



Portable T.30 Fax Protocol Engine

There are four major elements of a typical fax application: fax modems, T.30 protocol engine, image-conversion and management, and the application. If every fax transaction is successful the factor that leads most directly to customer satisfaction is the application. But there is nothing an effectively designed and implemented application can do to overcome poor performance of the underlying fax technology, and the system element most responsible for high connect rates, the biggest term in the performance equation, is the T.30 fax protocol engine. Commetrex's Portable T.30, with Always Connect Technology™, provides the highest connect rates in the industry, reflecting its 7 years of field deployment.

But the licensor of technology must be concerned with more than performance: portability and maintainability are major success factors. These attributes are affected by design, implementation, and documentation.

Portable T.30 was designed specifically for portability. All interfaces with external modules or sub-systems are implemented as exported interfaces. Moreover, the large state machine required to implement the entire T.30 protocol consists of approximately 2400 state machine "tuples", of which about 1200 are possible, resulting in a nearly unmaintainable system if implemented using traditional program logic. But Commetrex has developed an exclusive state-machine design tool based on a relational database and a graphical front end that allows maintenance and design customization through a T.30-specific graphical interface. State-machine "edge routines" are written, tested, and checked into a library, as required. These routines are subsequently used by the tool's code generator to create the runtime object. This approach makes design and

maintenance straightforward and virtually eliminates regression errors. Moreover, the tool isolates the designer of all but the edge routines from the specifics of the targeted environment.

Features

- Complete implementation of the ITU T.30 fax protocol
- Supports BFT, MMR, ECM, and sub-addressing
- Graphical design tool included
- OTF, OpenMedia, and MSP Media Gateway
- Object Code Licenses
- Source Code Licenses
- Field proven
- Hardware independent
- Fully documented

Benefits

- Broad market coverage
- Fast time-to-market
- Resource and vendor independent
- Low product-development costs
- Low maintenance costs



Overview

Portable T.30 is a fax protocol state machine portable across a number of operating systems and hardware platforms. In order to achieve portability Portable T.30 isolates system dependencies to "adaptation" packages. The process of porting T.30, then, requires the rewrite of the adaptation packages, not T.30.

This design partitioning has resulted in the following major components:

1. T.30 Core consisting of the T.30 state machine and HDLC support routines.
2. Event Preprocessor converts system events into T.30 Core events.
3. Hardware Adaptation consisting of routines for interfacing with the modems.
4. Timer Package provides timer service to the T.30 Core.
5. Image Data Service handles image data and conversions on-line.
6. Image I/O Service handles physical I/O of the image data.
7. API Package provides an interface for the controlling application.
8. Buffer Management Package manages a pool of image-data buffers.
9. Parameter Manager provides access to system parameters by name.

T.30 Core

The T.30 core consists of a finite state machine driven by events from the Event Preprocessor. The finite state machine implements all sections and protocols of the ITU T.30 specification. It makes use of the other services to implement the processing required by each event transition.

Portable T.30 maintains a context area for each separate T.30 session it is handling. This context is initialized when the application requests service and remains in existence until the application specifically terminates the session, giving Portable T.30 an inherent multi-channel capability.

Event Preprocessor

The Event Preprocessor controls the operation of the T.30 Core. It accepts or detects event conditions in the environment, converts these conditions into events for the T.30 core, and passes the event, along with the appropriate channel context, to the T.30 Core for processing.

The exact nature of the Event Preprocessor depends on the operating system used. Under NT it may run as a separate thread for each session. The input of un-processed events to this package depends on the hardware adaptation required and operating system services available.

Hardware Adaptation

The Hardware Adaptation layer presents a set of basic functions to Portable T.30 for controlling modem operation. There are a number of models available for this interface. Some of the design issues are

1. Mini-Sequences - Because timing required by T.30 between modem operations, some fax command sequences must be performed as one command if the system partitioning has the T.30 engine executing on a processor with significant command latency.
2. Fixed-Length Buffering – Some environments require a relatively large buffer for data transfer between the controlling environment and the modems, sometimes placing unique requirements on the formatting of data and the design of the modems.

Timer Package

The Timer Package implements timer-based events to trigger error recovery for T.30. A number of timers are specified in T.30, and in some cases the protocol depends on the expiration of a timer to move to a new state. The actual implementation of the times is operating-system and hardware dependent. In some cases, the hardware platform can generate timer events that can be routed through the Event Preprocessor to drive the Timer Package, in

others the operating system provides sufficient facilities.

Timers for T.30 support require a resolution of no more than one-half-second accuracy. The longest timer required is 30 seconds.

Image Data Service

The Image Data Service is responsible for handling data between external storage and Portable T.30. It handles all format conversions. BFT, ECM and MMR are treated as additional conversion formats.

