

# Design of Mixed Higher Layer Protocol Systems.

by Kent Lennartsson  
 KVASER AB  
 email: kent@kvaser.se, WWW:  
 www.kvaser.se

## Mixing of HLPs

The HLP covered in this text is CANKingdom, DeviceNet, SDS, J1939, CAL and CANopen.

CAN modules co-existing in a system must obey three rules:

- all modules must have the same baud rate.
- all modules must use the same physical medium.
- a unique CAN-id must only be transmitted from one single module.

Physical layer  
 Bit rate  
 Drivers ISO11898.  
 Connectors

1000kBit      CK, SDS

500 kBit      CK, SDS, DN, CANopen  
 250 kBit      CK, SDS, DN  
 125 kBit      CK, SDS, DN

## Connectors

It will never be possible to have any plug and play between modules designed for different HLPs.

But:  
 Modules can have a multi-HLP design.  
 With a network management it will be possible to solve the mixing of modules supporting different HLPs.

- There will be two levels of integration of modules.
- Diferent HLPs concurrently operating on the same CAN bus.
  - Dfferent HLPs interchanging information in one or both directions.

In the table below we can see some restrictions when combining the described HLPs. In most cases there must also be some consideration to be made in the network manager for the respective HLPs combined at the CAN-bus. The upper triangle indicate the action to take when having concurrent HLPs and the lower triangle describe the possibility to exchange data between the modules.

	CANKingdom	DeviceNet	SDS	CAL	CANopen	J1939
CK	*	OK if DeviceNet MACID !=0	OK if SDS Source address !=0	CK most move away from CANid = 0	See CAL.	OK
DN	OK with PDMS-CS and known IO-connections. Depending of KP16 and KP17 support in CK module.	*	Problem with autobauding. Carefully selection of module ID	No DN group 4 allowed. MACID !=0	See CAL. Very few MAC-ID allowed	Could be possible if DN device support CAN 2.0A
SDS	OK if CK module support KP18. Can be depending of KP16 and KP17 support in CK module.	Impossible	*	Problem with autobauding.	See CAL. Carefully selection of module ID	Could be possible if SDS device support CAN 2.0A
CAL	If DBT is predictable. Depending of KP16 and KP17 support in CK module.	Maybe	Impossible	*	CANopen is based on CAL	Could be possible if CAL device support CAN 2.0A
CAN-open	Depending of KP16 and KP17 support in CK module. To support synchronized behavior, CK module may need KP5, KP11 or KP12	Maybe	Impossible	In most cases	*	See CAL
J1939	Depending of KP16 and KP17 support in CK module.	Impossible	Impossible	Impossible	Impossible	*

## CANKingdom and DeviceNet

In CANKingdom it is possible to allow only approved nodes, because the King can at any time

remove a module from using the CAN-bus. In DeviceNet it is possible to combine several DeviceNet systems on a common CAN-bus as long as all modules have unique MAC-ID. If this

is necessary there must be a special software in the King to support such functionality. There is two solutions both demand the King to listen at all possible DUP-ID check messages. At reception of such DUP-ID message there are two ways to solve the problem:

1. All CANKingdom CAN-identifiers have to be moved away from the identifier range owned by this DeviceNet module dynamically installed into the system.
2. In a normal CANKingdom system, the King is the unlimited owner of the system and either is the module known and in that case no changes are necessary to make. If it is an "alien" module, it could be kicked out of the system by sending a DUP-ID check message back to the module.

#### *Concurrently operating systems.*

The CANKingdom Kings' module must know the MAC-IDs used by DeviceNet modules. From this information, a list of forbidden identifiers can be made and those identifiers must not be used in the CANKingdom part of the system. If MAC-ID 0 is not used and if all DeviceNet MAC-IDs are known in the system, the only task is to set up the new CAN-ID data base in the King to get a working system. There must be a DeviceNet module setting up the DeviceNet modules for communication if this is not integrated into the King (i.e. the system manager.)

#### *One system with cooperating modules.*

There is no support in a CANKingdom node for UCMM as described in DeviceNet. This process must be provided by the system manager according to the DeviceNet protocol. It is possible to integrated software in the King to set up all DeviceNet nodes and this is very simple if it is PDMS-CS modules. After the DeviceNet modules are set up for communication those modules will be prepared to transmit and receive information with certain CAN-identifiers with IO-connections. A CANKingdom module can be configured to transmit or receive any CAN-identifier, so this is not a problem when exchanging information between DeviceNet and CANKingdom modules. To exchange information, a transmitting module and its receivers must share a common data structure. The solution to this in CANKingdom is the possibility to have a number of data structures with the very same purpose and by King's Page 16 select the structure that fits to the DeviceNet data. DeviceNet specification has 500 pages describing different standard data types and a lot of them will be supported by CANKingdom nodes. Even if the CANKingdom node does not have predefined data structure that fits to the DeviceNet module it is possible to set up new data structures. This is

made by King's Page 17 commands that makes a linking of fundamental bits, bytes, words and data structures to different locations in a CAN message. This linking can be made both for received and transmitted CAN-messages.

