

Organic Sensor Networks for Pervasive Computing

Debashis Saha

MIS & Computer Science Group
IIM Calcutta, Joka, D. H. Road, Calcutta 700104, India
Email – ds@iimcal.ac.in
[Author for all correspondence]

Anirban Banerjee

IIT-Calcutta
Salt Lake, Block-AQ 13/1, Sector-5, Calcutta 700106, India
Email – anir_iit@yahoo.co.uk

Abstract—This research effort presents a viewpoint with regard to employing Organic Sensor Systems for effective and efficient Pervasive Computing. We deal specifically with criteria that are critical to the components used to build the underlying sensor network employing hybrid data collection and event sensing nodes, electronic as well as biological. We intend to discuss the low-level sensor architecture in concord with the IEEE 1451 family of interface specifications as well as highlight a generic BIOS, easily molded to fit standard 8-bit microprocessors which may be effectively deployed for realizing data fusion points in such hybrid sensor networks.

I. INTRODUCTION

Pervasive Computing (PerCom) [2] is a truly ubiquitous approach towards the development of an ambience wherein seamless integration of man and machine is envisioned as the final goal. To realize this objective, construction of a sensor network capable of satisfying all the implicit and explicit needs of the PerCom system we intend to conjure is but obvious. The requirements of such a PerCom system have been well defined by Weiser [1] and in [3]. The use of live biological organisms as the very basic data gatherers in such an *Organic Sensor Network* (OSN) is a new challenging idea, demanding a careful evaluation of the underpinnings needed to form a distributed measurement and control (DMC) framework. The use of bio-organisms in this novel manner unlocks a completely new perspective on the benefits to be accrued from such a system. *Context sensitiveness* and *context awareness*, the prime requirements of PerCom architecture, are built-in for free along with the basic levels of self organization and self-healing. These characteristics, woefully amiss in the electronic sensors mass produced for DMC applications tilt the scales in favor of OSNs. PerCom systems based on such live-bio-sensors based Pervasive Networks (PerNets), a.k.a. OSNs, have the upper hand on these issues, moreover the single most defining argument in favor of such PerCom systems, is *Cost*. Consider a simple argument, a rudimentary pico radio costs about \$1 in the U.S., even at this rate, to deploy a million or so nodes for observing a particular phenomenon involves a significant cost. Quite obviously to deploy a set of million or so bio-sensors in clusters are much lesser [3], [4]. Our work focuses attention towards the realization of this OSN from an architectural perspective. The resulting PerNet proves to be the basis of the final PerCom system we may choose to build. To construct the

required OSN, hybrid Link-Up Nodes (LuNs) [4] must be able to process data and control signals arriving from the completely biological or fully electronic transducers. The transducers used for this purpose must be able to communicate with each other, bringing us to the thorny issues regarding intercompatibility. Fortunately, however, the IEEE 1451 family of specifications is of solace in this regard. We will briefly define this architecture in the following section and also discuss some of the specific issues that need to be highlighted with reference to bio-sensor networks. Section III defines the actual bio-sensor architecture while we define the device level characteristics in Section IV, roundly summed up with an apposite conclusion. The Appendix contains the implementation code for a generic 8-bit microprocessor.

II. IEEE 1451 OVERVIEW

The IEEE 1451 architecture, Figure 1, consists of a family of standards for a networked smart transducer interface. This consists namely of two basic modules, i) a smart transducer information model, 1451.1 [15], targeting software based, network independent transducer application environments and ii) a standard digital interface and communication protocol, 1451.2 [5] and its prepared update [6] for accessing a transducer or a group of transducers via the microprocessor modeled by 1451.1.

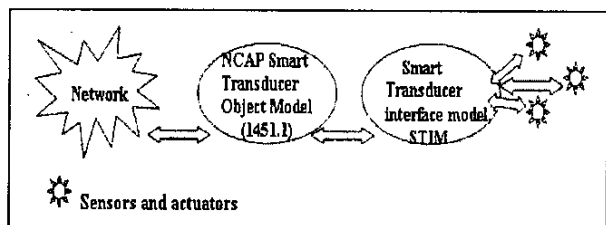


Figure 1. The IEEE 1451 architecture. The Network capable application processor (NCAP) module consists of the network hardware along with the necessary transducer interface drivers. Smart Transducer Interface Model (STIM) links up to the NCAP (1451.1) via the transducer interface specification (1451.2).

