PROSTHETIC PROTOCOL DEFINITION REALIZING AN APPLICATIVE PROFILE BASED UPON I²C STANDARD

Stefano Banzi*, Elena Mainardi*, Angelo Davalli+

** Dipartimento di Ingegneria - Università degli Studi di Ferrara*
+ Centro Protesi I.N.A.I.L. di Vigorso

Abstract: The paper will present the definition of a Prosthetic Bus, specifing an application profile based on I²C standard. Aim of this definition is to realize an upper limb prosthesis control oriented bus, wich grants an efficient sensors and actuators management, with the possibility both of centralized and distributed control, trought the transmissione and reception of high level command and data.
The enlarge possibility and reconfigurability of the system will be necessary aspects on following discussion. *Copyright © 2005 IFAC*

Keywords: protocols, application, byo-medical systems, sensors, motors, motion control

1. INTRODUCTION

In order to improve complex, innovative and simple control strategies for users with different levels of amputation, it has been necessary, in an active prosthesis for an upper limb, a network of autonomous intelligence coordinated by a master unit directly interfaced with the patient. Actually, the prosthesis used at the Prosthesis Center I.N.A.I.L. (Istituto Nazionale Assicurativo per Infortuni sul Lavoro, which has many European and sometimes Worldwide contacts) are realized with four mioelectric sensors, three motorizations related to hand, elbow and wrist, one positioning sensor for the elbow motor and a central control unit which realizes the data elaboration, the control of the movement, and also the communication with the outside world through serial interface RS-232 or Bluetooth® (A.Davalli et *al*, 2000)

Notwithstanding the high efficiency of the above described system, there's much more the need to carry out distributed control strategies to help the patients doing "complex" but necessary tasks as, for instance, eating. In order to do this, it's necessary to introduce a high number of sensors (positioning, speed, for the vocal control,…) and implement architectures with distributed intelligence not manageable with the actual system. Then it's obvious, considering also the introduction of a motor for the movement of the shoulder and by consequence the increasing of the architectural complexity, the needs to realize a Prosthetic Bus projected on purpose for a system derived by the one described.
This bus must be fully compatible with the actual reality, but guarantee the possibility of improvement of several aspects as type and number of sensors (position, sliding, temperature, vocal control,…) and actuators, interface system and control modules. This bus will use the I²C standard.

Fig.1: I.N.A.I.L. Prosthesis for upper limb.

The discussione concerning the protocol is articulute in the following way:

- In section 2 will be described the causes of the chioce of I²C bus.
- In section 3 will be presented a general overview on I²C bus.
- In section 4 will be discussed the specifications on prosthesis control applicative profile.
- In section 5 will be presented the conclusions of the work and suggested some future developments

2. BUS CHOICE

The choice of the I²C standard, as substrate to use for the protocol definition, occurred after a careful analysis of the existing standards, of their potentialities and of their limits. During the analysis we consider a series of specifications deriving from the used prosthetic system and from the actual and future needs of project.

The more pressing specification are:

- Lines number. This feature has a great importance because a high number of conductors would cause an unsustainable size inside the prosthesis and increase the chance of system breaks.
- Used voltage levels. As the previous one, this is a fundamental specification, because the maximal voltage of the rechargeable batteries in the prosthesis is 7.2V.
- Standardization. The high standardization on international scale of a bus facilitates the finding of the devices, widening the
- possibilities to realize effective and efficient controls.
- The presence of integrated controllers and transceivers in the used microcontrollers or similar, is guarantee of further reduction of space and possibility of breaks.

Other specifications, as the transferred frame format, the bus access control or the ISO/OSI levels implemented, have all been considered for completeness only, but they hadn't particular importance. Also the maximal speed of transmission hasn't played an essential rule in our choice because, notwithstanding the desired control type is real-time, the timings which we refer to, are related to the human physiology of the voluntary muscular activation and then their order is about ten milliseconds.

Carried out studies (A.S.Poulton, *et al,. 2002*) have explained how an excessive structured and rigidly defined bus as LONWorks Bus, doesn't allow to realize a profile suitable to manage an efficient prosthesis control. For the same reason, some buses oriented to the industrial control (Profibus, FIP, P-NET) or to the data transfer in computer science (USB, Firewire, Ethernet) have been rejected. Good choices, represented by CAN or SPI, have been rejected due to the high complexity of the protocol towards the need of not excessive data transmission (CAN) or the necessity of additional hardware for the device addressing (SPI). An optimal found solution, which offer a great quantity of sensors and microcontrollers with integrated controller, an efficient but not rigid package format, a reduced number of lines and TTL tension levels is the I²C standard (R. Busse, *2002*).