## **CANKingdom and SDS**

SDS and CANKingdom have the same philosophy with a system master and a number of slave modules. The difference is that in CANKingdom it is possible to have the King only during startup and configuration and during the running phase it is possible to have all modules running autonomously without the master. Some CANKingdom modules even support auto start with default setting in case the King is lost. The major problem is the autobauding that must be supported by the master. This can be solved by keeping the King in silent mode until the SDS system is up and running. A standard CANKingdom node will not communicate until told so by the King. This indicates that auto starting of CANKingdom modules is not possible together with SDS modules.

#### *Concurrently operating systems.*

If device address 0 is not used, it will only be necessary to ensure that CANKingdom modules do not use CAN-ids reserved and used by SDS modules. It may be necessary to have a system management to protect the SDS autobauding. There must also be a master of the SDS system because otherwise the SDS devices are of little use. The King is also necessary to start the CANKingdom modules. All necessary system software can easily be integrated in the King module.

#### *One system with cooperating modules.*

This is possible since it will always be possible to link CAN-id and data elements to and from a SDS device to a CANKingdom node. It is even possible to place data elements in the CAN-id in a CANKingdom module, as used in SDS, if the CANKingdom module support KP18. This is a very advance feature in a CANKingdom module and it is not available in low cost modules, unless the designer has made them to be configured for working together with SDS nodes.

## **CANKingdom and CAL**

Both CANKingdom and CAL use dynamic distribution of CAN-identifiers. The distribution in CANKingdom is initiated and performed by the King alone. In CAL, this process is initiated by

the network manager, but it is up to each module to perform the process under some overall control by the network manager. It must be possible to put some restrictions on the CAL network manager to select a limited range of CAN-id during this process. The CAN-ids outside this range can be used by the King in the CANKingdom part of the system.

#### *Concurrently operating systems.*

In CANKingdom, the CAN-id 0 is recommended for King pages. This is in conflict with CAL which uses this CAN-id for start and stop functions. The simple solution is to move King pages to another CAN-id but that could cause problems because all new modules will assume KP with CAN-id=0 and they must be reconfigured before they are installed into the system.

The best solution is to power up the CANKingdom modules first, then replace the default CAN-id for KP to open up for CAL system to use.

The database of CAN-ids for CANKingdom and CAL must be configured to ascertain that CAL and CANKingdom do not use the same CAN-ids.

#### *One system with cooperating modules.*

Both CANKingdom and CAL modules can produce and consume information from the CAN-bus without bothering about the location or the type of module at the other end of the communication. The main problem is to find modules with corresponding data structures. With CANKingdom modules supporting dynamic data linking it will always be possible to place and remove any data to and from a CAN message and this will increase the possibility. Probably will some CANKingdom modules support the data structures as described in CAL device specification.

## **CANKingdom and CANopen**

#### *Concurrently operating systems.*

CANopen is very much a combination of CAL and PDMS-CS and the consideration for CANKingdom will be the same as CAL with the addition of a number of CAN-id reserved by the assignment of module number to predefined CANopen modules.

#### *One system with cooperating modules.*

It will be more complicated if synchronous messages are used in the system. A CANKingdom node will probably not directly support such behavior. CANKingdom have a feature to link event to certain CAN-messages events with something called action-reaction. With this

function it may be possible to get a module to behave in a fashion that fits to CANopen.

## **CANKingdom and J1939**

#### *Concurrently operating systems.*

Because J1939 uses 29-bit arbitration fields and the module id is 8 bits, just one single module id owned by the CANKingdom part of the system will gain control of 2097152 different priorities and that should be enough for most CANKingdom systems. CANKingdom will normally use CAN-id 0 and this CAN-id is not used in J1939.

#### *One system with cooperating modules.*

The main problem when transferring data to and from J1939 is the fact that the data often include some status information. A byte could have a value +/-125. Absolute value 255 means no data available, absolute value 254 is an error indicator and absolute value 251-253 are reserved for future indications. Such definitions is made for 1, 2 and 4 byte words and ASCII data. It can be generally solved with KP16 in combination with KP17. It will be very easy if the module is adjusted for J1939 type of data in that case it can be made with KP16.

