Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 13 (8): 1269-1280, 2013
ISSN 1812-5654 / DOI: 10.3923/jas.2013.1269.1280

Design of the High-speed Communication System Based on the Serial Real Time Communication Specification II

[1]J.L. Gao, [1]X.X. Huang, [2]H.T. Song, [1]W. Chen and [3]N.H. Quan
[1]School of Automation, Guangdong University of Technology, Guangzhou, 510006, China
[2]School of Business Administration, South China University of Technology, Guangzhou, 510641, China
[3]Mindray Medical International Limited, Shenzhen, 518057, China

Abstract: To realize the high-speed, realtime and high-reliability communication between the controller and the servo motors, the high-speed communication system is developed. It is based on Sercos II (serial real time communication specification II). One modeling scheme fitting for the communication system based on Object-Oriented Technology (OOT) and Unified Modeling Language (UML) is presented. It gives in-depth discussion about modeling the system based on UML, designing of the software and hardware of Sercos II master/slave station, real-time performance calculation and consistency verification. Eventually, the system is applied in 3-axis numerical control machine tool. It can send the control instructions with 4 bytes to 1-6 servo motion axles and receive the feedback parameters with 4bytes within control cycle of 0.5 m sec.

Key words: SERCOS, servo driver, motion control, master/slave station

INTRODUCTION

The mainstreams of motion control system are open, network and digitization. The high-speed, high-precision, high-reliability communication among control unit and servo devices is the basis of ensuring the optimum performance of the whole motion control system. The traditional analog interface has the defects such as sensitivity to noise and EMI (Electro-Magnetic Interference), low resolution, unavoidable signal drift, complicated wire connection and so on. Thus, it is unable to satisfy the modern motion control.

Sercos II is the 2nd generation real-time serial data communication bus (GB/T18473-2001, 2002), which is applied to the controller, actuators, mechanical and electrical equipments and programmable controllers. Based on the advantages of entirely open interface, high transmission rate, rigorous real-time data synchronization and optimum anti-interface, comparing with other buses such as MACRO, Fire Wire, EtherCAT, etc., Sercos II can reliably realize the synchronal real-time multi-axis motion control and provide the high-performance interface (Xun and Yin, 2005; Chen and Yang, 2003; Liu and Huan, 2006). Nowadays, the Sercos nodes successfully applied in industry automation field have exceeded 2.5 million in developed countries. It is necessary to implement the complex initialization before system running. The technical difficulties on developing the Sercos II master and slave drivers restrict its promotion and application in China. The embedded system is different from other universal pure software or hardware systems, which is a collaborative design process of software and hardware. The major problem of analyzing and designing embedded system is listed: There is not an unified engineered description method to describe the structural and behavioral characteristics of the requirement, analysis, design, test and maintenance for embedded system (Chen and Chen, 2005; Zhu and Yu, 2004; Gu *et al*., 2010).

Therefore, present the modeling scheme fitting for the analysis and design of embedded systems based on object-oriented technology and unified modeling language. Then, develop the embedded high-speed communication system based on Sercos II, which realizes the data exchange between the control unit and servo devices. It gives in-depth discussion about modeling embedded systems based on UML, designing of the software and hardware of Sercos II master/slave station, real-time performance calculation, consistency verification and practical application in 3-axis CNC (Computer Numerical Control) machine tool.

MODELING METHODOLOGY

The object-oriented designing methodology for the embedded system is that all the operations are described by object and message (or event). The OOT is adopted during analysis and design stage to construct the system structure with clear hierarchy, fine portability, easy

Corresponding Author: H.T. Song, School of Business Administration, South China University of Technology, Guangzhou, 510641, China

expansibility and redefinition. The resources for embedded system are limited (e.g., memory, CPU, etc.) and specificity (e.g., many hardware devices are customized according to application). It is different from the traditional object-oriented software development (Chen and Chen, 2005; Zhu and Yu, 2004; Gu *et al*., 2010; Jia, 2007).

System use-case diagram: The Use-cases indicate an interactive mode between user and system, which captures the system functions and can be realized by mutual cooperation among objects. The participant of the system is just one user, which can be the control unit (master). The directly related Use-cases include the data transmitting and data receiving. Giving further decomposition according to system requirements, the System use-case diagram can be obtained as shown in Fig. 1.

System class diagram: Firstly, decompose the classes and the relations among them according to the requirement analysis results. The basic design idea is as follows: Extract the characteristics of those things related with the system to build appropriate classes on the basis of abandoning those irrelevant things of the system. The class diagram of the system is shown in Fig. 2, which includes two initiative classes: Arm and Sercon 816, five basic classes: Dpram, Interrupter, CtrlReg, TelegramProcesser, SerialInterface and State (state machine).

The initiative class Arm is the major control module in the system. It controls the operation of the object Sercon 816 and processes the messages and diagnostic information which come from the object Sercon 816. The Sercon 816 takes charge of processing protocol-related data and driving the underlying hardware, etc. The five basic classes are used in reading and writing the double-port RAM respectively, processing system interrupt, communicating in the serial interface, processing the Sercos II protocol telegram and controlling the registers of the chip Sercon 816. The State completes data processing and tasks scheduling in every initialization phases of Sercos II protocol. The basic classes can control the state machine to complete the system initialization and enter into the operation phase.

System state diagram: UML offers the hierarchical state chart, which is in favor of description complex embedded system (Gu *et al*., 2010; Jia, 2007). In each state, the system executes one or several processes solely or concurrently. When the condition is satisfied or stimulated by some external signal, the current process should be terminated/suspended and another process should be waked up. At the same time, the state-machine switches the current state to another and outputs a triggered signal. The life cycle of objects could be reflected through the state chart. What's more, not only it can greatly reduce the state numbers in state chart via "XOR" and "AND" mechanisms but also describe the uncertain system states.

The initialization of Sercos II and communication link are essential to the system. The initialization includes five phases, namely CP0-CP4. When the system is in abnormal state, it shall be switched to CP0 to rebuild communication link again according to Sercos II protocol standard. The basic state chart of the system is shown in Fig. 3.

