An Embedded Processor Based Data Acquisition System using Innovative Compression Algorithms and Flash Memory Technology

1) **Name:** **Vandana Verma**
Title: **Technical Marketing Engineer**
Company: **Intel Corporation**
Address: M/S CH10-82
5000 W. Chandler Blvd.
Chandler, AZ 85226
Telephone: (602)554-3928
Fax: (602)554-6167

2) **Name:** **Minda Zhang**
Title: **Sr. Software Engineer**
Company: **Intel Corporation**
Address: M/S CH10-21
5000 W. Chandler Blvd.
Chandler, AZ 85226
Telephone: (602)554-3759
Fax: (602)554-6167

Introduction

An Embedded Processor Based Data Acquisition System (EPBDAS) is a novice analog interface to the digital world. The Intel386EX embedded processor is used in the EPBDAS for compressing digital signal sampling files generated by the Analog-to-Digital (A/D) converter in the system. Flash memory, arranged in an array for nonvolatile data storage, is used for compressed digital signal file storage.

This paper presents an innovative compressed digital signal file format which allows $O(N\log_2 K)$ storage requirements for storing a size N digital signal sampling file with K different signal patterns in the file. This file format greatly increases the size of the data stored for signal processing.

EPBDAS System Components

A DAS is a system that is dedicated to the measurement and sampling of analog

signals for further processing. The analog input signals are preconditioned and translated to an encoded digital format. The signal is translated into the

analog-electrical domain and stored in the DAS.

The basic block diagram of our DAS is shown in figure 1.

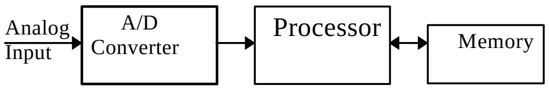

Figure 1. *Basic Block Diagram of a Data Acquisition System*

The analog-to-digital converter (ADC) translates the analog signal into an encoded digital format. The processor makes on-line programmable processing of the incoming data possible and it also supplies the clock information. Fulfilling the role of the recording device is memory.

The embedded processor is based on a PC platform and is used in conjunction with a timer, an interrupt controller, memory and the PC-104 extension bus. The embedded processor in the EPBDAS was chosen over a DSP processor so that General Software's Embedded DOS can be ported as the Real-Time Operating System (RTOS).

The low level hardware device drivers are provided by General Software's Embedded BIOS. This permits applications developed in the PC DOS environment to be run on the EPBDAS. The EPBDAS also has an environment that is totally compatible with a PC/DOS environment allowing the flash array to be hooked up with a PC so that the data stored in the flash array can be post processed on a PC. The embedded processor is obviously a more cost effective solution than a PC for the task of acquiring data.

The embedded processor was interfaced with the Flash array, our memory component in the DAS, via the PC-104 extension bus and Adtron's Input/Output (I/O) card as shown in Figure 2.

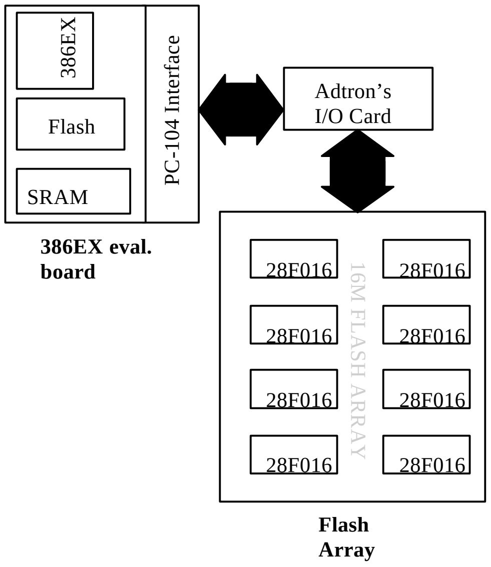

Figure 2. *Embedded processor to Flash Array interface*

We used flash memory in our DAS since flash memory is nonvolatile, electrically changeable, rugged and is offered in high densities. The Flash Array utilizes 16 Mbit memory components which have low-power operation and high read/write performance. The use of Flash memory to replace SRAM on the motherboard is explored.

Flash Memory Technology

The major categories in memory technologies are magnetic media, optical media, and solid state media. Solid state memory was developed to perform specific functions on chips. Data storage applications now have the choice of several different memory technologies. The original semiconductor memories were manufactured for either RAM or ROM functions and functioned as a binary storage device, storing on and off pulses of information in individual memory cells, memory cells being the smallest unit of solid state memory. Traditionally, EPROMs handled code and BIOS storage; disk drives stored the applications and user data files that get downloaded to system DRAM during execution. EEPROMs then replaced anything from SRAMs to EPROMs when flexibility and nonvolatility was required at low densities.

