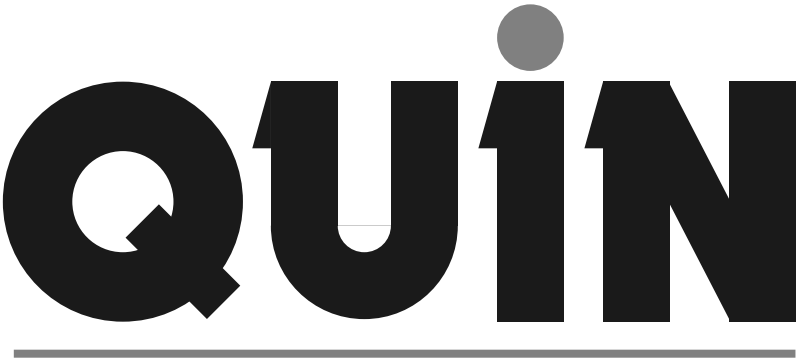


SYNCHROLINK

MAP LINKING BETWEEN PTS UNITS

MAN528

Issue 1

September 1996

Copyright Notice

Software Version

This manual reflects the following firmware/software versions:

- PTS Firmware version 1.8.1 or later

Important Notice

Quin Systems reserves the right to make changes without notice in the products described in this document in order to improve design or performance and for further product development. Examples given are for illustration only, and no responsibility is assumed for their suitability in particular applications.

Although every attempt has been made to ensure the accuracy of the information in this document, Quin Systems assumes no liability for inadvertent errors.

Suggestions for improvements in either the products or the documentation are welcome and should be addressed to:-

Quin Systems Limited
Oaklands Business Centre
Oaklands Park
Wokingham
Bershire
RG41 2FD

Telephone: 0118 977 1077
Facsimilie: 0118 977 6728
E-Mail: Support@Quin.co.uk
CompuServe: 100572, 1631

Contents

1. Introduction 5

2. Features of SynchroLink 6

3. Wiring Requirements for SynchroLink 7

4. Additional/Modified PTS Instructions for SynchroLink 8

4.1 CK<val> Configure ClocK....8
4.2 CN<node> Configure Node....8
4.3 CQ CANbus status Query....9
4.4 LK<val> master map LinK over CANbus....9
4.5 ML<node:channel> Map Link....10
4.6 XN<n> eXecute sequence on all other Nodes....10

5. Error Messages and Codes 11

1. Introduction

SynchroLink is the ability for PTS systems to perform map linking between separate PTS units (nodes). This is an advance on the use of a link encoder to daisy chain PTS units together. A high speed communications bus, CANbus, has been used for SynchroLink, providing real time map linking for multiple master/slave combinations. Currently SynchroLink is supported on the MiniPTS3 and the CPU360.

SynchroLink is a subset of ServoNet, a distributed Master/Slave configuration of PTS units utilising CANbus as the communication medium.

2. Features of SynchroLink

SynchroLink supports the following features:

- Up to 60 nodes on one SynchroLink network.
- Network length maximum 100m.
- Up to 8 map masters transmitting simultaneously across CANbus.
- Many map slaves can receive one map master signal (see next point).
- Each node can support up to 8 unique map masters/slaves (more than one channel can receive the same map slave information within a node, whilst only requiring one unique map slave reception from CANbus).
- Global sequence execute for use during motor error sequences or similar.
- Requires only a handful of extra PTS instructions to configure and use SynchroLink.
- Node number clash, clock signal configuration and network error detection provided.

3. Wiring Requirements for SynchroLink

SynchroLink uses the CANbus network and should be wired as follows:

Each PTS node has a male 9 pin sub-D type connector for CANbus. The cable should therefore have female 9 pin sub-D type connectors. The CANbus cable joins each node in turn (in the same manner as ethernet), with no cable spurs. Twisted pair cable is recommended. A 12V DC power supply is required to provide bus power (all PTS nodes are opto-isolated from the bus). Each end of the bus should be terminated with a 124Ω resistor placed between CAN H and CAN L.

