G.PAK Packet Voice DSP System

Adaptive Digital Technologies, Inc.
October 11, 2001
Issue 1

Revision History

Issue	Revision History	Date	Initials
1	Initial Creation	10/11/2001	SDK

Table of Contents

1. Introduction

The economics of the business of telecommunications is drawing together the ubiquitous traditional telephone system and the exploding data networking world. The convergence of these two worlds of technology seems to have gotten a slow start compared to today's fast paced world of technology. In actuality, the voice-data convergence is moving at a pace that is normal for the more methodical traditional telecommunications world. Together, the economic pull and the methodical technical development will ensure that, in the end, voice-over-data will be a booming success.

While many products and systems involve the combination of diverse areas of technology and fields of expertise, the convergence of voice and data stands out in many ways. The conventional wisdom had professed that the bursty nature of packet data networks was well suited for the world of computers, but ill suited to continuous transmission of voice. The very idea that voice could be carried by data networks was revolutionary and counterintuitive. The thought that continuous speech could be passed through the erratic packet-based data networks was hard to grasp. The new concept required a change in mindset and shift in paradigms.

The companies making voice equipment had little to no knowledge of the data networking technology and the data networking companies had little to no knowledge of the voice technology. In order to develop converged voice and data products, these companies had to become educated quickly. Companies augmented their knowledge base through training, hiring, and outright acquisition depending upon how quickly they needed to bridge the gap. Convergence occurred on many levels.

Digital Signal Processing (DSP) is an enabling technology that has been at the heart of voice communications for decades. DSP technology is a fundamental building block in voice-over-packet communication. Among other things, Digital Signal Processing is used to compress speech signals in order to make efficient use of bandwidth. DSP is also used to cancel echoes in telephone systems, a problem that is magnified in voice-over-data systems.

Founded in 1994, Adaptive Digital Technologies, Inc. has been developing and licensing the Digital Signal Processing (DSP) technology that enables efficient digital voice transmission. In the early years, Adaptive Digital's products were deployed in satellite systems, cellular and other wireless telephone systems, digital circuit multiplexers, and voice recorders. Having this technology, Adaptive Digital Technologies was well positioned to participate in the voice-over-packet explosion.

At Adaptive Digital, we have encountered two broad classes of customers. Those who want to license DSP software components and those who want to license DSP software solutions. When we speak about a DSP software component, we are referring to a software implementation of a DSP algorithm such as a voice coder, echo canceller, tone detector, etc. A DSP component is viewed by a customer as a black box with well-defined inputs and outputs.

When a customer licenses a DSP component, the customer takes on the responsibility of integrating components to create a DSP solution that meets their particular needs. The customer therefore leverages our existing technology without having to reinvent it.

When a customer licenses a DSP solution, the customer generally does not want to be involved in the DSP software development at all. The customer wants to view the entire DSP chip as a black box with well-defined inputs and outputs. Up until now, Adaptive Digital has supported these customers by developing custom DSP solutions on a customer-by-customer basis. We

have been providing high quality custom DSP solutions successfully for many years. Our customers benefit not only from leveraging our highly optimized, robust DSP components, but also from leveraging our experience as DSP solution providers.

The drawback of custom solutions is that they are time consuming and labor intensive. This is the economic motivation for Adaptive Digital's G.PAK product. In a nutshell, G.PAK is a scalable and configurable DSP solution framework that provides sufficient flexibility to be used in nearly any voice application. G.PAK enables designers to configure their own custom DSP software solution using a menu driven configuration tool. Using G.PAK significantly reduces time to market by eliminating the need for a custom solution.

2. Voice-Over-Packet Systems

Before describing how G.PAK fits into the voice-over-packet world, it is instructive to provide an overview of voice-over-packet systems.

Traditional modern telephone systems digitize voice signals and send the resulting digital information using circuit-switched networks. When a person makes a phone call, a circuit is established between the two parties. This circuit is a dedicated connection for the duration of the phone call. The circuit carries digitized speech information continuously at a constant data rate using a fixed bandwidth data channel.

