Technical Reference

Sutron Corporation
22400 Davis Drive
Sterling, Virginia 20164
TEL: (703) 406-2800
FAX: (703) 406-2801
WEB: http://www.sutron.com

Table Of Contents

Welcome to the Technical Reference

This document describes all the setup fields, and DDE/COM commands supported by the XConnect applications. This information can be used to build add-on products for XConnect, interface to existing XConnect servers with 3rd party software, or design custom setups and applications for situations beyond the scope of the standard setup and operator programs. Knowledge of the fields and setup information can used and tested with applications ranging from the very simple Microsoft Excel, Microsoft Word, Visual Basic, Intouch up to high level languages such as C and Pascal. The sections are organized by DDE information and Setup information. In general the DDE information is used to control and access an applications in real-time, whereas the setup information is used to define the default behaviour and operation of the application. A brief introduction is included to each application, and meant to be of a more technical nature than the information contained in the individual XConnect applications' help file.

This document describes an open, expandable and flexible standard for data collection and storage, which can grow as technology and requirements develop.

DDE Concepts

DDE stands for Dynamic Data Exchange and is a Windows based messaging protocol for sharing information between applications. DDE is the basis for all information exchange and commands in the PC Base 2 software. Much of the DDE interaction between applications is automatic and hidden from the user, but since DDE is supported by many non-Sutron applications it is very useful to understand the basics of DDE so that these applications can interact with the base station software.

A DDE message consists of four parts:

- the name of the **application** receiving the message
- the **topic** which the message is about
- an **item** under that topic
- **data**

The application name is typically limited to 8 characters, such as XCSETUP, XCRTU, or EXCEL. If NETDDE is installed, then the application name can also contain a network node name, such as \\WORKSTATION\XCSETUP would address the XC Setup running on network node WORKSTATION. The topic name will depend on the application, most of the topics are named DATA or CONTROL in XConnect. DATA topics usually supply or change information, whereas CONTROL topics usually initiate or check the status of actions. The data itself can be transferred in any format which the sending and receiving applications can agree upon. The most basic formats are called TEXT for tab delimited ASCII text and CSV for comma delimited ASCII text. Most applications support at least the TEXT format. These formats are also the same formats as used by the Windows clipboard, hence they are often called clipboard formats.

DDE supports seven basic commands:

- INITIATE -- is used to start a DDE conversation between two applications and requires an application name and a topic name to do so.
- TERMINATE -- is used to end a DDE conversation on a particular topic.
- ADVISE -- is used to request a hot or automatic link to a data item, so that whenever the data item changes the applications which requested the advise link will be updated automatically.
- UNADVISE -- terminates an advise link.
- REQUEST -- is used to request the current value of a data item.
- POKE -- is used to set the current value of a data item.
- EXECUTE -- is used to execute a macro or list of commands specified by the data item.

NOTE: The syntax and format of commands, data, and responses is not standardized in DDE and hence this online help exists to describe the formats adopted by Sutron.

Sutron software uses the same notation for DDE paths as used by Microsoft EXCEL: app|top!item to specify a DDE item or just app|top to specify a DDE topic where the character "|" vertical bar separates applications from topics and the character "!" exclamation point separates topics from items. Specifying a network source for a DDE path is performed by placing \\node\ before the application.

XConnect and OPC

The XConnect servers are automation servers built on Microsoft Windows COM technology. As a natural extension, XConnect has implemented OPC (OLE for Process Control) and XConnect applications such as XC Rtu, XC Decode are OPC server. Like DDE, OPC allows applications to share data via standard interface. In this case, OPC standardizes the communication for process control data. OPC is a communication standard.

When data acquisition devices conform to the OPC standard, you can use them with any OPC-enabled software application. And vice versa. You can therefore easily combine different devices from different manufacturers in one system, without the normal integration headaches. OPC gives you the freedom to add new hardware from third-party vendors to existing set-ups, or to replace a device, without worrying about compatibility with your chosen software. Select exactly the hardware and software you want for a particular application.

Based on Microsoft Windows technology, OPC now stands for Open Process Control. Previously it stood for OLE Process Control but today the OLE technology has been replaced by Active X.

The measurement and control hardware provides front-line data acquisition. As soon as the hardware device has collected the data it makes it available to software applications running under Windows. It presents the data according to the OPC standard, and is thus known as an OPC server. Each OPC server offers data in the same way. If the software application can understand the OPC format it can therefore access data from any OPC server device, making individual drivers for each piece of equipment obsolete.

OPC-enabled software include spreadsheets, databases, virtual instruments and SCADA (supervisory control and data acquisition) interfaces. These applications are known as OPC client software. You can also develop your own Visual Basic programs to exchange data with OPC servers. Your program will work with all OPC devices connected to your PC and on the network.

Each OPC server can simultaneously provide data for any number of OPC clients. Likewise multiple clients can at the same moment access any server: a robust method of communication. With OPC, measurement and control systems can share information and co-operate with other installations across factories, offices, laboratories, etc. The same data is therefore readily available to engineering, maintenance, management...in fact to anyone that requires up-to-the-minute data on which to base their decisions.

OPC allows "plug-and-play". All OPC devices will connect together and immediately work with the OPC client software. This has the potential to massively reduce installation and system configuration time. It also means that you can add devices without shutting down existing systems.

OPC Version

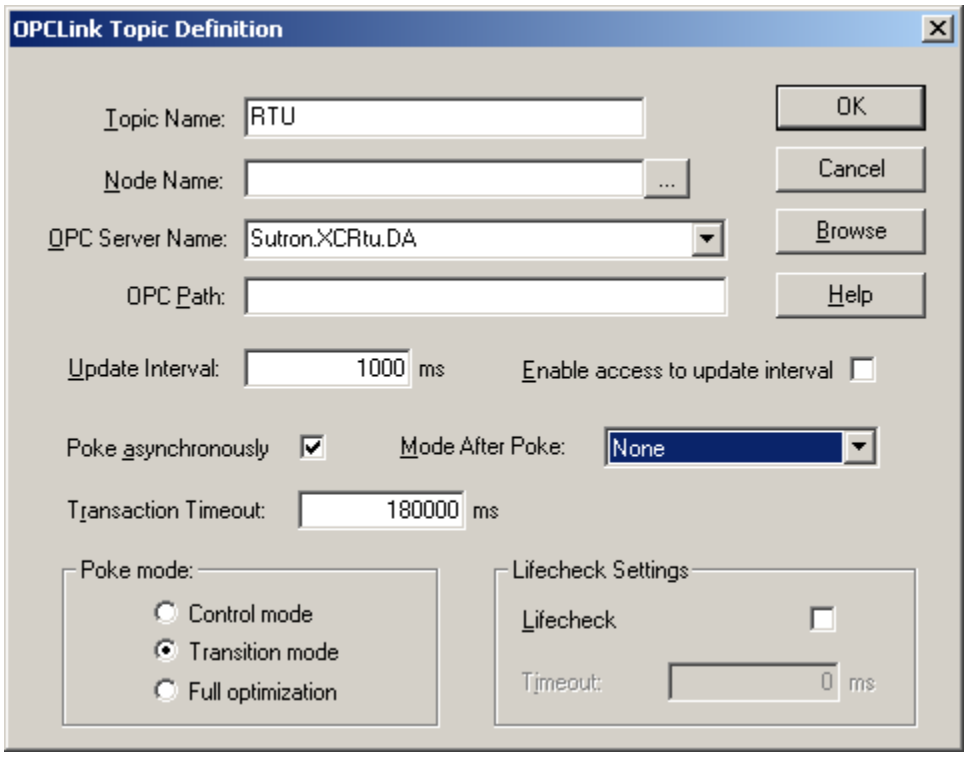
XConnect has implemented OPC Data Access Custom Interface Version 2. The details on interfaces implemented are listed below:

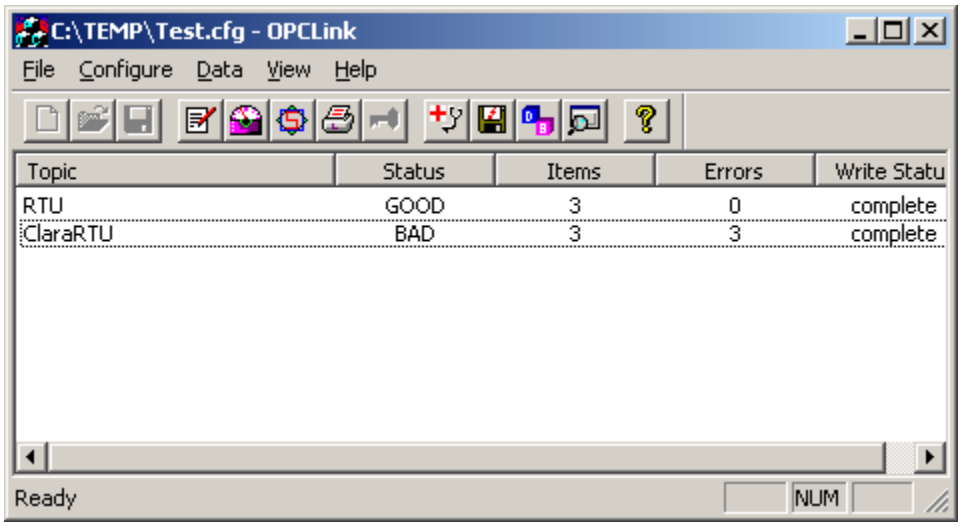
- OPC Server
  - IOPCServer
  - IOPCServerPublicGroups
  - IOPCBrowseServerAddressSpace
  - IOPCItemProperties
  - IConnectionPointContainer
  - IOPCCommon
- OPC Group
  - IOPCGroupStateMgt
  - IOPCPublicGroupStateMgt
  - IOPCASyncIO2

  - IOPCItemMgt
  - ICOnnectionPointContainer
  - IOPCSyncIO
- EnumOPCItemAttributes
  - IEnumOPCItemAttributes

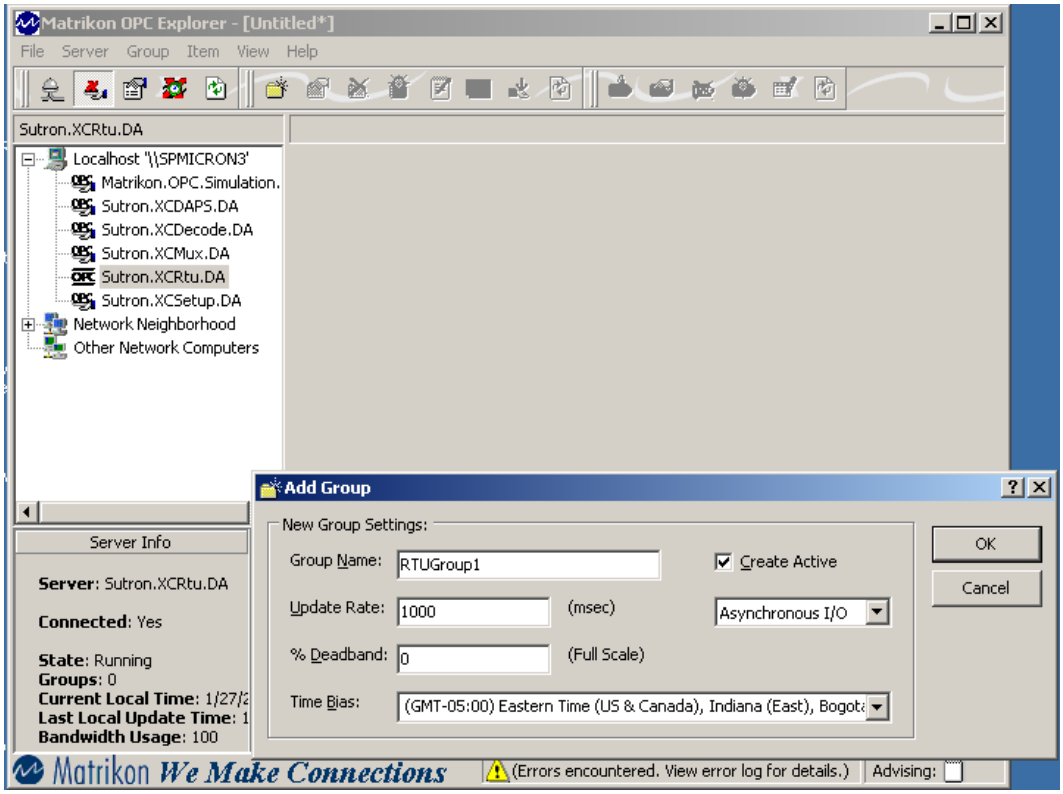
OPC Clients tested with XConnect OPC Servers

- Wonderware Intouch 8.xx/OPC Link 7.6.1.9 in compliance with OPC Data Access version 2.04.





