

APIs and development tools advance paging applications

MANJU NATH, TECHNICAL EDITOR

Flex and Hermes protocols enable paging applications ranging from a simple one-way pager to advanced two-way and voice paging. Understanding these protocols helps you quickly develop paging applications. Also, new APIs and development tools help you test and debug your designs.

Paging applications are getting more sophisticated everyday, thanks to the flexibility and digital nature of Flex and the European Radio Message System (Ermes), the leading paging protocols. Gone are the days of one-way paging, when a pager was a simple display. Two-way paging adds a response channel to a traditional one-way paging system. Two-way paging protocols can help you implement low-data-rate remote-control and inexpensive high-data-rate telemetry applications. For example, the protocols can read electricity and gas meters and perform follow-up action. They even let you replenish vending machines whose merchandise is running low. You can also remotely monitor vehicles and deploy paging-based streetlight control. Another applications is the ability to send and receive 2000-character e-mail messages.

With today's two-way and voice-paging applications, the paging devices' host requires a 16- to 32-bit μ C interface and an RTOS with well-defined application-programming interfaces (APIs). The pager-decoder IC (Figure 1) connects to an RF front end that converts a four-level audio signal into a 2-bit digital signal. The decoder uses control lines for warming up, operating, and shutting down a receiver in stages.

The pager-decoder IC detects a low-battery signal from an external detector during the receiver-control sequences. The IC interfaces to a host μ C through a serial interface to communicate demodulated pager-signal information. Although the pager-signal



NDC is the exclusive dealer for the OI electric voice pager, which runs on one AA battery. It uses Lernout & Hauspie's (www.lhs.com) low-bit-rate voice-compression technology to store and replay messages over the Flex Oneway protocol. Operators input messages by telephone to a voice-paging terminal with NDC Voice's VoiceOver technology. The unit stores as many as 30 20-second-long messages.

PAGING PROTOCOLS

@ a glance

- Both Flex and Ermes enable advanced paging applications.
- Application-programming interfaces and development tools ease the complexity of RF-data acquisition and help you quickly develop paging applications.

decoders interpret the transmitted pager data using the RF front end and Flex includes the Flex Stack application-programming interface (API) from Motorola, you still need to understand the bit-level nitty-gritty of pager protocols to be able to develop paging applications.

The first digital paging protocol was the Post Office Code Standard Address Generator (POCSAG), which emerged in 1981. It offered 512-, 1200-, and 2400-bps transmission, which the faster and more robust Flex and Ermes protocols have overtaken. Also, POCSAG cannot support growing pager applications because of channel capacity: POCSAG has a limit of 2 million addresses,

