COBALT User Guide
Integrating COBALT With TEMPO

Table Of Contents

COBALT User Guide

Integrating COBALT With TEMPO

This section includes some helpful tips on integrating COBALT with your TEMPO system; it assumes you are familiar with TEMPO. If you have detailed questions about any of the COBALT or TEMPO features mentioned here, please refer to their respective Reference Manuals.

A COBALT system uses the same client-server architecture as TEMPO. The COBALT server software runs on a dedicated computer and performs the real-time processing while the COBALT client software runs on a separate Windows-based computer.

A single TEMPO system can support up to eight COBALT systems. If you have multiple COBALT systems, each COBALT client must run on its own computer. The COBALT client program can run on the same computer as the TEMPO client or on a different computer.

In many ways, the COBALT client program, COBALT.EXE, looks and behaves like the TEMPO client program, TEMPOW.EXE. Many of the activities you will do with COBALT are similar to what you currently do with TEMPO. For instance, when you create a TEMPO protocol, you will create COBALT databases, lay out the COBALT screen, define KEY macros and save a COBALT configuration file which can later be loaded when you want to run that protocol.

The COBALT client also provides a set of commands that perform various functions on the COBALT system. Many of the COBALT client commands have similar meaning as their counterparts on the TEMPO client. For instance, the familiar TEMPO commands CLS, CLOAD, CSAVE, START, STOP, HSAVE and HZERO all have similar meaning on the COBALT client. Like TEMPO, the COBALT commands can be entered into the COBALT Command Window or executed from a command file. A new feature allows another client computer (i.e., a TEMPO client) to send commands to the COBALT client for execution.

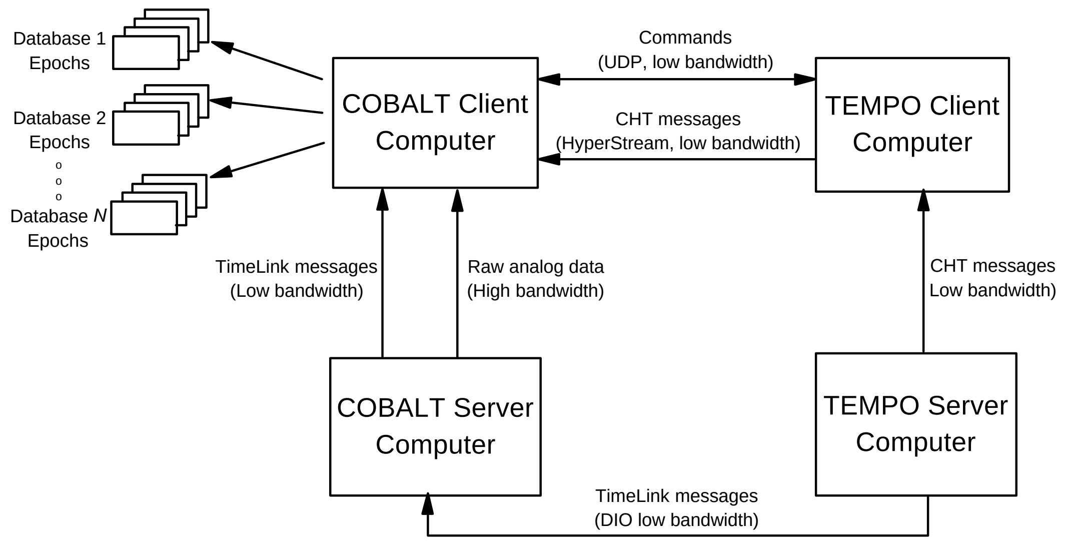
How COBALT and TEMPO Communicate

COBALT and TEMPO use two methods for communicating trigger and timing information:

1. The TEMPO client and COBALT client communicate cancel, harvest and trigger (CHT) information and commands via the local area network.
2. The TEMPO server and the COBALT server communicate timing information via the TimeLink ribbon cable and PCI-DIO boards.

Both links must be configured correctly in order for COBALT to function properly. When you set up your COBALT system, be sure to follow the procedure in the COBALT Reference Manual for installing the COBALT server.

The TimeLink connection is configured with SETUPTN. TEMPO server's parameters are listed under TEMPOServer/TimeLink. COBALT server's parameters are listed under COBALTServer*N*/TimeLink.