- Matrikon OPC Explorer (http://www.matrikonopc.com/products/opc-desktop-tools/opc-explorer.asp)



OPC Test Software

- Wonderware Intouch / OPC Link
- OPC Foundation Compliance Test Utility

XConnect Implementation

In general, the XConnect servers try to preserve the same syntax as DDE tag names. They follow the DDE data points as described in XConnect Tech Note. The DDE topic name is the first access path followed by a dot "." and then the item name.

- Because all DDE points can be read/write, all OPC tagnames are writable as well.
- All values returned are variants, so that they can be easily converted to other data types just as DDE.
- If the values for tagnames are updated, the clients are notified only when the values are changed. No notifications are sent out if the update is the exact same value as the old one.

Example 1

DDE - Application - XCRtu; Topic - Data; Item - XPERT.AT.

OPC - The sensor AT on station XPERT can be accessed as DATA.XPERT.AT.

Example 2

DDE - Application - XCRtu; Topic - Data; Data - $CURRENT.

OPC - The current string can be accessed as DATA.$CURRENT.

Example 3

DDE - Application - XCRtu; Topic - Control; Item - POLL,XPERT,CURRENT; Data - 10.

OPC - Set CONTROL.POLL,XPERT,CURRENT to value 10 will trigger the current poll to station XPERT.

Example 4

DDE - Application - XCRtu; Topic - Control; Item - POLL,XPERT,DATE; Data - 10.

OPC - To poll the station using Date poll, write/set the item CONTROL.POLL,XPERT,DATE to 10.

Example 5

OPC - Special item CONTROL.EXECUTE is available for other control functions. Set this item the value in the syntax as "key = value". Set CONTROL.EXECUTE to string POLL,XPERT,CURRENT=10 will trigger the current poll to station XPERT. This is a generic item that can be used to create additional tagnames in the OPC server address space.

Additional References

For more information, view documents listed below at http://www.opcfoundation.org.

- OPC Overview 1.0
- OPC Common Definitions and Interfaces 1.0
- OPC Data Access Custom Interface Standard 2.05

XC Setup

XC Setup

Introduction

The XC Setup is designed to isolate XConnect applications from the details of how setup information is stored and accessed. This allows the setup information for any application to be modified locally or remotely by any other application using DDE or NETDDE. The current implementation of the XC Setup stores information in the file XCONNECT.INI in an ASCII format, similiar to other Windows .INI files. Because applications often cache setup information in their own local memory, the XC Setup supports a method of notifying applications that setup information has been changed.

By default the setup is stored in the file XCONNECT.INI in your default Windows directory. There are two ways to modify which file is used. Changing the default file is useful if you need to maintain multiple setups. From the XC Setup main menu, select File|Open and select the desired filename.

The second way is to pass the filename on the command line to XC Setup. Since XC Setup is automatically started by XConnect server programs with an empty command line, you will have to make sure that you start XC Setup with the correct command line parameter before running other XConnect programs. You can select Options|Make INI Default from the XC Setup main menu to ensure the new filename is used as the default name (instead of XCONNECT.INI) when XC Setup is started

Warning: Although you can edit the XCONNECT.INI directly with the notepad, edit or any ASCII text file editor, **do not do so while the XC Setup is running - shut it down first!** You can make changes to the setup while the base station is running using DDE, but be aware that many applications cache the setup information and will not notice the changes you have made unless they are told to reset their caches or restart. See the discussion about "GENERAL,RESETSETUP".

Data is stored in the XCONNECT.INI files under a topic and an item, the general format is show below:

[topic]

item=some kind of ASCII printable data

There can be an unlimited number of topics in a setup and an unlimited number of items under any topic. Data must be less than 64K bytes, and must consist of characters between space (32) and tilde (127) in the ASCII sequence. There is one DDE topic under XC Setup.

DDE Information

DDE APPLICATION NAME: XCSETUP

DDE TOPIC NAME: DATA

ITEM	A	P	R	E	DATA	DESCRIPTION
topic,item	x	x	x		any string	Data in the XC Setup is stored under a topic and an item name, a topic and an item can be assigned any text value. Operation will fail if the data does not exist in the setup.
topic,*			x		topic list	When a "*" is specified for the item, a list of all items under the specified topic will be returned.
+topic,item		x			any string	Poking with a "+" sign before the topic will force the item to be added to the setup if it does not already exist.
+topic,item			x		+|-any string	Requesting with a "+" sign before the topic will cause a "+" sign to be inserted before the returned data if the

						data was accessed thru a template or a "-" sign if it was not.
-topic,item			x		any string	Requesting with a "-" before the topic sign will disable template searching.
ADD topic,item				x	n/a	Adds a topic and an item to the setup initializing it to blank.
DEL topic,item				x	n/a	Deletes an topic and an item from the setup.
UPDATE				x	n/a	Causes the XC Setup to write all changes in memory out to disk.

Operations: A-Advise/UnAdvise, P-Poke, R-Request, E-Execute

Setup Information

XC Setup does not use much setup information, its job is to supply setup information to other applications. One piece of information which all applications which use the XC Setup share in common via an advise link is the "GENERAL,RESETSETUP" item which when poked with the values below should cause the following actions:

1: Minor setup changes to status information such as station/sensor setups has occured, All applications should flush their setup information caches.

2: Major setup changes to configuration information such as COM ports and data files has occured, all applications should close active functions which rely on this information, re-read setup, and restart.

3: Reset DDE communications, useful when automatic links may have been terminated and/or re-started and applications are no longer in sync.

9: XC Setup has shutdown, and all users of setup information should also shutdown.

TOPIC	ITEM	DEFAULT	DESCRIPTION
[SETUPSERVER]	UpdateInterval	600	UpdateInterval contains the number seconds between disk writes performed by the XC Setup. If another value is not specified the XC Setup will write changed setup information out to disk every 10 minutes. If no data has been changed then the XC Setup does not write to disk. A long interval (such as 10 minutes) will reduce disk writing and help keep the system running faster (updating a Windows .INI file is a very disk intensive). The risk of a long interval is that updates might be lost if power is lost or the system crashes between updates. Client applications should instruct XC Setup to update after making significant changes to the setup.

The [GENERAL] topic contained in the following table contains setup items shared by all XConnect server applications:

TOPIC	ITEM	DEFAULT	DESCRIPTION
[GENERAL]	Version	2	Version number of setup file, as new features are introduced this number may be increased.
	DataPath		Default directory path to data files.
	SetupPath		Default directory path to setup files.
	TablePath		Default directory path to setup files.
	LogPath		Default directory path to store Logger's log files.
	RawPath		Default directory path to store raw (non-decoded) NESDIS message files.
	Use24Hour	No	When set to "Yes" midnite will be displayed as 24:00 hours, when set to "No" midnite will be displayed as 00:00 hours. This option does not effect how the data is stored.
	ResetSetup		see table above.
	INIFilename	XCONNECT.ini	Contains the file name of the setup file currently in use. The value is (read only) poking it will not change the setup file. To change the default setup file assign the filename to the DOS environment variable "XCONNECT.INI" before running windows, or pass the filename on the command line to XC Setup before starting other elements of XCONNECT. The default ini file can be changed from XC Setup. The name of the file will also be stored in the registry.
	GMT	0	Number of hours Greenwich Mean Time (GMT) is ahead of Local Time. Currently used by the GOES XC Mux program to time stamping NESDIS compatible files with GMT time. Also used by the GOES XC Decode program in order to extract the time stamp from the same files. If you only ever going to use NESDIS files retrieved from NESDIS or from users in other time zones, then it is important to enter the correct GMT offset for your time zone. Use a negative value to indicate the Greenwich Mean Time is behind Local Time.
	Stations		List of defined stations. ex: PumpPlant,MetSite,StageSite

	DecodeAll	No	Set to Yes will causes the XC Decode to at least partially decode all messages regardless of whether the SatID of the message is defined. This is useful when debugging because the decoder can decode the data quality information without needing setup information, but is undesirable in a final system because it causes the decoder to perform more processing than is necessary.
	UseNoValue	No	If Yes, then a value decode by XC Rtu or XC Decode as nil or missing will be replaced with the NoValueIndicator.
	NoValueIndicator	998877.0	If UseNoValue is Yes, the value for NoValueIndicator will be used for any value decode as missing or nil.

XC Rtu

XC Rtu

Introduction

The XC Rtu is responsible for all communications with the RTUs. The XC Rtu supports automatic polling for archived or current data when is stored in to a real time RAM resident data base. The data values are then available to other programs such as Wonderware Intouch for display. XC Rtu also supports many manually initiated operation such as uploading and downloading setups, starting and stopping RTUs, and other SSP operations. Uninitiated messages such as alarms are also supported. Setup information for the stations is stored in the XConnect.INI file and accessed via XC Setup.

XC Rtu has 4 DDE topics:

1. Control
2. Data
3. Auto
4. Command
5. DataServe

DDE Information

DDE APPLICATION NAME: XCRTU

DDE TOPIC NAME: CONTROL

The following codes are used for the control items, 10, 11 and 12 can be DDE poked to a control item, and 1 through 9 are returned as a status from a DDE request once the operation begins.

0	Initial State, No Data
1	Operation OK
2	Operation Failed
3	Disconnecting from RTU
8	Operation in Progress
9	Connecting to an RTU
10	Do an operation, such as connect to an RTU
11	Disconnect from an RTU
12	Cancel an opertion. Eliminates all occurrences of an operation from the message queue and if it is in progress, aborts the current operation.

ITEM	A	P	R	E	DATA	DESCRIPTION
CONNECT,rtu	x	x	x		1,2,3,9,10,11,12	Poke it to: Connect or Disconnect from an RTU. Request it to: Check on manual or automatic

ITEM	A	P	R	E	DATA	DESCRIPTION
						connect status. All control items will automatically connect before sending a command, so you only need to use this control when you want to make, verify, and hold a connection - such as when performing control loop operation at the RTU. Poking an 11 does not cancel the operation, instead it causes a low priority Disconnect command to be placed in the que. The disconnect will take place after all other other operations have been completed. Poking a 12 can be used to cancel a connect operation.
POLL,rtu,CURRENT	x	x	x		1,2,3,8..12	Request current data from an RTU.
POLL,rtu,DATE	x	x	x		1,2,3,8..12	Request archive data from an RTU based on the LastPoll date stored in the setup.
POLL,rtu,NEWEST	x	x	x		1,2,3,8..12	Request archive data from an RTU based on the date of last poll stored in the RTU.
POLL,rtu,MANUAL,startdate,stopdate	x	x	x		1,2,3,8..12	Request archive data from an RTU from startdate to stopdate. Date format is: MM/DD/YYYY HH:MM:SS
POLL,rtu,MANUAL,startdate,stopdate,mask	x	x	x		1,2,3,8..12	Request archive data from an RTU from startdate to stopdate, but only those log entries where the record id matches the character mask. (Not supported by the 8200)
POLLGROUP,group	x	x	x		1,2,10	Causes a poll group to be polled immediately. The poll group will also be polled at its regularly scheduled time. Only enabled poll groups can be polled, but they do not have to be in the