The following two standards 1451.3 and 1451.4 extend the possible single attached configurations to embedded distributed multi drop systems and to mixed mode communication

protocols for analog transducers. The 1451.1 deals with an object oriented definition of the NCAP, which is in fact an object oriented embodiment of a smart transducer device itself. This model includes the definition of all application level access to network resources and transducer hardware. The object model encompasses a set of object classes, attributes, behaviors that provide a concise description of a transducer and the networked environment to which it may link up to. This allows for a hardware independent as well as application independent architecture to be developed easily. Block and base classes are defined in the standard, to define the transducer device. The 1451.1 defines four component classes offering patterns for one physical block, one or more transducer blocks, function blocks and network blocks. Each block class may include specific base classes from the model. The base classes include parameters, actions, events and files and provide component classes. All classes in the model have an abstract or root class from which they are derived. This abstract class includes several attributes and methods which are common to all classes in the model and offer a definition facility to be used for instantiation and deletion of concrete classes. In addition methods for getting or setting attributes within each class is also provided [7].

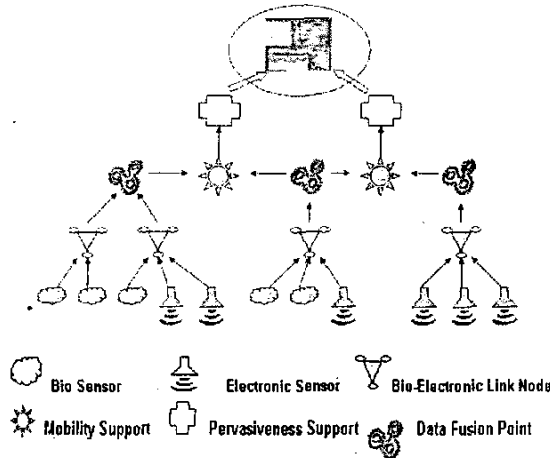


Figure 2. The topological description of a Pervasive system built on an intercompatible sensor network architecture. The bio sensors as well as the electronic sensors perform the low-level data gathering and the bio-electronic link node channels these streams towards the local data fusion points. Mobility support conjoined with pervasiveness support provides the overall PerCom "ambience" to such a system.

III. THE OSN ARCHITECTURE

The OSN architecture, Figure 2, details out the generic hierarchical organization of such a network. The lowest level of this network can comprise of completely biological or only electronic components. Interestingly the most promising application scenarios are realized by means of the hybrid-electronic-biological LuNs. The figure also projects a heterogeneous sensor cluster consisting of both electronic and biological agents communicating with the LuN. The Object oriented approach mooted by the IEEE 1451 family of

standards focuses quite significantly quite and definitely favorably on developing a modular view of the device and its networking aspects. A traditional centralized controller based DMC system development makes use of the master-slave method for systems design. In the aforementioned design, a node or device is designated as the master controller of all subordinate devices. Figure 3a, illustrates a master executing system processing by polling slave devices under its control for information. As the master controller gains information it sends out control signals to each device needing attention on a one-to-one basis. Quite clearly the controller becomes a bottleneck should information transfer to higher levels be necessary [8]. Integrating this type of architecture with higher level information models clearly becomes overloaded and convoluted. Soft Control, a relatively recent perspective on distributed design methodologies does away with hardware controllers and interfaces with software based control [9]. Soft control based systems use scalable general purpose hardware that facilitates data communication using higher level network interfaces. The software based characteristics of soft control facilitate the move to distributed architectures. This approach in comparison to the traditional use of inflexible application specific hardware and protocols is a marked improvement. Figure 3b details a distributed decentralized design for the same case as illustrated by Figure 3a. Furthermore an OSN specific modeling is described by Figure 3c. A little information when shared can go a long way, but the question that arises is, when and among whom must information be shared. This architecture underlines the tiny nuance which is critical when scalability of such a model comes into question. Maximum interaction is allowed at the lowest levels which are in conformance with most conventional views on distributed architectures however interaction between the lowest levels and the subsequent LuN levels is summarily restricted to only information summary and not hordes of raw data packets traversing the respective links.