Packet data networks were designed to carry a different type of data – primarily computer data. Computers send and receive information as needed rather than continuously. It would therefore be inefficient to dedicate fixed bandwidth circuits between every pair of computers that needed to communicate. Instead, computers send packets of information as needed into a network. The network is responsible for routing packets to their intended destination. The actual communication channels within a network are shared amongst all packets. The amount of time it takes a packet to reach its destination depends upon the bandwidth of the network and the amount of network traffic. The delay in transmitting a packet from point A to point B is therefore not constant.

This non-uniform delay is one of a number of issues that must be circumvented in order to carry voice over packet data networks. It is necessary to make the flow of information appear continuous from end to end. Otherwise, the resulting speech will be distorted. A related issue is echo. As the end-to-end delay increases, echo becomes far more perceptible. The function of the echo canceller becomes far more important.

Another issue in carrying voice over packet networks is channel capacity. It is desirable to carry as many conversations over a given bandwidth network as is possible. Speech compression is therefore used heavily in voice-over-packet systems.

There are numerous variations of packet data networks such as IP, ATM, and Frame Relay. Nevertheless, there are some fundamental similarities between these network types and hence similarities in the functionality required to pass voice across these networks.

Figure 1 depicts the general architecture of a network that includes voice-over-packet capability.

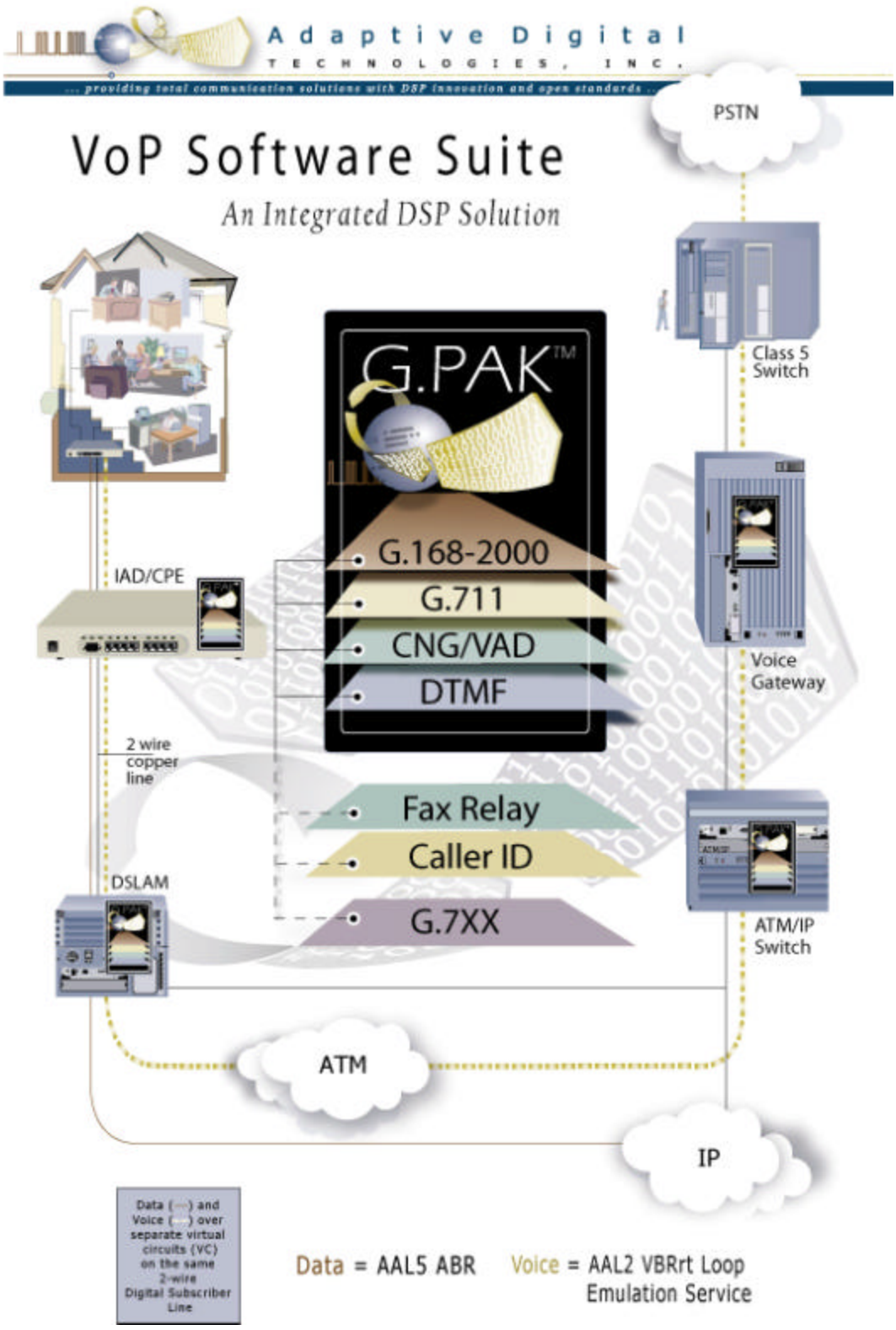

Figure 1: Voice-Over-Packet System

