIME COMMUNICATION

The highest layer in EtherCAT protocol is application layer which can be directly used by control task. The application layer contains some application-layer protocols, such as CANopen over EtherCAT, Servo Drive over EtherCAT, Ethernet over EtherCAT, and so on[10]. In this paper, a robot control system based on LinuxCNC will adopt CoE protocols to complete realtime communication program. In CoE, process data objects and service data objects are supported to transfer periodic dense data and aperiodic uncertain data respectively.

4.1 EtherCAT Master Module

Under the help of EtherLab, which works as a real time kernel module attached to the open source operating system Linux communicating with peripherals devices by a EtherCAT technology[11], the entire control system use IgH EtherCAT Master for Linux as an EtherCAT device drive. General overview of the master architecture introduced by EtherLab in this paper is shown in figure 5.

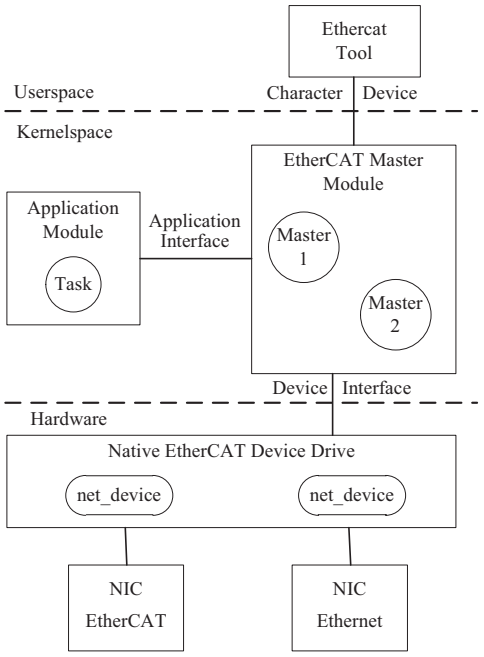

Figure 5 master architecture

The task in application module running in kernel space transfers data and command through EtherCAT master instance. A master module can include more than master instance, but if you want to create some master instances to execute task, the PC must have more than one NIC[12]. EtherCAT tool can run in command-line interface, so it is convenient to examine and configure master instance. If you want specific NIC worked as an EtherCAT device, you should support configure information when load EtherCAT module.

4.2 Application-layer Protocol:CoE

CANopen over EtherCAT features PDOs and SDOs as well as an Object Dictionary, the CANopen state machine and fully supports all CANopen device profiles[13]. CoE uses object dictionary to map the corresponding PDO, in this way, the master station and slave station can exchange command data and status data periodically. Object dictionary supports application program interface to application program, and when you read or write PDO or SDO communication object, it will automatically realize CoE communication, like socket in TCP/IP protocol.

The master station sends rational turns of servo motor to the slave station and the slave station returns feedback value to master station, so we design PDO entries and map it to object dictionary. For each entry of object dictionary, the index contains a 16 bits index and 8 bits subindex.

0x6000~0x6fff is usually designed for MISO data, and 0x7000~0x7fff is usually designed for MOSI data[14]. According guild regulations, the article assigns PDO entries and some entries are shown in table 1.

Table1. PDO Entry in Object Dictionary

Entry Index	Entry Subindex	Variable Name	Data Type
0x6000	0x01	svr_alm	Uint8
0x6000	0x02	svr_limit	Uint8
0x6000	0x03	svr_ioin	Uint8
0x6000	0x04	svr_absrdy	Uint8
…	…	…	…
0x7000	0x01	svr_on	Uint8
0x7000	0x02	svr_clr	Uint8
0x7000	0x03	svr_ioout	Uint8
0x7000	0x04	svr_absreq	Uint8
…	…	…	…

After designing PDO entries, we should map PDO entries in area of data object to PDO communication object in object dictionary. 0x1600~0x17ff is retained for RxPDO mapping, and 0x1A00~0x1bff is retained for TxPDO mapping. CoE assign PDO communication object with the help of synchronization manager, for example in table 2.