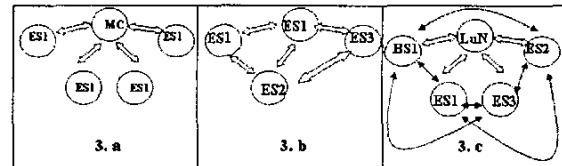


Figure 3. a) Master Controller (MC) coordinates the various activities of the subordinate electronic sensors: type 1 (ES1), this completely centralized approach is flawed as has been discussed in the text. b) All the ESs are deployed according to a more flat policy leading to a much more distributed approach to the application system. Note that standards such as the IEEE 1451 go a long way in acting as collyrium to allow ESs of different manufacturers to cooperate together, easily. c) The bio-sensor specific model clearly illustrates that the hierarchical structure must be maintained between the LuN and the ESs or the Bio-sensor (BS) while affording a singular level of completely distributed architecture at the lowest level.

One of the most clearly visible requirements of this architecture is that it will depend greatly on the link-up technology used to conjoin the various LuNs at the secondary levels. Wireless technologies such as IEEE 802.11x or Bluetooth provide a stable solution to this requirement. To build, a wired, PerCom system which would leverage this bio-

sensor network, would not only be a sheer waste of resources but an absolute thumbs down in terms of achieving the levels of scalability that sensor networks should be aiming for. One particularly interesting issue when considering wireless implementation of an IEEE 1451 enabled PerCom system is where to integrate the wireless communication interface. Two clear possibilities emerge [10]. One, the wireless communication electronics could be integrated into the STIM, see Figure 4, which consists of sensors and/or actuators, signal conditioning circuitry, and digital data output. By incorporating the wireless communications at this low level it is possible to use these sensors virtually anywhere. The other case is to have one or more STIMs hard wired to a NCAP, sensor network node. NCAP nodes allow for multiple sensors to be attached to the network using one common point of access. This allows the communication portion of the sensor to be taken out of the sensors themselves and distributed among a cluster of sensors using a specific piece of hardware: The STIMs would be networked with the NCAP, hard-wired, as in general nodes; however the NCAP node has a wireless link to the main network via a gateway. By removing the wireless hardware from the STIMs, overall cost of the sensor systems is reduced. Also current application areas could be retrofitted with such systems without major replacements.

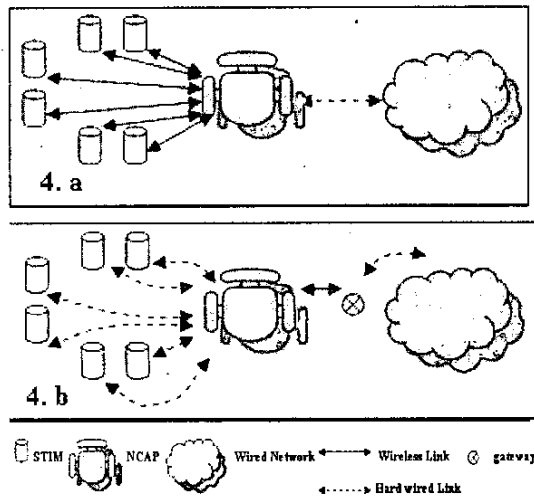


Figure 4. The two ways to place the wireless communication micro-structure. a) Wireless STIMs b) Wireless NCAPs

IV. DEVICE PERSPECTIVE

This section specifically deals with a generic application level device, the LuN which integrates the signals generated by the live bio-sensors actually deployed over the phenomenon, developing an application driven temperature measurement system and tools starting with the study of the various physical design principles of sensors finally leading to the information processing implementation including intrinsic and extrinsic data communication. The initial impediment to the following discussion is obviously the issue of selecting a strategy to allow