The two categories of equipment that we address are the access device and the gateway. The access device is located “near” the end user. It connects the user to a gateway, which is connected to other remote gateways via a packet network. Gateways are also connected to the Public Switched Telephone Network (PSTN) to enable users to make phone calls via the traditional network.

As can be seen in figure 1, G.PAK software is used in many pieces of the voice-over-packet network.

It is clear that voice-over-packet communications requires the translation between traditional digitized voice streams and packetized voice data. There are a number of channel configurations possible in systems of this type. These channel configurations therefore are supported by the G.PAK software and are discussed in the sections that follow.

2.1.1 PCM-Packet.

This routing links a PCM time slot with a network data stream. The encoding process of this link takes data from a PCM time slot, encodes and packetizes it and places the packetized data in a network data stream. The decoding process of this link takes packetized data from the network data stream, decodes it and places it onto the PCM time slot.

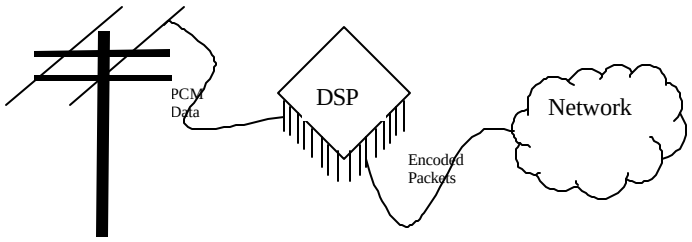

Figure 2: PCM to Packet Translation

2.1.2 PCM-PCM

This routing links a PCM time slot with a second PCM time slot. The encoding process of this link takes data from the first PCM time slot and places it on the second PCM time slot. The decoding process of this link reverses this transfer by taking data from the second PCM time slot and placing it on the first PCM time slot. This routing is somewhat trivial, but can be used for time slot interchanging or mu-law to a-law transcoding.

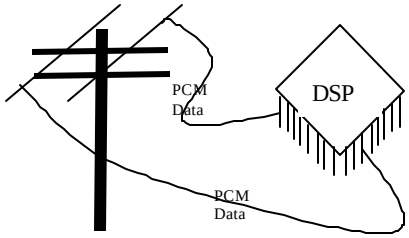

Figure 3: PCM to PCM Routing

2.1.3 N*PCM-Packet

This routing links N PCM time slots of circuit data (rather than voice) with a network data stream. The encoding process of this link takes data from N separate PCM time slots, multiplexes and packetizes it and places the packetized data in a network data stream. The decoding process of this link takes packetized data from the network data stream, demultiplexes it and places it onto the N PCM time slots.