System activity diagram: Activity diagram is one special type of state chart, which can describe the workflow and concurrent behaviors. The activity diagram of the system is shown in Fig. 4, which mainly includes the workflow and concurrent behaviors of the three objects, namely Arm, Sercon 816 and Servo.

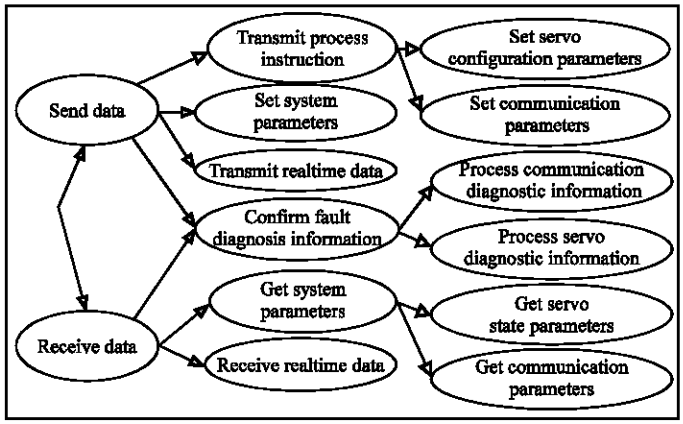

Fig. 1: System use-case diagram for Sercos II communication system

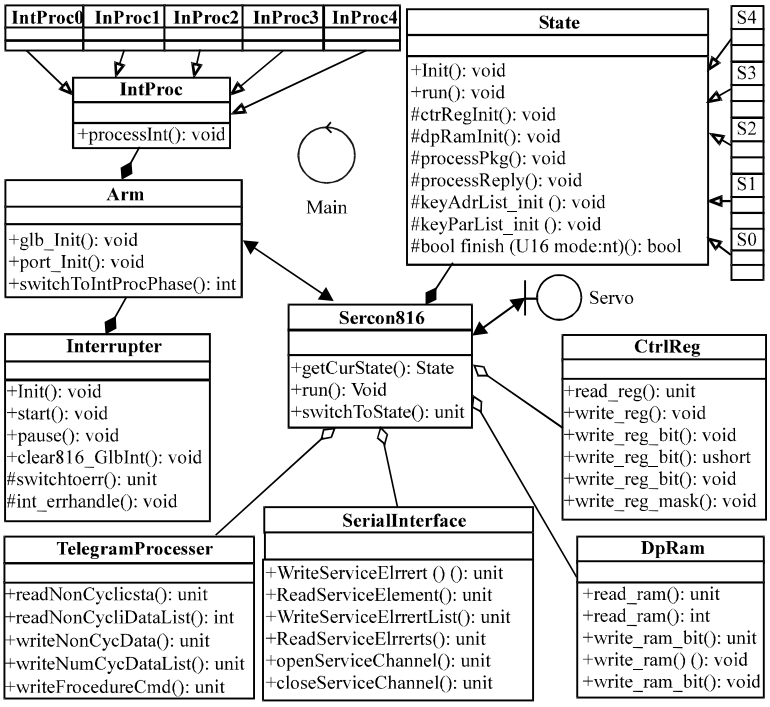

Fig. 2: Class diagram for Sercos II communication system

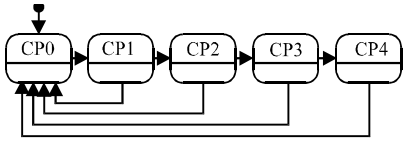

Fig. 3: State diagram for Sercos II initialization process

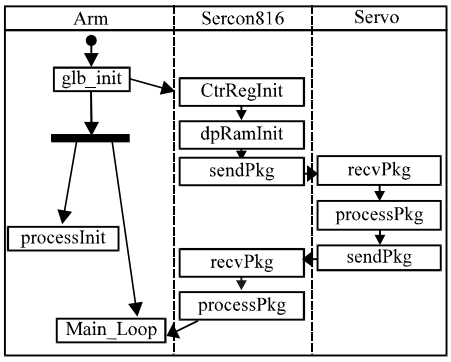

Fig. 4: Sercos II communication system activity diagram

System sequence diagram: Sequence diagram takes time as sequence and gives further analysis to the workflow of system via mutual operation among objects. After obtaining the classes fit for system requirements, the Use-case can be realized via the interactions of these classes (objects) based on the thought of Use-case driving. The sequence diagram is shown in Fig. 5.

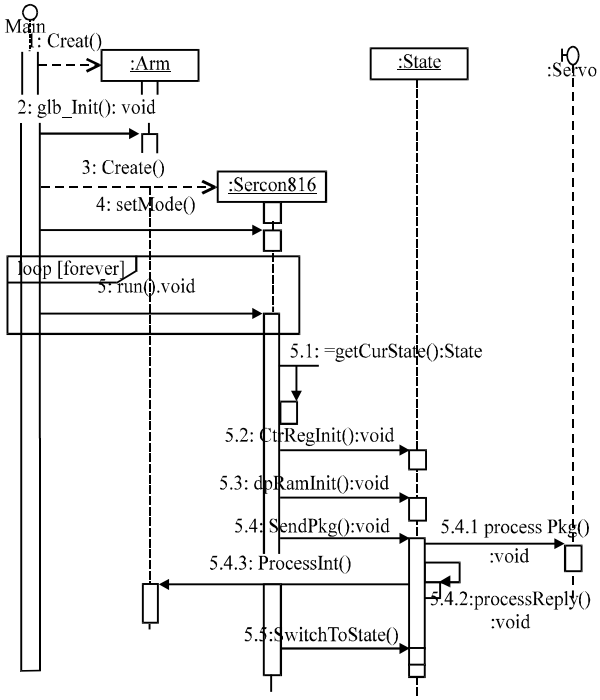

Fig. 5: Sequence diagram for Sercos II communication system

Firstly, the function "main" establishes the object Arm of the control unit and implements the initialization work. Secondly, it creates the object Sercon 816, sets the

operation mode of the master or slave. Finally, it enters into the cyclic process, namely the interaction between Sercon 816 and state-machine (state), to realize the initialization of Sercos II communication, sends messages to servo devices, receives and disposes messages from servo devices. After the link for the current phase had been established, the function "SwitchToState" would be called to switch to the next phase, in turn, from CP0 to CP4. Then it will enter into the normal operation phase.

