Addressable Decoder with Downloadable Operation

Mack Daily
Zenith Electronics Corporation

ABSTRACT

There was a time when a set top converter could take a non-remote TV and turn it into a wonder of convenience. Now the converter has turned that fancy, hi-tech, 32 function remote control TV into a color monitor.

There is a need to make the converter desirable again, to offer features that a TV set can't support, to make the decoder a convenience.

This paper follows the design of a new CATV converter from the initial marketing concept, to the design that is in production today. It tries to shed some light on the decisions which lead to that gap between what was asked for and what is finally produced.

INITIAL MARKETING CONCEPT

Here's the wish list:

1) Color On Screen Display (OSD).

A new color on screen display will be the user interface. A menu system will allow the user to select options. Instructions for using the converter will be spelled out in plain English (or Spanish) on the screen.

2) Electronic Program Guides.

An electronic program guide will be loaded and updated continuously. This guide will be used as one would use the guide in the Sunday paper. The user finds the day and time and the converter shows what is on. Two weeks of information was deemed sufficient.

3) Messaging.

Messages to groups or individuals can be sent. Anything from "HAPPY BIRTHDAY" to "PAY YOUR BILL" to "DON'T FORGET! THE CONCERT YOU ORDERED IS ON TONIGHT".

4) Flexibility.

Think about the future. The system needs to be independent of the video scrambling system to allow migration from present to future systems.

HARDWARE

Hardware Part 1: Fulfill the List

Referring to Figure 1, a block diagram of a cable decoder is fairly generic if one is concerned only with the operation of the microcontroller.

The tuning microcontroller is the control center. It provides an interface with the user having IR and keyboard for input, LED and OSD for feedback. The tuning micro controls the tuner. Depending upon the complexity of the system, it may have input concerning descrambling. This implies the micro receives and interprets data. With or without the tuning micro's help, the descrambling portion decides authorization and manipulates the video for the desired output.

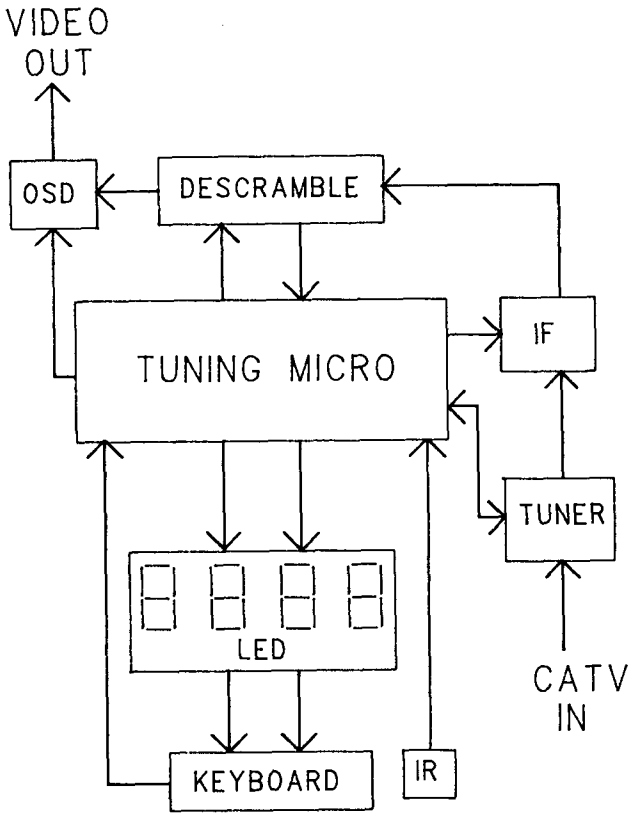


Figure 1
A Generic CATV Converter

Given our generic converter, how do we add the needed features? Easy, leave the present converter alone, it works! We still need all the tuning micro's tasks done, we're just throwing a couple of screens up now and then. An additional microcontroller is needed to control the new electronic program guide, and it needs a large memory in which to store the guide data. If the new micro reads the LED segment drivers of the existing tuning micro while an LED digit is being scanned, it can tell what that digit is displaying. These LED's tell if the converter is authorized, or can buy a program, etc. Using this information, the new micro will be able to show an OSD with a more detailed explanation of the situation. The new micro will receive the IR input, and control the keyboard, as it will be deciding what actions user inputs are calling for. The original tuning micro can be controlled by synthesized IR commands applied to its IR input.