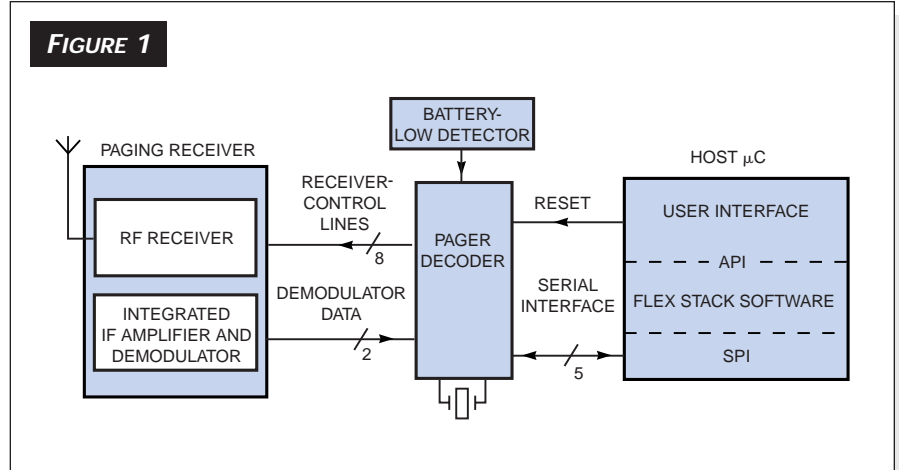


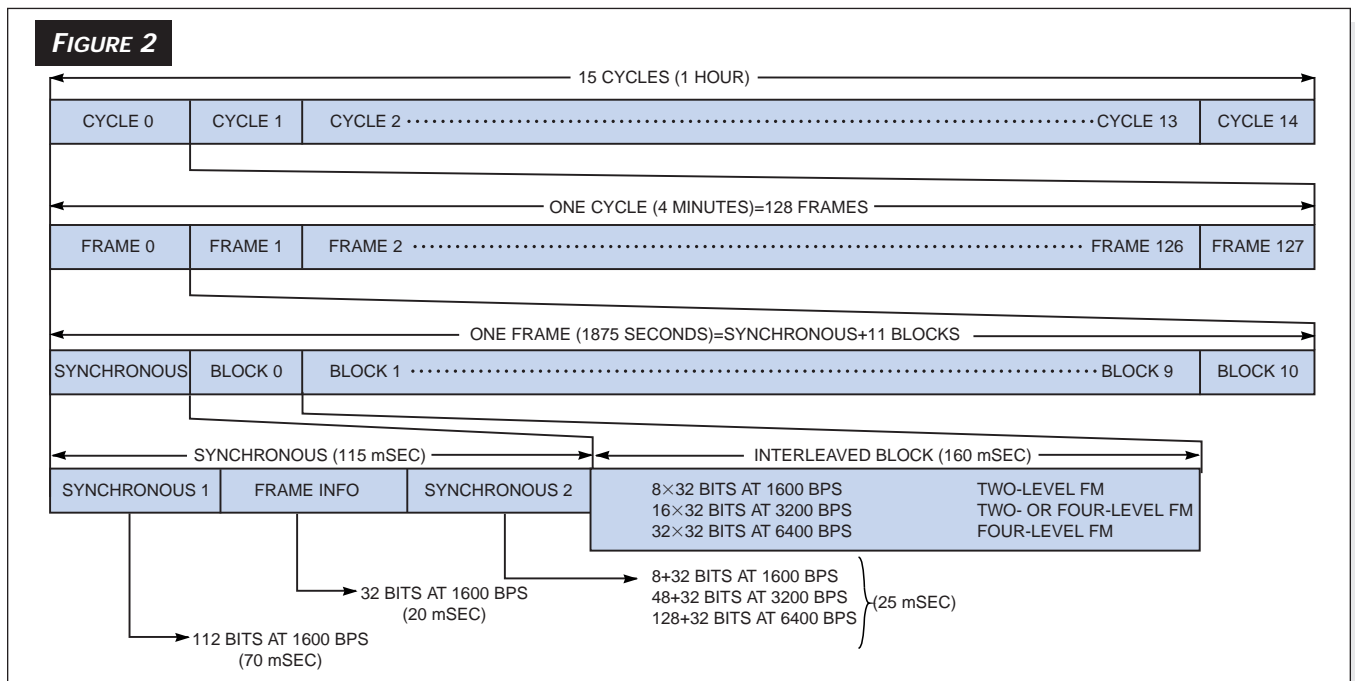
FIGURE 1 The pager-decoder IC connects to an RF front end that converts a four-level audio signal into a 2-bit digital signal.

whereas Flex supports 5 billion.

With POCSAG, fading-protection performance degrades at high speeds. Flex and Ermes, on the other hand, protect against fading even at high speeds by including 2-bit error correction and data interleaving of transmitted code words. Battery life is also a major issue in paging applications. Both Ermes and Flex offer five times the battery life of POCSAG. These advantages have com-

bined to make Flex the leading paging standard in the United States and most of Asia. Ermes, meanwhile, enjoys strong support in the European pager market, although Flex is now beginning to challenge that support.

A Flex signal transmitted on a radio channel comprises a series of 4-minute cycles, each having 128 frames at 1.875 seconds/frame (Figure 2). You can assign a pager to process any number of



A Flex signal transmitted on a radio channel comprises a series of 4-minute cycles, each cycle having 128 frames at 1.875 seconds/frame.

frames. The protocol performs battery saving for unassigned frames. The Flex signal can assign additional frames to the pager using collapse, fragmentation, temporary addressing, or carry-on information within the Flex signal.