for a unified external communication interface. The IEC-Fieldbus to Intranet gateway architecture could have been a possible solution, regrettably the failure of the standardization initiative negated any possibility of concentrating in this domain. The IEEE 1451 architecture is the framework we turn to for development of the system we propose to describe. We therefore move on to describe a generic temperature measurement sensor architecture. Any temperature measurement application is based on temperature sensors, in our case intelligent devices equipped with microprocessors. Each of the proposed intelligent sensors contains a NCAP implementation based on the 8-bit Rabbit microprocessor [11] that are able to provide for a realization of the IEEE 1451.1 object model for intranet connectivity, and similarly access to the IEEE 1451.2 interface for linking up with similar temperature sensors. It must be kept in mind that since we are specifically dealing with bio-sensors we would need to use techniques for the detection of Indole [12] such as those being offered by Agdia (www.agdia.com), Microcyte (www.microcyte.com). Flow cytometry [13] one of the alternatives to regular chemical sniffers such as those offered by (www.professionalequipment.com) can be employed by sensors which use reflected laser beams and diffractive lenses for their purpose. These sensitive sensors often employ nitride membranes and an opto-electronic read-out system. Measured values are transformed to related thick nitride layer deflections. In our case we need to use materials which are employed in commercial sensors for the detection of Indole [4], [3]. The sensing system combines two principles that provide both high precision and ample range of temperature measurements. For laser based sensors, large displacements may be measured by the position of reflected focused laser beams. Small position changes are measured by one side layer diffractive lens principle [7]. Sensor output signal is conditioned in digital by ADuC812 one chip microcomputer, which provides the IEEE 1451.2 interface as one of its communication ports. The ADuC812 microprocessor, a basic building block of the intelligent temperature sensor electronics, includes on chip high performance multiplexers, analog-to-digital converters (ADCs), digital-to-analog (DACs), FLASH program and data storage memory, and industrial standard 8052 microcontroller core, and supports several standard serial ports. The microcontroller may also utilize nonvolatile memory containing a Transducer Electronic Data Sheet (TEDS) field and ten-wire transducer independent interface (TII) that bolsters IEEE 1451.2 [5]. A possible modular design is depicted in Figure 5. The actual Bio-sensor is referred to as a STIM. The STIM contains a Positioning and sensing device (PSD) with possibly two analog laser beam/current transducers (XDCR), if we employ laser based flow cytometry instruments at the sensor level. And ADuC812 with TEDS storing sensor specifications, circuits necessary for signal conditioning, ADCs and logic circuitry to facilitate communication between the STIM and NCAP. The NCAP or the networked host can request TEDS data and can initiate sensor readings via the TII [5]. The PerCom architecture using these nodes, LuNs and Bio-sensors have to link up to networks such as the internet for a truly ubiquitous user experience. The internal intranet architecture can be based on a JAVA based architecture while utilizing a web server as a half gateway. A JAVA applet handles the client side for the client-server communication

paradigm, and the half gateway provides for the server side capabilities allowing for the clients to connect, subscribe and communicate to the intelligent sensors. JAVA can effortlessly support client-server application architectures as the core JAVA specification, rich in APIs, contains the same for TCP/IP too. The JAVA applet utilizes the java.net package to support client server application distribution allowing the applet client to access intelligent sensors and other nodes too. All transducers can be linked up directly to an internet environment.

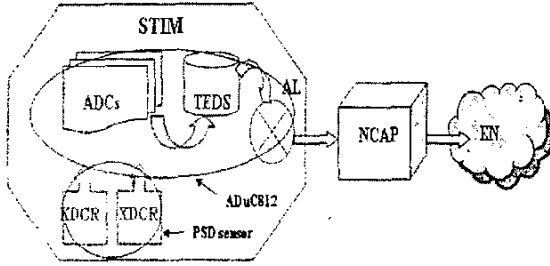


Figure 5. A possible Bio-Sensor LuN implementation, AL: Addressing Logic, EN: Extraneous Network.

We demonstrate a Dynamic C (www.rabbitmicroprocessors.com) code based implementation of a generic BIOS on the system discussed. It supports the 1451.1 and 1451.2 specifications, is modeled for a bio-sensor environment and is inspired by the universal Rabbit BIOS system specification.

V. CONCLUSION

This research effort has duly focused attention on the architectural perspective of an OSN enabled PerCom system as envisioned in [1], [2], [3] and [4]. A concise and pertinent view of an OSN in relevance to futuristic PerCom systems has been discussed. We have provided insights into the network characteristics of such a PerNet, employing LuNs and hybrid clusters of completely biological as well as electronic sensors. The Pervasive system developed on this hybrid bio-sensor network can successfully employ the biological as well as the truly electronic nodes to best serve its needs. This paper comprehensively details the network centric views concentrating on critical issues as intercompatibility between Commercial-Off-The-Shelf (COTS) components. The following BIOS code inconclusively proves the apposite feasibility of implementing the ideas discussed throughout this paper.

REFERENCES

[1] Weiser, M., "The Computer for the 21st Century," Scientific Am., Sept., 1991, pp. 94-104; reprinted in IEEE Pervasive Computing, Jan./Mar., 2002
 [2] Saha, D., and Mukherjee, A., "Pervasive Computing: A Paradigm for the 21st Century," IEEE Computer Mag., March, 2003.