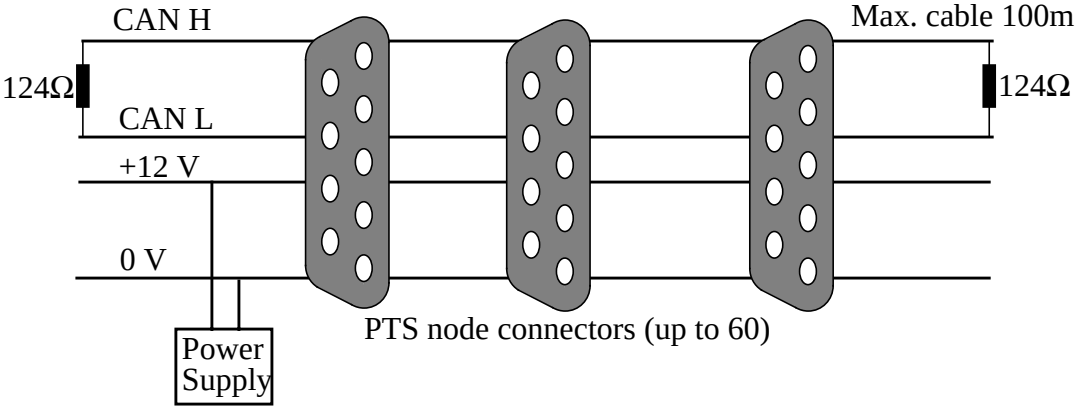
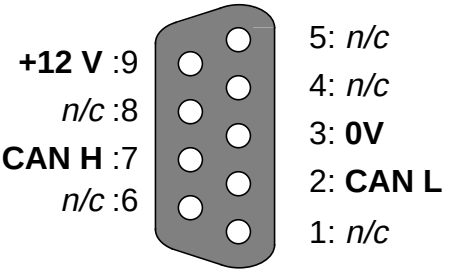


FIG.1: CANBUS WIRING

Each female 9 pin sub-D type connector should be wired as below:



n/c: not connected

FIG.2: CONNECTOR PINOUT

4. Additional/Modified PTS Instructions for SynchroLink

4.1 CK<val> Configure ClocK

Value 0 (default) or 1

SynchroLink requires clock synchronisation between nodes for position mapping to work. Therefore one node must be a clock *master* (CK1) and all other nodes must be clock *slaves* (CK0). The clock signal uses up approximately 3.5% of the available CANbus bandwidth.

When bus off (CN0) changing CK has no effect. The value of CK is applied when CN is set to a non-zero number and the node attempts to go CANbus on. If another clock master is detected on the bus then an error message (error 130) will be displayed and CK will revert to zero.

When bus on (CN non-zero) changing CK will have immediate effect. CK1 will attempt to make the node a clock master. If no other node is clock master then this will be successful, otherwise an error message (error 130) will be printed. CK0 will turn the node into a clock slave.

$ABC=CK will place either a zero or one in variable $ABC, to indicate clock slave and clock master respectively.

NOTE: there must always be a clock master for SynchroLink to work. Clock slave nodes check for the presence of a clock master and will not perform mapping across CANbus unless the clock signal is present (error 131 followed by map update timeout errors (error 98) will be reported if clock signal is not available). Therefore always configure one node to be the SynchroLink clock master. Use the CQ function to check for the presence of a clock signal.

4.2 CN<node> Configure Node

Value 0 (default) to 60 inclusive

To assign a node number and to go on bus for SynchroLink use CN<n> where <n> is the node number to be used. Setting CN0 switches SynchroLink off for this node. Setting CN<n> will start SynchroLink. If another node already uses <n> as its node number an error will be reported (error 132) and the node will go bus off (CN will also revert to zero). $ABC=CN will place the current node number (0 to 60) in the variable $ABC.

If at any time the node goes bus off (loss of CANbus power or loss of CANbus cable integrity) each affected node will report an error (error 126) and try to go back on bus immediately. If that fails the node will continue to try and regain the bus, waiting 1 minute between each attempt. No error messages will be displayed for success or failure of these attempts. The CQ function can be used to query the status of a node at any time.

There is no requirement for nodes to be numbered consecutively, or to start at node number 1; a node numbering scheme that suits the application should be implemented.

4.3 CQ CANbus status Query

Value none or binary code flags

To query the SynchroLink status of the node use CQ. CQ<return> displays the current status of the node. This information includes the node number being used, whether the node is clock master, the status of the CANbus interface and which node is clock master if the current node isn't.

The CQ function can display extra diagnostic information to aid CANbus configuration:

Bit	Meaning
0	Display this nodes bus usage
1	Find all nodes on CANbus
2	Display global bus load
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved

Bit 0 displays the number of unique map links being transmitted and received by this node. A maximum of eight transmissions/receptions is available.

Bit 1 counts all the nodes responding on CANbus. For a node to respond it must have been configured with CN non-zero, have successfully initialised its CANbus interface and be bus on (receiving power to the transceivers, and the CANbus cable be sound).

Bit 2 gives an indication of the percentage bus load on CANbus due to map links. Each master map link accross CANbus occupies approximately 5% of the available bandwidth, with the clock signal occupying a futher 3.5%. The figure presented is calculated by querying each node for the number of map transmissions.