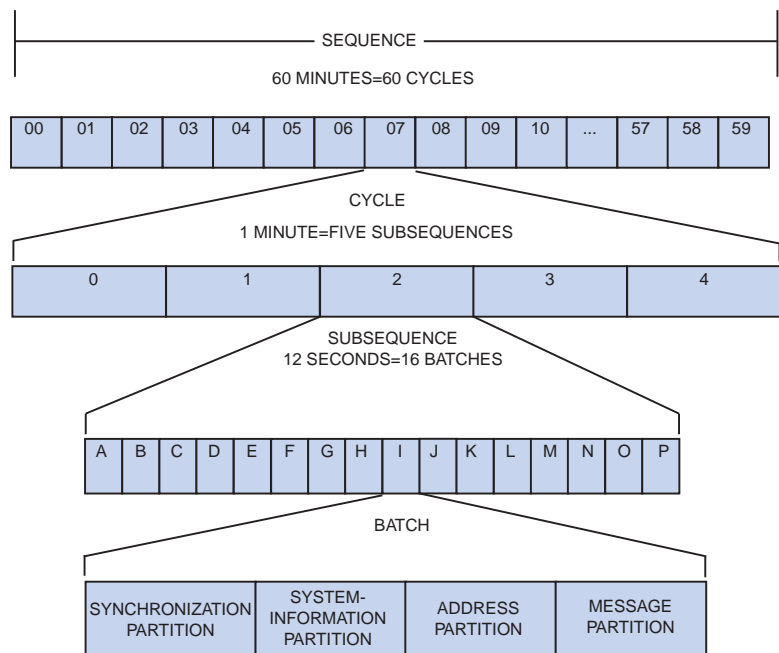
Each Flex frame has a synchronization portion followed by an 11-block data portion, with each block lasting 160 msec. The synchronization portion indicates the rate—1600, 3200, or 6400 bps—at which the data portion transmits. In Phase A—a single phase of information transmitting at 1600 bps—the protocol transmits at 1600 bps and 1600 symbols/second using two-level FSK modulation. In phases A and B—two concurrent streams of data transmitting at 1600 bps—the protocol transmits at 3200 bps and either 1600 symbols/second using four-level FSK modulation or 3200 symbols/second using two-level FSK modulation. In phases A, B, C, and D—four concurrent streams of information transmitting at 6400 bps—the protocol transmits information at 3200 symbols/second using four-level FSK modulation.

Each block has eight interleaved words/phase; thus, every frame has 88 words, numbered zero through 87. Each word has information contained within an error-correcting code, which allows for bit-error correction and detection. The protocol organizes the 88 words in each phase into block-information, address, vector, message, and idle fields. The boundaries between the fields are independent of the block boundaries. Furthermore, at 3200 and 6400 bps, the information in one phase is independent of the information in a concurrent phase, and the boundaries between the fields of one phase are unrelated to the boundaries between the fields in a concurrent phase.

The synchronization portion comprises a first synchronous signal at 1600 bps, a 7-bit frame-information word having the frame number zero through 127, a 4-bit cycle numbered zero through 14, and a second synchronous signal at the data rate of the interleaved portion. The block-information field contains block-information words for determining time and date information and paging-system information.

The address field contains addresses

FIGURE 3



An Ermes signal transmitted on a radio channel comprises a series of 60-minute sequences, or cycles.

assigned to paging devices. Addresses identify information sent to individual paging devices or groups of paging devices. An address can be either a “short” one-word address or a “long” two-word address. Information in the Flex signal can indicate that an address is a priority address. An address can be “tone-only,” which has no additional information associated with it. If an address is not tone-only, an associated vector word appears in the vector field. A vector is a code word that contains information on the message length and its location. Information in the Flex signal indicates the location of the vector word in the vector field associated with the address.