SERCOS II COMMUNICATION SYSTEM HARDWARE DESIGN

System overall structure: The overall system structure is shown in Fig. 6. Here, the Sercos II master is developed by us and the slaves can be motion controllers, IO (Input and Output) modules and servo drivers and so on. The master and slaves are cascaded successively through two optical fibers with "One In, One Out" to form one SERCOS II bus loop and realize the bidirectional data communication (Wang and Yang, 2007; Yu *et al*., 2008; Ma and Huan, 2008). The master (Control unit) sends instructions to the servo devices via the Sercos II bus. The slaves receive the data instructions from the bus and then run their inner motor-controlled program to drive the relevant motor to operate as the instructions from the master, which is executed by the microprocessor of the servo driver. In the meanwhile, the servo driver feeds its real-time operation state of the servo motor back to the master via the bus, in order to realize the high-speed real-time synchronous control among the control unit and the multi servo devices.

Master/Slave hardware design: The hardware block diagram is shown in Fig. 7. The S3C44B0 is the microprocessor for the master. The chip SN74LVC164245 realizes level translation from 5-3.3V between the Sercon 816 and S3C44B0. The optical fiber transceiver is

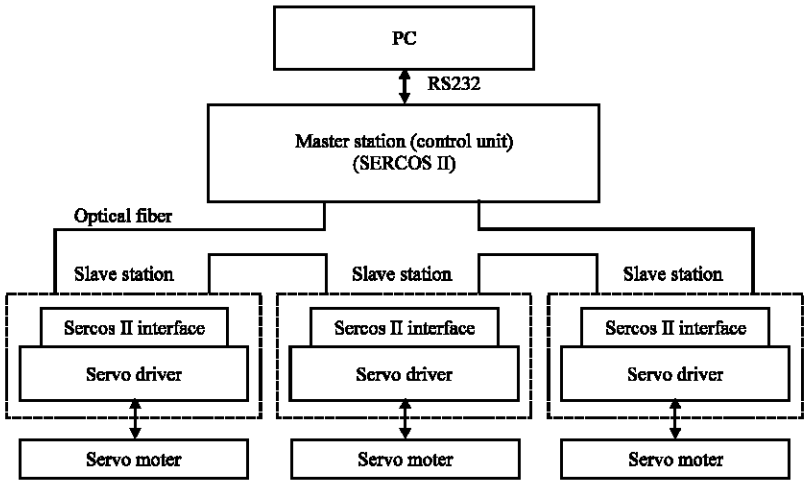

Fig. 6: System topology based on Sercos II

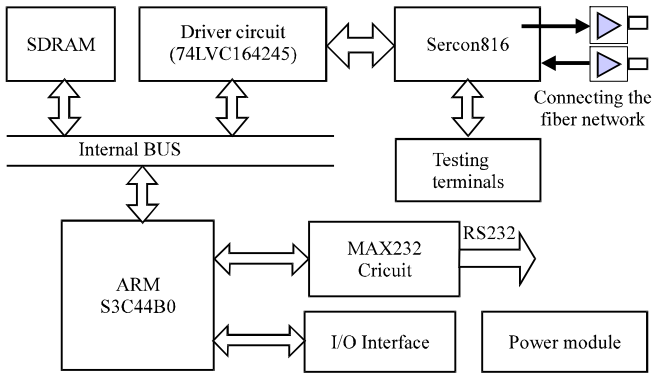

Fig. 7: Hardware scheme for Sercos II master/slave station communication unit

HFBR-1506AMZ and HFBR-2506AMZ respectively. Sercon 816 is the 2nd generation Sercos bus controller chip, designed by the European Sercos Association, which integrates the Sercos II physical layer and data link layer (Yu *et al*., 2005).

SERCOS II COMMUNICATION SYSTEM SOFTWARE DESIGN

Sercos II interface protocol stack: Sercos II protocol defines three types of telegrams: Master Synchronous Telegram (MST), Master Data Telegram (MDT) and amplifier (Servo driver) telegram (AT). The master sends the MST to all slaves as broadcast mode to begin a communication cycle. This MST is used as the reference clock of all the slaves to ensure they begin their control cycle synchronously. Every slave samples its real time value, inserts it into the AT and feeds back to the master within its own time slots. The Sercos II cyclic running sequence is shown in Fig. 8 (Xun and Yin, 2005; Chen and Yang, 2003).

The Sercos II interface protocol stack is shown in Fig. 9, in which its physical layer and data link layer are realized by Sercon 816 in form of hardware. The control program can establish the bidirectional communication between the master and slaves through the master/slave driver and Sercon 816 to control multi servo devices moving synchronously and feed back their real-time value.