## **DeviceNet and SDS**

#### *Concurrently operating systems.*

The major problem when combining SDS nodes with DeviceNet is the autobauding used in SDS nodes. This autobauding can cause the DeviceNet modules to go off-bus. The opposite is also true if the DeviceNet modules starts to communicate during the autobauding it could cause problems and in worst case disable the SDS from bus activity.

The system designer must ensure that the modules do not have node numbers that cause conflicts in the use of the CAN-identifiers. We have not found any general algorithm to get a certain set of node numbers that can be used. We found a limited rule that should solve the problem for most systems. If the SDS nodes are limited to odd node numbers in the range {1..63} it will be possible to use following range of node numbers for the DeviceNet nodes {0,2,4,6,16,18,20,22,32,34,36,38,48,50,52,54}. If the DeviceNet modules are limited to PDMS-CS without supporting bit-strobing it will only be a question of selecting a unique node number in the range {0..63} for all modules DeviceNet and SDS.

#### *One system with cooperating modules.*

It is not possible to directly exchange data between DeviceNet and SDS modules. The main reason for this is the fact that SDS have placed information in the CAN-id and that is not possible in DeviceNet. There might be exceptions to this, but they are probably very rare.

## **DeviceNet and CAL**

It is possible to select CAN-identifiers in CAL and use them at locations not used by DeviceNet modules. Some CAN-IDs identifiers are predefined in CAL and there must not be any DeviceNet nodes using these CAN-IDs. All DeviceNet group 4 messages are reserved by CAL for network management. DeviceNet also use the group 4 messages for network management and if this process is placed in one module common for CAL and DeviceNet it may be possible to solve this problem. CAL has also reserved CAN-id 0 which restricts the usage of MAC-ID 0 at DeviceNet modules.

### *Concurrently operating systems.*

In a normal system there will be a fixed number of modules and they are known in advance. If this is known, it will be possible to select appropriate MAC-ID number for all DeviceNet modules. Then it is only a question of setting up the CAL database with CAN-id not interfering with the CAN-ids used by the DeviceNet part of the system.

### *One system with cooperating modules.*

Both DeviceNet and CAL have described data structures to be used at certain type of devices. If the two specifications are compatible in some data structures, it will be possible to link IO-channels between two such modules. Due to the different specification for bidirectional communication this will not be possible.

## **DeviceNet and CANopen**

This is very complicated because it is like mixing DeviceNet, CAL and SDS at the same time. There are only a few module ids that could be used and the algorithm to pick the right ones is very complicated.

## **J1939 with DeviceNet, SDS, CAL or CANopen**

This is not possible because J1939 is based on extended CAN. DeviceNet, SDS, CAL and

CANopen are specified for standard CAN. It could be possible to have J1939 concurrent to the other protocols if the other protocol devices have a CAN-controller compliant to CAN spec. 2.0A, but such information is not generally available for standard devices, because the information is not necessary for the device to fulfill the specification.

## **SDS and CAL**

The main problem is to ascertain that the SDS nodes are properly performing the autobauding. It is possible to select CAN identifiers in CAL and use them at locations not used by SDS modules. Some CAN-IDs identifiers are predefined in CAL and there must not be any SDS nodes placed on top of those CAN-IDs. This can be solved by having SDS module ids in the range {1..63}.

### *Concurrently operating systems.*

In a normal system there will be a fixed number of modules and they are known in advance. If this is known, it will be possible to select appropriate module id for all SDS modules. Then it is only a question of setting up the CAL database with CAN-id not interfering with the CAN-ids used by the SDS part of the system.

### *One system with cooperating modules.*

This is not possible because SDS have data in the CAN-id and that is not described or solved in CAL.

## **SDS and CANopen**

Most part here is the same as for CAL, but due to the PDMS-CS type of module there must be some extended restrictions for used module-ids. If the SDS modules are restricted to the range {1..15} it will be possible to place CANopen nodes in the range {0..7}, but other combinations are possible.

## **CAL and CANopen**

### *Concurrently operating systems.*

CANopen is based on CAL, so this should be no problem.

### *One system with cooperating modules.*

There is a great chance that interchange of data is possible. There is some behavior specified in CANopen which is normally not included in the CAL module.

## Philosophy in CAN HLP

We have now listed a number of rules to be used when combining some HLP. To understand the problem and how to get around the problem you have to understand the fundamental parts in the different HLP's. This also is a background to understand why there is a number of different HLPs for CAN.

*CANKingdom* was design to give means to the system designer to manipulate the modules to behave according to his will without removing any of the possibilities provided by CAN.