A pager can perform battery saving at the end of the address field when the pager’s addresses are undetected. The vector field comprises a series of vector words. Depending upon the type of message, a vector word—or words in a long address—may either contain all of the information necessary for the mes-

sage or indicate the location of message words in the message field comprising the message information. Short addresses have one associated vector word in the vector field. Long addresses have one associated vector word in the vector field followed by the first message code word of the call. The message field comprises a series of information words containing the message information. The message information may be in ASCII, BCD, or binary, depending upon the message type.

Ermes protocol

An Ermes signal transmitted on a radio channel comprises a series of 60-minute sequences, or cycles (**Figure 3**). The protocol coordinates sequences with the universal time coordinate (UTC), so a new sequence commences each hour. UTC coordinates a paging cycle of exactly 1 minute, allowing coordination among networks. Receivers may listen to one cycle or a few cycles in a sequence to reduce bat-

PAGING PROTOCOLS

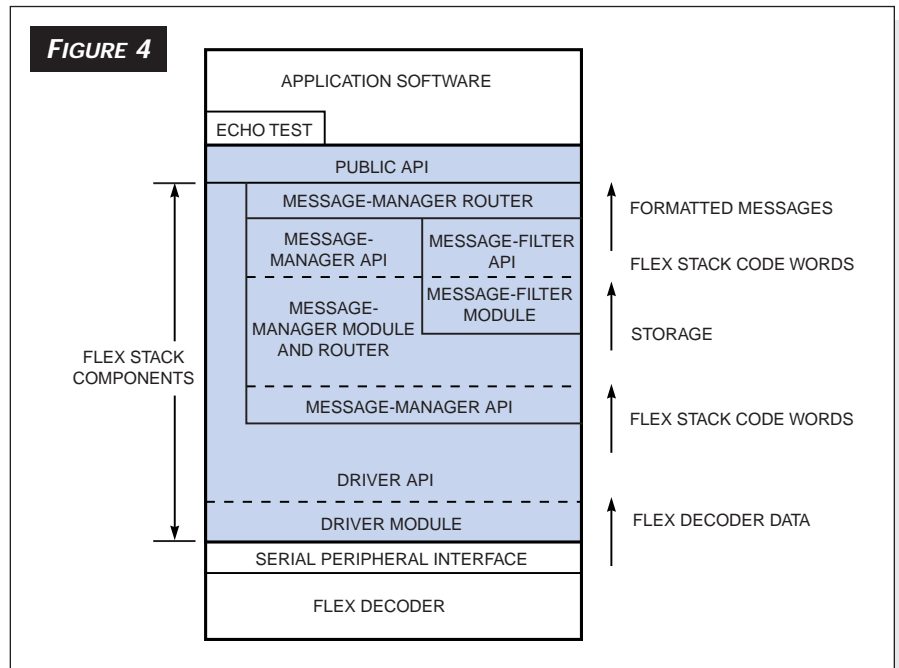
tery-power consumption. Each cycle comprises five subsequences commencing at 12-second intervals. To allow coordination among networks, the subsequence 0 transmits first after the UTC minute marker. Each subsequence contains 16 batches, A through P. Each batch subdivides into the synchronization, system-information, address, and message partitions.

The first 15 batches in every subsequence have 154 code words, and the final batch in every subsequence has 190 code words. This scheme enables messages in time-division-multiplexed networks to complete within a subsequence. A time batch shifts the time of transmission of a batch on each frequency channel with respect to the other batches. Consequently, a receiver can, if necessary, step through the paging channels without losing any messages.

The receiver population contains 16 groups, and each receiver belongs to one of the 16 batch types, according to the four least significant bits of its basic radio-identity code. Further radio-identity codes that this receiver uses should be of the same batch type. Each receiver is initially addressed only in its own batch-type transmission. Upon detecting its initial address, the receiver should wait on the same channel for the message to transmit. The message can be in the same batch, in any subsequent batch of the same subsequence, or in the following subsequence. Also, an initial address can transmit more than once in the same batch.