The workflow of Sercos II interface protocol stack is as below (Xun and Yin, 2005; Chen and Yang, 2003): The S3C44B0 of control unit writes instruction into the dual

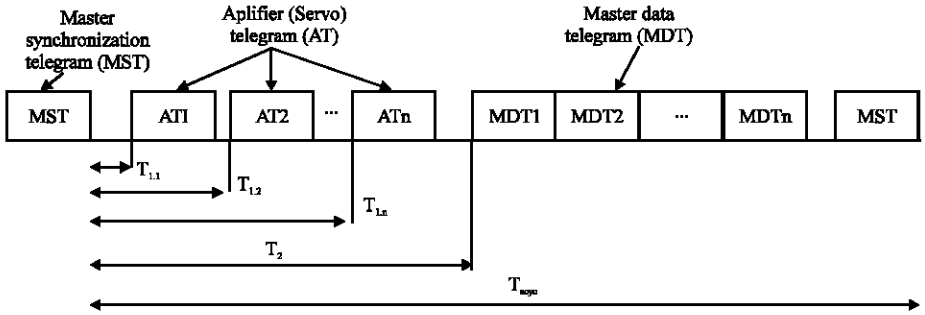

Fig. 8: Cyclic timing sequence for Sercos II communication

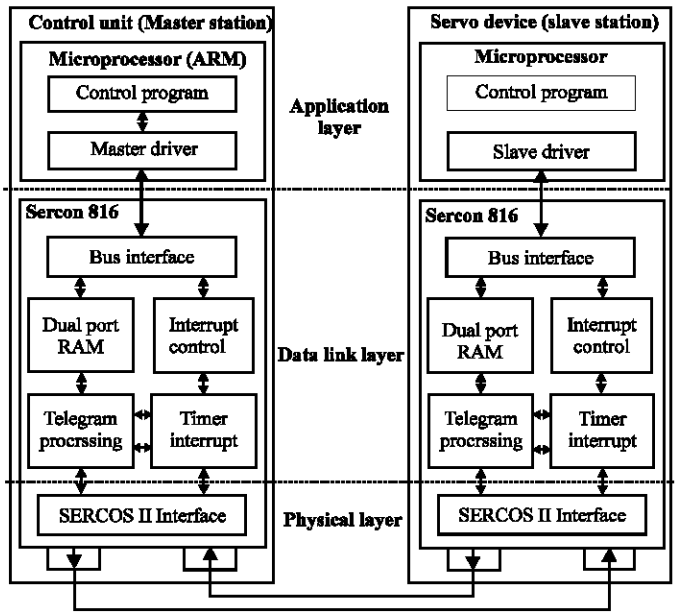

Fig. 9: Sercos II master/slave station interface protocol stack

port RAM of Sercon 816. Sercon 816 reads the instruction from the double-port RAM, assembles it into the preset MDT, completes its serialization and sends the MDT to the related slave. On the side of the slave, Sercon 816 receives and extracts the instruction from the MDT, then writes it into its dual port RAM. The microprocessor of servo device reads the instruction from the dual port RAM, controls the servo motor and writes the sampled actual value into the dual port RAM. Sercon 816 reads the actual value from dual port RAM, inserts it into AT, completes its serialization and feeds back the AT to the master. Finally, Sercon 816 of the master receives and extracts the actual value from the AT and writes it into the dual port RAM. The microprocessor of control unit reads the actual value from the dual port RAM and completes the corresponding processing and so on.

Design of the master driver: It includes the modules such as ARM initialization, interrupt handle, Sercon 816 initialization, telegram processing module, diagnosis module, etc. The ARM initialization is a precondition for the whole system, which mainly completes the initialization of S3C44B0. The master communication initialization flowchart is shown in Fig. 10.

Sercon 816 initialization: It includes the initialization of its internal register and dual port RAM. Its major task is configuring the parameters of Sercos II according to

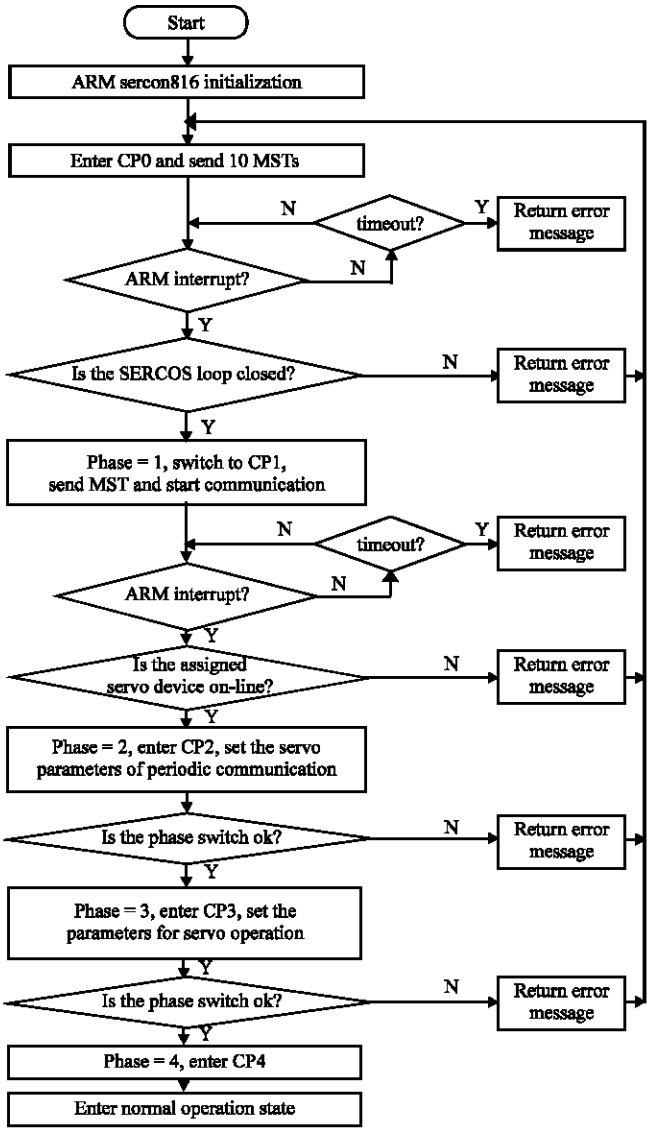

Fig. 10: Sercos II master station initialization flow chart

different initialization phases, such as communication nodes of the system, time slots of the communication cycle and data telegram type, etc. These parameters are essential to ensure the Sercon 816 transmits and receives the Sercos II data telegrams accurately. The process is as follows (GB/T18473-2001, 2002; Xun and Yin, 2005; Chen and Yang, 2003).