*DeviceNet* was design as a CAN link of distributed IO units with larger number IO's to a central PLC. DeviceNet also includes more advanced network features to get the same functionality as you would have on a computer network, but most modules only support the limited PDMS-CS.

*SDS* was intended for simple devices where each device had only one IO, for example a solenoid, a proximity switch, light gate, etc.

*CAL* was designed to make it possible to get a communication between complex independent units via a CAN-bus in the same way it is possible to connect devices to a office network. The first implementation was in an X-ray machine.

*CANopen* is based on CAL, but with some restrictions to make it possible to have simpler and low-cost devices like the PDMS-CS devices for DeviceNet.

*J1939* was design for modules to be used in a truck and trailer and further on were also other vehicles included like buses and agriculture units using the same kind of basic equipment.

*Propriety* is very common and if you have a small fixed system produced in high volume it is not necessary to have a protocol except the function in CAN itself.

The main problem for HLP in CAN is to secure that CAN-id are distributed to all modules without violating the CAN specification. To have any use of the CAN bus the HLP must also include tools to link producers of CAN-messages to consumers of CAN-messages. This is the minimum support, to have any use of the data in the CAN-message there must be a common definition of the data between producer and consumer of the data. This is also included in most HLP or specifications related to the HLP.

At least during configuration it is necessary to have a bidirectional communication link between the Device and some kind of configuration tool and most HLP also use such communication link during normal operation. Because CAN only support messages with up to 8 byte of data, there is normally a description in HLP how to send larger package of data.

### *CANKingdom*

This HLP is very much directed to configuration of modules to make them suitable to the system demands. The modules will need no knowledge about the system or other modules. The main feature is that all modules can dynamically be assigned any CAN identifier.

There is no standard data structures or devices in CANKingdom, but there is means to select one data structure out of many and link that to a CAN-id for transmission or reception on the CAN-bus. By this it is possible to make low-cost module supporting only a few device type like defined in DeviceNet, SDS, J1939, CAL or proprietary specification. It is also possible to make a more expensive module supporting a number of specifications to make it compatible to DeviceNet, SDS, J1939, CAL and number of proprietary specifications. It is even possible to dynamically define data-structures to be used in a CAN-message from a set of data-primitives. The selected solution will depend of the market and customer to be supported.

It is also possible to adjust the real-time performance by setting timing restriction of the transmission and to synchronize transmission to a reception of messages. It is also possible to place CK-modules in groups to make it possible to start/stop a group and even change the used CAN-id (priority) in the transmitter and all receivers with one single command.

Any baud rate is allowed as supported by the CAN-controllers and to make it possible to get hold of a module with unknown baud rate will all modules be set to 125kbit during 200 ms after power on scanning for a specific CAN-message (CAN-id = 2031 and all 8 byte of data filled with 0xAA).

The CAN hardware supported remote request message is of course supported.

### *DeviceNet*

The philosophy in DeviceNet is that a node will get a resource of 31 identifiers to be used by the module. Every node is the master of those 31

identifiers and no other module can force the module to use them in a specific way, but it is possible to make a kindly request. The key to the unique 31 CAN-identifiers is the module-id, called the MAC-ID in the range {0..63}.

Any module in the system can get in contact with another module and establish a bidirectional link to this node. A number of such bidirectional links can be established as long as there is free identifiers left from the pool of 31. By this definition is DeviceNet a true multi-master system because all modules are equal, within the range of 31 CAN identifiers to be used.

It is also possible to have IO connections which is standard CAN messages with one transmitter of information and one or many receivers. Those general connections are made in a resource demanding way which makes it necessary to have a lot of RAM, ROM and CPU power. To solve this problem have a predefined setting been specified called the "The predefined master slave connection set" PDMS-CS. This make it possible to have the module connected as a distributed IO with a predefined bidirectional communication to set and get data from the module. Such a module will only be connected to one other module the master.

A number of devices have been described in the specification to make it possible to have the same data structures common for a number of different suppliers of nodes. This part is 50% of the specification or almost 500 pages. The description is divided in different functions like Discrete IO, AC-drivers, bar code scanner, encoder etc. By definition is the producer of information the "client" the receiver(s) are the server(s). If you have temperature sensor producing temperature values trough an IO connection to a PLC controlling a system, then is the sensor the client and the PLC the server. The CAN hardware supported remote request message is not allowed because you do not have that in real data communication.