Flex versus Ermes

Even though both Flex and Ermes offer near-identical bit rates, crucial differences exist between the two protocols. Unlike the Ermes protocol that operates within the 169.425- to 169-MHz band, Flex is not wedded to one band. Flex and its variants offer multiple bit rates, providing a business advantage to carriers that want to roll out new system rates (Table 1). Initially, a carrier having relatively few customers can cover an area with a few transmitters running at 1600 bps. As the number of customers grows, the carrier can add a few more transmitters and double its



Available in C source code, HC08 assembler code, and pseudocode, Flex Stack runs on the host processor to manage communication with the Flex decoder and to interpret and format the Flex Stack code words that the decoder passes to the host.

speed. When traffic grows further and income is steady, the carrier can add transmitters and operate at 6400 bps. ReFlex and the newest Flex versions let you add channels to a system and have the pagers spread over the available channels.

In contrast, to roll out an Ermes system, an operator must install all the final transmitters from the start. Initially, a few customers must pay the full infrastructure and recurring site costs.

Both Ermes and Flex are synchronous, time-slotted protocols; these time slots are “batches” in Ermes and “frames” in Flex. The protocols assign pagers to a subset of time slots and usually switch off most of the electronics the rest of the time. More slots in a protocol means shorter battery life. The pager system knows the slot-selection rules of each pager and sends commands to a pager only in its assigned slots.

Both protocols use the same trick to increase battery life. A pager’s address tags commands to the pager and puts addresses early in the time slot. If a pager does not decode its address at the

start of the slot, the pager switches off its receiver and goes back to sleep. Ermes uses 18 data bits plus 12 check bits, and Flex uses 21 data bits, 9 check bits, and 1 parity bit for error-correction codes. Both protocols can correct 2 bad bits and detect 3 bad bits. Both protocols use interleaving to be more robust under the several-milliseconds fades that characterize radio transmission.

Because of the differences in address encoding, Flex seems slightly more robust than Ermes because Flex encodes all addresses using interleaving and error correction. In Ermes, all addresses appear twice, first without interleaving and later with interleaving. Hence, the initial address in Ermes is more vulnerable than an address in Flex. So, when an Ermes pager misses its initial address and does not look for a second copy, the pager misses a message.

However, sophisticated Ermes pagers can compensate for this drawback by staying awake longer. Because the Ermes pagers do not miss any messages, they have shorter battery life than do Flex pagers. Another advantage of Flex

PAGING PROTOCOLS

is that it provides a smooth migration path to ReFlex one- and two-way paging, which enhances system efficiency and lets you develop new applications.

APIs help you build applications

Available in C source code, HC08 assembler code, and pseudocode, Motorola's Flex Stack runs on the host processor to manage communication with the Flex decoder and to interpret and format the Flex Stack code words that the decoder passes to the host (Figure 4). Ermes has no equivalent APIs. However, Ginjet offers a demo program to help customers in Ermes software

design, according to the company's product manager, Charles Lin.

The Flex Stack comprises public and intermodule APIs that control the Flex decoder and manage the raw Flex data. This approach means that the software insulates you from having detailed knowledge of Flex decoder hardware. Also, by adhering to the public API, you do not have to rework software for revisions to the decoder, protocol, or other software components.

Flex Stack supports integrating a Flex protocol receiver into a variety of products, ranging from a simple pager device to sophisticated devices, such as PCs,

personal digital assistants, VCRs, or other similar devices, that require advanced message-receiving or data-retrieval capabilities. Because these products have hardware resources from an 8-bit μ C with little RAM and ROM to a 32-bit μ C with megabytes of storage, Motorola provides Flex Stack as a set of modules rather than a monolithic set of calls. This approach provides more flexibility to select the appropriate hardware and software components to support your application.