Time slot calculation for telegram: The external system clock frequency f_{sclk} is 64 MHZ and the main clock frequency f_{mclk} is 32 MHZ. The baud rate B can be 2, 4, 6 or 16 Mbit sec^{-1}. $T_{bit} = 1/B$. The jitter time J_t is 1us and the loop time delay T_{cycdel} is 1 us. The corresponding time slots can be calculated as follows:

Frequency division factor:

$$M_{clkdiv} = f_{mclk}/1\text{-}1 = 31 \quad (1)$$

Internal time delay compensation:

$$T_{start} = 5.5\times T_{bit}+3.5/f_{mclk} = 0.4531\text{us} \quad (2)$$

Initial value of the counter:

$$T_{cntst} = T_{start}/1+1 = \lfloor 0.453+1 \rfloor = \lfloor 1.45 \rfloor = 1\text{us} \quad (3)$$

Initial value of pre-configured frequency divider:

$$M_{clkst} = (T_{cntst}\text{-}T_{start})\times f_{mclk} = \lceil (1\text{-}0.4531)\times 32 \rceil = \lceil 17.5 \rceil = 18 \quad (4)$$

MST time window:

- Lower limit of communication cycle jitter time:

$$J_{tscyc1} = T_{cntst}\text{-}J_t = \lfloor 1\text{-}1 \rfloor = 0\text{ us} \quad (5)$$

- Upper limit of communication cycle jitter time:

$$J_{tscyc2} = T_{cntst}+J_t = \lceil 8.0 \rceil = 8\text{ us} \quad (6)$$

In which, MST time window is determined by $T_{scyc}+J_{tscyc1}$ and $T_{scyc}+J_{tscyc2}$ and T_{scyc} is the communication cycle.

Time window of receiving telegram:

- Time delay of receiving telegram:

$$T_{rdelnom} = 29.5\times T_{bit}+T_{cycdel}+2.5/f_{mclk} = 2.9218\text{ us} \quad (7)$$

- Lower limit of receiving telegram jitter time:

$$J_{trdel1} = T_{rdelnom}\text{-}J_t\text{-}T_{bit}/2\text{-}0.5/f_{mclk}\text{-}1 = \lfloor 0.875 \rfloor = 0\text{ us} \quad (8)$$

- Upper limit of receiving telegram jitter time:

$$J_{trdel2} = T_{rdelnom}+J_t+T_{bit}/2+0.5/f_{mclk}+1 = \lfloor 4.97 \rfloor = 5\text{ us} \quad (9)$$

- The maximum jitter time:

$$J_t = T_{rdelnom}\text{-}T_{bit}/2\text{-}0.5/f_{mclk}\text{-}1 = 1.875\text{ us} \quad (10)$$

Parameter configuration for master:

- f_{mclk} = 32 MHZ, B = 2, 4, 8, 16 Mbit sec^{-1}
- Six servo devices with node address 1, 2, 3, 4, 5,6 and service channels are 0, 1, 2, 3, 4, 5, respectively
- Communication cycle is 300 us in phase CP0-CP2 and 500 us in CP3-CP4
- Cyclic synchronous signal CON_CLK is "1" within 50-60 us after MST
- Start time of AT is 100 us in CP0-CP2 and that of ATn is 8+ (n-1)×40 (n = 0-6 refers to the servo device numbers) in CP3-CP4
- Start time of MDT is 600 us in CP0-CP2 and it is 450 us in CP3-CP4
- Sampling time for servo device feedback is 479 us
- Effective instruction time for servo devices is 487 us
- Terminal time of the telegram is 492 us

Master completes its initialization by loading the relevant parameters. By the way of address plus offset, the ARM accesses the two mapped storage areas with the initial address of dual port RAM: 0×0A000000 and Sercon 816 control registers: 0×0A002000. Through the driver functions of writing dual port RAM and control registers, the driver loads the initialization parameters. The driver functions are as below:

Algorithm

```
//"Write" operation for unsigned short integer variables of dual port RAM
void CDpRam::write_ram (Word adr, Word value)
{ Word *p = Null;
   p=(Word*)(_Bank4_DRomAdd+adr*2);
   *p=value;
}
//"Write" operation for control registers
void CCtrlReg::write_reg(U16 reg_num, U16 value)
{ _sm_reg *sm_reg;
   sm_reg = (_sm_reg*)_Bank4_ StartAdd;
   if (reg_number < = max_reg)
   { sm_reg->n[reg_number] = value;}
}
```

The Sercon 816 initialization process includes loading and verifying the systematic and customized parameters for the master as below:

- Set the default values of the key system parameters
- Load the key customized parameters defined by the user

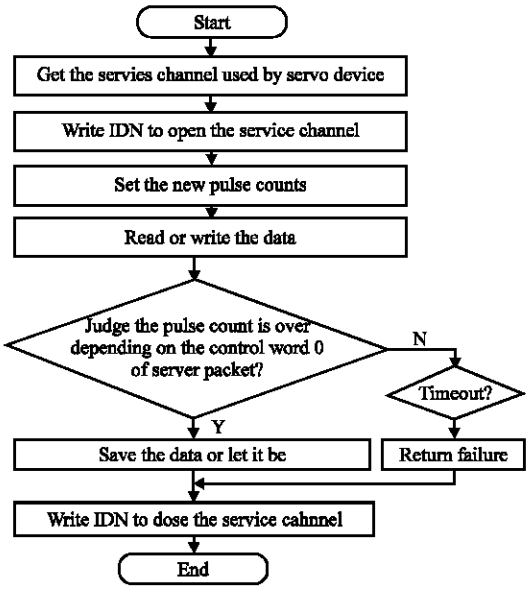

Fig. 11: Sercos II non-cyclic data transmission flow chart

- Load the configured parameters in phase CP2 and IDNs (identification numbers)
- Load the configured parameters in phase CP3 and IDNs
- Verify time slots and address parameters

Sercos II telegram processing: Mainly includes the driver for non-cyclic data transmission. According to the Sercos II protocol, the non-cyclic data transmission includes the system parameters and process instructions as shown in Fig. 11. Each parameter and process instruction is saved or sent as the IDN and each IDN corresponds to one unique data block (Cao *et al*., 2002; Danaher Motion Kollmorgen, 2003). It can configure all the essential parameters for SRCOS II and start the specific functions of the servo device by means of the processing mechanism.