Some labs have many TEMPO systems. When the COBALT client is run, it needs to know on which computer its TEMPO client is running. The COBALT client obtains this information from the .TN file created by SETUPTN (i.e., TEMPOClient/Network/clientname parameter). When you create a new .TN file, this parameter defaults to the computer on which you are running SETUPTN. When you create the COBALT server diskette, the TN file is copied to the COBALT server diskette and, thus, onto the COBALT server. When a COBALT client connects to that COBALT server, the COBALT client downloads the .TN file from the COBALT server and uses the TEMPOClient/Network/clientname parameter to locate its TEMPO client.

In most cases, you will run the TEMPO client on the same computer that you created the .TN file and the COBALT client will look on that computer for its TEMPO client. Thus, when a COBALT client connects to a COBALT server, it will discover where the corresponding TEMPO client is.

But sometimes, you may want to configure the COBALT client to connect to a TEMPO client on a different computer from the one you ran SETUPTN on. In this case, you will need to tell the COBALT client the name of the TEMPO client computer. This is done by setting TEMPOClient/Network/clientname parameter with SETUPTN.

COBALT Commands

The following table summarizes some of the more frequently used commands available on the COBALT client. A complete list of all COBALT client commands is in the COBALT Reference Manual.

Command	Meaning
`hsave`	Save database
`hzero`	Zero database data
`hremove`	Delete database
`hcopy`	Copy database format
`hopen`	Create a database
`hfile`	Set database save file
`htitle`	Set database title

Command	Meaning
`hedit`	Modify database parameters
`stop`	Stop acquisition
`start`	Start acquisition
`speed`	Set acquisition rate
`exit`	Close the client
`connect`	Connect to COBALT server
`disconnect`	Disconnect from COBALT server
`flush`	Write output log file
`log`	Open output log file
`cls`	Clear message window
`msg`	Add message to LOG file
`key`	Define KEY macro
`cclear`	Clear COBALT configuration
`cload`	Load COBALT configuration
`csave`	Save COBALT configuration
`edit`	Edit a file
`dos`	Execute DOS command (wait)
`dosx`	Execute DOS command (no wait)
`cd`	Change current directory
`cmd`	Execute command file
`show`	Show status display
`remote`	Send command to remote client
`remotecfg`	Configure REMOTE command
`udpsrv`	Start command server

The REMOTE Command

The purpose of the REMOTE command is to allow you to control multiple client computers from one location. This is useful when you have one or more COBALT systems connected to a TEMPO system. For example, from the TEMPO Command Window, you can use the REMOTE command to START and STOP the data acquisition or HSAVE all databases on all COBALT client computers.

The REMOTE command is available on both TEMPO and COBALT client computers. It is automatically configured by the TEMPO and COBALT clients. You can use the REMOTECFG command to configure additional client computers; see COBALT Reference Manual for detailed information on the REMOTE and REMOTECFG commands.

The REMOTE command sends a command from one client computer to one or more other client computers.

The REMOTE command is followed by a command line which is executed on the remote client computer(s).

The syntax for the REMOTE command is:

```
REMOTE remoteCommand
```

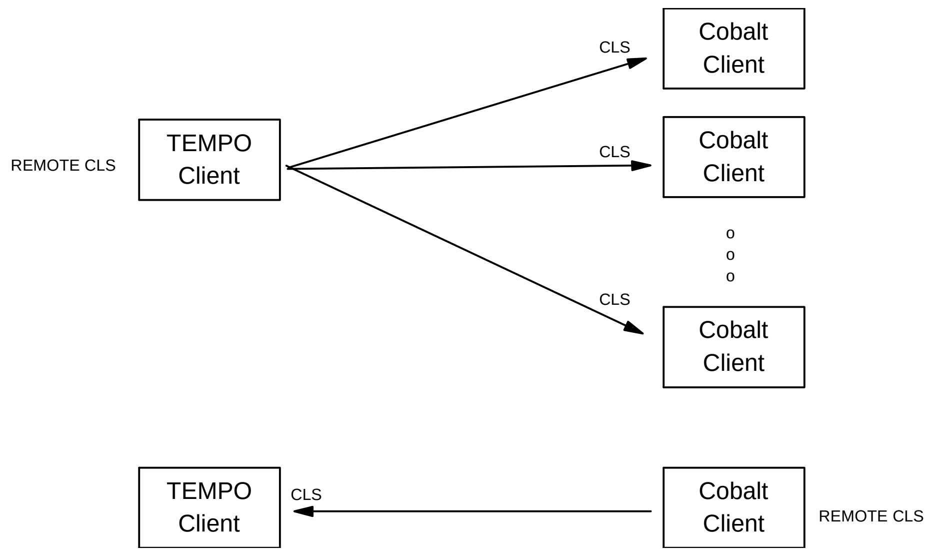
where

remoteCommand is the command to execute on one or more remote client computer(s).