Flex Stack comprises driver, message-manager, and message-filter modules and a public API (PAPI). The driver

TABLE 1—PAGER PROTOCOLS

Protocol	Description	Applications	Operating frequency	Infrastructure requirements	Roaming support	Outbound channel (kHz)	Outbound signaling speed	Inbound channel	Inbound signaling speed
POCSAG	Low-speed, one-way	One-way, 4- and 7-bit numeric and alpha	Any available paging frequency	Existing	No	25	512, 1200, or 2400 bps	Not applicable	Not applicable
Ermes	European, one-way	One-way, 4- and 7-bit, binary numeric and alpha	169.425 to 169.00 MHz	Mostly new	Yes, between Ermes systems	25	6250 bps	Not applicable	Not applicable
Flex	High-speed, one-way	One-way, 4- and 7-bit, binary and symbolic-character numeric and alpha	Any available paging frequency	Modest upgrade typical	Yes	25	6250 bps	Not applicable	Not applicable
ReFlex 25	Two-way, messaging and data	Two-way, 4- and 7-bit, binary short messages	929 to 932, 940 to 941 MHz out, 896 to 902 MHz in	Modest transmitting upgrade or new transmitting plus new receiving	Yes	25 or 50	1.6, 3.2, or 6.4 kbps in three 25-kHz carriers in 50-kHz channel	12.5 kHz in 896 to 902 MHz	800, 1600, 6400, or 9600 bp
ReFlex 50	Two-way, messaging and data	Two-way, 4- and 7-bit, binary short messages	929 to 932, 940 to 941 MHz out, 896 to 902 MHz in	Modest transmitting upgrade or new transmitting plus new receiving	Yes	50	As high as 25.6 kbps	12.5 kHz in 901 to 902 MHz	9600 bps
InFlexion Voice	Advanced voice messaging supporting seven subchannels with three carrier channels	Voice paging with acknowledge back	930 to 931, 940 to 941 MHz out, 896 to 902 MHz in	Modest upgrade to new ReFlex	Yes	50	Digitally processed compression	12.5 kHz in 896 to 902 MHz	800, 1600, 6400, or 9600 bps
InFlexion Data	High-speed, two-way data supporting as many as seven subchannels, each as fast as 16 kbps; 50-kHz channel capacity as fast as 112 kbps	Two-way, 4- and 7-bit, numeric and data	930 to 931, 940 to 941 MHz out, 896 to 902 MHz in	Modest upgrade to InFlexion	Yes	50	4-, 8-, 12-, or 16-kbps/subchannel	12.5 kHz in 896 to 902 MHz	800, 1600, 6400, or 9600 bps

PAGING PROTOCOLS

receives data from the Flex decoder and builds raw message data from received data streams. The message manager stores and manages message data and routes calls to the appropriate API. The message filter formats raw message data according to the required format. The data format can be ASCII, binary, or ideographic character symbols.

The PAPI provides a high-level interface that allows the host software to manage message data, message notification, and the Flex decoder using 11 function calls. The PAPI's function calls include message-access calls, which comprise routines that access message attributes and formatted and unformatted message data; driver calls, which initialize and control Flex decoders and provide the means for initiating user actions based on the occurrence of events; and notification calls, which take action when selected events occur and are used to create new messages. In addition to the PAPI calls, each module has its own set of intermodule API calls that interface with other modules.

Implementing the Flex Stack

Motorola recommends implementing the Flex Stack software via the message and the driver models. By using the driver, message-manager, and message-filter modules and adhering to the PAPI calls, you can implement the message model. This approach works best for paging applications without RAM and ROM constraints. Using the message model simplifies implementation because you can access received mes-

sages using the PAPI's simple open, read, and close calls.

The Flex decoders receive the appropriate data that is specified when the designer initializes the chip. The designer determines the initialization parameters based on the product requirements. The decoders decode the signal and generate 32-bit packets of control information and data and send them to driver module using a small, μ P-dependent interrupt-service routine. The driver module interprets the chip packets and creates Flex Stack code words. Each Flex Stack code word contains 3 bytes of compressed message and status data. A full message comprises one or more code words. The Flex Stack delivers code words and additional status information to the message manager using intermodule API calls. The message manager accumulates the code words and notifies the host software when it has stored the message. To access a message, the host software issues PAPI calls to indicate which message to get. The message manager integrates a router, which directs or initiates intermodule calls to retrieve and interpret the sequence of code words that constitute the requested message. The filtered or unfiltered stream of message data then returns to the host application either filtered or unfiltered.