Critical time slot for cyclic transmission: The system will enter into the cyclic transmission in CP3-CP4. How to define the key time slots for cyclic transmission is as follows (GB/T18473-2001, 2002; Xun and Yin, 2005; Chen and Yang, 2003):

- **Transmission starting time for AT:** The master needs the parameter t_{1min} from all servo devices in CP2. The servo device with minimum parameter t_{1min} is deemed as the first slave (m = 1), namely:

$$t_{1.1} = t_{1min.1} \quad (11)$$

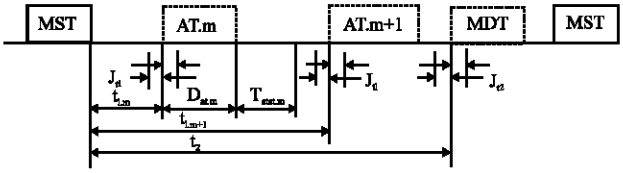

Fig. 12: SercosII data packet interval sequence sketch map

The master should be ready for receiving the AT, at the moment of $t_{1min.1}$-J_{t1} after it had sent the MST. Otherwise, one time slot greater than $t_{1.1}$ shall be defined. After the time slot used in the first servo device had been defined, the master can allocate the time slot for the next servo device (m = 2), namely:

$$t_{1.m+1} > = t_{1.m} + D_{at.m} + t_{atat.m} + 2 \times J_{t1} \quad (12)$$

The $t_{atat.m}$ refers to the minimum time interval for the slave m and $D_{at.m}$ refers to the maximum time interval for AT.m, which are shown in Fig.12.

- **Transmission starting time for MDT:** After determining the time slot for AT, the time slot for MDT can be calculated by the formula as below:

$$t_2 > = t_{1.m} + D_{at.m} + t_{atat.m} + J_{t1} + J_{t2} \quad (13)$$

$$t_2 < = t_{scyc} - D_{MDT} - D_{MST} - \max\{t_{mtsy}\} - J_{t1} - J_{tscyc} \quad (14)$$

In which, D_{MDT} refers to the maximum duration for MDT, D_{MST} refers to the maximum duration for MST and t_{mtsy} refers to the maximum time interval between MDT and MST.

- **The maximum telegram duration:** The maximum duration for MST:

$$0D_{MST} = (2 \times 8 + 4 \times 9.6) \times T_{bit} \quad (15)$$

The maximum duration D_{AT} for AT:

$$D_{AT} = (2 \times 8 + (7 + N_{bytes}) \times 9.6) \times T_{bit} \quad (16)$$

N_{bytes} refers to the number of bytes for the configurable data in AT.

The maximum duration for MDT:

$$DMDT = (2 \times 8 + 3 \times 9.6 + 4 \times 9.6 \times M + 9.6 \times \sum_{m=1}^{m=M} M_{byts}) \times Tbit \quad (17)$$

Here, M refers to the number of servo devices in loop; M_{bytes} refers to the number of bytes for configurable data correlated with single servo device in MDT.