Note that the *remoteCommand* is not executed on the client computer that executes the REMOTE command.

The REMOTE command is behaves slightly differently on the TEMPO client and COBALT client computers (see the REMOTECFG command). On the TEMPO client computer, REMOTE sends the *remoteCommand* to all COBALT client computers. For example, when you type REMOTE CLS into the TEMPO Command Window, the "CLS" command is executed on all COBALT client computers (and not on the TEMPO client computer).

On the COBALT client computer, the REMOTE command sends *remoteCommand* to the TEMPO client computer. For example, if you type REMOTE CLS into the COBALT Command Window, the "CLS" command is executed on the TEMPO client computer (and not on the COBALT client computer).



Using the REMOTE command on TEMPO and COBALT client computers

On the TEMPO client computer, the REMOTE command is like any other TEMPO command in that you can invoke it using any of the usual ways:

1. Type the REMOTE command directly into the TEMPO Command Line Window,
2. Execute the REMOTE command from a dialog button,
3. Execute the REMOTE command in a TEMPO command file,
4. Add a call to the system() or systemf() function in your protocol,
5. Execute the REMOTE command from an external program via DDE or Symphony,
6. Execute the REMOTE command from a Key macro.

For example, when you want to start the acquisition on your TEMPO and COBALT computers, you could type START into the Command Window on each client computer.

Alternatively, you can type the following commands in the TEMPO Command Window:

```
START                                 First start acquisition on the TEMPO system
REMOTE START                          Then start acquisition on all COBALT systems
```

Thus, you can centralized the control of your entire system (TEMPO + COBALT computers) from one place (i.e., the TEMPO client computer).

It is possible to restrict the computers to which REMOTE sends the *remoteCommand*. It is also possible to use the REMOTE command to send commands from a TEMPO client computer to one or more other TEMPO client computers, allowing you to coordinate multiple TEMPO systems from one location. See the COBALT Reference Manual for more information about the REMOTE and REMOTECFG commands.

Creating, Saving and Loading COBALT Configurations

The COBALT client program uses a COBALT Configuration File for each of your TEMPO protocols. The COBALT Configuration File stores protocol specific information such as the databases you want to use with the protocol and the layout of the graphs on the screen.

Before you run one of your TEMPO protocols, you will CLOAD that protocol's COBALT Configuration File. The default file extension for a COBALT Configuration File is .CCF (the default extension for a TEMPO Protocol Configuration File is .PCF).

The COBALT client uses the CCLEAR, CSAVE and CLOAD commands to clear, save and load a COBALT configuration. If the extension is not specified in CSAVE or CLOAD, COBALT's CLOAD command defaults the extension to .CCF.

On the COBALT client computer, after manually creating your COBALT databases and graph layout, use CSAVE to save the configuration to a COBALT Configuration File. For instance, CSAVE MYPRO saves the COBALT configuration to MYPRO.CCF on the COBALT client computer.

The command "CLOAD MYPRO" loads the MYPRO.CCF COBALT Configuration File stored on the COBALT client computer.

On the TEMPO client, you can use the REMOTE command to execute CLOAD on all of your COBALT client computers. For example, typing into the TEMPO Command Window:

```
REMOTE CLOAD MYPRO
```

causes each COBALT client computer to execute "CLOAD MYPRO".

A convenient way to load all COBALT configurations is to add a dialog button to your protocol's main dialog. Another way is to have the protocol execute a call to the system("REMOTE CLOAD MYPRO") PCL function. When using system() or systemf(), be sure to have your protocol wait for the commands to complete before continuing.

```
systemf("REMOTE cload mypro\n");          // Load all COBALT configurations
systemf("REMOTE start\n");                // Start all COBALT acquisitions
while (systemf()) nexttick;               // Give all COBALT systems a chance to start
```

```
    ...                                          // All COBALT systems are loaded and acquiring data
```

Note that each COBALT client computer uses its own MYPRO.CCF file so each COBALT client computer can have a different configuration.