[3] Saha, D., and Banerjee, A., "Live Bio-Sensor Nodes for Pervasive Computing," AAW-HPC 2003, Hyderabad, India, 2003.
 [4] Saha, D., and Banerjee, A., "Pervasive Computing with Live Bio-Sensors and Conventional Sensors," IJCS, Vol. 6, No. 2, December, 2003.
 [5] IEEE 1451.2, standard for a smart transducer interface for sensors and actuators- Transducer to microprocessor communication protocols and transducer electronic data sheet (TEDS) formats, IEEE, New York, September 1997.
 [6] Johnson, R. N., and Woods, S. P., "Proposed enhancement to the IEEE 1451.2 standard for smart transducers," Sensors, Vol. 18, No. 9, 2001, pp 74-87.
 [7] Sveda, M., and Vrba, R., "Integrated Smart Sensor Networking Framework for Sensor Based Appliances," IEEE Sensors Journal, Vol. 3, No. 5, October, 2003.
 [8] Lee, K. B., and Schneeman, R. D., "Distributed Measurement and Control based on the IEEE 1451 Smart Transducer Interface Standards," Proc. IMTC 1999, Venice, Italy, 1999.
 [9] Dieraur, P., "PLCs giving way to smart control", InTech Mag., ISA Services Inc. Research Triangle Park, NC, March 1998.
 [10] Gilsinn, J. D., and Lee, K., "Wireless interfaces for IEEE 1451 Sensor Networks," Sicon 2001, Rosemont, Illinois, USA, 2001.
 [11] www.rabbitsemiconductor.com
 [12] Heal, R. D. and Parsons, A. T. (2002), "Novel intercellular communication system in Escherichia coli that confers antibiotic resistance between physically separated populations", Journal of Applied Microbiology, 2002.
 [13] Davey, H. M., Davey, C. L., and Kell, D. B., "On the determination of the size of microbial cells using flow cytometry," Flow Cytometry in microbiology, Lloyd, D., Editor. 1993, Springer-Verlag: London, p. 49-65.
 [14] Small, T. and Haas, Z. J., "The Shared Wireless Infostation Model - A New Ad Hoc Networking Paradigm (or Where there is a Whale, there is a Way)," Proc. of the ACM MobiHoc 2003, Annapolis, Maryland, 2003.
 [15] IEEE 1451.1, standard for a smart transducer interface for sensors and actuators- network capable application processor information model, IEEE, New York, April 2000.