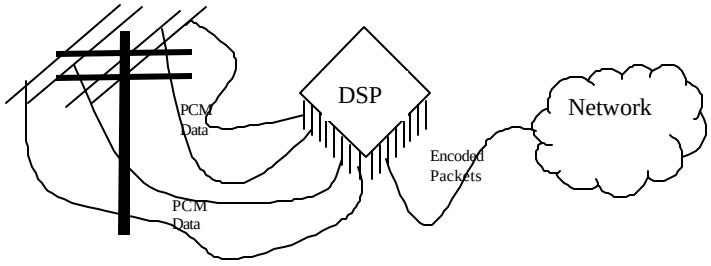

Figure 4: Circuit Data to Packet Data Translation

2.1.4 Packet-Packet

This routing links a network data stream with a second network data stream. In this case, the DSP translates one packet type to another packet type. This might involve translation of one media stream type to another media stream type. This translation can occur in a gateway that is connecting two users whose protocols are not compatible.

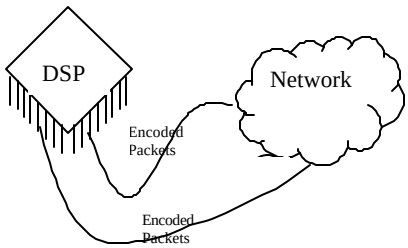

Figure 5: Packet to Packet Translation

3. Using G.PAK to Develop DSP Solutions

3.1 Overview

G.PAK is a scalable and configurable DSP solution framework that provides sufficient flexibility to be used in nearly any voice application. G.PAK enables designers to configure their own custom DSP software solution using a menu driven configuration tool.

Figure 6 depicts a typical G.PAK use in a voice-over-packet gateway application.

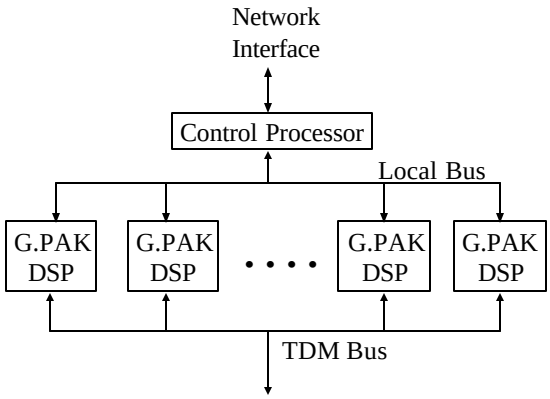

Figure 6: Typical G.PAK Application

The Control Processor is responsible for controlling the G.PAK enabled DSPs. At reset time, the Control Processor downloads the G.PAK software to the DSPs. The G.PAK control software provides an API that facilitates control of and communication with the G.PAK enabled DSP.

Each time it is necessary to initialize a voice connection, the control processor assigns a connection to one of the G.PAK enabled DSPs using a function in the G.PAK API. The G.PAK DSP is given the channel's configuration, which includes identifiers indicating the network port as well as the TDM port.

While a call is active, the control processor writes payloads to and reads payloads from the G.PAK DSP. Once again, G.PAK API functions are included for this purpose.

G.PAK consists of a set of DSP algorithm component libraries and a configurable software framework that integrates the algorithms into a complete DSP system. Figure 7 depicts the fully configured G.PAK software stack.