While awkward, this approach could fulfill the requirements. However, it had became apparent that simply switching between watching programs and showing screens wasn't enough. Entering favorite channels and parental control had to be made easier. No one likes current VCR timers. All those connections to the tuning micro drove the pin count of our new micro way up.

It was apparent this system could not have 2 microcontrollers, both programmed to be in control.

Hardware Part 2: A Division of Labor

1) The Conditional Access Microcontroller:

The tuning microcontroller was made a slave, and became the Conditional Access Microcontroller. The Conditional Access Micro controls the LED's, keyboard scanning, and contains whatever video descrambling capability the original tuning micro had. The descrambling duties of the micro vary from scrambling

system to system, therefore the Conditional Access Micro is hardware and firmware specific to a scrambling system. The Conditional Access Micro receives in-band data (data which is transmitted within the bandwidth of the tuned channel) as before, but except for program authorizations, the data is passed to a master microcontroller. In fact, the only independent function of the Conditional Access Micro is program authorizations. The decision making process as to whether to descramble is the same as in the original system.

2) The Dialog Processor Microcontroller:

The new master of the converter is called the Dialog Processor. Except for program authorizations, the Dialog Processor runs the show. It can access a large external RAM. This RAM, where program guides and more are stored, has battery back up. An EEPROM provides non-volatile storage of critical data. Parental control and channel tuning information are stored in the EEPROM. The new color on screen display is under the direct control of the Dialog Processor. The display can overlay characters on live video, or generate entire screens.

With a 2-way expansion bus, the Dialog Processor will be able to read and write future accessories. A special accessory now available is a plug-in IR transmitter. The Dialog Processor can transmit low power IR commands, controlling a VCR, TV or whatever.

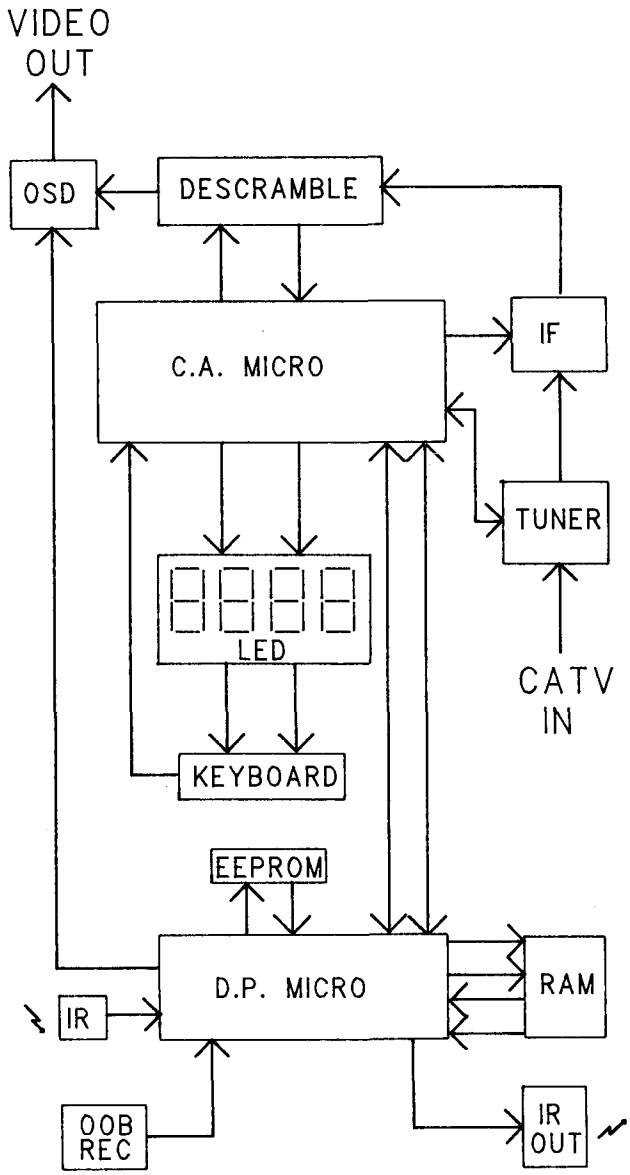


Figure 2
The New System

The Conditional Access Micro is considered just another peripheral by the Dialog Processor. The Conditional Access Micro and Dialog

Processor communicate over a custom 2-way serial bus. This bus allows each micro to determine at what speed it can comfortably send and receive data, but each micro is required to constantly send data, whether it really has anything new to say or not.

The commands passed back and forth were designed to be independent of any Conditional Access Micro hardware. The Conditional Access Micro also takes care of any timing involved with command execution. For example, the Dialog Processor provides the Conditional Access Micro with two displays for each LED digit. The Conditional Access Micro drives the LED, automatically alternating between the two displays.