ITEM	A	P	R	E	DATA	DESCRIPTION
						automatic poll list. The return status only indicates start of polling not completion or final success. To find out whether the poll worked you will have to check the poll status for each station individually (i.e. POLL,rtu,DATE or other option). Reasons for this function failing include the Poll Monitor Window has been shutdown or that the specified poll group does not exist or is not enabled.
SENDTAG,rtu,sensor1[:a-b],etc...	x	x	x		1,2,3,8..12	Send the current value of specified sensors to an RTU. A numeric range can be used to help specify multiple sensor values to send. For instance RainFall:1-3 would be the same as listing out: "RainFall:1,RainFall:2, RainFall:3".
GETTAG,rtu,sensor1[:a-b],etc...	x	x	x		1,2,3,8..12	Requests the current value of specified sensors from an RTU. A numeric range can be used to help specify multiple sensor values to get.
SENDSETUP,rtu	x	x	x		1,2,3,8..12	Sends the setup stored in rtu.$SETUP to the RTU. The format is always 9000.OBJ. This method exists to allow downloading a setup previously placed in memory with GETSETUP (but is rarely used). **8200/8210/9000 only**
SENDSETUP,rtu,filename	x	x	x		1,2,3,8..12	Sends the setup stored in filename to the RTU. **8200/8210/9000 only**
SENDSETUP,rtu,*	x	x	x		1,2,3,8..12	Sends the setup stored in the default setup file to the RTU. **8200/8210/9000 only**
GETSETUP,rtu	x	x	x		1,2,3,8..12	Requests the setup from an RTU and stores it to rtu.$SETUP. The format is always 9000 .OBJ. This exists to allow uploading a

ITEM	A	P	R	E	DATA	DESCRIPTION
						setup without creating a disk file (but is rarely used).
GETSETUP,rtu,filename	x	x	x		1,2,3,8..12	Requests the setup from an RTU and stores it to filename. Handles 9000 .OBJ format or 8200 .SET and .BAS format. For an 8200, pass the name of the .SET file, the basic file is assumed to have the same name with an extension of .BAS. **8200/8210/9000 only**
GETSETUP,rtu,*	x	x	x		1,2,3,8..12	Requests the setup from an RTU and stores it to the default setup file. For an 8200, the basic portion of the setup will be saved under the same name as the setup file with an extension of .BAS. **8200/8210/9000 only**
MAIL,rtu[,message]	x	x	x		1,2,3,8..12	Sends a mail message containing rtu.$OutBox to the RTU. If the optional message is included, then that message is sent instead of the contents of rtu.$OutBox, however it will be sent in upper case.
SETCLOCK,rtu	x	x	x		1,2,3,8..12	Sets the clock of the RTU to the current time of the PC.
SELECTCONTROL,rtu,sensor:a-b	x	x	x		1,2,3,8..12	Loads sensors values in to an RTU, and checks to see if the RTU will accept a BeginControl. SelectControl and BeginControl are used to implement a check back before operate control. **9000 only.**
BEGINCONTROL,rtu	x	x	x		1,2,3,8..12	If SelectControl worked then BeginControl is used to eval the sensor to start the operation. **9000 only.**
SELFTEST,rtu	x	x	x		1,2,3,8..12	Instructs the RTU to perform a self test. **8200/8210/9000 only.**
CLEARSTATUS,rtu	x	x	x		1,2,3,8..12	Clears the system status of the RTU. **8200/8210/9000 only.**

ITEM	A	P	R	E	DATA	DESCRIPTION
GETSTATUS,rtu	x	x	x		1,2,3,8..12	Requests the system status of the RTU.
RESET,rtu	x	x	x		1,2,3,8..12	Causes the RTU to perform a hard reset. **8200/8210/9000 only.**
ERASESETUP,rtu	x	x	x		1,2,3,8..12	Causes the setup in the RTU to be erased. **8200/8210/9000 only.**
START,rtu	x	x	x		1,2,3,8..12	Start RTU processing and recording.
STOP,rtu	x	x	x		1,2,3,8..12	Stops RTU processing and recording.
STARTTAG,rtu,tag	x	x	x		1,2,3,8..12	Runs the start code of a tag in the 9000. **9000 only.**
EVALTAG,rtu,tag	x	x	x		1,2,3,8..12	Runs the eval code of a tag in the 9000, causes a sensor to be measured in the 8200.
STOPTAG,rtu,tag	x	x	x		1,2,3,8..12	Runs the stop code of a tag in the 9000.
RESETSETUP	x	x	x		1,2,3,8..12	Causes the XC Rtu to flush its setup information cache and re-read station information from XC Setup
RESETSETUP,ALL	x	x	x		1,2,3,8..12	Causes the XC Rtu to shutdown and restart all communications monitors and polling, re-reads all setup information.
GETBERTSTATUS,rtu	x	x	x		1,2,3,8..12	Requests the current BERT status of an RTU and places the information in to rtu.$BERTGOOD, rtu.$BERTBAD, and rtu.$BERTDIST.
GETBERT,rtu	x	x	x		1,2,3,8..12	Requests that an RTU send a BERT message. A BERT message is a 1024 byte message consisting of 256 four-byte blocks designed to detect errors in communication links. The results are available by requesting RTUSERVE|COMMAND!COMINFO,comport,xxx where xxx is BERTGOOD, BERTBAD, and BERTDIST.

ITEM	A	P	R	E	DATA	DESCRIPTION
SENDBERT,rtu	x	x	x		1,2,3,8..12	Sends a BERT message to an RTU. A BERT message is non-addressable, so other RTUs within range will also be able to here the BERT message.
CLEARBERT,rtu	x	x	x		1,2,3,8..12	Clears the BERT status of an RTU, setting them to 0.

Operations: A-Advise/UnAdvise, P-Poke, R-Request, E-Execute

Message Priority

When SSP messages are dispatched via a scheduled poll or on-demand, they are entered in the message queue of the appropriate Communication Monitor in XC Rtu. The are executed from the queue based on the following rules:

1. Priority
2. Station Name (alphabetically)
3. Operation code (control command)

Priority Options

Some commands can be scheduled with a specific priority. To trigger a command add the priority before the item (command): %EPOLL,rtu,CURRENT.

pri_Emergency	%E
pri_Override	
pri_High	%H
pri_Medium	%M
pri_Low	%L
pri_AutoPoll	
pri_OnPoll	%O
pri_AfterPoll	%A
pri_None	
pri_Illegal	

Operation Code Priority Options

See detailed description of operation codes above.

qop_SSPConnectRTU
qop_SSPDisconnectRTU
qop_SSPPollCurrent

qop_SSPPollNoWait
qop_SSPPollNewest
qop_SSPPollDate
qop_SSPPollManual
qop_SSPSendSetup
qop_SSPGetSetup
qop_SSPMail
qop_SSPSetClock
qop_SSPSelectControl
qop_SSPBeginControl
qop_SSPSelfTest
qop_SSPClearStatus
qop_SSPGetStatus
qop_SSPReset
qop_SSPEraseSetup
qop_SSPStart
qop_SSPStop
qop_SSPStartTag
qop_SSPStopTag
qop_SSPEvalTag
qop_SSPSendTag
qop_SSPGetTag
qop_SSPBertStatusReq
qop_SSPBertClear
qop_SSPBertReq
qop_SSPBertSend
qop_SSPSendRDI
qop_SSPGetRDI
qop_None

DDE APPLICATION NAME: XCRTU

DDE TOPIC NAME: DATA

Sensor alarm values are made up of one or more alarm characters (alarms are the second value in a tag), below is a table of alarm characters:

ALARM CHAR	DESCRIPTION	ALARM CHAR	DESCRIPTION

OK	Everything is normal	1	Intermediate Low Limit Alarm (9000)
H	High Limit Alarm	T	Control Timeout (9000)
L	Low Limit Alarm	F	Control Failure (9000)
R	Rate Of Change Alarm (8200/8210)	S	Control Software Error (9000)
+	Trend Rising (8200/8210)	C	Control Hardware Error (9000)
-	Trend Falling (8200/8210)	D	Digital Alarm (9000)
A	In Alert	I	Alarm Inhibited (9000)
h	Intermediate High Limit Alarm (9000)		

ITEM	A	P	R	E	DATA	DESCRIPTION
station.sensor	x	x	x		a number, an alarm string, a character, or a boolean "Yes" or "No" depending on the type of sensor.	Current real time value of a particular sensor from a station. Results of a current data poll or a get tag operation will update these values. You must set these values before performing a send tag operation
station.sensor :n	x	x	x		same as station.sensor	Same as station.sensor but allows for multiple values under the same sensor name.
station.$LASTUPDATE	x	x	x		mm/dd/yyyy hh:mm:ss	Time and date of last received current data poll.
station.$LASTALARM	x	x	x		mm/dd/yyyy hh:mm:ss	Time and date of last received alarm.
station.$TIMETAG	x	x	x		mm/dd/yyyy hh:mm:ss,sensor,data, sensor,data, ...	Last time tag data received consisting of a date and time stamp followed by as many sensor names and data values as exist in the log record received from the station.
station.$LASTPOLL	x	x	x	x	mm/dd/yyyy hh:mm:ss	Time and date of last data from a timetag (date) poll.
station.$SETUP	x	x	x		9000 SDL object file format	Setup received from a station, or setup to send to a station as used by the SENDSETUP/GETSETUP commands.
station.$STATUS	x	x	x		number	A 32-bit bit masked number returned by the GETSTATUS

ITEM	A	P	R	E	DATA	DESCRIPTION
						command.
station.$INBOX	x	x	x		string	Last mail message received from the station.
station.$OUTBOX	x	x	x		string	Mail message to send to the station using the MAIL command.
station.$RXCOUNT	x	x	x		number	Total number of good messages received from this station addressed to the master station.
station.$RXTOTAL	x	x	x		number	Total number of good messages received from this station addressed to the master station.
station.$RXBAD	x	x	x		number	Total number of bad messages from this station, where the from station part of the header was ok but data was lost elsewhere resulting in a CRC error.
station.$TXCOUNT	x	x	x		number	Total number of messages sent to this station.
station.$TXBAD	x	x	x		number	Total number of times this station failed to respond to a request for information even after all retries were attempted.
station.$POLLSTATUS	x	x	x		0,1,2	Status of the last POLL control performed on this station. 0 - means no poll has been performed yet 1 - means the last poll was good 2 - means the last poll was bad This status can be more useful than the results of the POLL control itself for detecting communication problems, because the POLL control is forever changing value from 10 down to 1 or 2 as polls cycle over and over.
comport.$DECODE	x	x	x		to,from,seqnumstatus, opcode,data	Last decoded message from the specified comport (COM1..COM9). This field only updates when the SSP Decoder window is open, although it will update will the screen is paused.

ITEM	A	P	R	E	DATA	DESCRIPTION
station.$BERTGOOD	x	x	x		number	Number of good BERT blocks received by this station. There are 256 BERT blocks in a single BERT message. This is set by first polling for BERT statistics using the GETBERTSTATUS control.
station.$BERTDIST	x	x	x		32-bit number	Error distribution of blocks in the BERT message. There are 32 bits in this base10 number representing 256 blocks, so each bit represents an error in 8 blocks. The most significant bit represents the first block, so that when the number is displayed in hexadecimal or binary errors in early blocks will show up at the beginning and errors in later blocks show up at the end. A value of 0 would mean that no errors have occured, a value of 4294967295 (FFFFFFFF16) means that errors are distributed in possibly all the blocks.
$TIMETAG	x	x	x		mm/dd/yyyy hh:mm:ss,station.sensor,data, station.sensor,data	Last time tag data received from any station consisting of a date and time stamp followed by pairs of station.sensor names and data values.
$REALTIME	x	x	x		mm/dd/yyyy hh:mm:ss,station.sensor,data, station.sensor,data,	Last alarm or current data received from any station consisting of a date and time stamp followed by pairs of station.sensor names and data values . Current data results from a current data poll, whereas alarm data occurs randomly.
$ALARM	x	x	x		mm/dd/yyyy hh:mm:ss,station.sensor,data, station.sensor,data	Last alarm data received from any station consisting of a date and time stamp followed by pairs of station.sensor names and data values.
$CURRENT	x	x	x		mm/dd/yyyy hh:mm:ss,station.sensor,data, station.sensor,data,	Last current data received from any station consisting of a date and time stamp followed by pairs of station.sensor names and data values.