3. I²C BUS

I²C (Inter-Integrated Circuits) is configured as a Multi/Master-Multi/Slave bus, with CSMA/CA access to the physical mean. It's formed by two bidirectional lines (SCL, Serial Clock; SDA, Serial Data) connected to a positive tension through two pull-up resistors, and to the transceivers of the devices with open-collector technology (open-drain).

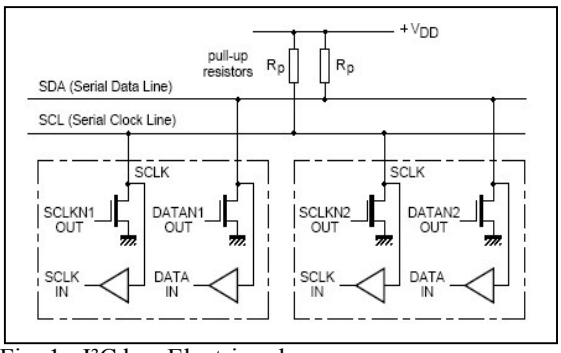

Fig. 1: I²C bus Electric scheme

The connection of the devices with the two bus lines, through an open-collector configuration, allows an hardware synchronization of the clocks of the masters. This configuration realizes, in fact, a logic AND between the clocks inserted in the SCL line. The data can be exchanged with a speed which varies from 100Kbit/s (standard-mode) to 400Kbit/s (fast mode), the number of connectable devices depends only on the capacitive limit of the bus of 400pF. Due to the different technologies connectable to the bus (CMOS, NMOS, Bipolar,…), the logic level 0 and 1 are not fixed, but they depends on the battery voltage. The communication starts with a Start character which is immediately followed by the address of the device interested by the message (in our case we'll use an addressing with 7 bits followed by 1 writing/reading bit). Every transmitted byte will be followed by a bit of acknowledge (ACK) of the receiving device. The package ends after the sending of the Stop character by the master.

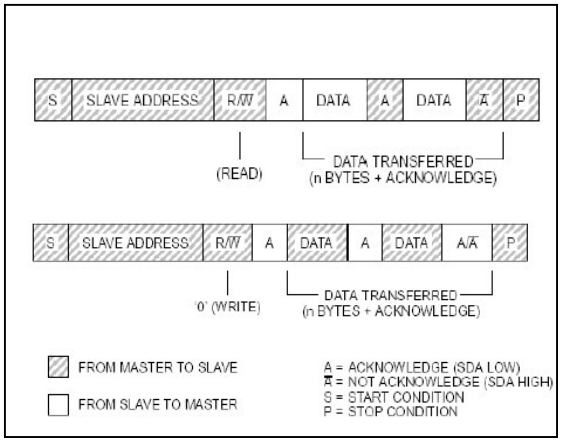

Fig.2: I²C standard frame

For the reasons explained above, I²C is very versatile implementing an open frame and not oriented towards any particular application (Philips Semiconductors, *2000*). But this generality makes necessary a greater attention in the definitions of the prosthetic profile specifications. Besides defining that series of specific commands for the actuators control and for the data communication of the sensors to the controller, it will be necessary to realize techniques of harror handling (as an algorithm for the CRC calculation) present in the LLC of the level 2 ISO/OSI standard, but not present in the considered standard (ISO/IEC 7498-1, *1994*).

4. INAIL Prosthetic Bus (IPB)

INAIL Prosthetic Bus (IPB) is, in fact, a communication protocol which amplify and complete the functionality of the I²C bus. Essentially it maintains the structure to realize a Master/Slave type architecture with access to the physical mean through the polling of the master control unit; we also define four types of packages, a series of commands for the distributed and centralized control of a prosthesis for an upper limb, and an errors and transferred data management which allows to optimize the band of the conductor and the fault-confinement. The general package for data exchange is composed of a start character immediately followed by the address (as the I²C specifications forecast), an information byte, six bytes of useful data, a CRC byte and a stop character. The following fields will be examined separately afterward.

The choice of the Master/Slave architecture with polling access has the purpose to make deterministic the timings in which every peripheral device is interrogated or commanded by the central controller. As already explained, in fact, the not urgent real time of the arificial arm controlled by a human, doesn't make necessary multi/master policy for critical actuators or sensors. Knowing the acquisition timing of the signals, which must be compatible with the sample frequencies of Shannon, gains then a great importance in order to avoid the loss of useful information for the control.

The main purpose is to make a safe communication of data and commands between a central control unit and an indeterminate number of peripheral units with the task to condition the entry signals from the sensors actually present or forecast, realize the received commands from the central unit, or make a distributed control, receiving from the central controller an angular position set-point only.

4.1 Packages used by IPB

In order to realize this, as explained before, we have defined four types of package; the element which characterizes the package is the byte immediately after the address, called *information field*. This field contains information related to the following bytes , distinguishing command packages from data transfer packages.
In particular, the first two bits of this field communicate to the addressed device if it's going to receive a simple command (for instance a reset), a parameters command (for instance a speed set-point) or some useful data, for instance, an on-line calibration.
The eighth bit of the addressing field yet indicates, where required, the necessity of the Master to obtain information from a slave (for instance the data related to a sensor or the device status); in that event it won't be necessary the sending of the information field.