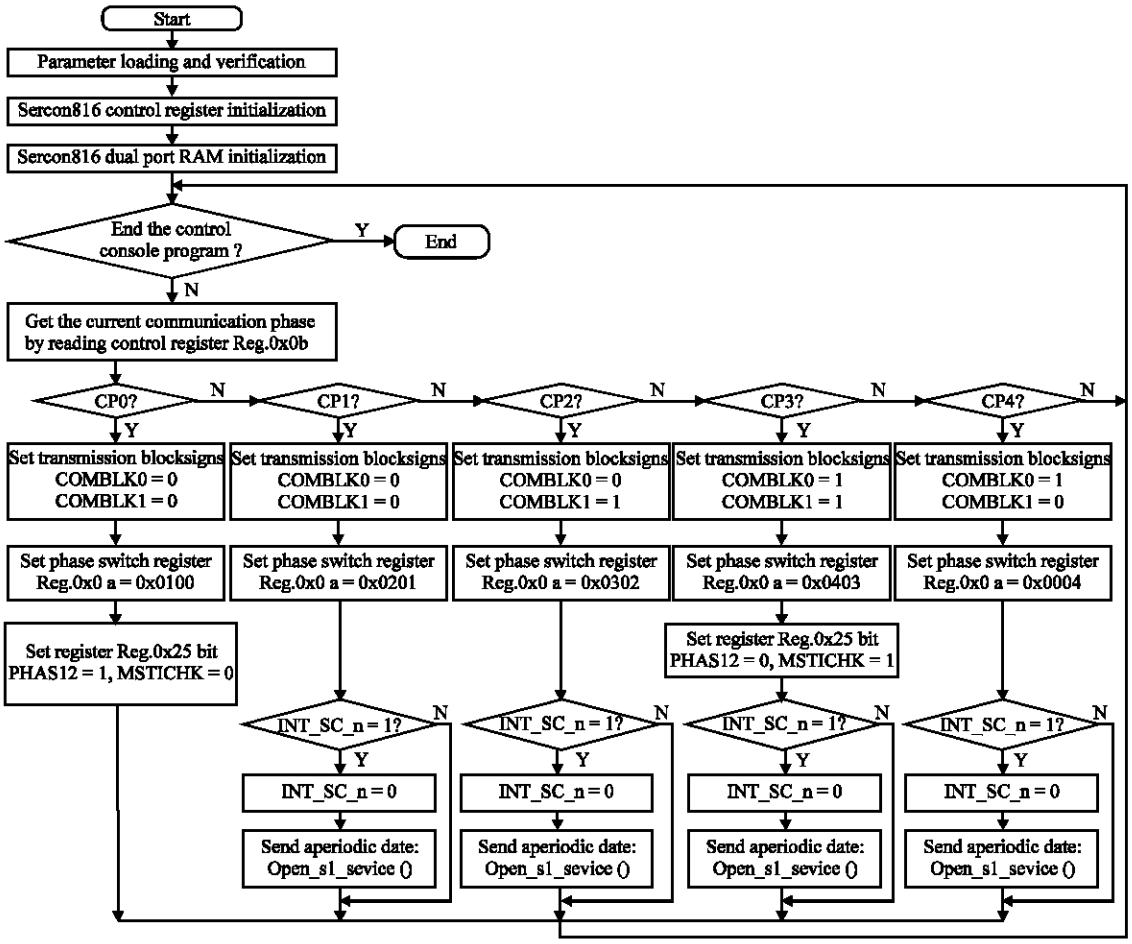

Fig. 13: Sercos II slave station initialization flow chart

Design of the slave driver: The critical tasks for the slaves are to respond and process the requests for the master and to feed back the relevant results or diagnostic information to the master. Similar with the master, the slave driver includes Sercon 816 initialization, telegram processing module and so on. During the slave initialization, it initiates the Sercon 816 first. Then, the driver will enter into a loop body to wait for processing read-write requests and instructions from master. The slaves initialization shall be implemented in ascending order as CP0-CP4 in the loop body, then enter into the normal running phase. When the communication shows errors or receives instruction for switching to CP0 from the master, the driver will restart the process from CP0. The initialization for slave is shown in Fig. 13, which can be divided into three steps:

Step 1: Read the register 0×0b: bit7-0 of the dual port RAM, obtain the phase message in the MST received and enter into the corresponding processing branches based on this message

Step 2: Set the correlated control registers in the branch (corresponding to the different communication phase)

Step 3: Read the register 0×05: bit7~0 and obtain the interrupt mark INT_SC_n of service channels (n refers to the service channel number from 0-7)

SYSTEM VERIFICATION and TEST

System verification: The verification scenario is shown in Fig. 14. It can verify whether the application program is satisfied with Sercos II standard or not. The master adopts the module developed by ourselves, which interconnects with the servo devices from Lust Company in German by optical fiber.

CP0-CP4 verification is as follows:

- **CP0 phase:** The master sends only MSTs and checks if the fiber ring is closed. The master sends 10 MSTs to all slaves in the fiber ring. It can not switch to CP1

Fig. 14: Scenario for Sercos II communication system verification

until the master receives all the 10 MSTs feedback through the loop continuously

- **CP1 phase:**
  - When the master sends MDTs to all the servo devices in turn, they should feed back the corresponding AT in next cycle
  - In CP0 and CP1, the master should send MDT at 500-700 us after MST
- **CP2 phase:** The master sends parameters for cyclic communication such as T_{scyc} (IDN00002) to all servo devices in loop through non-cyclic data transmission:
  - Allow the master to send MST with process instruction "CP3 transition check" (IDN00127), only after all servo devices give responses to this process instruction
  - Before answering the instruction and switching to CP3, the master can't send any other data to servo devices
- **CP3 phase:**
  - Send MST including CP3 phase message
  - Send MDT at t_2 (IDN00089) set in CP2 after MST
  - Before switching to CP4, the master must send the process instruction "CP4 transition check (IDN00128)". It is not allowed to switch to CP4 until all servo devices give correct responses
- **CP4 phase:**
  - Send MST including CP4 phase message.
  - Send MDT at t_2 set in CP2 after MST
  - The only difference between CP3 and CP4 is that the cyclic data is not yet valid in CP3

Table 1: IDN parameters for Sercos II master station reading

Type	Contents
IDN00003	Shortest AT transmit starting time (t_{1min})
IDN00004	Transmit/Receive trans time (t_{ATMT})
IDN00005	Min feedback processing time (t_2)
IDN00087	Transmit-Transmit recovery time (t_{ATAT})
IDN00088	Receive-Receive recovery time (t_{MTSY})
IDN00185	Max length of AT configurable Data
IDN00186	Max length of MDT configurable Data
IDN00187	List of AT configurable Data IDNs
IDN00188	List of MDT configurable Data IDNs

Table 2: IDN parameters for Sercos II master station writing

Type	Contents
IDN00001	Control unit cycle time (t_{ncyc})
IDN00002	Communication cycle time (t_{scyc})
IDN00006	AT transmission starting time (t_1)
IDN00008	Command effective time (t_3)
IDN00009	Position of Data record in MDT
IDN00010	MDT length
IDN00015	Telegram type parameter
IDN00016	Configuration list of AT cyclic Data
IDN00024	Configuration list of MDT cyclic Data
IDN00032	Primary operation mode
IDN00089	MDT transmission starting Time (t_2)
IDN00127	CP3 transition check
IDN00128	CP4 transition check

  - Phase transition in descending-order
  - When execute the descending-order transition CP1-CP4 to CP0, the master must send MST with CP0 message

Service channel verification: It is mainly verify the operation states of service channel with 2 bytes data during CP2-CP4. The master imitates different behaviors and then compares with the results obtained by the on-line servo management software. The handshake timeout is not allowed. The IDNs in Table 1 and 2 (Danaher Motion Kollmorgen, 2003) should be verified to implement comprehensive verifications.