Building COBALT Databases

Like TEMPO, COBALT accumulates epochs of data into one or more memory (RAM) databases. Unlike TEMPO, which builds its databases on the server computer and downloads them to the TEMPO client computer, COBALT builds its databases on the client computer. The COBALT server computer transmits a continuous stream of analog data directly to the COBALT client computer and the COBALT client computer extracts the data it needs and builds the COBALT databases based on trigger , cancel and harvest information it receives from the TEMPO system.

After a number of epochs have been collected into one or more COBALT databases, you can save the databases to a disk file. The memory database(s) can then be zeroed and more epochs can be accumulated into the memory databases.

The familiar TEMPO commands HEDIT, HOPEN, HLOAD and HCOPY commands are available on the COBALT client computer for creating or modifying the parameters for a COBALT database. You can also use the COBALT client's Database menu to access the HEDIT dialog and manipulate the COBALT databases.

The 12 and 16 bit analog append (XAPP) databases are supported by COBALT. At present, other database types such as XSUM databases are not supported by COBALT. (Please refer to Symphony's Spike Triggered Averaging program for creating peri-event databases.)

All of the database parameters are identical in meaning to the TEMPO databases.

Starting and Stopping COBALT Acquisition

The COBALT client uses the START and STOP commands to control the starting and stopping of data acquisition on the COBALT system.

You can also start and stop COBALT acquisition from your TEMPO client computer. For instance, you can have a START dialog button that you use to START the acquisition and a STOP dialog button to stop the acquisition.

START dialog button:

```
    start                                 Starts acquisition on TEMPO system
    REMOTE start                          Starts acquisition on all COBALT systems
```

STOP dialog button:

```
    REMOTE stop                           Stops acquisition on all COBALT systems
    stop                                  Stops acquisition on TEMPO system
```

It is helpful to adopt conventions that you use with all of your protocols. For example, on the TEMPO client you can define the F1 key (see KEY Macro command in the TEMPO Reference Manual) to display a dialog that lets you configure the protocol, start and stop the acquisition. This dialog can be accessed by typing the F1 key after CLOADing the TEMPO Protocol Configuration File. In the F1 dialog, you can have a button that invokes the REMOTE CLOAD command to load all COBALT Configuration Files. You can also have START and STOP buttons that start and stop acquisition on both TEMPO and all COBALT systems.

Triggering, Canceling, Harvesting COBALT Database Epochs

COBALT conforms to the same rules as TEMPO for triggering, harvesting and canceling database epoch accumulations. When your protocol executes the TRIGGER *tag* PCL statement, all COBALT databases with the matching tag will accumulate an epoch of data.

When your protocol sets a CANCEL bit (see cancel_set() PCL function), all pending COBALT triggers that span that moment in time will be canceled.

The HARVEST PCL statement also works the same for COBALT as for TEMPO: You can force a premature accumulation of data on your COBALT systems by executing the HARVEST statement from your protocol.

The TimeLink cable that connects the TEMPO server to all COBALT servers is used to convey timing information which is used by each COBALT system to digitally collate data it collects with the time of the trigger on the TEMPO system.

COBALT.PRO Include File

There are a number of COBALT specific protocol definitions that you can use in your protocol to monitor your COBALT databases. To use them, you must add the following line at the beginning of your TEMPO protocol:

```
#include cobalt.pro
```

The COBALT.PRO file is in your C:\TEMPO directory. It is well documented and describes each protocol variable it defines.

Checking COBALT Database Epoch Counts and Pending Triggers

There are inherent delays in the processing of triggers on each COBALT system due to the fact that they are processed on computers separate from the TEMPO server computer. These delays do not affect the accuracy or temporal alignment of the triggers because COBALT performs the alignment based on digital time stamps.

In TEMPO, the Pending Trigger Request Queue (see PCL Chapter in TEMPO Reference Manual) holds all pending triggers before they have matured. PCL functions such as getPendingDBTriggers() and GetPendingTAGTriggers() are used by the protocol to determine if all pending triggers have matured. Before saving databases to disk, your protocol will typically check the TEMPO Pending Trigger Request Queue to make sure all pending triggers have matured.

In COBALT, databases are accumulated in a similar way but on separate computers from the TEMPO system. So a new mechanism is required to check that all pending triggers have matured for a COBALT database.

To verify that all pending triggers have matured for a particular COBALT database, certain values in several protocol arrays (see COBALT.PRO) are set by each COBALT system. The protocol can check these arrays at any time to obtain information on the current status of each COBALT database on each COBALT system.

