ISSN 1993-9981 Методи та прилади контролю якості, № 19, 2007

УДК 681.3

ОСОБЛИВОСТІ ПРОГРАМУВАННЯ ПРИ РОЗРОБЛЕННІ ВІБРОДІАГНОСТИЧНИХ СИСТЕМ

В.А. Ровінський, О.В. Євчук

Івано-Франківський національний технічний університет нафти і газу, вул. Карпатська, 15, м. Івано-Франківськ, тел. (8-03422) 480-00, e-mail: victor_rovinskiy@yahoo.com

Рассмотрен способ повышения точности формирования временных интервалов при программировании модулей сбора и обработки вибродиагностической информации с использованием звукового кодека ЭВМ. Проведено сравнение точности формирования временных интервалов с помощью системных функций Windows и функций, синхронизированных ASIO-драйвером ЭВМ. Анализ результатов свидетельствует об преимуществах механизма синхронизации, используемого ASIO-драйверами по сравнению с обычными способами формирования временных интервалов.

There is considered a method for increasing the precision of time interval generating in the data acquisition and processing modules programming using computer audio codec. The precision of time interval generating using Windows system functions and ASIO driver synchronized functions was compared. The analysis of results shows the advantages of ASIO driver synchronization mechanism.

При проведенні вимірювань віброакустичних процесів на об'єктах нафтогазового комплексу за допомогою дослідних систем, ввід даних до ЕОМ може здійснюватись за допомогою аналогових кодеків, якими оснащені більшість плат вводу аналогової інформації сучасних комп'ютерів [1].

Практично усі сучасні ПЕОМ обладнані власними модулями АЦП/ЦАП, які використовуються для вводу і відтворення звукової інформації. Їх можна успішно використати для вводу і виводу дослідницької аналогової інформації, оминувши значні труднощі, створення А/Ц модулів. Це зумовлено тим, що високоякісне звуковідтворення потребує дотримання всіх вимог схемотехнічного проектування, включаючи добір компонентів, розробки топології друкованих плат із забезпеченням мінімальних перехресних зв'язків та написання драйверів, що гарантують синхронність та неперервність передачі потоків даних. Виготовлені в умовах масового виробництва звукозаписуючі і відтворюючі системи мають суттєво меншу вартість, ніж аналогічні промислові пристрої. Оскільки частотний діапазон таких пристроїв (від 5 Гц до 96 кГц) дозволяє проводити запис більшості віброакустичних процесів, то вони і можуть бути успішно використані для таких потреб.

Існує значна кількість програм, призначених для професійного звукозапису та звуковідтворення (наприклад, SpectraLAB, Nuendo, CoolEdit), які дозволяють здійснювати запис та попередню обробку даних в реальному часі. Після цього такі дані можуть бути збережені у вигляді текстових файлів формату ASCII для подальшого аналізу в середовищах типу MathCAD або MathLAB. Останній має вбудовані функції для роботи зі звуковими платами, однак, як показують проведені авторами спостереження, вони мають недостатню часову стабільність і при довготривалих записах даних можуть викликати втрату працездатності системи. Використання ж програм, призначених для професійного звукозапису, дозволяє здійснювати запис даних протягом багатьох годин. Недоліком використання типового програмного забезпечення є те, що за допомогою нього можна здійснювати тільки запис або відтворення коливних процесів, а також одержувати типові характеристики, на зразок спектра сигналу, одержаного за рахунок використання функцій швидкого перетворення Фур'є. Якщо ж потрібне застосування рідше вживаних функцій та базисів (Уолша, вейвлет-, або Z-перетворень), або необхідно, щоб програма додатково виконувала керування зовнішнім обладнанням, доводиться створювати спеціальне програмне забезпечення. Деякі

особливості створення такого програмного забезпечення наведені в [2]. Подібним випадком, що породжує суттєві проблеми при програмуванні є створення програмного забезпечення для вібродіагностування та балансування роторів газоперекачувальних агрегатів, де крім синхронного запису вібросигналу, потрібний ще і запис сигналу від оптодавача, встановленого на роторі, що дозволяє вимірювати його кутове положення. В такому разі нагальною є проблема вимірювання часових інтервалів між спрацюваннями оптодавача. Цю проблему можна розв'язати за допомогою зовнішнього пристрою – вимірювача часових інтервалів, однак такий спосіб теж не можна назвати простим, оскільки потрібна фазова корекція записаних віброакустичних сигналів із опорними часовими інтервалами від оптодавача.