Tuning information is communicated to the Conditional Access Micro in the form of PLL data. This allows the tuning of any combination of channels. The Conditional Access Micro indicates to the Dialog Processor any keyboard action, as well as any commands sent via the in-band data channel. The task of receiving IR, a function which the original tuning micro had performed, has not been given to the Conditional Access Micro, but rather is done by the Dialog Processor. This speeds IR response by eliminating the Conditional Access Micro to Dialog Processor bus delay.

The Dialog Processor also has an out-of-band data receiver. Out-of-band data is data transmitted independently of the bandwidth of the channel tuned. The format of the in-band data is a function of the scrambling system. For previously existing systems, that format probably has no provision for loading a two week program guide. Even if the guide data can be fitted into the format, speed will be a problem. In the future, the in-band data rates capable with digital video may well overwhelm the Dialog Processor, but for the present even the "penny pinchers" in the marketing department agree that loading a 2 week guide using in-band data at 2 bytes per vertical is too slow.

SOFTWARE

The Need for Flexibility

At this point, we had a hardware system into which we could load and store a lot of data. The restriction was the framework hard wired into the Dialog Processor for displaying the data. Screen after screen could be displayed, but actions within a screen were not flexible. For example, consider the 2 screens shown in Figure 3.

The user hits "ENTER". Do we add channel 7 or 19 if we're highlighting the upper right corner of the screen? How did we get the upper right highlighted in the first place? To code the Dialog Processor to work with either screen is doable, but it has to be planned. Considering the amount of information to be made available, it was certain we would not cover all the options.

The solution is to load not only screens, but the behavior of the decoder with any given input. One option

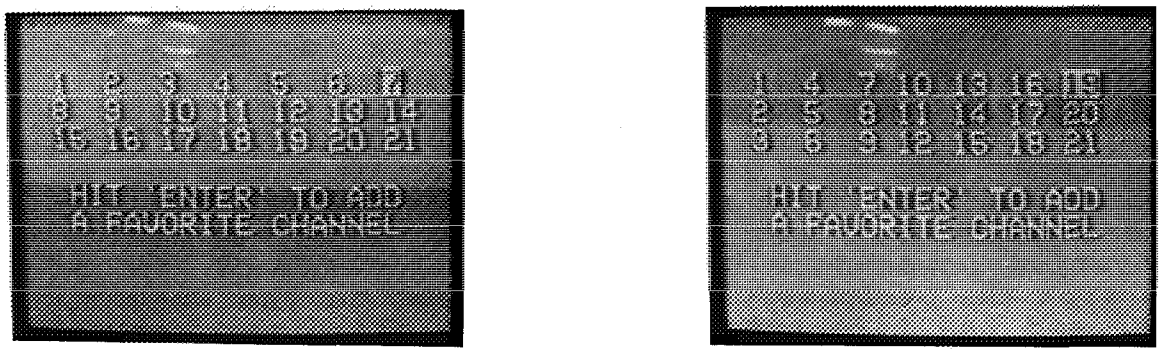


Figure 3
Two Screens for the Same Function

is to load the actual opcodes of the micro. This requires a micro which can run a program out of external memory. It also ties you to micros which will run that code. What about the possibility of loading the wrong code and locking the device in an endless loop? Lastly, who is going to support this code?

The Dialog

In order to avoid the mentioned pitfalls, but gain the needed flexibility, we developed our own "CATV decoder programming language". The Dialog Processor was programmed to interpret the language. The combination of on screen display data and behavior codes was called the Dialog, hence the name Dialog Processor. The interpreter is a firmware function of the Dialog Processor, the dialog to be interpreted is loaded via out-of-band data and stored in the external RAM.

It was necessary to develop an architecture, commands, and the associated tools for program development. There were also two programs to write, the interpreter and the dialog, neither of which could be reliably developed independently.

Downloading a Dialog

Hardcoded into the Dialog Processor is the task of out-of-band data reception and the associated memory management of the external RAM. Before being transmitted, the data is sorted into records which consist of a variable number of like sized files. Depending on what is presently in the RAM, the actual addresses into which data is stored will vary from decoder to decoder. Therefore, a dialog will refer to the Nth entry of record X to recover data. In addition to keeping track of where all downloaded data resides, the memory manager coalesces current data to lower addresses. Thus, all available memory resides in a block above an address which the memory manager calculates.