The four packages are:

- Package for data or command with parameters sending (DC, Data or Command packet). These data or commands can be sent by the Master to the slave during every

cycle, and then every ten milliseconds (various tests show that, nowadays, an execution cycle of the program doesn't last more than 14 milliseconds), or one time every different program cycles or only when necessary (for instance in the event of on-line calibration of the device). In the first situation, the loss of a package won't turn out critic, because the same data will be transmitted again in the immediately following cycle. The receiving device, after elaborating the CRC, will decide to use or not the received data. In the second situation, instead, it will be a necessity of the master to obtain a confirmation from the slave, of the effective correct data reception. The protocol forecast, then, a further interrogation of the master in order to obtain a *status check* packet. In the event of missed answer, the central controller will send again the package.

- Package for the data request (RD, Request Data packet). It's a packaet which uses a writing addressing of the slave (by the eighth bit of I²C addressing byte); this device won't needs other commands to transmit *all* elaborated data to master. So, in this packet, an *information field* is not necessary. Immediatly following *data field*, slave device will transmit a CRC calculated on data sended only. Afetr that, master will verify the correctnnes of requested and obtained packet, and will re-request to peripheral device the packet, in case of critical data is corrupted.
- Command packet without parameters (NDC, No Data Command). An example for this case should be the out-of serviceing of a damaged device. In this particular case, data field is not necessary and information field is immediatly followed by CRC. A reserved bit marks to slave, in case of critical information, the need of send a *status check* to master to confirm the reception.
- The last protocol forecast packet class is *status check* (SC packet). This packet, more to confirm a correct data reception, also indicate slave device activity or inactivity to central controller.

3.2 Service commands and motion control commands

With quoted above packets, and specifically with information field, it's possible to exchance, on physical mean, all commands granting master device both to manage sensors and actuators, and supervise on network correct working, and, if necessary, shut-off or reset badworking devices.
So, with thirty commands (ten reserved by protocol and twenty programmable by user), it's possible to completly manage the prosthetic arm. Ten reserved commands, indeed sufficient for a centralized and distributed control, are subdivided into service commands and motion control commands.

Service Command.

- Acivity check: on startup this command is sent to all peripheral devices wich have to respond with a *status check* packet. If they don't, master device signals an allarm then updates his active devices table. After startup, the command can be sent to one or more devices, to request activity/inactivity state, or as a confirmation for received data.
- Bus-off setting: used to disactivate a peripherad device due to unexpected events or comunication errors. After sending this command, master module dimamically updates his active devices table.
- Bus-on setting/Reset: necessary to reset a device or set it in bus-on mode. In this case also, afeter sending the command, master module dinamically updates active devices table.
- Calibration: command useful to modify parameter in a peripheral device.
- Generic data transmission: to comunicate to slave a generic data sending. Slave, after reception, will take care of process data. This command should be usefull, for instance, to comunicate information to external world interface devices.

Motion control commands.

- Right motor rotation with a speed of x% relatively to maximum speed (where x is a command parameter ranged between 0 and 100). This command, to send in case of centralized control, needs a transmission every program cycle so that related motor to activate, always has got information on his tasks. In case of DC motor, x directly mean the duty cycle of PWM wave. Due to low speed on the requested arm movemet (order of few ten rad/s), an error on receiving this tipe of command won't cause badworking on entire system.
- Left motor rotation with a speed of x% relatively to maximum speed (where x is a command parameter ranged beetween 0 and 100). Like previous command excepted for rotation reverse.
- Right motor rotation with an ω angular speed (ω is a parameter in rad/s). This command is utilizable in case of presence of an agular speed sensor only, or if it's possible to calculate angular speed in any way. In this case also, due to ditributed motion control, master (which has to generate speed profile of controlled motor)

needs to send to slave device a value of ω every progam cycle.

- Left motor rotation with an ω angular speed (ω is a parameter in rad/s). Like previous command excepted for rotation reverse.
- $X°$ angle motor positioning (x is a parameter ranged beetween 0 and 360). This command is utilizable in case of distributed motion control. Obviously, an absolute position sensor is necessary on board. Master device can send the command every cycle (as overseen cases), to generate a trajectory, or once. If sent once, due to information importance, master needs a slave confirmation with a *status check* packet.

Through overseen command set, and through the possibility by user to extend it to realize specific applications (for instance immediate stop commands or emergency commands), or to control specific devices, it's possible to obtain an open control network, versatile and functional as regards to considered prothesic system.