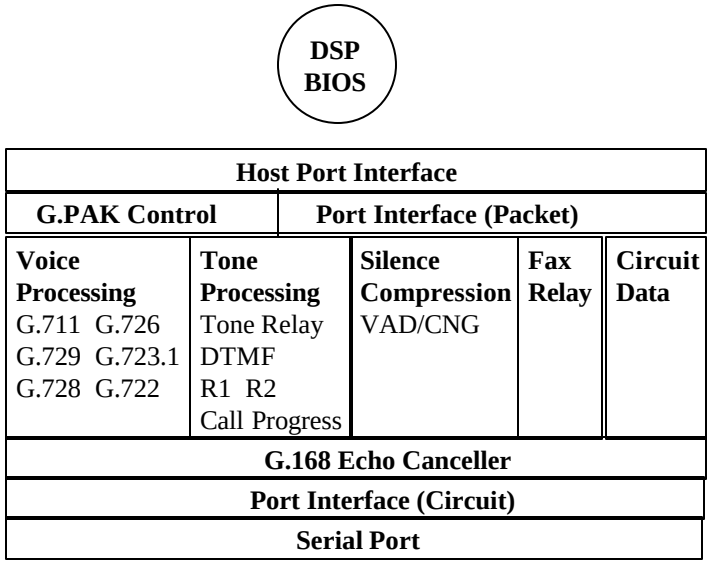

Figure 7: G.PAK Software Stack

Data flow diagrams of a fully configured G.PAK are shown in figures 8 and 9 below.