APPENDIX

```

Cofunc void RChanTeds(Void)
{
    Error=0;
    //Initialize Error value
    auto int A,B;
    Waitfor ((TICK_TIMER-Delay)>(Teds.EndFrameDetectLatency*1000));
    //Suspend until subsequent
    If(ConditionNACK()) //
        transmission
    {
        EndNACK=1;
        MWait=Teds.TedsHoldOffTime; AssertNIOE ();
        Cas=TICK_TIMER;
        Wfd{Error=WriteData(2);} //
        Write the data and channel
        If(!Error) wfd{Error=ReadData(4);} //
        Read data length
        If(!Error) wfd{Error=Readdata(Length);} // Get
        the data
        NegateNIOE();
        If(Error) Return; // wait
        max time when STIM must

        //negate NACK after NIOE or Error
        waitfor((ConditionNACK()==1)||((TICK_TIMER-
        Delay)>(Teds.STIMHandshakeTime*1000)));
        if((Error==0)&&(ConditionNACK()==0)){Error=11;}
        Delay=TICK_TIMER; //Set
        time for next transmit
    }
}
    
```

```

//Check checksum
if (!Error) Error=CountChecksum();
if((Error!=0)&&(Error!=12))return;
//Delay until Checksum OK
pChanTeds=(char*)&ChanTeds; //non-
permanent pointer
memset(&ChanTeds,0,sizeof(ChanTeds));
//deleting
PozBuf=Buff;
PozBuf+=2; //first
two bytes are reserved for

address of the functions
Memcpy(pChanTeds,PozBuf,sizeof(ChanTeds)); //save
data
ActualChan=Buff[1]; //save actual
ChanTeds in root memory
//swap bytes because rabbit writes long=0x0A in mem like '0X0A 00 00 00
and TIL goes first in the highest //byte
CorrectChanTeds(); //if not
calibration write is necessary
if(ChanTeds.CalKey==0)LengthChanCal[ActualChan-1]; // check
value of chantededs with teds
// write data into flash . address is adrChan + offset by num.chan and size of
chantededs
Error=WriteFlash2((AdrChan+((int)Buff[1]-1)*sizeof(ChanTeds));
If(!Error)wfd{LengthDatChannels();}
NewByte=0;
} else{Error=6;}
}
RTeds
Cofunc void RTEDS(void);
Cofunc void RTEDS(void)
{
auto int B,k;
auto unsigned int Conversionint;
auto unsigned long memory, conversion;
auto unsigned char xmemStore.FlashStore;
auto char*pConversion;
Error=0;
Waitfor ((TICK_TIMER-Delay)>(Teds.EndFrameDetectLatency*1000));
If(ConditionNACK())
{
EndNACK=1;
MWait=Teds.TedsHoldOffTime; AssertNIOE ();
Cas=TICK_TIMER;
Wfd{Error=WriteData(2);} // Write
the data
If(!Error) wfd{Error=ReadData(4);} // Read
data length
If(!Error) wfd{Error=Readdata(Length);} //Read
data
NegateNIOE();
Delay=TICK_TIMER; //Store
time for next transmit
If(Error)return;
// wait max time when STIM must negate NACK after NIOE or Error//
waitfor((ConditionNACK()==1)||((TICK_TIMER-
Delay)>(Teds.STIMHandshakeTime*1000)));
if((Error==0)&&(ConditionNACK()==0)){Error=11;}
//Checksum Error
if(!Error) Error=CountChecksum();
if(Error!=0)&&(Error!=12))return;
// read first LenBuff bytes into Buff if it is'nt in inxmem or flash
if(LengthBuf>LenBuff)
{
Memory=0;
xmemStore=Pomxmem=ActualBufxmem;
FlashStore=PomFlash=ActualBufFlash;
MoveBuffer();
Else Memory=LengthBuf; //Store read values
from Buff to Teds
pTeds=(char*)&Teds; // Store pointer for
reference
memset((&Teds,0,sizeof(Teds));
PozBuf=Buff;
PozBuf+=2; //first 2 bytes:
address of function
Memory-=2;
Memcpy(pMetaTeds.PozBuf,74);
PozBuf+=74;
pTeds+=74Memory-=74; //realign position
Conversion=Teds.Length;
Teds.Length=htonl(Conversion);
Conversionint=Teds.WorstChanDataRepet;
Teds.WorstChanDataRepet=htons(Conversionint);
pConversion=(char*)&Teds.NULLrTEDSLen); // pointer set
for(B=1;B<=12;B++)
{
memcpy(&Conversion,pConversion,4);
Conversion=htonl(Conversion);
Memcpy(pConversion,&Conversion,4);
pConversion+=4;
}
Conversionint=Teds.ChanGroupDataSubLength;
Teds.ChanGroupDataSubLength=htons(Conversionint);
If(Teds.ChanGroupDataSubLength!=0)
{
memcpy(&NumChanGroup,PozBuf,1);
Save NumChanGroupPozBuf++;memory--;
if(Memory==0) MoveBuffer(); // read subsequent
data from buffer
pGroup=(char*)&Group;//new group
for(B=0;B<NumChanGroup;B++)
{
memset(&Group,0,sizeof(Group));
*pGroup=*BuffPosf;
PozBuf++;pGroup++;Memory--;
if(Memory==0) MoveBuffer();
*pGroup=*BuffPosf; //save
NumGroupMembers
PozBuf++;pGroup++;Memory--;
if(Memory==0) MoveBuffer();
for(k=0;k<Group.NumGroupmembers;k++)
{
*pGroup=*BuffPosf;
PozBuf++;pGroup++;Memory--;
if(Memory==0) MoveBuffer();
}
//save data into Flash, address is AdrGroup + offset num. Group and size of
Group
Error=WriteFlash2(AdrGroup+(B*sizeof(Group)),&Group,sizeof(Group));
pGroup=(char*)&Group;
} //end for B<NumChanGroup
} //end if ChanGroupDataSubLength!=0
*Teds=*BuffPos;
PozBuf++;pTeds++;Memory--;
if(Memory==0) MoveBuffer();
*Teds=*BuffPos; // save
checksum
Conversionint=MetaTed.CheckSumTEDS;
MetaTed.CheckSumTEDS=htons(Conversionint); //data values
saved position returned
ActualBufxmem=xmemStore;
ActualBufFlash=FlashStore;
NextInterrupt=(unsigned int)(1/(2*Teds.MaxDataRate*TaktTimerB));
If(NextInterrupt<0X1E) NextInterrupt=0X1E;
If(NextInterrupt>0XFF) NextInterrupt=0XFF;
//Teds are nor saved, already in root memory,only each group
NewByte=0;} else {Error=6;}

```