3.3 Data exchanging management

Since every device can exchange variable lenght (from 1 bit to 2 bytes) and nature data (values from thermical or myoelectrical sensors, microswitch, values from position sensors,...) it's necessary, for central controller, to know the correct composition of every data packet from slaves. To do that, every Slave Unit is entirely described by a table stored both in in controller and slave memory, and called *Data Reconstruction Table* (DRT).
Data position in the packet, data lenght (in bit), the name of the system variables referred to data, and an empty field where will be recorded the values once received are recorded in this table. For instance, a controller has to receive informations from a slave acquiring data from four myoelectric sensors (1 data bytes every one), and a temperature sensor (2 data bytes). So in master there will be the following five columns (one for every data to receive from specific slave) table:

With this simple data reconstruction strategy, every

TABLE I
FIVE ELEMENTS DATA RECONSTRUCTION TABLE
TAB.N = SLAVE_N_NAME

Position 1	**Position 2**	**Position 3**	**Position 4**	**Position 5**
8 bits	*8 bits*	*8 bits*	*8 bits*	*16 bits*
Myoel_1	*Myoel_2*	*Myoel_3*	*Myoel_4*	*Thermo*
Data Entry	*Data Entry*	*Data Entry*	*Data Entry*	*Data Entry*

device is univocally specified, and it's so possible to send data packets including all process data related to a peripheral moduls. With DRT all message interpretation errors will be avoided and the band of the conducturs will be entirely utlilized.

3.4 Error handling and Fault Confinement

IPB specification overcasts three error types recognition and handling: acknowledge error, form error and CRC error.

Acknowledge bit is sent, as I²C protocol specify, every data or addres byte received. Every non-significant voltage level on SDA or SCL lines, and a consequent error in exchanging data due to a physical problem, will be notified with a not-acknowledge and the system will take care of handle the problem.

Form error is, otherwise, noticed when a violation of possible bit combinations on *information fields* occurs. If a binary combination not corresponding to any protocol or user programmed command is recognized, a form error is notified.

CRC (cyclic redundance Check) error is noticed if redundance code calculated by receiver device doesn't match with code calulated by trasmitter on the same bits sequence and sent on *CRC field*. A difference between codes prove, in fact, that at least one wrong bit is received. Laboratory tests show as the extremely short distance among network modules (order of ten centimeters) and the low comunication speed (100kbit/s), cause insignificant number of trasmission errors.
Further, since a single error on a generic packet reception isn't absolutely critical due to mecanical and physiological inertness (gear and muscles activations) better choice is a 1 byte cylic redundance code, to avoid slowing the comunication with useless overhead. In this case, standard CRC eighth degree polynomial is:

$$x^8 + x^2 + x + 1 \qquad (1)$$

corresponding to binary sequence 10000011.

If one ot these error types is noticed in the system, an error counter (EC) is increased; when EC reach a programmable value (from 1 to 255) depending by module importance, bugged device will be bus-offed and an alarm cast.

A bus-off module will answer in proper way the periodic master interrogation, with regard to slaves status, sending a *status check* packet and the central controller will take care of update active device table. This error handling and fault confinement strategies allow both to track comunication errors among devices for statistics considerations, and to manage

the network in dinamical way, reducing necessity for external intervents. Excluding dinamically incative devices make possible, in fact, to eliminate comunication dead times and motion control errors. A complete master knowledge about entire network, is necessary before startup to fulfil discussed error handling.

4. CONCLUSIONS AND FUTURE DEVELOPEMENT

Throught specification obtaind from study on I.N.A.I.L. prosthetic systems, a new protocol, motion control oriented and grantig both centralized and decentralized motors control, has been realized.
Possibility of new commands programming and exchanging more data on a single packet, take into consideration future requirements, granting a wide expandibility of existing system, but also a project "ex novo" of a new upper limb prosthesis generation including shoulder motorization.
Recognition and handling of three types comunication errors and fault confinement strategies, throught active devices table, make motion control secure in efficient way, considering prosthesic robustness and ininfluenzabiltà by few wrong packets.
Future developements overcast error probabilility studies, related to every network module, and protocol extension by introducing an entire system plug and play.
To do this it's necessary that master controller is able to auto-update a table of all devices connected to the network (actives and unactives ones). With this feature it's possible to avoid a new programming due to each new slave introduced.

REFERENCES

A. Davalli, R. Sacchetti, S. Fanin, G. Avanzolini, E. Urbano (2000). *Biofeedback for upper limb myoelectric prostheses.* Tecnology and Disability, Vol 13.

A.S. Poulton, P.J. Kyberd, D.Gow, and L. Sandsjö (2002). *Experience with the intelligent hybrid arm system. P*rooceding of MEC 2002.

Philips Semiconductors (2000) . *I²C Bus Specification.*

ISO/IEC 7498-1 (1994). Information technology – Open Systems Interconnection -- Basic Reference Model: The Basic Model