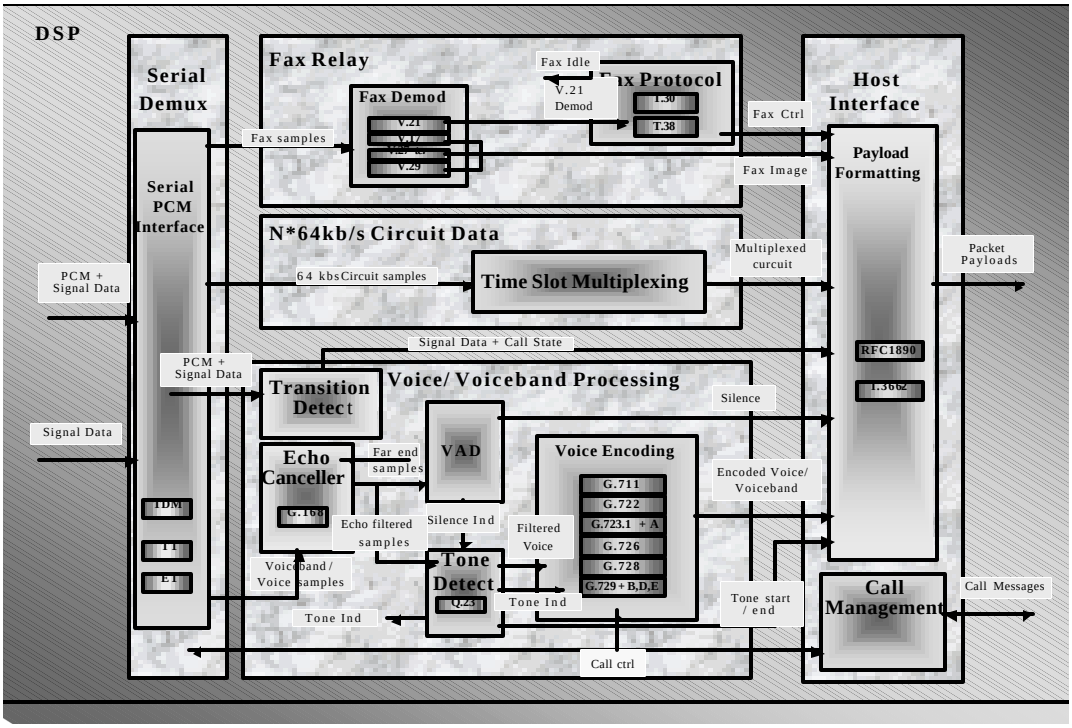

Figure 8: Call Encoding Block Diagram

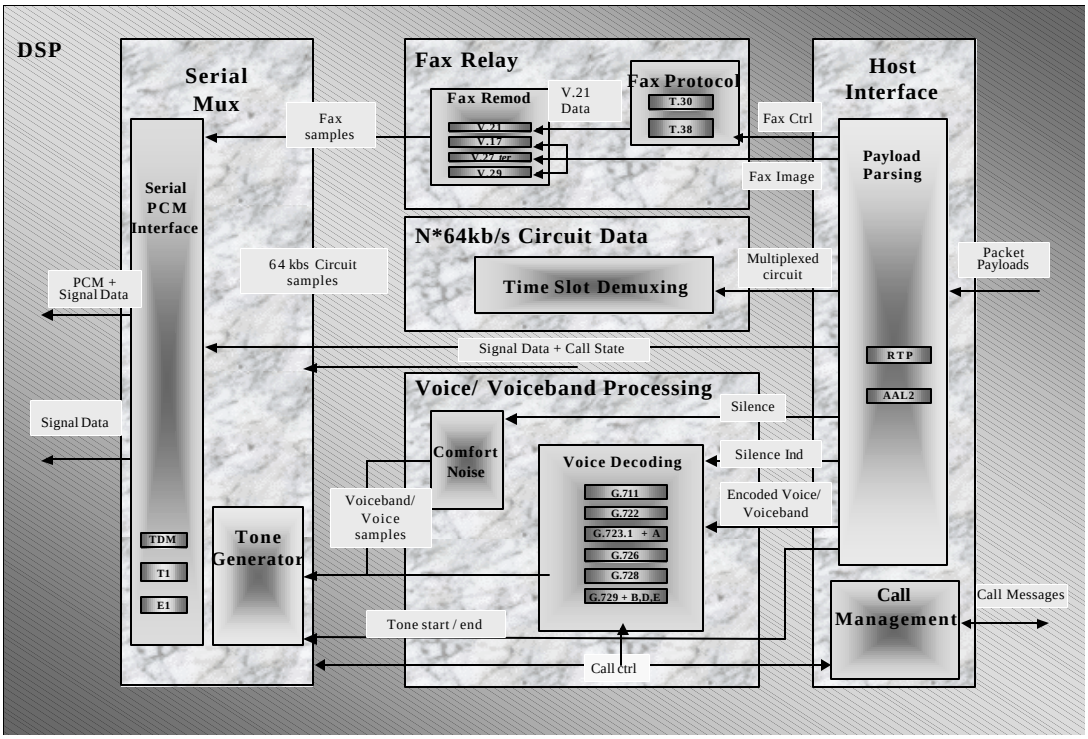

Figure 9: Call Decoding Block Diagram

3.2 Configuring G.PAK

G.PAK can be configured to include as many algorithms as are required for the user's application. By including only the necessary components, the DSP memory and CPU utilization are kept to a minimum. Stated differently, the channel density for a given DSP core is maximized.

The G.PAK configuration utility is a windows-based tool that allows the user to select the appropriate components and configuration. The tool then builds a custom DSP software image for the user's requested configuration. The software image is loaded into the DSP, usually via the DSP's host port interface using a control processor.

The configuration options are listed in table 1.

Configurable Options	Options
Processor	5402, 5409, 5410, 5416, 5421, 5441
Port A Type	TDM, Packet
Port B Type	TDM, Packet
Companding Mode	Mu-law, A-Law, Linear
Packet Profile	RTP, I366.2
G.168 Echo Canceller (Port A)	Enable/Disabled Long/Short Tail Tail Length Number of Reflectors Length of Reflector
G.168 Echo Canceller (Port B)	Enable/Disabled Long/Short Tail Tail Length Number of Reflectors Length of Reflector
G.726	Enable/Disable
VAD/CNG	Enable/Disable
AGC	Enable/Disable
Tone Relay	Enable/Disable
Tone Detection	Enable/Disable DTMF R1 R2 Call Progress
Tone Generation	Enable/Disable
G.729	Enable/Disable Annex A Annex B
G.723.1	Enable/Disable
G.728	Enable/Disable
G.722	Enable/Disable
Circuit Data	Enable/Disable
Signaling	Enable/Disable
Fax Relay	Enable/Disable

Table 1: Configuration Options

The configuration process is shown in figure 10.