Однією з важливих проблем при створенні програмного забезпечення, які працюють із аналоговими кодеками в середовищі операційної системи (ОС) Microsoft Windows™, є забезпечення синхронізації потоків та процесів. Це є особливо важливим у зв'язку з тим, що ОС Windows не є операційною системою реального часу, однак для неї створена велика кількість програмного забезпечення, в тому числі і для обробки сигналів. Ця операційна система реалізує принцип багатозадачності, що дозволяє імітувати одночасне виконання декількох процесів шляхом швидкого переключення між ними, яке здійснюється за допомогою диспетчера задач. Диспетчер задач Windows кожні ≈20 мс переглядає об'єкти ядра "потік" і відмічає ті з них, які можуть одержувати процесорний час. Далі відбувається вибір одного з таких потоків, завантаження його контексту в регістри процесора, після чого відбувається короткочасне виконання цього програмного потоку. [3]. Загальна проблема полягає в тому, що аналіз потоків відбувається через приблизний, а не точний час 20 мс. Це призводить до того, що реально управління деякому потоку може передаватись із затримкою в 70-100 мс, причому порядок передач управління потокам із однаковим пріорітетом не є почерговий. Наприклад, для потоків з номерами 1,2,3 виклики можуть виглядати так: 1,1 ,3 ,2,2, 3, 3,3, 1,2,2,1, ... , тобто за деякий період часу середня кількість викликів для кожного з потоків є однаковою, але порядок викликів –псевдовипадковий.

При створенні програмного забезпечення для вібродіагностичних систем, які орієнтовані на використання ОС Windows, зручним є використання механізму ASIO-драйверів, який реалізований у вигляді бібліотек (SDK), створених фірмою Steinberg (ФРН) [4]. Бібліотеки реалізовані за допомогою Microsoft Visual C++ і забезпечують більш точну витримку часових інтервалів при роботі із аудіокодеками. Вбудовані в звукові плати апаратні таймери є незалежними джерелами часових інтервалів і можуть бути використані при часових вимірюваннях. Робота з апаратурою розміщеною на звукових платах є значно простішою у випадку використання бібліотек ASIO SDK. Принцип дії механізму може бути проілюстрований за допомогою напрямленого графа станів скінченного автомата (рис.1).

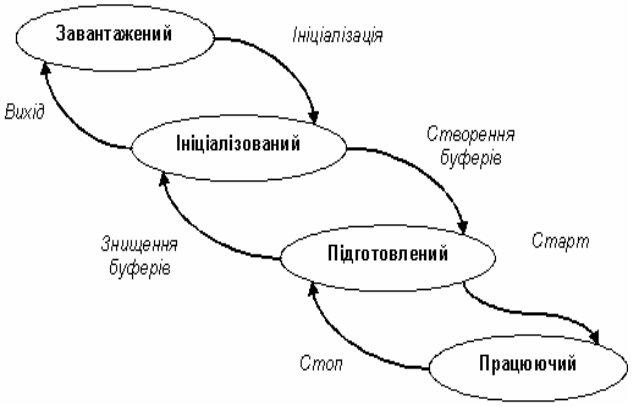

Рисунок 1 – Граф станів скінченного автомата

Можливі стани автомата (рис. 1):

Завантажений – код драйвера є доступним для зовнішніх програм;

Ініціалізований – драйвер завантажений програмою і може приймати від неї запити. Апаратна частина при цьому не обов'язково може бути ініціалазованою;

Підготовлений – аудіобуфери сформовані, а програмне забезпечення підготовлене для роботи;

Працюючий – апаратна частина функціонує і відбувається обмін даними.