Operations: A-Advise/UnAdvise, P-Poke, R-Request, E-Execute

DDE APPLICATION NAME: XCRTU

DDE TOPIC NAME: AUTO

The AUTO topic allows access to the exact same points as in the DATA topic with the addition that points can be automatically sent to field RTUs when poked, or polled from the RTU when an advise link is created. The AUTOPOLL and AUTOUPDATE parameters (settable in the XCONNECT.INI file under the [RTUServer] topic or changable in real-time via the COMMAND topic), enable and control how often SENDTAG and GETTAG controls are are sent to RTUs. An additional feature of the AUTO topic is that polled data will not overwrite poked data until a SENDTAG control has been generated with the poked data. This ensures that the data you poke is the data that is sent, whereas in the DATA topic if a poll for current data might replace the data you intended to send to the station before you had the chance to issue a SENDTAG control.

ITEM	A	P	R	E	DATA	DESCRIPTION
station.sensor	x	x	x		a number, an alarm string, a character, or a boolean "Yes" or "No" depending on the type of sensor.	Current real time value of a particular sensor from a station. Results of a current data poll or a get tag operation will update these values. Poking a value will initiate a SENDTAG control, advising a data point will initiate GETTAG controls to fetch the current value of a data point.

Operations: A-Advise/UnAdvise, P-Poke, R-Request, E-Execute

DDE APPLICATION NAME: XCRTU

DDE TOPIC NAME: COMMAND

ITEM	A	P	R	E	DATA	DESCRIPTION
AUTOPOLL		x	x		number	Determines how often (in seconds) GETTAG controls will be issued to RTUs in order to update hot-linked data points in the AUTO topic. A value of 0.0 disables automatic polling. A negative number will cause a single poll to take place using a delay based on the absolute value, and then AUTOPOLL will be reset to 0.
AUTOUPDATE		x	x		number	Determines how long the XC Rtu will wait before issuing SENDTAG control to update RTUs after a data point in the AUTO topic has been changed. A value of 0.0 disables automatic updating. A negative number will cause a single pol to take place using a delay based on the absolute value, and then AUTOUPDATE will be reset to 0.
POLLINFO,GROUP			x		groupnum	Group number of next poll group to be polled.

ITEM	A	P	R	E	DATA	DESCRIPTION
POLLINFO,TIME			x		mm/dd/yyyy hh:mm:ss	Date and time that next poll will take place.
POLLINFO,STATUS			x		lines-of-text	Contents of the Poll Group Scheduler's status display list box.
COMINFO,comport,STATUS where comport is COM1,COM2,...			x		lines-of-text	Contents of the status list box from the Communication Monitor for the selected comport.
COMINFO,comport,ERROR			x		lines-of-text	Contents of the error status list box from the Communication Monitor for the selected comport.
COMINFO,comport,MESSAGE			x		lines-of-text	Contents of the message que list box from the Communication Monitor for the selected comport.
COMINFO,comport,STATS			x		tx,rx,rx count,rx total, rxbad,txcount,txrepeat	Contents of all statistic boxes from the Communication Monitor for the selected comport'. Each statisitic is described below (this option lets you grab them all at once instead of one at a time).
COMINFO,comport,TX			x		"Yes" or "No"	Status of the TX lite from the Communication Monitor for the selected comport'. Returns "Yes" if an SSP message is being sent.
COMINFO,comport,RX			x		"Yes" or "No"	Status of the RX lite from the Communication Monitor for the selected comport'. Returns "Yes" if an SSP message is being received.
COMINFO,comport,RXCOUNT			x		number	Value of the RXCOUNT box from the Communication Monitor for the selected comport'. This is the number of received messages addressed to the base station.
COMINFO,comport,RXTOTAL			x		number	Value of the RXTOTAL box from the Communication Monitor for the selected comport'. This is the number of received messages addressed to anyone.
COMINFO,comport,RXBAD			x		number	Value of the RXBAD box from the Communication Monitor for the selected comport'. This is the number of bad messages

ITEM	A	P	R	E	DATA	DESCRIPTION
						received. Bad messages occur because of CRC errors, dropped characters, and protocol errors.
COMINFO,comport,TXCOUNT			x		number	Value of the TXCOUNT box from the Communication Monitor for the selected comport. This is the number of transmitted messages.
COMINFO,comport,TXBAD			x		number	Value of the TXBAD box from the Communication Monitor for the selected comport'. This is the number of transmitted messages which failed because they could not be sent or an acknowledgment was never received.
COMINFO,comport,TXREPEAT			x		number	Value of the TXREPEAT box from the Communication Monitor for the selected comport'. This is the number of repeated messages (messages stored and forwarded to another station or com port).
COMINFO,comport,BERTGOOD			x		number	Number of good BERT blocks received on this com port. There are 256 BERT blocks in a single BERT message.
COMINFO,comport,BERTBAD			x		number	Number of bad BERT blocks received by on this com port. Bad blocks are accumulated by counting the number of missing blocks between two good blocks.
COMINFO,comport,BERTDIST			x		32-bit number	Error distribution of BERT blocks received on this com port. There are 32 bits in this base10 number representing 256 blocks, so each bit represents an error in 8 blocks. The most significant bit represents the first block, so that when the number is displayed in hexadecimal or binary errors in early blocks will show up at the beginning and errors in later blocks show up at the end. A value of 0 would mean that no errors have occured, a value of 4294967295 (FFFFFFFF16) means that errors are distributed in possibly all the blocks.

ITEM	A	P	R	E	DATA	DESCRIPTION
COMINFO,comport,STATS		x			0	Poking a zero causes all communication statistics to be cleared for the Communications Monitor on the selected comport.
ABORT,comport		x	x		0,1	Poking a 1 will abort all operations performed by the communications monitor for the selected comport. All operations in the message que will be flushed (after the current operation has been aborted). Poking a 0 will clear the abort flag and return operations to normal. Requesting the value will return the status of the abort flag as set by software or manually by the operator: 1=Aborting, 0=Normal.
CANCEL,comport		x	x		0,1	Poking a 1 will cancel the current operation(s) in progress by the communications monitor for the selected comport. Poking a 0 will clear the cancel flag and return operations to normal. The cancel flag will automatically return to 0 after the current operation(s) is canceled. The message que is not cleared. Requesting the value will return the status of the cancel flag as set by software: 1=Canceling, 0=Normal. Canceling a communications monitor which is aborted has no effect.
PROTOCOLON,comport		x	x		No,Yes	Poking PROTOCOLON with Yes, On, or 1 turns the protocol analyzer window on for the specified com port - poking No, Off, or 0 turns it off. Requesting returns the current state of the protocol analyzer window: Yes if its on or No if its off.
DECODEON,comport		x	x		No,Yes	Poking DECODEON with Yes, On, or 1 turns the SSP decoder window on for the specified com port - poking No, Off, or 0 turns it off. Requesting returns the current state of the SSP decoder window: Yes if its on or No if its off.

Setup Information

TOPIC	ITEM	DEFAULT	DESCRIPTION
[RTUServer]	MasterID	XCONNECT	SSP Unit ID of your Base Station
	ListenID		List of Unit ID's to listen in and parse messages addressed to, but not respond to. This allows one base station to capture messages being sent to one or more base stations without having to poll for the data itself. This is not a perfect method of data collection since the listening base station may drop a message, which the master base station receives ok - and hence does not perform a retry. ex: Master1,Master2,Master3
	ComPort		List of defined com ports. ex: COM1,COM2,COM3
	PollGroups		List of defined poll groups. ex: 1,2,3,4,6
	Run		List of programs to run on start up before the XC Rtu starts to collect data. ex: EXCEL.EXE
	AutoPoll	0.000	Determines how often (in seconds) GETTAG controls will be issued to RTUs in order to update hot-linked data points in the AUTO topic. A value of 0.0 disables automatic polling.
	AutoUpdate	0.000	Determines how long the XC Rtu will wait before issuing SENDTAG control to update RTUs after a data point in the AUTO topic has been changed. A value of 0.0 disables automatic updating.
	RePollLimit	00:01:00	When XC Rtu polls for time tag data from a unit and the response contains decoded sensors which are empty then XC Rtu will only update the time of LastPoll if the time stamp is older than RePollLimit. This handles two conflicting situations ... 1) The RTU may still be collecting data which was empty and a repolling later will collect it, and 2) The entry may have been skipped and all the re-polling in the world may never retrieve a value. Since an RTU usually logs data in "real time" and rarely logs data back in time, RePollLimit should be set to at least the longest logging interval in the system. RePollLimit has a second use in

			conjunction with the TimeLimit option under [POLLGROUP:n]. When a TimeLimited poll receives no data, the LastPoll date will be automatically advanced anyway until it reaches RePollLimit minutes before the current time.
	ListenUpdate	No	Causes XC Rtu to store the Last Poll date for time tag data that was received thru a ListenID.

TOPIC	ITEM	DEFAULT	DESCRIPTION
[COM*n*] where *n*=1,2,3...9	Type	None	Type of device connected to com port, one of: None, Direct, Radio, Modem, or Hub. Radio implies a Sutron Base Station LOS Radio Modem or compatible. Modem implies a Hayes compatible AT-command set telephone modem. Direct implies a direct hookup to an RTU. Hub implies a message concentrator is connected to the port (similar to a direct connect, except logging-in and software handshake are not used). A Hub collects messages from a single port and re-transmits them out multiple ports at the same time. The Hub then collects the responses and ships them back to the base station. This allows for high speed parallel communications to multiple stations at once. See "NoWait" option below.
	Baud	1200	Baud rate of device
	DialRetries	0	Number of times to re-dial a number if cannot get thru on the first attempt.
	SSPRetries	2	Number of times to retry an SSP message if a response is not received.
	NumberOfRings	2	Number of rings to answer the phone.
	CarrierDelay	1.0	Number of seconds to raise carrier before transmitting data on a radio.
	NullDelay	0.0	Number of seconds of null characters to send before sending data. Can be used to allow an RTU to wake-up when using modems without CD.
	UnKeyDelay	0.0	Number of seconds to hold carrier after data has been transmitted.
	ReplyDelay	0.0	Number of seconds to pause before sending an SSP message. This value is sometimes needed to allow a radio repeater time to settle before

			responding to a request.
	ConnectTimeout	30.0	Number of seconds of inactivity to hold a manually requested connection before hanging up.
	NoConnectTimeout	60.0	Number of seconds to wait for the device to become available to attempt a connection.
	DialTimeout	60.0	Number of seconds to wait for a phone number to be dialed and the modems to connect.
	AutoTimeout	1.0	Number of seconds of inactivity to hold an automatic connection before hanging up.
	AckDelay	10.0	Number of seconds to wait for an acknowledgement after a message has been sent.
	ResponseTimeout	30.0	Number of seconds to wait for a response once a request has been made, should include slack time for the field unit to timeout and re-transmit a dropped message.
	TouchTone	Yes	Enables tone dialing with a modem, otherwise "No" would cause the modem to pulse dial.
	Handshake	Yes	Enables software handshaking and synchronization before a message is sent.
	HWHandshake	Yes when Type is Modem	Enables RTS/CTS hardware handshaking on the serial port. This is often necessary for high speed telephone modems.
	ModemInitString		String to be sent to the modem when the modem is being initialized, do not include the AT. ex: M2L3 would enable the modem speaker all the time and set the volume to the highest level.
	HangUpDelay	0	Number of seconds to delay after the telephone modem hangs up/disconnects.
	ModemHangupString		String to be sent to the modem when the modem is being hung-up, do not include the AT. ex: M1L2 would put the modem speaker back to normal.
	ControlLine		DEFAULT Sets the serial port control

			line to use for detecting a connection or detecting carrier, should be one of: DEFAULT, NONE, CTS, DSR, or CD. The DEFAULT setting uses NONE for a direct connect or CD for a modem or radio.
	NoWait	No	Setting NoWait to Yes will allow SSP messages to be sent to the device while other messages are still being processed. However, simultaneous messages to the same station will still remain in the message queue to prevent confusing an RTU which can only handle one conversation at a time. This option is meant to be used when the com port is connected to a Hub, and allows the Hub to operate on more than one message at a time. If NoWait is enabled on a modem or radio link, collisions will occur resulting in decreased throughput and transmission timeouts.
	AutoBaud	Yes	Causes the baud rate of the communications port to be switched to match the "CONNECT xxxx" message returned by a modem after dialing and connecting. Some modems (like older Microcom models) report one rate but stay at the original rate and should have this option set to No.
	ConnectDelay	0.0	Number of seconds to pause after connecting to the device before sending characters. Some combinations of modems & modem protocols may require a few seconds of delay before characters are sent after dialing-out and connecting. When characters are sent to soon sometimes all following characters are dropped and on some modems a disconnect may occur.
	MaxParsers	16	Controls the maximum number of messages which will be issued at the same time when using NoWait=Yes on a Hub based system. Performance bogs down when too many parsers are running monitoring the performance of multiple messages. This options allows you to "throttle" back the system so too many messages are not dispatched. As a rule of thumb, a setting of 8 is appropriate for a 386/33 level machine, 16 is good for a 486/33 machine. You can tell if the Base Station is overloaded by monitoring whether it responds to incoming messages within a few seconds.
	TXPort	600	For **Virtual ports** using IP address, port number of socket to transmit SSP

			messages.
	RXPort	500	For **Virtual ports** using IP address, port number of socket to receive SSP messages.
	Protoco	UDP	For **Virtual ports** using IP address, transport protocol to use.