4.4 LK<val> master map LinK over CANbus

Value 0 (default) or 1

For map linking across CANbus a map master is required. This is created by setting LK1 on the desired channel. Each master map link over CANbus uses up approximately 5% of the available bandwidth [CANbus map slaves do not use bus bandwidth as they receive messages only]. Testing has found that up to 8 map masters will be supported by CANbus (equivalent to 40% bus loading). The number of map slaves is limited by the CANbus interface on each node (maximum 8 unique CANbus map masters/slaves per node).

$ABC=LK will place zero or one in variable $ABC indicating map master off and map master on respectively.

4.5 ML<node:channel> Map Link

Value: node=1 to 60 inclusive, channel=1 to 48 inclusive

The ML command has been extended for SynchroLink. Using a node:channel format it is possible to specify that the required map master is on another node. [The standard form of ML<channel> still functions and performs a map link within the node] The existence of this master is not checked, however map timeout errors will occur if the master is not transmitting data. Any one node will support up to 8 CANbus map masters/slaves.

In addition to ML, two other functions must be called to complete the map link on the map slave axis, above the normal requirements:

Master axis, channel 5 on node 3:

CK1/CN3/CH5/SB20000/AR256/LK1/ZC/PC

Slave axis, channel 8 on node 5:

CK0/CN5/CH8/MP20000/MM256/SB1000/ML3:5/PC/XM56

In the above example channel 5 on node 3 is to be the map master for channel 8 on node 5. First node 3 is configured (here it is also the clock master). The master axis is then initialised by setting the bounds (SB) and the analogue range (AR, optional) before exporting the map data on CANbus using LK1. Secondly node 5 is configured. Channel 8 requires knowledge of the master bound (MP, using the SB value entered on channel 5, node 3), the master analogue range (MM, using the AR value entered on channel 5, node 3 [optional]) and its own bound (SB). The map link is established by ML3:5, and then the axis put into mapping using map table 56.

$ABC=ML will place a value of 256*node + channel in variable $ABC. In this manner it is possible to query and program ML using variables. For example to set ML to node 3 channel 5 the following are valid: ML3:5, ML773, $ABC=773/ML=$ABC.

4.6 XN<n> eXecute sequence on all other Nodes

Value 1 to 255 inclusive

This function provides the ability for one node to request the execution of the said sequence number on all other nodes. This is equivalent to typing XS<n> on terminals connected to the serial ports of all other nodes and pressing all the enter keys at the same time. XN<n> does NOT perform an XS<n> on the source node. XN is a global broadcast with no confirmation from receiving nodes, and therefore no guarantee of reception (and execution of sequence <n> on remote node) can be given.

If sequence <n> does not exist on a receiving node no error is generated, the request is ignored. XN<n> will generate errors if two or more XN's are received or transmitted before the firmware has finished operating the first XN function.

XN<n> is primarily intended for use in motor error sequences. If one channel on a node experiences a motor error the XN function should be used to inform all other nodes and perform a controlled shutdown of the plant, for example the XN could call the motor error sequence on other nodes. Other uses of XN are not recommended, due to the lack of acknowledgment between transmitter and receiver.

5. Error Messages and Codes

Eleven additional error codes have been added to the PTS language for SynchroLink.

Error	Description	Comments
123	No CAN chip fitted to board	SynchroLink functionality requires a CAN interface IC to be fitted to the circuit board
124	CAN chip stuck in reset	Hardware fault on circuit board, contact Quin Systems
125	No CAN network transceiver power	Check CAN cabling
126	This node is CANbus OFF	Reported when node goes off bus due to errors or when already bus off and a command (such as LK) is issued
127	Unable to log onto CANbus	Either there is insufficient bus idle for CAN chip to synchronise with other bus traffic or the wiring is faulty
128	No free CAN mailbox	All eight mailboxes are in use: no more SynchroLink map links available
129	No CAN mailbox assigned	ML,LK or UL function failed
130	Node <n> is already clock master	Only one clock master is required
131	No Clock signal being received	Node is not receiving the clock signal that should be being broadcast from one node
132	Another node already uses our node number	Attempt to go bus on with CN<n> failed as <n> is already used, try another number
133	XN(s) lost on CANbus	Over ten XN's arrived before they could be processed

Table 1: CANbus Error codes and descriptions

INDEX

C

CK 8
Clock 8
CN 8
Connections 7
CPU360 5
CQ 9

D

Diagnostics 9

E

error codes 11

L

LK 9

M

Map master 9
Map slave 10
MiniPTS3 5
ML 10

N

Nodes 8

P

Power supply 7

S

Sequences 10
ServoNet 5
Status 9

T

Terminators 7

W

Wiring 7

X

XN 10, 11