The verification process is as below:

- Master reads and writes all the IDNs which should be verified
- There is no error in the read-write process of the master and it is the same as the value read by Lust servo parameter management software
- The master reads and writes the IDN with upper and lower limit and it is no error in this domain values When exceeds this range, the master should report errors
- Valid t_3 for instructions set by the master shall be same as all the online servo devices and meet the formulas:

$$t_3<t_{scyc} \quad (18)$$

$$t_3<t_2+D_{MDT}-J_{t2}-J_{tscyc} \quad (19)$$

Fig. 15: System machining platform based on Sercos II communication system

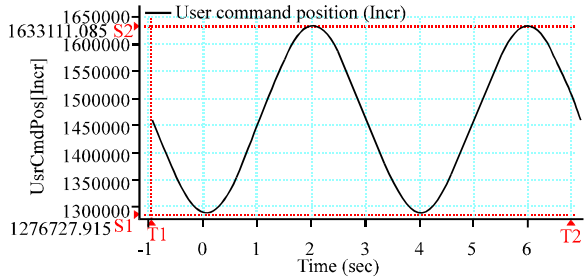

Fig. 16: Received the instruction curve of Sercos II slave staion (servo driver)

	Millimetre	Position 1-Spherome 3				
Axis	Standard value plus	Tolerance	Negative Tolerance	Mensuration	Deviatin	Overproof
D	38.0000	0.0000	-0.0200	37.9813	-0.0187	0.0000
Roundness	0.0000	0.0.150	0.0000	0.0140	0.0140	0.0000
Profile of surface	Millimetre	0.015	Only the shape	Least Square		
Property	Standard value plus	Tolerance	Negative Tolerance	Mensuration	Deviatin	Overproof
Spherome 3	0.0000	0.0150		0.0140	0.0140	0.0000

Fig. 17: Machining accuracy for sphere based on Sercos II communication system

$$t_3 > t_2 + D_{MDT} + t_{MTSGmax} + J_{t2} + J_{tscyc} \quad (20)$$

The t_4 for feedback sampled value set by the master shall be same as all the online servo devices and meet the formulas as below:

$$t_4 > 0 \quad (21)$$

$$t_4 < t_{scyc} - t_{5max} - 2 \times J_{tscyc} \quad (22)$$

System operation and test: The actual machining scene on the Sercos II communication system developed by ourselves in the 3-axis CNC SDS9-6CNCH is shown in Fig. 15. The CNC takes 32-bit ARM9 and 32-bit DSP as core to control the servo devices with Sercos II interface and the feed speed is greater than 30 m min^{-1}. After system initialization, the main operation mode (IDN00032) is set as 0×0003, namely position mode.

The instruction curve from the digital oscilloscope of the servo management software is shown in Fig. 16. It can be seen the update cycle of servo device is 0.5 m sec, which is same as the communication cycle set by master. The sinusoidal instruction curve is consistent with the preset values.

The curved surface machining accuracy of sphere work piece measured by coordinate measuring machine is shown in Fig. 17. The maximum deviation of sphere diameter and roundness are 0.0187 and 0.014 mm, respectively, which all meets the requirements for machining accuracy.

According to the tests, it is indicated that the Sercos II communication module developed by ourselves is consistent with the Sercos-II protocol. The real-time and reliability of the master consist of the communication module plus CNC and the slave consist of the servo drivers can meet the requirements of the Sercos II communication system. For performance, it can send control instructions with 4 bytes to 1-6 servo motion axles and receive the feedback parameters with 4 bytes within the control cycle of 0.5 m sec.

CONCLUSION

The modeling scheme for analysis and design of embedded system is presented based on UML visual modeling mechanism. From the developer's perspective, discuss the design and realization process of hardware and software of the high-speed digital communication

system based on Sercos II in details. The functions and performances of the system are verified and tested through the actual machining application in 3-axis CNC machine tool. The system reliably realized the multi-axle's real-time synchronous communication between the control unit and servo drivers. It can be used in the high-speed, high-precision, real-time data transmission domain such as machine tool, automatic production line, shaft less drivers, etc. It has positive significance to the popularization and application of Sercos II in domestic.

ACKNOWLEDGMENTS

This study was supported by the Guangdong Province University-Industry Cooperation Project [2011B090400577], the Guangdong Science and Technology Plan Project [2011B040300002].

REFERENCES

Cao, X.L., Y.N. Zhao, Z.H. Yang, J. Wang and P. Jia, 2002. An open industrial robot based on SERCOS. Proceedings of the 4th World Congress on Intelligent Control and Automation, Volume 2, June 10-14, 2002, Shanghai, China, pp: 1466-1470.

Chen, W.F. and J.W. Yang, 2003. Open Numerical Control System and Application Technology of Sercos Interface. China Machine Press, Beijing, China.

Chen, F. and Q.A. Chen, 2005. Embedded system development technology based on object-oriented. Microcontrollers Embedded Syst., 9: 16-18.

Danaher Motion Kollmorgen, 2003. SERCOS IDN reference manual, version 6.3.3. M-SS-011-0504, United States of America.

GB/T18473-2001, 2002. Electrical Equipment of Industrial Machines-Serial Data Link for Real-Time Communication Between Controls and Drives. China Standard Press, Beijing, China.

Gu, W.J., Y.L. Wang, E.Y. Shier and Y.X. Money, 2010. Design and application of the embedded model based on UML. J. Chongqing Coll. Electon. Eng., 19: 147-149.

Jia, L.N., 2007. Design of embedded software based on UML. Sci. Technol. Inform., 27: 535-538.

Liu, Y.Q. and J. Huan, 2006. Communication protocol for open CNC based on fieldbus. Manuf. Autom., 28: 50-53.

Ma, F.K. and J. Huan, 2008. Research of active SERCOS card based on PCI and DSP. Mod. Mach. Tool Autom. Manuf. Tech., 1: 30-32.

Wang, S.Y. and J.W. Yang, 2007. Design of Sercos bus slave interface card based on TMS320F2812. Manuf. Auto., 29: 94-96.

Xun, J. and X.F. Yin, 2005. Design and Application of Digital Servo Communication Protocol Sercos Driver. Beijing University of Aeronautics and Astronautics Press, Beijing, China.

Yu, Y., J.W. Yang and K. Cui, 2005. SERCOS bus controller Sercon 816 and its application. Int. Electron. Elem., 10: 38-41.

Yu, J., Y. Feng and J.F. Zheng, 2008. A sercos-based distributed control system. Tech. Auto. Applic., 27: 44-71.

Zhu, C.G. and S.L. Yu, 2004. Design method of embedded systems based on object-oriented technology. Microcont. Embedded Syst., 5: 9-11.