TOPIC	ITEM	DEFAULT	DESCRIPTION
[POLLGROUP:*n*] where *n*=1,2,3,4...	Enable	No	Set to "Yes" to enable polling for this group.
	BaseTime	00:00:00	Offset from the top of the hour to start polling data from the group.
	Interval	00:00:00	How often to poll the group.
	Stations		List of stations to poll and/or send time syncs to in this group.
	Clients		List of applications which must be running before this group will be polled. The names must match the window title of the application which is usually not the same as the executable file name or the DDE server name. ex: XC Export
	TimeSync	No	Set to "Yes" to cause a SETCLOCK message and a poll request to be sent to each station. Set to "Only" to to cause only the SETCLOCK message to be sent. A SETCLOCK message will correct the RTU's clock with the current time in the master station PCs. 8200's will update right away, while 9000's must perform a timesync operation before they will update.
	PollType	None	When set to something other than None, this option will override the default PollType assigned to each station. For more information on valid values see PollType under the [Station] topic below.

	TimeLimit	00:00:00	If set to a value other than 00:00:00, time limit will limit the amount of data that will be collected by a poll by date. This is very useful in preventing a station which has gotten very far behind in polling from hogging the communications port. ex: 08:00:00 would limit date polls to collecting no more than 8 hours of data at one time. The actual time required to collect 8 hours worth of data will vary depending on the communications rate and the logging interval of the station. Sometimes when a field station has been down it may contain no data for the poll'ed time period. XC Rtu will automatically advance the date of LastPoll in this case until the polling catches up with real time. The time that determines when XC Rtu stops automatically advancing is determined by RePollLimit in the [RTUServer] topic.
	Control		Allows the actual control performed on each station in the poll group to be defined explicitly, so that pollgroups which perform actions other than archive or current data polls can be performed. Any control topic item can be used. Use "%p" to indicate where the station name should appear in the control if needed. ex: GETSETUP,%p,* would cause the pollgroup to perform the GETSETUP control on each station in the stations list.

TOPIC	ITEM	DEFAULT	DESCRIPTION
[STATION:*station*] where *station* is the name of the station being defined.	Enable	No	Set to "Yes" to enable station for decoding.
	SiteType		Type of station being defined, can be one of: 8200, 8210, 9000, 9210, XPERT.
	PollType		Determines how the unit will be polled, can be one of the following: DATE, NEWEST, CURRENT, NOWAIT. DATE collects archived data starting after the LastPoll date up until now. NEWEST collects archived data not

			previously sent as remembered by the RTU. CURRENT collects last measured values only. NOWAIT peforms a current data poll, but does not wait for the response. This is meant to be used when polling through a Hub which will buffer the polls send them out to the field stations simultaneously, and then send back the responses as they arrive. Do not use the NOWAIT option on anything other than a Hub, or massive message congestion will occur.
	ComPort		List of com ports which the station can be reached through in the order they should be used. ex: COM1,COM2 would try COM1 first and if it failed would then try COM2.
	RadioPath		List of SSP paths for reaching the station, one for each ComPort defined. In the simplest form a SSP path is just the UnitID as defined above, in a more complex system it can include store and forward repeater paths. For instance in a marginal radio system we may first try to reach RTU01 directly, but if that fails, we may try to route the message through RTU02 which is closer first and have it relay the message to RTU01 as shown below: ex: RTU01,RTU02_RTU01 the "_" underscore characters seperates stations in a repeater path.
	AckDelay	defaults to the value specified under the [COMn] topic.	List of acknowledgement timeout values in seconds for each RadioPath as defined above. Each AckDelay value defines how long the base station will wait for a response to a request before timing out and retrying. The longer the path to the RTU or the slower the link the longer delay you will need. Can be as short as 2 seconds for a direct connect, 5 to 15 seconds for a radio or modem, and a multiple of that for each store and forward repeater involved. ex: 10,15
	ResponseTimeout	defaults to the value specified under the [COMn]	List of response timeout values in seconds for each RadioPath as defined above. Each ResponseTimeout value defines how long the base station will wait for a response to a request before timing out and retrying. The

		topic.	longer the path to the RTU or the slower the link the longer delay you will need. The ResponseTimeout is used in cases where the RTU is acting like a master in the respect that the RTU is sending data and expecting an ack back from the PC. The RTU will automatically retry according to the Ack Delay and SSP Retries set in the RTU. The ResponseTimeout must allow time for the RTU to attempt all of its retries, which can be determined by: AckDelay * (SSPRetries + 1). Can be as short as 15 seconds for a direct connect, 30 to 60 seconds for a radio or modem, and a multiple of that for each store and forward repeater involved. ex: 30,60
	UnitID		The Unit ID or Name of the station as defined at the station - can be different than the station name. ex: RTU01
	SetupFile		Default filename for storing and retrieving setups for this station. ex: C:\BASE\SETUPS\RTU01.SET
	PhoneNumber PhoneNumber2		The primary and secondary phone numbers to dial to reach the station if it is connected to a modem. You can include any extra characters which your modem understands. For instance adding a comma to the phone number will cause a short delay in dialing. ex: 1-800-555-1212
	UserName		This is the user name you wish the base station to use when logging in, for 8200's the UserName is the same as the UnitID when logging in is enabled. We recommend setting up user names and passwords in your RTUs, not only for security reasons, but because it will help the base station login more reliably.
	Password		This is the password the base station will need to use to login, for 8200's the Password will be blank if an external modem is used; otherwise it is defined in the modem setup menu.
	LoginReqd	No	Tells the base station whether it will need to login with a UserName and Password to access the station.

	AutoProto	No	Set this to "Yes" when a modem or direct connection to the RTU is always in SSP protocol mode and login is not required.
	QuickProto	Yes	All recent 8200's and 9000's support this feature which allows the base station to switch directly to SSP protocol mode when logging in by placing a "/C" after the user name. This allows the base station to avoid having to traverse the RTU menus and select the "A"-Application Menu for entering protocol mode.
	Sensors		List of sensor names defined for the station, does not have to match the names used in the RTU. ex: Air Temp,WindSpeed,WindDir,Battery
	LastPoll	12/31/1984 00:00:00	Stores the date and time of the last poll as used when polling the station for archived data by date. You can move the date back to cause old data to be repolled, or set it ahead to cause data to be skipped.
	Concentrate	No	When set to Yes, tells XC Rtu that the station is a concentrator of other station's data in its log. If the first item in a log record from a concentrator is the unit id of a station in the setup then the log record will be interpreted as if it had come from that unit. The record id will be switched to 0, which is the default for 8200's. If the first item does not match any existing units, then it will be processed normally as a record id belonging to the station. Currently the only device which supports concentration is the 9000 RTU. With special SDL software 9000's can poll 8200's for current data and concentrate this information in to the 9000's log. 9000's can only concentrate current data, because they do not support polling for time tag data. A better method of concentration is to use multiple base stations, and share data with Poll Data. This is not always possible if the concentrator needs to run off batteries or at remote locations.
	HoursAhead	0	Set this to the number of hours the RTU's time is ahead of the local time at the base station. Use a negative number to specify the number of hours the RTU is behind the base

			station. This amount will be added to outgoing and subtracted from incoming timestamp's in messages processed by XC Rtu. For GOES messages there is a single GMT offset and HoursAhead is not used. ex: (Base station is EST and RTU is PST) HoursAhead=-3 ex: (Base station is EST and RTU is GMT) HoursAhead=6
	RDIBaseTime	00:00:00	Used for **EDACs systems only**, offset from the top of the hour when the 8210/XPERT will turn on the EDACs radio.
	RDIInterval	00:00:00	Used for **EDACs systems only**, how often the 8210/XPERT will turn on the EDACs radio.
	RDIDwell	00:00:00	Used for **EDACs systems only**, how long the 8210/XPERT will leave the EDACS radio turned on.

TOPIC	ITEM	DEFAULT	DESCRIPTION
[UNITS]	unitid=station		Each station defined in the setup must have an entry in the UNITS topic to allow translation from a UnitID to a Station Name. The UnitID is the name of the RTU as set in the RTU, the Station Name is the name of the RTU as set in the base station. If they are the same name, then no entry is necessary. ex: RTU01=Potomac River Stage

TOPIC	ITEM	DEFAULT	DESCRIPTION
[TAGINFO:station]	tagname=sensor		Each sensor defined in the a station setup must have an entry in the TAGINFO:station topic to allow translation from a tag name to a sensor name. The tag name is the name of the parameter as set in the RTU, the sensor name is the name as set in the base station. If they are the same name, then no entry is necessary. ex: Analog1=Air Temperature

TOPIC	ITEM	DEFAULT	DESCRIPTION
[LOGINFO:station]	recordid=sensor1 ,sensor2,sensor3 ...		Each sensor you wish to collect archived data for should be listed in a LOGINFO:station topic. The record id identifies the type of log record which are transmitted by the station (always "0" for an 8200), and contains a list of sensor names in the order they appear in the log. For the 9000 the record id is the first item in a log entry and must be either a unique character or string. For instance if you had an 8200 named MET5 which was logging 3 parameters: Air Temp, Wind Speed, and Wind Dir in that order, you would need the following entry: [LOGINGO:MET5] 0=Air Temp,Wind Speed,Wind Dir