Команди переходу між станами (рис. 1):

Ініціалізація – Ініціалізація драйвера для використання зовнішніми програмами за допомогою виклику функції ASOIInit();

Створення буферів – Виділення пам'яті для буферів даних та асоціації апаратних ресурсів для роботи з аудіо каналами. Здійснюється за допомогою виклику функції ASIOCreateBuffers();

Старт – Запуск потокових процесів. ASIOStart();

Стоп – Зупинка потокових процесів. ASIOStop();

Знищення буферів – Деактивація апаратних ресурсів та знищення буферів даних в пам'яті ЕОМ. Здійснюється за допомогою функції ASIODisposeBuffers();

Вихід– Вилучення ресурсів виділених драйверу при ініціалізації. Виконується функцією ASIOExit().

Після ініціалізації ASIO-драйвера, програмному забезпеченню стають доступні наступні функції:

ASIOGetChannels() – запит на кількість доступних аудіоканалів;

ASIOGetBufferSize() – запит на розмір підтримуваних буферів даних;

ASIOCanSampleRate() – запит на можливість використання деякої частоти дискретизації;

ASIOGetSampleRate() – запит поточної частоти дискретизації;

ASIOGetClockSources() – запит переліку джерел тактування;

ASIOGetChannelInfo() – запит інформації про аудіоканал;

ASIOSetSampleRate() – задання частоти дискретизації;

ASIOSetClockSource() – задання джерела тактування;

ASIOGetLatencies() – запит значення затримок.

Потоки даних стартують після виклику функції ASIOStart(). Перед стартом апаратного забезпечення драйвер здійснює один або більше викликів функції bufferSwitch() або bufferSwitchTimeInfo() для заповнення перших вихідних буферів. Рівномірність та нерозривність потоків даних забезпечується наступним способом: вхідні і вихідні буфери даних складаються з двох частин (з індексами 0 та 1). Тому поки апаратне забезпечення записує або відтворює одну половину буферів, програма заповнює другу (при відтворенні) або зчитує частину (при запису) даних. Запит на виклик функції bufferSwitch() формує драйвер за сигналами від апаратної частини. Таким чином забезпечується часова рівномірність запису/відтворення даних. Синхронізація процесів проілюстрована в табл.1.

Таблиця 1 – Часові співвідношення при роботі ASIO-драйверів

№ виклику bufferSwitch()	0	1	2	3	4	5	6
Індекс буфера	0	1	0	1	0	1	0
Системний час (мсек)	2000	2000	2023	2046	2069	2092	2116
Позиція вибірки	0	1024	2048	3072	4096	5120	6144

Для аналізу синхронності роботи програмних часозадаючих систем авторами був проведений експеримент, який полягав у тому, що в багатопоточній програмі формувались дві часові події з періодом в 46 мсек – одна з використанням системної часозадаючої функції Sleep(), робота якої подібна до роботи програмних таймерів ОС, а інша – за рахунок викликів функції bufferSwitch(), ініціалізація яких відбувається під дією апаратних подій створених апаратно. На рис.2 наведені результати експерименту.