Interfacing the Dialog to the User

The main program loop of the Dialog Processor handles the hardware specific portion of communicating with the OSD chip, the Conditional Access Micro, and other periphery.

While the IR input is sampled and decoded by firmware, a special downloaded record is predefined to contain two IR formats. By loading this record, the decoder may be made to respond to up to 3 IR transmitters. The main loop updates a real time clock and 2 timers which are set by interpretive commands. The inputs from the periphery, IR inputs, and the time-out of the timers, are presented to the interpreter as transitions.

Interpreting a Dialog

Actions taken due to the presented transitions are determined by the dialog loaded via out-of-band data. The Dialog Processor only interprets the dialog, which is changeable.

For a "slide show" type of application, the interpreter is used as a state driven machine. Given the current screen and the transition, the next screen is known. This allows the dialog to change screens by simply supplying a pointer to the new screen. Referring to Figure 4, the screens represent the states. The state number is given in the upper right of the square. An arrow points to the next state, given a transition corresponding to the arrow's label. Any transition without an associated arrow has no effect on the state. Transitions may also be defined to cause an action regardless of the current state. "POWER", for instance, could turn the box off from any state.

This efficient way of stepping allows development of tools with which a nontechnical person may develop their own slide show. The new decoder has 32,512 states available for this type of application.

For any given input, a more complex action may be required. "CHANNEL UP", for instance, requires finding the next valid channel, and communicating to the Conditional Access micro new tuning and LED information. For these types of actions, interpretive commands were developed. The commands are an assembly language code which the interpreter runs.

The interpreter features a design to prevent a corrupted dialog from locking the micro. The dialog is also prevented from modifying itself.

From a dead start, at least minimal functionality was required from the decoder. This basic mode was written as a dialog and at start up it is loaded into the RAM. From that point it is executed as if it were a downloaded dialog. This was not necessarily the most efficient way to code the basic operation, but it allowed testing and enhancement of the interpreter using a real application.

In order to allow for personalized messaging, dialogs are sent with addressing information. A packet may be globally addressed, addressed to members of a specific group, or individually addressed. Each box has a unique address and can be a member of up to 8 groups. A packet may also be encoded and a decoder must be authorized to receive it. Packet trans-