The Image Data Service treats image data as a sequence of documents each consisting of a series of pages. T.30 will renegotiate between documents in order to transfer new NSF information and possibly to change document size, etc. In the case of BFT and ECM a page is a single high-speed modem operation rather than an actual page. The Image Data Service separates the physical storage interface from the image-formatting function, permitting the use of alternative file formats and data sources other than disk files to be used for supplying/receiving data.

Image I/O Service

The Image I/O Service handles the physical I/O of the image data. Some implementations use the "C" runtime library to access TIFF-F-formatted files. Other implementations may use other image data containers or network-based file storage.

API Package

The API Package is divided into front-end and back-end sections. The front-end section collects the parameters required for each call and packs the parameters into a datagram. The datagrams are routed to the back-end portion. The back-end portion unpacks the parameters and issues the call to Portable T.30 by configuring the context area and generating events.

The front end of the API is linked to the application. The back end is linked with Portable

T.30. The application and Portable T.30 might execute on different nodes on a network. Another form would place the application and API front end on the host and the portable T.30 and back end on the scalar processor on a hardware board.

Buffer Management Package

T.30 requires a number of large buffers for image data. In Portable T.30 the facility filling a buffer creates the buffer to pass to T.30. It is the responsibility of Portable T.30 to call the appropriate release function when use of the buffer is complete. For example, in transmit, T.30 requests a buffer of data from Image Data Services. Image Data Services allocates a buffer, fills it with data and returns a pointer to the buffer to T.30. T.30 passes the buffer unchanged to the Hardware Adaptation Layer for queuing to the modem resource. Once the data have been passed to the modem resource, the Hardware Adaptation Layer releases the buffer to Buffer Management. The hardware Adaptation Layer will allocate and queue buffers to the modem resource as required. Each time a buffer is filled, T.30 will be notified by event. It will request a pointer to the filled buffer from the Hardware Adaptation Layer. This buffer is passed to Image Data Services for processing. The buffer is returned to buffer poll by Image Data Services.

This package allocates a number of buffers as required by the implementations of the Hardware Adaptation Layer and the Image Data Services. The size of these buffers is dependent on the buffering restrictions for the modem resource.

Parameter Manager

Portable T.30 makes use of a number of system configuration parameters. It would be cumbersome to specify all parameters with each call. Also various versions of Portable T.30 require special configuration parameters, this facility provides the adaptation layers with an interface to read version-specific parameters without changes to the application or other components of Portable T.30

FSM Design Tool

A Finite State Machine (FSM) is a state machine that is composed of a finite number of states and events. A state machine is a set of tuples, with each tuple composed of an old state, event, new state, and action. Commetrex has developed a design tool to support the creation of complex FSMs which defines the FSM in an SQL database. Each record in the database documents a tuple, defining the action to be taken when the machine is in the old state and the event is detected. The tuple record lists the new state the machine assumes and the action it performs. The actions are defined in the tuple by the associated module name. The design tool is used to create a database of these tuples and translate the data base into C code.

The FSM Design Tool is a GUI-based program designed to allow simple and intuitive development, viewing, and modification of any FSM. This tool is designed to be the only

interface necessary for design and maintenance of the FSM. The developer has the option of quickly and easily navigating the FSM, the same way a program would, by using events to jump from state to state, or to navigate sequentially through the database table. The FSM Design Tool flexibly handles database access, version control issues, and the spawning of the user's desired editor program.

The FSM database contains five tables, each consisting of several fields (see below). Two of the tables have a unique index, or key, defined to ensure that there are no duplicate records as well as to facilitate the querying of information from the database. Only the MODULE table and the FSM table are modified through the FSM Design Tool. The STATES, EVENTS, and VARIANTS tables are used to populate the selection lists for associated fields. These are static tables created at installation time.

NAME	SHORT_DESC	DESCRIPTION	TYPE	CODE
char[12] †	char[32]	varchar[1024]	char[2]	smallint †

EVENTS

NAME	SHORT_DESC	DESCRIPTION	TYPE	CODE
char[12] †	char[32]	varchar[1024]	char[2]	smallint †

MODULE

MODULE	REF_CNT	CODE	CODE_GEN	CODE_TIME	CODE_STATUS
char[8] †	smallint	char[13]	DEC(5,2)	char[18]	char[10]

NAME	SHORT_DESC	DESCRIPTION	CODE
char[12] †	char[32]	varchar[1024]	smallint †

FSM

OLD_STATE	EVENT	VARIANT	NEW_STATE	T30_REFERENCE
char[12] †	char[12] †	char[12] †	char[12]	char[80]

MODULE	ROOT	ACTOR	LASTCHANGE_DATE	LASTCHANGE_TIME
char[8]	char[8]	char[8]	DATE	TIME

Commetrex Corporation

6400 Atlantic Blvd., Suite 190

Norcross, GA 30071

Voice: (770) 449-7775, Fax: (770) 242-7353

<http://www.commetrex.com>

e-mail: marketing@commetrex.com