Table2. PDO Mapping

Sync Manager Index	PDO Communication Object Index	PDO Entry Index
0x1C12	0x1600~0x1605	0x7000,0x7010, 0x7020,0x7030, 0x7040,0x7050
0x1C13	0x1a00~0x1a05	0x6000,0x6010, 0x6020,0x6030, 0x6040,0x6050

4.3 Realize Communication Component in HAL

Before loading a real time component of CoE application program to specific thread, we need firstly configure master station and slave station. The configure information of master is contained in /etc/sysconfigure/ethercat file. The slave station and CoE object dictionary is configured and released in EXTRA_SETUP() and EXTRA_CLEANUP() function. These two functions are optional section of a HAL component.

The main steps of EXTRA_SETUP() function is shown in figure 6. Each function can be found in ecrt.h file. PDO image is a set of all PDO entries, and PDO domain is a technology to manage PDO image. The application will directly operate PDO domain.

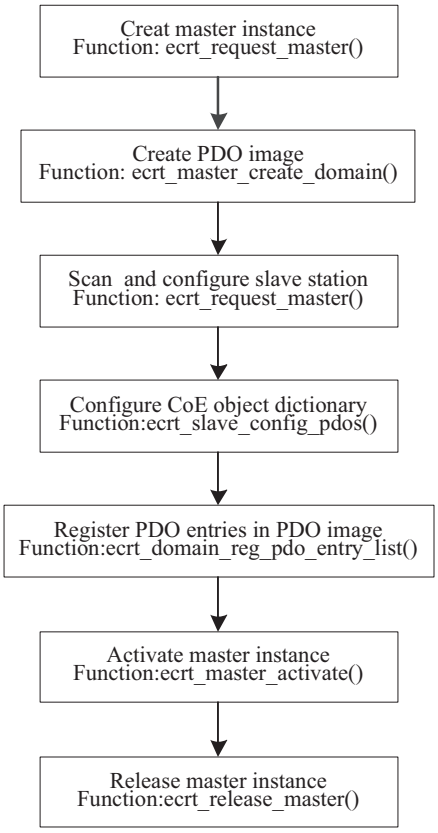

Figure 6 steps order of EXTRA_SETUP function

The process of receiving data, processing data and sending data is contained in FUCTION() HAL component. To guarantee the right order of master's operation, the classic spin lock technology is introduced to implement the mutually exclusive access to master instance. The main process of realizing FUNCTION component is in section below. As shown in figure 7, it can explain the main procedures of master station worked in LinuxCNC environment.

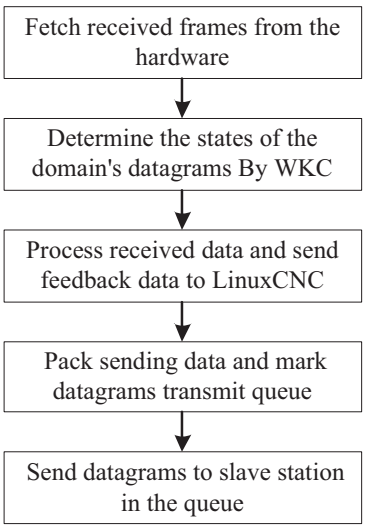

Figure 7 main procedures of master station

5 EXPERIMENT

The controlled member is a six-axis industrial robot, and each axis is drove by servo motor. The robot control system, modified from LinuxCNC 2.5.3, runs in industrial PC(Intel J1900, 2 GHz) with real time operation system based on Debain 7.0 and RTAI package. The servo motor is an AC servo motor of Yaskawa Σ-7 series supporting absolute encoder. The experiment aim to that master station, industrial PC, send request of reading absolute encoder counter, then slave station, AC servo motor control board, responses request and sends counter value to master station.

Because of the introduction of absolute encoder, it is not essential to zero when running the machine. In running process, the absolute encoder saves current position. If the robot suddenly loses power or normally power off, whilst robot reboot, the control system can quickly get current position of joint, so the control system can plan route without going back to the zero. The result of experiment is shown in figure 7 and figure 8. The figure 8 shows that the robot is normally running, and the figure 9 shows that the real position of each joint through the absolute encoder.