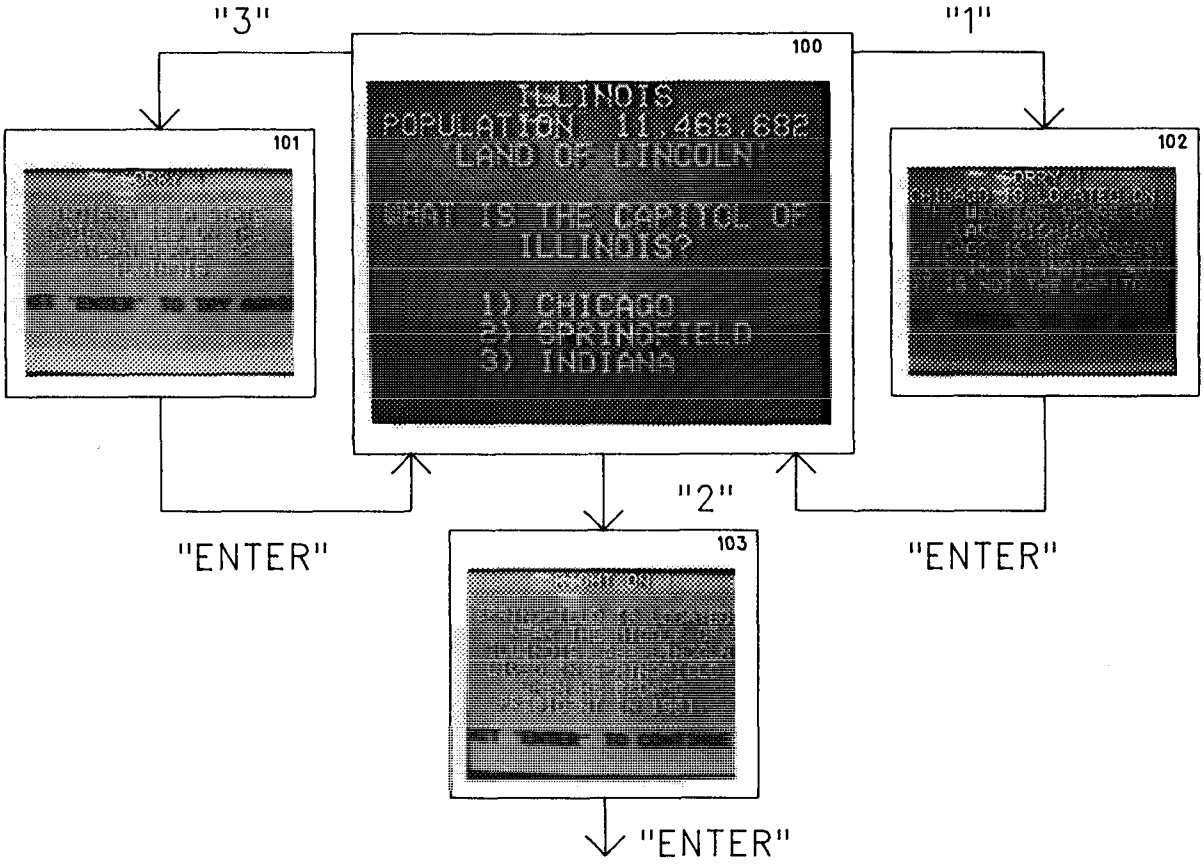


Figure 4
State Transition Diagram

mission includes a 16 bit CRC to prevent accepting bad data.

CONCLUSION

The end result is a field programmable cable box. The box contains 2 microcontrollers. One micro replaces the functions that a tuning micro traditionally performs, ie. tuner control, LED drive, keyboard scan, and any video descrambling control. The second micro has 3 main functions.

First, it manages a large external RAM into which data received from an out-of-band source is stored. This data is referred to as a dialog. Secondly, the micro performs housekeeping tasks of time keeping, and communication with peripherals. Housekeeping also includes collection and indicating the presence of certain events called transitions which may require a response. Thirdly, an interpreter, given a transition and the state in which the box currently sits, performs a corresponding series of instructions indicated in the dialog.

By using state transitions to page through screens, tools have been developed which offer the service provider the chance to design his own dialog. State transitions make efficient use of the

available RAM, and require only that pointers from state to state be supplied.

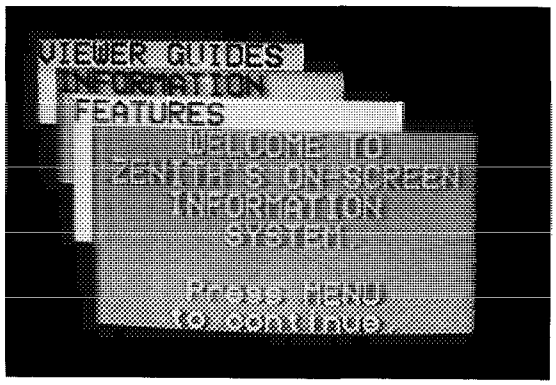
Though not something to be changed recklessly, the most basic operations of the box are downloadable.

The OSD displays a field of text 11 rows by 24 columns. By taking advantage of the colors and characters available, a fairly complex display may be generated. A bit mapped display of either 192 bits X 18, or 96 bits X 36 may also be displayed. A bit mapped display may be scaled horizontally to extend the display across the screen, but the vertical size may not be altered. Examples of a text and a bit mapped display are shown in Figure 5.

The box was designed for 2 weeks of program guides, but it can do more. Stock prices, scores, even download your own games.

A self diagnostic screen indicates both the presence and signal strength of in and out-of-band signals. ROM code revisions of the Conditional Access Micro and Dialog Processor, as well as the size of the external RAM are also displayed. The diagnostic screen is shown in Figure 6.

This design is the result of the work of the entire Zenith CATV department. I think we've designed what marketing really wanted, but were afraid to ask for. The content of the dialog will determine whether the decoder becomes a convenience for the user.



Text Field OSD

Bit Mapped OSD
Figure 5

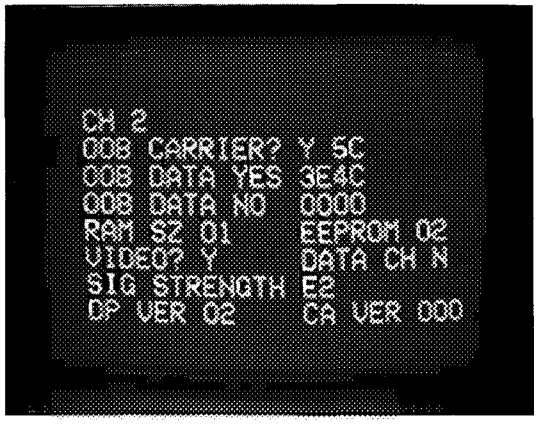


Figure 6
The Diagnostic Screen