The COBALT.PRO file defines the following protocol variables and arrays:

```
int NCOBALTS = 8;                          // Maximum number of COBALT systems
int NCOBALTDBS = 255;                      // Maximum number of databases on one COBALT system

int cobaltTick[NCOBALTS];                  // COBALT tick time for each COBALT system
int cobaltDBnEpochs[NCOBALTS, NCOBALTDBS];   // Number of DB epochs in each COBALT database
int cobaltDBnPending[NCOBALTS, NCOBALTDBS];  // Number of pending triggers for each COBALT db
```

These arrays contain information that is automatically updated by each COBALT system: cobaltTick[.], cobaltDBnEpochs[.,.] and cobaltDBnPending[.,.] whenever a COBALT database changes.

- The **cobaltTick[.]** array holds the current time (in TEMPO ticks) of each COBALT system. You can compare this value with TEMPO's tick time (see PCL's tick() function) to determine if the COBALT system has processed all TRIGGER, HARVEST and cancellations up to a particular point in time (see example below). Each COBALT system updates its tick time in this array periodically (approximately every 100 ms).
- The **cobaltDBnEpochs[. , .]** matrix contains the epoch count for each COBALT database. The protocol uses this matrix to determine the number of epochs in each COBALT database. When an epoch is accumulated into a COBALT database, the COBALT system updates the corresponding value in this matrix.
- The **cobaltDBnPending[. , .]** matrix contains the number of pending triggers for each COBALT database. The protocol uses this matrix to determine the number of pending trigger requests for a particular COBALT database. When the COBALT system receives a TRIGGER request, it increments the corresponding value in this matrix. When the trigger matures, the COBALT system decrements the corresponding value in this matrix.

The following PCL code illustrates how your protocol can test whether all pending triggers have matured for a particular COBALT database.

```
// WaitForCobaltDB - Wait for all triggers for a COBALT database to mature
// (accumulate or be canceled).
//
// IN
//      nCobalt         Cobalt Client System Number [0,2,..,NCOBALTS)
//      nDb             Cobalt Database on COBALT client [0,2,..,NCOBALTDBS)
//
// OUT
//      The WaitForCobaltDB() process returns when the COBALT system has processed
//      all triggers for specified database up to the time the WaitForCobaltDB()
//      process was spawned.

process WaitForCobaltDB(int nCobalt, int nDb)
{
    int     currentTick;                // Current TEMPO tick time

currentTick = tick();

// Wait for COBALT to process all triggers up to this point in time

while (cobaltTick[nCobalt] <= currentTick)
    nexttick;

// Wait for COBALT to process all triggers for this db

while (cobaltDBnPending[nCobalt, nDb])
    nexttick;
}
```

In the above example, the WaitForCobaltDB process sets currentTick to the current TEMPO tick time. It then waits until a particular COBALT system has processed all triggers up to that point in time. This insures that the last trigger request made by the protocol has been processed by the COBALT system. Finally, the WaitForCOBALTDB process waits for all pending triggers to mature for the requested database. Collectively, this insures that all triggers up to currentTick time have been processed by the COBALT system and are mature for the particular database.

Saving and Zeroing COBALT Databases

The HSAVE command saves one or more memory databases to a disk file (in .HTB format) on the COBALT client computer.

The TEMPO system can use the REMOTE command to send the HSAVE command to the COBALT client computer. For instance, your protocol can execute system("REMOTE HSAVE") PCL function to save all COBALT databases.

If you want your protocol to automatically save your COBALT databases, it can spawn and wait for the WaitForCobaltDB() process above before executing the COBALT HSAVE command. This insures that there are no pending triggers when COBALT saves the database (see the CCQTEMPO.PRO sample protocol).

```
int     nCobalt;
int     nDB;

nCobalt = 0;
nDB = 0;

spawnwait WaitForCOBALTDB(nCobalt, nDB);
systemf("remote hsave %d\n", nDB);
systemf("remote hzero %d\n", nDB);
while (system()) nexttick;
```

The COBALT HTB file format is identical to TEMPO's HTB file format. Note that COBALT supports a higher acquisition rate than TEMPO so, while the HTB file formats are identical, the TEMPO client may not be able to load and display HTB database files created by COBALT.

The COBALT client HZERO command sets the epoch count to zero for one or more COBALT databases.

The CCQTEMPO Protocol Example

The CCQTEMPO sample protocol (CCQTEMPO.PRO, CCQTEMPO.PCF and CCQCOBAL.CCF) illustrates the use of many of the techniques mentioned in this section. You can refer to this specific example if you have a question about how to implement particular feature.