TOPIC	ITEM	DEFAULT	DESCRIPTION
[SENSOR:station:sensor]	Enable	No	Set to "Yes" to enable sensor for decoding.
	TagName		This is the name of the sensor as defined inside the RTU. An optional ":value#" can be placed after the tag name to select a different tag value then the default of 1. This information is used for processing send tag and get tag requests. For the 9000 the following values are defined (by convention): :1 - Most recent data for a tag :2 - Alarm Status for a tag :3-n - User defined values For the 8200 the following values are defined: :1 - Most recent data for a sensor :2 - Alarm Status for a sensor :3 - Slope of the sensor :4 - Offset of the sensor :5 - Right digits for the sensor :6 - Measurment counter note: XC Rtu swaps values 2 & 6 as received from an 8200 so that the alarm status will be stored in value 2

			for consistency with the 9000. ex: WindSpeed1 for the current reading of WindSpeed ex: WindSpeed1:2 for the alarm status of WindSpeed
	Equation		Contains an equation expressed in terms of "X" which will be applied to incoming sensor data before it is stored. The expression is not case sensitive. Applies to sensor data from current data polls, archived data polls, and get tag requests. Uses the syntax of the BASIC language. The following functions are available: ARCTAN, COS, EXP, LOG, LN, SIN, SQRT, TAN, ABS, ROUND, TRUNC, INT and RND. INT(X) returns the integer part of X where as TRUNC and ROUND truncate and round the value. RND(X) returns a random number from 0 to X-1. To raise a number to a power use the "^" (X^2 would square X). Comparison can be performed using <, >, <=, >=, <>, and =. The result of a comparison is 1 for true or 0 for false. The following bitwise boolean operators are supported: AND, OR, XOR, SHL, and SHR. The last two are shift-left and shift-right. For instance (X SHL 4) would shift X left by 4 bits. AND & OR can also be used in logical expressions. For instance "(X>100) OR (X<50)" would result in 1 if X is above 100 or below 50 - otherwise 0. The NOT operator is logical not bitwise. This means that NOT 0 is 1 and NOT 1 is 0. Also, the NOT of any non-zero number is 0. An additional function called TABLE exists which can perform a table lookup using an external file of data points. The syntax for a table lookup is: TABLE(tablefile,expression) where tablefile is the filename that contains the data points, and expression is any valid equation or just "X". See the section on Lookup Table Format after this section for information on how to build a lookup

			table file. You can also check to see whether a real time value is defined with "?". For instance ?TEST has a value of 1 if test is defined and 0 if not. Equation (examples) ex: (X AND 128) <> 0 results in a 1 if bit 7 in X is set or 0 if bit 7 is clear. The bit mask 128 is 2^7. This assumes bit 0 is the least signicant bit. In general, the bit mask for any bit N is 2^N. Equations can also contain references to real time values: ex: (X+RTU01.AirTemp)/2 would average an incoming air temperature with its most recent value. ex: X*9/5+32 would convert readings from oC to oF. ex: SIN(X)+COS(X)+X^3+LOG(X) example with multiple functions. ex: TABLE(MYTAB,X) translates X based on a lookup table contained in MYTAB.TBL in the directory specified by TablePath. ex: (X>1000)*1000 + (X<=1000)*X would limit the value so that it could never be greater than 1000.
	RightDigits	0	Number of decimal digits to be used when displaying the sensor value.
	Raw		If non-blank this is a list of additional sensors to process the raw data from this sensor through. Besides allowing the benefit of additional processing, Raw can also be used to store the same data in to multiple data files. ex: TempF,TempK would cause raw data to be processed by sensors named TempF, and TempK.
	RoundOffTime	00:00:00	Determines the interval at which time stamps are rounded off to. For instance with the round off time is 15 minute : 12:48 would be rounded to 12:45 and 12:53 would be rounded to 13:00.

DDE APPLICATION NAME: XCRTU

DDE TOPIC NAME: DATASERVE

The DataServe topic is only valid when using **PCBASE2 binary data files** as your data storage.

ITEM	A	P	R	E	DATA	DESCRIPTION
POINT,station.sensor,date			x		number	Retrieves or stores a number to/from for a given station and sensor at a particular date and time. The data file the sensor is stored in is determined automatically from the station and sensor. ex: POINT,Site5.AirTemp,Now would store or retrieve the current value stored for sensor AirTemp from station Site5.
BLOCK,datafile,startdate, stopdate[,interval][,sensorlist]			x		r1c1,r1c2<cr><lf> r2c1,r2c2<cr><lf> r3c1,r3c2<cr><lf>	Requests data from a data file from start date to stop date and returns a table of comma (if csv format requested) or tab (if text format requested) seperarated columns and <cr> <lf> delimeted rows of ascii numbers. The interval can be specified to limit the request to only certain time intervals in the data file, and the sensor list can be specified to request data for only certain sensors in the data file. If the sensor list is specified then the interval must also be specified - although an interval of 00:00:00 can be used to select all data. ex: BLOCK,METDAT,Yesterday,Now,01:00:00,Site5.AirTemp would request hourly air temperature data for station Site5 from the data file METDAT from yesterday until now.
SENSOR,datafile			x		station.sensor1,station.sensor2, station.sensor(n)	Returns a list of all sensors in a data file including the station the sensors are from.
INFO,datafile			x		"desc",startdate,enddate,basetime, interval,use24hr, numsensors	Returns information about a data file including: the data files description, the start date of the file, the stop date and time, the base time, the interval, the Use24Hour flag, and the number of sensors count.
FILES			x		filelist	Returns a list of data file names currently supported by the data server.
CREATE,datafile,"desc",startda			x		1 - OK	Attempts to create a data file and

ITEM	A	P	R	E	DATA	DESCRIPTION
te,basetime,interval, use24hr,station.sensor1,rd1,--station.sensor2,rd2,...					2 - FAILED 3 - Already Exists	returns either 1, 2, 3 to indicate the status as listed to the left. You must pass the name of the data file to create, a description of the file (inside double quotes), a start date for the file (no time - assumed 00:00:00), a base time (normally 00:00:00 to sync timestamps with the top of the hour), an interval which should be greater than or equal to the smallest time duration you expect in your data stream between data points, the use24hr flag which determines whether days start at 00:00:00 "No" or end at 24:00:00 "Yes" , and finally a list of station.sensors and right digits you wish to store in the data file. The number of right digits for each sensor is used to determine the number of decimal digits used to display data from that sensor. The use24hr flag has an addition effect in that when set to "Yes" time stamps are rounded up to the next higher time stamp, and when set to "No" time stamps are rounded down the the next lower time stamp. For instance if a data file stores hourly data, a piece of data with a time stamp of 1:15 would be stored in the 2:00 slot if the use24hr flag is "Yes" or in the 1:00 slot if the use24hr flag is "No". This is due to the difference in what data belongs in which day according to the different schemes and how the RTUs time stamp their data. ex: CREATE,METDAT,"Meteorological Data",This Year,00:00:00,01:00:00,No,Site5.Air Temp,2,Site5.BaroPress,1 would create a data file called METDAT with a description of "Meterological Data", would allocate room for data starting at the beggining of the current year, with a base time of 00:00:00, a data interval of 1 hour, use24hr off, and with sensors from station Site5 called AirTemp and BaroPress with right digits of 2 and 1.
DELETE,datafile			x		1 - OK 2 - Does not exist	Deletes a data file and returns a result code.
SPLIT,datafile,date			x		1 - OK	Splits a data file at the specified date in to two files - the original file will contain data on and after the date,

ITEM	A	P	R	E	DATA	DESCRIPTION
					2 - FAILED	and a new file with the extension ".OLD" (instead of ".DAT") will be created which contains data before the date. This is a useful function for archiving old data, and keeping the data files trimmed down. ex: SPLIT,METDATA,This Month-6/0/0 would keep the last 6 months worth of data plus the current month in the data file and place the rest in "METDATA.OLD".
ADJUST,datafile,date			x		1 - OK 2 - FAILED	Adjusts the start date of a data file to the new data as passed. ex: ADJUST,METDATA,This Month would throw away all data in the data file METDATA before the start of this month, or if the data file previously started after the start of the month will allow data to be stored earlier.
CHANGE,datafile,"desc",startdate,basetime, interval,use24hr,station.sensor1,rd1,--station.sensor2,rd2,...			x		1 - OK 2 - FAILED	Attempts to change or create a data file and returns either 1, or 2 to indicate the status as listed to the left. Documentation for each parameter is discussed on the CREATE command. Change will retain all sensors which have the same name in the new file as in the old. If the interval is increased then some data will be lost. If the interval is decreased then existing data will be replicated to fill in the holes. Changing the use24hr flag may result in data being offset by one time interval.
ACTIVATE,datafile			x		1 - OK 2 - FAILED 3 - Already Exists	Adds an existing data file in the directory specified by the data file path to the Data Server. The filename will be automatically added to the setup so that the file will be opened the next time Data Server is started. This can be used to add a file which has been copied from another machine.
DEACTIVATE,datafile			x		1 - OK 2 - Does not exist	Removes a data file from control of the Data Server, but does not delete the file or any of its data. The file can be added back later with the activate command.
EXIST,station.sensor			x		data file name	

Operations: A-Advise/UnAdvise, P-Poke, R-Request, E-Execute

XC Mux

XC Mux

Introduction

The XC Mux program polls for data from the Sutron receive site Mux hardware and outputs NES/DAPS format files. Alternatively the captured output of the Mux h/w can also be read in from disk by the Mux program and converted to DAPS format. The DAPS files may then be processed by the XC Decode program so that the actual sensor data can be extracted. The XC Mux software displays a status screen, the contents of the last message processed, the current output file, a button for enabling the display of all data in the status screen (usually only error and status messages are display), and a button for displaying the raw communications with the hardware. Menu selections allow captured mux files to read-in and processed, resetting the mux h/w, and setting the clock in the mux (the mux clock is automatically checked once an hour). Menu options also exist for copying the following information to the clipboard and for creating a data link for capturing the information in logger for example: current status screen, current NES message, current mux message, last mux message with errors. The software also supports the installation of a true time clock connected to the mux. When a true time clock is available, the software will automatically update the PC's time to the true time; when not available the mux time is corrected to match the PC's time. The mux software can also be shut down for short periods of time with out missing messages, since the mux hardware is capable of buffering several messages. All messages received by the software are output, no filtering is performed.

DDE Information

DDE APPLICATION NAME: XCMUX

DDE TOPIC NAME: DATA

ITEM	A	P	R	E	DATA	DESCRIPTION
$BAD_DATA	x	x	x		text	Line received from the Mux which could not be interpreted or contained only partial data.
$MUX_DATA	x	x	x		text	Line of data received from the Mux in the Mux's format. Although the mux sends multiple lines for one message, they have all been concatenated for this DDE point.
$DEMOD	x	x	x		number	Number of the demod that the last message was received on.
LAST_DATE	x	x	x		mm/dd/yyyy hh:mm:ss	Time and date of the last message
$CHANNEL	x	x	x		number	Channel number of the last message.
$SATID	x	x	x		8-digit-hex-number	Satellite (Platform) ID of the last message.
$DATA_QUALITY	x	x	x		text	Data quality of last message: Good, Short Msg, Bad Chan, Parity Err, or ?.
$NES_DATA	x	x	x		text	Last message received converted to NES data format. See appendix A for more

ITEM	A	P	R	E	DATA	DESCRIPTION
						information on this format.
$FREQUENCY_ERROR	x	x	x		integer	Frequency error of last message (Hz).
$SIGNAL_STRENGTH	x	x	x		integer	Signal strength of last message (dB).
$MODULATION	x	x	x		integer	Modulation or phase deviation of last message (degrees).
$SIGNAL_NOISE	x	x	x		integer	Signal to noise ratio of last message.
$PARITY	x	x	x		integer	Number of parity errors in last message.
$OUTPUT	x	x	x		text	Allows manual commands to be sent to the Mux. When non-blank the software will send this string to the Mux. After being sent the field will be set back to blank.
$INPUT	x	x	x		text	Allows manual queries to be sent to the Mux. When non-blank the software will send this string up to 3 times to the Mux, and wait for a response. After the command is complete the field will be set back to blank. The item $RESULT will be set to the string returned from the Mux.
$RESULT	x	x	x		text	Contains the result of the last manual query or blank if the last query failed.
demod	x	x	x		channel,sattelite	demod can take on values ranging from '0' to '9' and 'A' to 'F'. Poking a demod will change the channel and satellite stamped on messages received from that demod. The demod is not re-programmed, that must be done by poking to $OUTPUT. The value will be set to the default contained in the setup initially or after a reset.
$RESET	x	x	x		list-of-commands	List of strings to be sent to the Mux when reset. Poking will cause the string(s) to be sent to the Mux. On startup the value is set according to the Reset option under the [Mux] setup section. This allows the Mux (and Demods) to be initialized when the s/w is started to a known state, and then changed

ITEM	A	P	R	E	DATA	DESCRIPTION
						on the fly to another state. If the user performs a reset mux command, then the Mux (and Demods) will be automatically re-initialized to the new setting.
STATUS	x	x	x		lines-of-text	Returns the contents of the Mux status display.