### *SDS*

All SDS nodes is assigned a number in the range {0..125} called the device address and this number must be unique for all modules in the system. This node number is used as a base for selecting a set of CAN identifiers that can be used by this module. Each modules will get access to 16 different CAN-ids for communication to and from the master. The master will normally scan all expected modules to get and set data to the module, but it is possible to program most

modules to produce information at IO event or timer event.

A SDS node will always make autobauding and the SDS master must be very careful when starting communication after power failure to make it possible for the modules to perform the autobauding successful.

The CAN hardware supported remote request message is not allowed because you do not have that in real data communication.

### *CAL*

The CAN identifiers 0 and 2016-2031 are reserved for network management and also 1761-2015 is reserved for special management functions. All other CAN-id are free to use and they are grouped into 8 groups with 220 CAN-ids to get 8 different priorities. All CAN-ids to be used in the system will be dynamically assigned to each module by the network manager.

During start up the network manager will try to get in contact with all possible modules in the system. When the network manager gets contact a bidirectional link will be established. Through this link the module will send a 14 byte ASCII string identifying a data type to be sent on the CAN-bus. The Network manager will look up this variable in a data base and assign a CAN-id to the data type and send back the CAN-id to be used. When all assignments are made to the module the network manager will move to the next module. If another module is asking for the same data type it will first be checked that there are no two transmitters of the same CAN-id, otherwise the second module will get the already assigned CAN-id and the two modules are then linked together.

This process is using CAN-id 2023 and 2024 for all modules. To use the same CAN-id in more than one module violates the CAN-specification, but by having the assignment controlled by one single network manager and with some time-out protection it is possible to remove the use of the CAN-id from one module before it is used within the next module. This is also the reason why the communication have to start from the network manager. There is a number of device standards describing data structures and behavior of such device stored in a common global data base.

### *CANopen*

This standard is based on CAL. CAL have in every module a complicated network management because the dynamically CAN-id assignment is launched from the module with the network

manager as a data base for the system. In addition to the CAN-id reserved in CAL CANopen has also reserved CAN-id 1-7 and 1740-1760. To remove the expensive dynamically assignment have something similar to PDMS-CS described in DeviceNet been included. As in DeviceNet and SDS is the Node Number used as the base for the selection of a number of identifiers that is own by the module.

### *J1939*

This HLP is designed for buses and trucks and very much of the specification is about functions and module behaviors useful in a truck. All 29-bit a reserved for specific use:

8-bit SA, is Source address

8-bit PS, is PDU specific, DA or GE

8-bit PF, PDU format.

1-bit DP, Data page

1-bit reserved

3-bit for 8 level priority.

If you need to read and write specific PDUs it is possible by an index in the first 3 bytes of the data bytes. Normally is the source address defined and fixed, as well as the baud rate. It is possible to change the source address via the CAN-bus and there is also included a address claim process; if the source address claim fail it is possible to claim a new address.

## Summary

There is a number of different HLP and it will be so forever and new propriety protocols are made every day all over the world. Do not view this as a big problem, change it into a opportunity to get low cost modules with a mix of functionality. In most cases it will be possible to integrate them into the system as long as you make a good system design.

## References

- GS052103 Application Layer Protocol Specification November 22, 1996 and related documents.
- DeviceNet specification Version 2.0, February 1997.
- J1939/21 Data Link Layer 1994-07.
- CAN application Layer draft standard 201... 207 Version 1.1 February 1996.
- CANopen draft standard 301 revision 3.0, 30/10 1996.

- CANKingdom specification Version 3.01, 1/1 1996.

## Abbreviations

PDMS-CS	Predefined Master-Slave set. A number of predefined IO connections in a small device to make it possible to make a simple device without the UCMM.
HLP	Higher Layer Protocol
CAN	Controller Area Network in general, in this case the specific protocol CAN defined by Bosch.
UCMM	Unconnected Message Manager. This is a predefined communication channel to make it possible to set up new communication channels.
KP1, KPxx	Abbreviation of Kings Page ## a set of defined data structure sent by the King/Master to set up parameters and variable in the module to make the suitable to the communication on the CAN-bus.
CAN-id	The 11-bits used for arbitration. There is other names used in different HLP. CAL COB-id; CANKingdom Envelopes; SDS, DeviceNet and J1939 have names to different parts of this CAN-id.
MAC-ID	The module identification number in a DeviceNet system.
SDS	The Honeywell HLP Smart Distributed System.
CAL	The CAN HLP from Philips medical systems CAN Application Layer.
J1939	A SAE standard for CAN in buses and trucks.
DA	Destination address as used in J1939 CAN-id to get point to point communication.
GE	Group Extension as used in J1939 CAN-id to address a group and not a device.
COB	A CAL abbreviation of a complete CAN-message.