Flash memory demonstrates the technical ability to displace each of the existing memories to a varying degree. With properties that include nonvolatility and in-system updateability Flash memories utilization ties directly to cost and design considerations. Flash memory combines the high speed of DRAM, the nonvolatility of hard drives and floppy disk drives, the updateability of RAM or EEPROM, and the high density of ROM and is programmable on a bit-by-

bit resolution. Erasure is accomplished on a block-by-block level. A Command Register architecture results in SRAM like command write
timings to flash components. Fast access times make Flash memory an excellent choice for data storage.

Flash array

The fundamentals of the Flash array interface are defined by PCMCIA 2.0 specification. A linear mapped memory design is used for the Flash array so that the embedded processor would have direct access to the entire memory array. The hardware and software implementation of a linearly mapped memory addressing is simpler than most other memory mapping techniques.

The flash components used were Intel's 16 Mbit SmartVoltage^{TM}, the 28F016SV product. These components permit us to use a single power supply of 5V for Vcc and Vpp. The Flash Array consists of eight components. The array is highly standardized because it's electrical interface is according to PCMCIA 2.0 specifications. This supports various capabilities like automated write and erase operations, and reset. The Ready/Busy signal is used because it frees up the host system to perform additional tasks after initiating an operation.

Digital signal file compression

The data is stored in the flash array after it is compressed to optimize memory by removing redundancy from the digital signal data. The size of a digital signal file, where all the samples are recorded, is considerably larger than the number of different signal patterns in the file. Since all the digital samples generated by the ADC must be quantized to a discrete set of amplitude levels, and sampling must be large enough such that the sampling process will not result in any major loss of spectral information in the signal sequence. The compressed digital signal file format that is proposed is depicted in Figure 3.

File Header

The first *22* bytes of a compressed digital signal file is a file header which holds information specifying a magic number, the time and date at which this compressed file was created, the resolution order, and pointers. The resolution order makes it possible for signals $x(n)$ and $x(m)$ to be treated similarly, if their absolute difference is within the resolution order. The pointers point to the signal pattern area and checksum byte which is at the end of the file. The file header data structure is shown in Figure 4.

Pointers

Immediately after the file header is the section which contains all the pointers pointing to each recording block in the recording area of the compressed file. Each pointer uses *10* bytes to specify the offset, in bytes, of the first sample recording, in the specified recording block. As depicted in Figure 1, it defines a *unit* and *a size.* The *unit* specifies the number of bits used for each sample recording in the block pointed to, and *size* determines the total number of signal recordings in the block. The pointer data structure is depicted in Figure 5.

Recording Area

All the recording blocks are stored in the recording area shown in Figure 1. Although the signal samples in the different blocks use different units to record, they can be recovered from the signal pattern area. The data structure for