Operations: A-Advise/UnAdvise, P-Poke, R-Request, E-Execute

Setup Information

TOPIC	ITEM	DEFAULT	DESCRIPTION
[MUX]	Run		List of programs to run when the Mux program is started and before it starts to process data.
	ComPort		Com port that the mux is connect to the PC on, COM1 thru COM9.
	Baud	9600	Baud rate of com port connected to DRGS. 9600 is default for 100-Baud DDRGS, 19200 is default for HDR DDRGS. **NOT USED FOR DSR DRGS.**
	DDRGSType	100-BPS DDRGS	Type of DRGS XC Mux is communicating with. AVailable DRGS types are: '100-BPS DDRGS', 'HDR DDRGS', 'DSR DRGS'
	RawPath	same as used in the RawPath entry in [General]	Directory to store raw NES files in to as messages are received from the mux.
	RawFile	yyyymmdd.RAW	Determines the file to append the raw NES files in to as messages are received from the mux. The lower case letters "yy", "yyyy", "mm", "dd" "hh", "mi" can be included in the name to instruct the software to create separate minute, hourly, daily, monthly, or yearly files. Optionally "#nn" can be added to specify that a new file should be created every nn minutes (when using "mi"), hours (when using "hh"), or days (all other). "#nn" can appear anywhere in the string, where nn must be two digits with leading zero (if necessary). The unit of time is determined by the smallest time specified. The dating of the raw files is important for a couple of reasons, easy backup,

TOPIC	ITEM	DEFAULT	DESCRIPTION
			purging, and distribution of raw data; and re-processing of raw data thru the decoder. If the files are allowed to grow too large these processes will become more difficult or require excessive time to perform. The default setting yyyymmdd.RAW creates a separate file for each and every day. ex: MYDATdd.RAW would create the file MYDAT01.RAW on the first day of the month, MYDAT02.RAW on the second day, and so on until the next month when MYDAT01 would be used again. If you did not delete MYDAT01 then it would be appended to and continue to grow. This method might be desirable if you wanted to make sure that no more than 31 files would be created. ex: NESmm.RAW would create the file NES01.RAW to contain all the data for January, NES02.RAW to contain all the data for February, and so on until the next year. This method might be preferable if your system is not storing high quantities of data. ex: yymmddhh.RAW#02 would create a new output file every 2 hours. If the "#02" was left off the, end then a new file would be created every hour. ex: mmddhhmi.RAW#15 would create a new output file every 15 minutes. Since DOS file names are limited to 8 characters the year had to be left out to make room for minutes.
	Reset		List of commands to be sent to the Mux (or Demods) upon reset. Also sets the initial value for the DDE field $RESET. ex: P5MC,P5C1231 would set demod 5 to computer mode, and set its channel to 123. **NOT USED WITH DSR DRGS TYPE.**
	Listen	No	When set to Yes, Xc Mux will act as a listen and send no commands to the DRGS. It is expected there is another PC also running XC Mux that is communicating with the DRGS.

TOPIC	ITEM	DEFAULT	DESCRIPTION

XC Decode

XC Decode

Introduction

XC Decode is a program which automatically or manually reads in ASCII files of NESDIS formatted or Sutron extended raw satellite data messages and extracts sensor values. As the values are decoded they are made available in XC Decode's database for realtime access or logging to disk. Setting up the XC Decode includes describing which stations map to which satellite ID's, how to decode a particular channel, and how to locate and extract sensor values from a message. XC Decode automatically stores the position at which it completed processing a file in to a 10 byte file of the same name but with an extension of ".POS". The ".POS" file is used so that the XC Decode can detect whether futher data has been appended to a message file so it can continue where it left off. To cause the XC Decode to re-process a file from the beginning you must delete the related ".POS" file first.

XC Decode has one DDe topic: Data.

DDE Information

DDE APPLICATION NAME: XCDECODE

DDE TOPIC NAME: DATA

ITEM	A	P	R	E	DATA	DESCRIPTION
station.sensor	x	x	x		a number, an alarm string, a character, or a boolean "Yes" or "No" depending on the type of sensor.	Most recent value of a particular sensor from a station. Data decoded from both selftimed or random transmission are considered most recent as long as the time the data was received is later than the previous data.
station.sensor :n	x	x	x		same as station.sensor	Same as station.sensor but allows for multiple values under the same sensor name.
station.sensor.$LASTUPDATE	x	x	x		mm/dd/yyyy hh:mm:ss	Time and date of most recent data received for a sensor from a station. This field is used as the basis of comparison to determine whether a newly decoded piece of data is more recent than a previous value.
station.$LASTUPDATE	x	x	x		mm/dd/yyyy hh:mm:ss	Time and date of most recent data received for a station.
station.$TIMETAG	x	x	x		mm/dd/yyyy hh:mm:ss,sensor,data , sensor,data, ...	Last time tag data received from a station consisting of a date and time stamp followed by pairs of sensor names and data values collected at that time.
station.$decodetype	x	x	x		mm/dd/yyyy hh:mm:ss,sensor,data , sensor,data, ...	Same format as station.$TIMETAG except only contains data received from the station which was decoded

ITEM	A	P	R	E	DATA	DESCRIPTION
						using decodetype. For instance station.$GOESRANDOM would contain the last random mesage from "station".
$TIMETAG	x	x	x		mm/dd/yyyy hh:mm:ss,station.sensor,data, station.sensor,data, ...	Last time tag data decoded from any station consisting of a date and time stamp followed by pairs of station.sensor names and data values.
$ *decodetype*	x	x	x		mm/dd/yyyy hh:mm:ss,station.sensor,data, station.sensor,data, ...	Same format as $TIMETAG except only contains data which was decoded using decodetype. For instance $GOESRANDOM would contain the last random message from any station.
$REALTIME	x	x	x		mm/dd/yyyy hh:mm:ss,station.sensor,data, station.sensor,data, ...	Most recent data decoded from any station consisting of a date and time stamp followed by pairs of station.sensor names and data values. Data is considered most recent if the time stamp of the data is later than previous data received for a sensor.
$QUALITY	x		x		mm/dd/yyyy hh:mm:ss,station.sensor,data, station.sensor,data, ...	All XC Decode quality messages are stored in this item, allowing these messages to be easily logged with Logger or in the GOESQC data table.
$STATUS	x		x		text	
$ERROR	x		x		text	
station.$CHANNEL	x	x	x		number	Satellite channel number of last message decoded from a station.
station.$FAILURE_CODE	x	x	x		71 = G - Good 63 = ? - Parity Errors 87 = W - Wrong Chan 68 = D - Dup Chan 65 = A - Addr Error 66 = B - Bad Address 84 = T - Time error 85 = U - Unexpected 77 = M - Missing Msg 73 = I - Invalid	ASCIIi numeric value of the failure code of the last message decoded from a station.

ITEM	A	P	R	E	DATA	DESCRIPTION
					Address 78 = N - Incomp PDT 81 = Q - Bad Qual 67 = C - Compare Err 83 = S - Short Msg	
station.$SIGNAL_STRENGTH	x	x	x		32-57	Signal strength of the last message decoded from a station in dB, normally a message will be in the range 44 to 48.
station.$FREQUENCY_ERROR	x	x	x		-500 to +500 Hz	Frequency error of the last message decoded from a station in 50 Hz increments. +/- 250 Hz is the normal range.
station.$MODULATION	x	x	x		49,60,71 degrees	Phase modulation of the last message decoded from a station. 60 degrees is normal, a value of 49 or 71 degrees may indicate a problem.
station.$DATA_QUALITY	x	x	x		0 - Normal 1 - Fair 2 - Poor	Modulation quality of the last message decoded from a station. 0 is normal, 1 or 2 indicate that there may be some errors in the transmission.
station.$GOES_SPACECRAFT	x	x	x		E - GOES East W - GOES West C - GOES Central A - Argos	Indicates the location of the spacecraft which received the last message decoded from a station.
station.$ORIGINAL_LENGTH	x	x	x		number	Number of data bytes in the last message decoded from a station according to the length stored in the message.
station.$ACTUAL_LENGTH	x	x	x		number	Number of data bytes in the last message decoded from a station actually contained in the message.
station.$PARITY	x	x	x		number	Number of parity errors detected in the last message decoded from a station.
station.$SATID	x	x	x		number	Satellite ID of the last message decoded from a station.
DECODE filename				x	n/a	Executing the decode command will cause the passed filename to be decoded. The filename can

ITEM	A	P	R	E	DATA	DESCRIPTION
						include a path and/or wildcard characters. ex: DECODE C:\RAW*.RAW would decode all non-decoded data in every file with an extension of "RAW" in the directory "C:\RAW".
STATUS			x		lines-of-text	Returns the contents of the XC Decode status display.
RROR			x		lines-of-text	Returns the contents of the XC Decode error display.

Operations: A-Advise/UnAdvise, P-Poke, R-Request, E-Execute

Setup Information

TOPIC	ITEM	DEFAULT	DESCRIPTION
[Decoder]	Run		List of programs to run on start up before the XC Decode starts to process data. ex: EXCEL.EXE
	Clients		List of applications which must be running before files will be decoded. The names must match the window title of the application, which is usually similiar, but not the same as the executable file name or the DDE server name. ex: Microsoft Excel
	AutoPath		Any files which match the AutoPath will checked for new data and/or decoded automatically based on the AutoInterval. The AutoPath may contain more that one directory to search. ex: C:\MESSAGES*.RAW,C:\RAW*.RAW Would automatically decode files ending in the extension ".RAW" contained in the directories MESSAGES and RAW on drive C:.
	AutoInterval	00:01:00	Determines how often files specified by the AutoPath are check and/or processed.

TOPIC	ITEM	DEFAULT	DESCRIPTION
[STATION:station]	Enable	No	Enables decoding of sensors from this

			station.
	SiteType		Defines the type of station being decoded, so that smart decoding if available for the unit can be used. The following sitetypes have been defined: 8200 - Sutron 8200 Data Logger 8210 - Sutron 8210 RTU 9000 - Sutron 9000 RTU 9210 - Sutron 9210 (XLITE) RTU XPERT - Sutron XPERT RTU
	SatID		8 digit hexidecimal number representing the Satellite ID of the station.
	Sensors		List of sensors used by this station and defined in the setup.