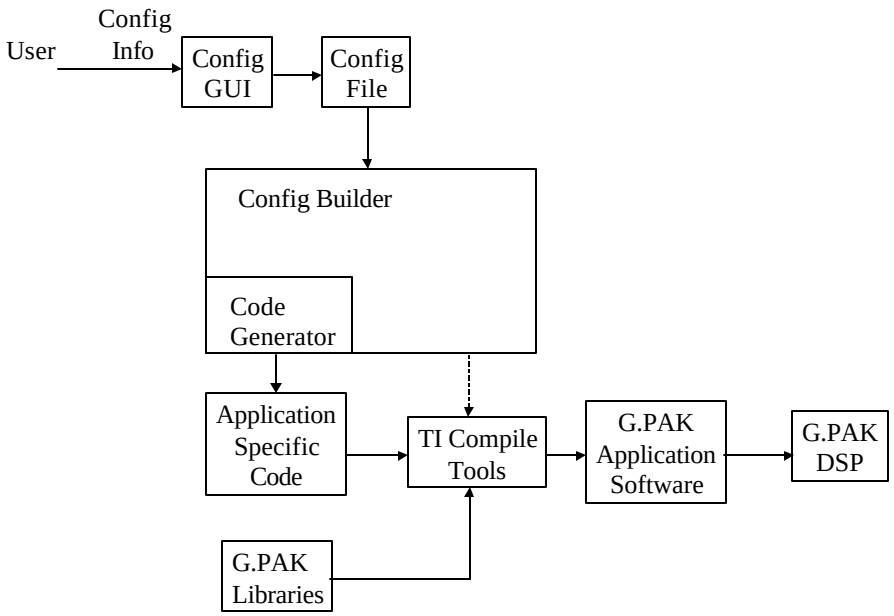

Figure 10: G.PAK Configuration Process

The user enters the application specific configuration using the G.PAK configuration program which is simplified with a graphical user interface (GUI). The GUI creates a configuration file containing all the user's configuration information. This configuration file is fed into the configuration builder, which in turn generates the application specific code. The configuration builder also orchestrates the building of the G.PAK application software using TI Compile Tools (compiler, assembler, and linker). The result is a G.PAK Application Software file, which can be downloaded into the G.PAK DSP in the target system. The whole procedure is automated under control of the configuration GUI.

3.3 Interfacing to G.PAK

Interfacing a host application to a G.PAK enabled DSP is facilitated by the G.PAK host API. This API runs on the control processor. The G.PAK host API enables the control processor to do the following operations on the G.PAK DSP:

- Set up and tear down channels
- Write and read packet payloads
- Write control information and read status information

Since the hardware interface between the control processor and DSP is left to the customer, the details of that interface are handled by a driver which is provided by the user. The function of the customer-supplied driver is needed to read and write DSP memory via the Host Port Interface. Figure 11 depicts the operation of the G.PAK API.

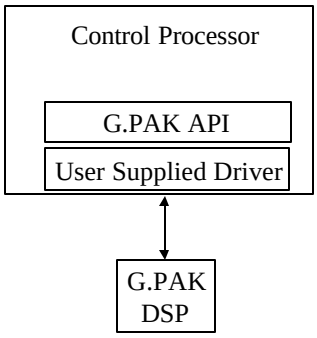

Figure 11: G.PAK API

The following pseudo-code is indicative of how a host application would use G.PAK. The G.PAK operations are facilitated by the G.PAK API.

Configuring a Channel:

```
gpakConfigureChannel( . . )     /* Configure a DSP's Channel. */
```

When a New Network Packet is Received:

```
gpakSendPayloadToDsp( . . )  /* Send a Payload to a DSP's Port. */
```

To Retrieve an Outbound Payload:

```
gpakGetPayloadFromDsp( . . ) /* Read a Payload from a DSP. */
```

Tearing Down A Channel

```
gpakTearDownChannel( . . )    /* Tear Down a DSP's Channel. */
```

Retrieving Status:

```
gpakGetSystemStatus( . . )      /* Read a DSP's System Status. */
gpakGetChannelStatus( . . )     /* Read a DSP's Channel Status. */
```