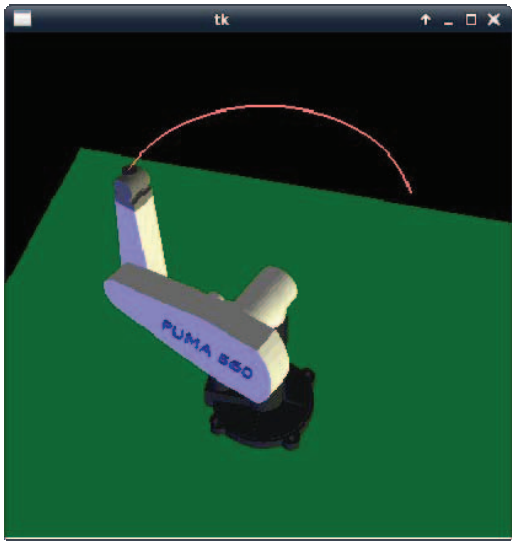

Figure 8 industrial robot running chart

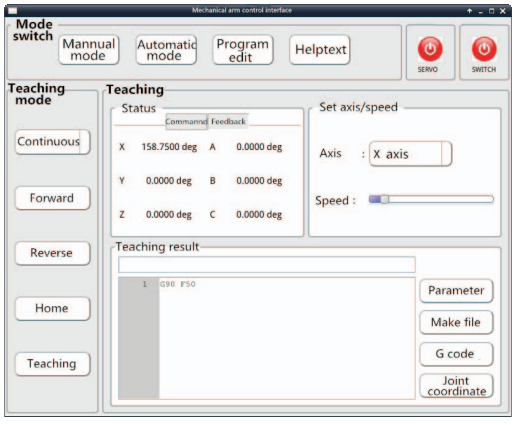

Figure 9 Industrial robot control plane

6 CONCLUSION

This paper introduces and analysis the EtherCAT protocol and CoE protocol. Base on LinuxCNC, the paper realizes the communication component in HAL. After experiment, it has been proven that the realtime communication technology in this paper is useful and effective.

REFERENCES

[1] Wang Tianran, Qu Daokui, Open System Architecture for Control System for Industrial Robot[J]. Robot, Vol.24, No.3, 256-261,2002.

[2] Liu Yanqiang, Wang Jian,Shan, Chunrong.Study on the Multi-axis Motion Controller Based on EtherCAT[J]. Manufacturing Technology & Machine Tool, No.6, 106-109,2008.

[3] Li Muguo, Wang Lei, Wang Jing, Industry Ethernet Data Gathering System Based on EtherCAT[J].Computer Engineering,Vol.36, No.3,237-239,2010.

[4] MA Chunmin, KANG Cunfeng, HUANG Xudong. Research on the EtherCAT master under Linux[J]. Manufacturing Automation, Vol.33, No.4,78-82,2010.

[5] Huan Ji, Xiao Wenlei, Liu Yanqiang, Redundancy and hot swap technology in industrial Ethernet EtherCAT[J]. Journal of Beijing University of Aeronautics and Astronautics, Vol.35, No.2, 158-161,2009.

[6] Wang Lei, LI Muguo, WANG Jing, Research of fieldlevel realtime Ethernet networked control system based on EtherCAT protocol[J]. Computer Engineering and Design, Vol.32, No.7, 2294-2297, 2011.

[7] The LinuxCNC Team, LinuxCNC Integrator Manual V2.6[DB/OL]. http://www.linuxcnc.org, 2016.

[8] Guo Xietao, Research of Robot Core Control Algorithm Based on LinuxCNC[D]. South China University of Technology, Guangzhou, 2014.

[9] The LinuxCNC Team, LinuxCNC HAL Manual V2.6[DB/OL]. http://www.linuxcnc.org, 2016.

[10] Xu Ji, Liu Yanqiang, EtherCAT-Industrial Ethernet Fieldbus and Its Driver Design and Application[M]. Beihang University Press, Beijing, 2010.

[11] Su Bing-en, Research and Development of six-axis Industrial Robot Control System Bsed on EtherCAT[D], South China University of Technology, Guangzhou, 2013.

[12] The EtherLAB Team, IgH EtherCAT Master 1.5.2 Documentation[DB/OL]. http://www.etherlab.org, 2016.

[13] Martin Rostan, Beckhoff, CANopen over EtherCAT-taking a CAN technology to the next level, Proceedings of IEEE International Conference on Communications, 2005.

[14] CHEN Zai-ping, WANG Feng, The Research of Structural Analysis and Slave Node Based on CANopen[J]. Manufacturing Automation, Vol.32, No.2, 27-31, 2010.