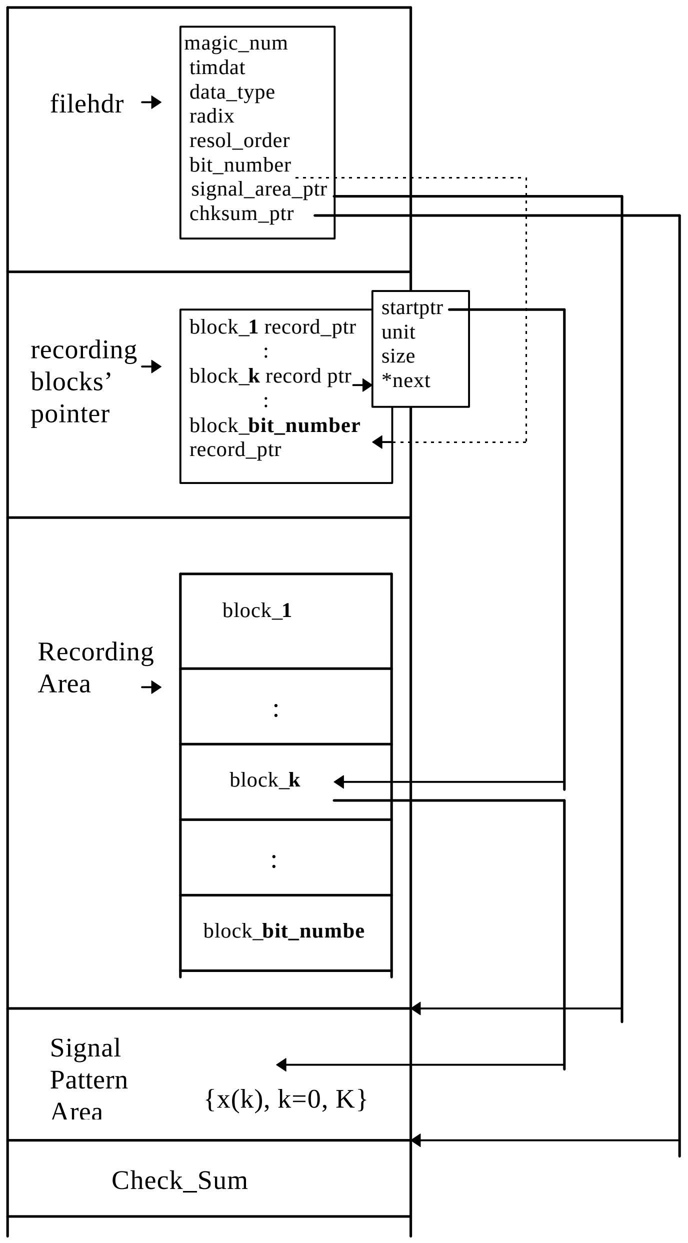

Figure3. *Compressed digital signal file format*

signal type and signal pattern are shown in Figure 6.

The compression algorithm which compresses a digital signal file requires *O(NK)* complexity, where *N* is the size of the diagonal file and *K* is the number of patterns in the file. The storage requirement for this compressed file is *O(N log K)*, assuming $N >> K$. The complexity could be greatly reduced, if the advanced technique is employed, such as branch prediction for comparing incoming signal samples with existing signal patterns.

It is worth pointing out that this compressed file format will greatly reduce the requirement for file storage. The following special case furnishes a good illustration of this.

```
struct  filehdr         {
   unsigned short   magic_num;
                    /*  magic  number  to  indicate  file
being               compressed */
   long    timdat;  /* # of seconds since GMT 00:00:00
                    Jan. 1, 1970 */
   unsigned short   data_type;
                    /* data type for each signal in original
                    file */
   unsigned short   radix;
                    /* radix position in fixed point
                    representation */
   unsigned short   resol_order;
                    /* the order of resolution parameter */
   unsigned short   bit_number;
                    /* the max number of bits needed for
a                   recording */
    long   signal_area_ptr;
                    /* offset of signal pattern area in this
                    file */
    long   chksum_ptr;
                    /* offset of check_sum for this
                    compressed file */
                    };

#define   FILHDR             struct filehdr
#define   FILHSZ             sizeof(FILHDR)
```

Figure 4. *File header data structure*

```
struct  record_ptr               {
                         /* recording block's pointer */
 long       startptr;
                         /* offset of starting  for current
                         recording block*/
 unsigned short          unit;
                         /* number of bits for each
                         recording */
 long       size;
                         /*   number   of   recordings   in
current                  block */
 struct record_ptr       *next
                         /* points to next signal recording
                         block */
                                  };

#define   RECPTR                  struct record_ptr
#define   RECPSZ                  sizeof(RECPTR)
```

Figure 5. *Recorder point data structure*

```
union     signal_type           {
   int    int_sig;   /* integer type */
   long   long_sig;/* long integer type */
   long   fix_sig;   /* 32_bits fixed point type */
   real   float_sig;/* 64_bit floating point type */
  double  double_sig;
                     /* 80_bit floating point type */
                                }

#define   SIGNAL               union signal_type

struct    sig_pattern          {
   SIGNAL            sig;
                     /*  signal  in  the  signal  pattern  area
*/
struct sig_pattern    *next;
                     /* points to next signal pattern */
```

Figure 6. *Signal data structure*

For instance, if a digital signal file records N signal samples $\{x(n), n=0, N\}$, each sample is a double floating point number which requires 80 bits, and signal samples have 16 different patterns. Thus the storage requirement for the original digital file is about $80*N$ bits. However, the compressed digital file will needs $O(N \log_2 K) = O(N* \log_2 16) = O(4*N)$ bits for the storage.

Summary

A highly integrated and cost effective DAS solution which allows for increased memory capacity as and when required, is proposed. The compression algorithm proposed can save storage space by a factor of 20x. Flash memory technology is the best choice since it is fast, nonvolatile in-system updateable and is available in high densities.

Filename:	ICPA_PAF.DOC
Directory:	C:\TESTDOCS\DOCS
Template:	C:\WINDOWS\WINWORD6\TEMPLATE\NORMAL.DOT
Title:	Subject: Abstract for ICSPAT '95
Subject:	
Author:	Stuart Levy
Keywords:	
Comments:	
Creation Date:	07/26/95 3:43 PM
Revision Number:	21
Last Saved On:	11/01/95 4:47 PM
Last Saved By:	Stuart Levy
Total Editing Time:	227 Minutes
Last Printed On:	12/18/95 5:27 PM
As of Last Complete Printing	
Number of Pages:	6
Number of Words:	1,393 (approx.)
Number of Characters:	7,943 (approx.)