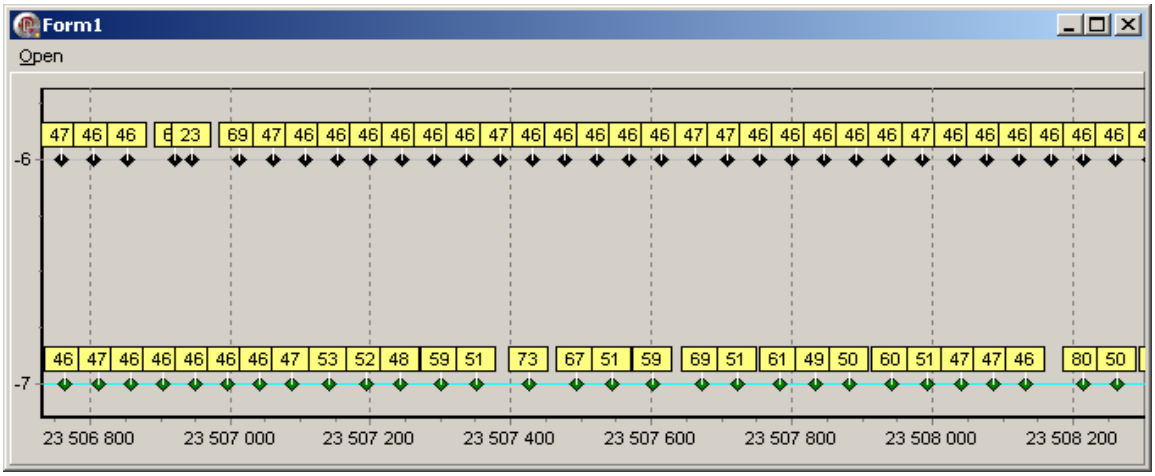

Рисунок 2 – Результати порівняльного експерименту з генерації часових інтервалів

На рис. 2 (рівень -6) – часові затримки в 46 мсек між двома викликами функції bufferSwitch(); (рівень -7) – часові затримки в 46 мсек, реалізовані програмними засобами Windows. По осі абсцис відкладений абсолютний час в мксек.

З рис.2 видно, що часові інтервали, сформовані за допомогою механізму ASIO-драйвера (рівень -6) є більш постійними по відношенню до програмних часових затримок (на рівні -7).

Спрощений приклад роботи з бібліотеками ASIO SDK:

```
// конструктор ASIO драйвера
AsioSample::AsioSample (LPUNKNOWN pUnk,
    HRESULT *phr)
: CUnknown("ASIOSAMPLE", pUnk, phr),
asioList()
{
        int j;
        lowernum=-1;

// вибір драйвера нижнього рівня
// із списка доступних драйверів
        int n=asioList.asioGetNumDev();
        for(j=0;j<n;j++)
        {
                if(lowernum==-1)
if(!asioList.asioOpenDriver(j,(void **)&pILower))
        {
                lowernum=j;
        }
}
        if(lowernum==-1)
    MessageBox(NULL,"No ASIO drivers found",
"ASIO error",MB_OK);
}

// функція створення буферів
ASIOError AsioSample::createBuffers
(ASIOBufferInfo *bufferInfos,
long numChannels,
long bufferSize, ASIOCallbacks *callbacks)
{
        ASIOError e=ASE_OK;
// збережемо адресу функції, наданої хостом:
        clb=*callbacks;
        to_low=clb;
// передамо драйверу нижнього рівня
// адресу нашої функції:
        to_low.bufferSwitch=&newBS;
e=pILower->createBuffers(bufferInfos,
        numChannels,bufferSize);

        return e;
}

void newBS(long index,ASIOBool pn)
// ця функція викликається періодично
// драйвером нижнього рівня
{
// необхідний нам корисний код:
// ...
// виклик функції, наданої хостом
AsioGlobal->clb.bufferSwitch(index,pn);

}
```

Як видно із наведеного вище, реалізація програмного коду не є складною і тому може бути рекомендована для використання при розробці дослідних систем акустичного та вібродіагностичного контролю, а також у випадках необхідності забезпечення часової синхронізації програмних процесів.

Література

1. Гук М. Аппаратные средства IBM PC. Энциклопедия. 3-е изд. – СПб.: Питер, 2006.- 1027с.

2. Євчук О.В., Ровінський В.А., Стрілецький Ю.Й. Використання аналогових кодеків для наукових досліджень // Методи та прилади контролю якості. – 2006, №17. – С.110-115.

3. Рихтер Дж.Windows для профессионалов: создание эффективных Win32 приложений с учетом специфики 64-разрядной версии Windows/Пер, англ - 4-е изд. - СПб; Питер; М.: Издательско-торговый дом "Русская Редакция", 2001. - 752 с.

4. Steinberg Audio Streaming Input Output Specification. – Steinberg Media Technologies GmbH, 2005. – 49p.