Unfiltered messages return in Flex Stack code-word format. Filtered messages return as ASCII, numeric, binary, or any other data format. You can use the Echo test code to verify and certify that messages can be built. Echo test

routine echoes back messages on the SPI bus for comparison with the originating message. Using the message-model implementation to read filtered messages insulates you from Flex protocol issues. Also, applications that are written to the PAPI are also protected from revisions to the Flex protocol and the Flex decoders.

If you plan to develop an embedded-pager application that has limited RAM and ROM (driver model), you can implement only those Flex Stack components that meet your requirements. For instance, you might want to write your own message manager to suit a product requirement. But this approach involves a lot of hard work, because you also have to manage the intermodule API calls that bind the host software to the module. Motorola warns that this technique is more susceptible to changes in the Flex protocol, decoders, and software applications.

Testing and debugging pagers

Few companies supply paging-development kits. For testing pager designs, you can use lab-based Ermes and Flex transmission capabilities. Motorola offers the \$395 Flex development kit comprising the Flex development board and a receiver board for building applications. Ginjet offers the EN622 Ermes pager encoder system comprising the EN622S protocol-emulation program, which interprets the command file and then transfers it into the Ermes code-word format, and the EN622H pop-up-menu program that

FOR MORE INFORMATION...

For more information on products such as those described in this article, circle the appropriate numbers on the Information Retrieval Service card or use *EDN's* Express Request service. When you contact any of the following manufacturers directly, please let them know you read about their products in *EDN*.

Ermes
www.ermesmousg.org
Circle No. 301

Ginjet Technology Corp
Taipei, Taiwan
+886 2 2506 3439
fax +886 2 2507 2064
www.gj.com.tw
Circle No. 302

Motorola Semiconductors
Boynton Beach, FL
1-800-542-7882
1-561-667-3299
www.mot.com/flexstack
Circle No. 303

NDC Voice Corp
San Diego, CA
1-619-268-4900
fax 1-619-636-6586
www.ndcwireless.com
Circle No. 304

VOTE . . .
Please also use the Information Retrieval Service card to rate this article (circle one):
High Interest 582
Medium Interest 583
Low Interest 584

PAGING PROTOCOLS

runs under MS-DOS. The program supports the ETS 300 133-4 air-interface specification and provides facilities to meet the ETS 300 133-5 receiver-conformance specification.

Using EN622S, you can define as many as 100 initial address calls and as many as 9000 message characters in one command file. You can also define as many as 3000 error patterns for modification of the generated Ermes code-word data. The EN622S supports numeric, alphanumeric, and transparent-data paging. The EN622S provides code-encoding and code-interleaving subfunctions. The system has 20 command-template files that let you test for conformance to the receiver specification. The EN622 is suitable for R&D and production testing of Ermes pager designs.

The EN622S sends the generated Ermes code-word data to the EN622H interface board via the PC's printer port. The EN622H converts the Ermes code-word data into the dc-modulated signal

output and 2-bit digital signal output. The dc-modulated signal output port connects with the dc-modulated input port of the RF signal generator, which generates the Ermes RF signal for user applications. Also, using the 2-bit digital output data, you can directly test or debug the Ermes pagers' digital board. Ginjet also offers customized solutions for developing Ermes-based paging applications.

The situation for Flex-based application developers is better. Motorola is developing Flex suite of presentation- and application-layer standards that further decouple applications from the protocol details. The suite will provide encryption, compression, e-mail, and file transportation on the Flex protocol.

EDN

References

1. Maloney, Mike, "Protocols and applications will be key to successful advanced messaging," *Wireless Systems Design*, January 1997.

2. Flex Stacy Oneway software-development guide, Version 3.0, Sept 10, 1997.

Acknowledgments

Many RF design engineers contributed to this article. I would especially like to thank Jheroen Dorenbosch, product architect at Motorola, for his valuable insights on paging protocols. Also thanks to Allen Kwan and Nelson Au of Motorola and HC Ma, managing director of Cirkisys (www.cirkisys.com), for sharing information on paging applications.

You can reach Technical Editor NS Manju Nath at +852 2965-1555, fax +852-2976-0706, nsmanjunath@cahners.com.hk.