TOPIC	ITEM	DEFAULT	DESCRIPTION
[SATID]	satid=station		[SATID] is a translation table used to translate Satellite ID's to station names. All satellite ID's to be decoded must be defined and have a matching station name. example: [SATID] 1234ABCD=HerndonCreek FA28478E=RestonDam

TOPIC	ITEM	DEFAULT	DESCRIPTION
[CHANNEL]	channel=goestype	<=100 = GoesTimed >100 = GoesRandom	[CHANNEL] is a translation table used to determine from the channel a message was received on what type of decode processing should be used. For a GOES transmission a station usually needs to be decoded differently when the message is from a self timed channel versus a random alarm channel. Random alarm message are often much shorter. See the definition of [decodetype:station:sensor] to see how decoding a sensor is handled. Be sure to use 3 digits with leading zeros when defining the channel. example:

TOPIC	ITEM	DEFAULT	DESCRIPTION
			[CHANNEL] 055=GoesTimed 167=GoesRandom 500=Argos

TOPIC	ITEM	DEFAULT	DESCRIPTION
[*decodetype*:station:sensor]	Enable	No	Enables a sensor to be decoded when a message from station is received on a channel which requires decodetype processing. example: [GoesTimed:HerndonCreek:Stage] Enable=Yes
	Format	6BIT	Defines the type of data contained in the message. 6BIT and U6BIT are always used in GOES Random messages and often used in GOES Self timed messages. 8BIT and U8BIT are used in ARGOS messages. ASCII is used in GOES Self timed SHEF or Standard Decimal Format messages. 6BIT - Signed 6 Bit Binary U6BIT - Unsigned 6 Bit Binary 8BIT - Signed 8 Bit Binary U8BIT - Unsigned 8 Bit Binary ASCII - Signed or unsigned number
	RelativeTime	No	Set this to Yes when this definition is a continuation of a decode and the time stamp where the previous decode table left off should be used. See the More option for information on how to continue a decode with another table.
	RelativePos	No	Set this to Yes when this decode table is a continuation of a decode and you want the decode position to continue where the previous decode table left off. In some cases even with a continuation you will want to set this to NO in order to restart decoding at position 1. See the More option for information on how to continue a decode with another table.
	AddBase	No	Setting this to Yes causes any decoded values to have a base value added to them. The base value is the value of the last decoded sensor in a previous decode table. This option allows you to decode messages where the first value is total count and the

TOPIC	ITEM	DEFAULT	DESCRIPTION
			rest are just increments to that total count. See the More option for information on how to continue a decode with another table.
	Size	3	Size is the length of the data to be decoded in bytes. For 6BIT data 3 bytes is the same as 18 bits. For 8BIT data 3 bytes is the same as 24 bits.
	BitSize	0	BitSize is the length of the data to be decoded in bits. There is no need to use BitSize unless data is not encoded on byte boundaries.
	Bound1	0	Determines the position to start searching for a BlockID.
	Bound2	0	Determines the position not to seach past when looking for a BlockID in a message.
	BlockID		This is a string which if specified is searched for in the message from Bound1 to Bound2 before decoding begins. If found the decode position is set to the character immediately after the BlockID. If the BlockID is left blank then no searching is performed and the initial decode position is set to the first data character in the message.
	Pos	1	Determines the byte position of the sensor in the message. The first character at the current decode position is Pos 1.
	BitPos	1	Determines the bit position of the sensor in the message. The most significant bit in a byte is at BitPos 1, the least significant bit in a byte is at position 8. BitPos is not needed unless the decode message is bit encoded.
	Next	3	Determines how many bytes to increment the decode position to point to the next value of a sensor in a message.
	BitNext	0	Determines how many bits to increment the decode position to point to the next value of a sensor in a message. BitNext is not needed unless the decode message is bit encoded.
	Amount	0	Determines how many values for this sensor are contained in this message.
	DeltaTime	00:15:00	This is the time betwen sensor values. If most recent data appears first in the message (which is the most common case) then DeltaTime should be negative. example: DeltaTime=-00:15:00

TOPIC	ITEM	DEFAULT	DESCRIPTION
			for typical 15 minute GOES data
	RoundOffTime	00:15:00	Determines the interval at which time stamps are rounded off to. For instance with the default 15 minute value for RoundOffTime: 12:48 would be rounded to 12:45 and 12:53 would be rounded to 13:00.
	OffsetTime	00:00:00	OffsetTime is the amount of time between when a DCP collects data and when it transmits it. Default time stamping of a sensor value is performed by subtracting the OffsetTime from the time the message was received and then rounding off to the nearest RoundOffTime interval.
	More		Allows continuation of the decode of this sensor with a new XC Decode table by specifying the name of the table to use. You may need to use continuation decode table if the sensor values are irregular in some way. example: More=GoesTimed:HerndonCreek:MoreStage would continue decoding using the table information found under the topic [GoesTimed:HerndonCreek:MoreStage]. The name MoreStage is arbitrary and does not have to be defined as a sensor. In fact there is no requirement on the other names as well, except that the topic must contain decode table information.
	SkipBlank	0	Will move the decode position past SkipBlank space characters after the block id in the message. This is useful when decoding standard decimal messages from the 8200/8210/9210/XPERT. An example of a typical sensor format of an 8200 ASCII message is: ":AT 0 #15 75.2 76.3" In order to skip past the offset of 0, and the interval of #15 to reach the first sensor value of 75.2, set SkipBlank to 3 - because there are 3 space characters.
	SkipComma	0	Will move the decode position past SkipComma comma characters after the block id in the message. This is useful when decoding other vendor platform DCP messages. An example of a typical sensor format of an ASCII message is:

TOPIC	ITEM	DEFAULT	DESCRIPTION
			"23.6,100.7,75.2,76.3" In order to skip past to read the third value, set SkipComma to 2.
	SkipLF	0	Will move the decode position past SkipLF LF characters after the block id in the message. This is useful when decoding other vendor platform DCP messages. An example of a typical sensor format of an ASCII message is: "05.391 05.410 05.391 15778.6 15778.0 15772.4 14.34 14.11 14.21 " In order to read the values on third line, set SkipLF to 2.

TOPIC	ITEM	DEFAULT	DESCRIPTION
[SENSOR:station:sensor]	Enable	No	Set to Yes to allow the sensor to be decoded and processed.
	TagName		to be used for automatic 8200/8210/9210/XPERT SHEF decoding.
	Raw		If non-blank this is a list of additional sensors to process the raw data from this sensor through. Besides allowing the benefit of additional processing, Raw can also be used to store the same data in to multiple data files. ex: TempF,TempK would cause raw data to be processed by sensors named TempF, and TempK.
	Equation		Allows an equation to be applied to incoming data to convert to engineering units or other processing.Table conversions and n-th order polynomials and trig functions are available. See the Equation field defined earlier in this document under the RTU Server program on page 32. An additional field called ASCII may be referenced. ASCII will be 1 when an ASCII message is being decoded, or 0 when a binary message is being decoded. This can be used as a flag to scale sensor values received in binary, but not in ASCII messages. ex: X/100+ 500 would move the decimal point of a sensor over two to the left value and add an offset of 500.

TOPIC	ITEM	DEFAULT	DESCRIPTION
			ex: (ASCII=1)*(X*0.234+10.6)+(ASCII=0)*X converts the sensor with one equation when it is received in an ASCII message, and another when it is binary. Equations can also contain references to real time values: ex: (X+RTU01.AirTemp)/2 would average the air temperature with its most recent value. However, since GOES messages are usually decoded oldest data last, this would always result in averaging the current value with the most recent.
	RightDigits	0	Determines the number of right decimal digits to retain when displaying or storing this number.

Templates

All topics can contain a Template item. A template contains the name of a new topic to search if an item cannot be found in the current topic. For instance if you are creating a station definition called STATION:NEW which is the same as STATION:OLD except for a few differences, you could set "STATION:NEW,Template" to the string "STATION:OLD". Then for instance if you requested "STATION:NEW,SiteType" and the site type did not exist in station NEW, then the XC Setup would look for it under "STATION:OLD,SiteType". Likewise if SiteType could not be found under station OLD then the XC Setup would examine "STATION:OLD,Template" if it was not blank and continue the search with that topic.

What the XCONNECT.INI file would look like:

[STATION:OLD]

SiteType=8200

ComPort=Com1

AckDelay=10.0

ResponeTimeout=30.0

UnitID=UNIT1

RadioPath=UNIT1

PhoneNumber=555-1212

Sensors=AirTemp,RainFall,WindSpeed,WindDir

[STATION:NEW]

; Default topics in station new to values contained in station old!

Template=STATION:OLD

; We only have to add information which is different:

UnitID=UNIT2

RadioPath=UNIT2

PhoneNumber=555-1213

Templates are very powerful because they allow you to concentrate like information in one place and reduce the amount of setup information in the setup file. Also in the case above station NEW did not have to be based on an existing "real" station, it could just as well be based on a fake topic which contains station general information. Template searching relies only on string substitution it does not rely on context information.

There are some limitations and special behaviour regarding template information. For one thing template linked items cannot be hot-linked, they can only be requested. Second template linked items can not be poked. If you want to change the base definition you will have to poke that item. If you wish to change the highest level value you will have to add a new item to that topic and then set it. This introduces an interesting difference between linked and non-linked items - that if you can request an item, but not change it then it must be linked.

Another special feature of the template handler in the XC Setup is the ability to associate a child topic with a father topic's template. For example, the sensor topic is a child of the topic station (since sensors belong to stations). When you tell the XC Setup to make this association it will ensure that any matching templates belonging to the father topic will also be inherited by the child. In simpler terms, if station NEW is based on station OLD and you do not explicitly define a sensor for station NEW then it will automatically be treated the same as the sensor defined in station OLD.

Below are the typical associations used for the RTU Server and other applications:

TOPIC	ENTRY	DESCRIPTION
[TEMPLATE]	ChildTopic=FatherTopic	This is the general format for associating any template of a child topic with the template of a father topic.
	Sensor=Station	This causes unsuccessful requests for information under the topic [SENSOR:station:sensor] to look further using the sensor information contained in the station specified in the template contained in the topic [STATION:station].
	TagInfo=Station	This causes unsuccessful requests for information under the topic [TAGINFO:station] to look further using the tag information contained in the station specified in the template contained in the topic [STATION:station].
	LogInfo=Station	This causes unsuccessful requests for information under the topic [LOGINFO:station] to look further using the log information contained in the station specified in the template contained in the topic [STATION:station].
	GoesRandom=Station	This causes unsuccessful requests for information under the topic [GOESRANDOM:station:sensor] to look further using the GOES random information contained in the station specified in the template contained in the topic [STATION:station].
	GoesTimed=Station	This causes unsuccessful requests for information under the topic [GOESTIMED:station:sensor] to look further using the GOES timed information contained in the station specified in the template contained in the topic [STATION:station].

NESDIS Header Field Description

```
CE122654023652359596G46-3NN052WFF00029@DEAap@JAAap@JAAap@JAgri□
1       2 3  4 5 6 78 9 ABC  DE F    G                        H
```

Field #	Byte Length	Range	Description
1	8	0-9,A-F	Satellite ID, Platform ID, DCP Address
2	2	00-99	Year message was received relative to GMT
3	3	000-365	Julian day message was received relative to GMT
4	2	00-23	Hour message was received relative to GMT
5	2	00-59	Minute message was received relative to GMT
6	2	00-59	Second message was received relative to GMT
7	1	G,?,W,A,B,T,U, M,I,N,Q,C,S	Failure Code G - Good Message ? - Message received with parity errors M - Missing message S - Short or Truncated message (**Sutron Only**) **THE FOLLOWING CODES ARE USED BY NESDIS ONLY:** W - Message received on wrong channel D - Message received on multiple channel (duplicate) A - Message received with address error(s) (correctable) B - Message received with bad address (not correctable) T - Message received late/early (time error) U - Unexpected message I - Invalid address N - PDT entry for platform not complete Q - Bad quality measurements C - Comparison error on test transmission Note: The PDT is NESIDS's platform description table.
8	2	32-57	Signal Strength of received message in dB. Normal operation is 44-48. Reliable data can be received as low as 37 if not other signal problem exists.
9	2	0,1,2...9,A -1,-2...-9,-A	Frequency Offset in 50 Hz increments. 0=0-49 Hz, 1=40-99 Hz,...8=400-449 Hz, A=500-549 Hz. A minus sign indicates the same range only below the center frequency.
A	1	N,H,L	Modulation Index or Phase Deviation N = Normal, 60 +/- 5 degrees H = High, > 70 degrees L = Low, < 50 degrees
B	1	N,H,P	Modulation Quality N = Normal, error rate better than $1x10^{-6}$ F = Fair, error rate between $1x10^{-4}$ and $1x10^{-6}$ P = Poor, error rate worse than $1x10^{-4}$
C	3	000-999	Channel message was received on

Field #	Byte Length	Range	Description
D	1	E,W,C,A	Satellite Spacecraft Location E - GOES East W = GOES West C = GOES Central (not currently used) THE FOLLOW CODES HAVE BEEN DEFINED BY SUTRON TO SUPPORT NON-GOES SATELLITES: A = ARGOS (Never see this from NESDIS)
E	2	FF	Uplink carrier status (not used, always FF)
F	5	00000-15750	Message Data Length (plus 4 bytes to account for satellite ID which used to be contained in the message + 1 byte for the EOT stored as a blank space at the end). In the example, there are acutally only 24 bytes of data, however, 24+4+1 or **29** is the actual number stored in the data length.
G	Message Data Length minus 5	!-~,CR,LF, SPACE	Message data. Can be continued on multiple lines separated by a CR and a LF.
H	1	☐	Blank space representing EOT (End of Transmission) at end of message.