การเขียนโปรแกรมติดต่อกับ ET-PCI8255 V3
ตอนที่ 3 การส่งออกและนำเข้า

ศุภชัย บุศราทิจ
คณะเทคโนโลยีสารสนเทศ มหาวิทยาลัยราชภัฏเพชรบุรี

บทความตอนที่ 3 นี้เป็นบทความสุดท้ายสำหรับการเขียนโปรแกรมกับ ET-PCI8255 V3 เนื่องจากเป็นตัวอย่างที่ครอบคลุมการติดต่อกับการ์ดได้ครบในทุกด้านแล้ว นั่นคือ จะทำการอ่านข้อมูลจากพอร์ต PA3, PB3 และ PC3 ของ 8255 ตัวที่ 3 แล้วทำการส่งข้อมูลไปยัง PA1, PB1 และ PC1 ของ 8255 ตัวแรก โดยการเขียนโปรแกรมนั่นยังคงใช้งานตัวตั้งเวลาเหมือนกับบทความตอนที่ 2 และใช้ Microsoft Visual C++ .NET 2003 เป็นตัวแปลภาษา

หลักการทำงานของโปรแกรมตัวอย่างนั้นมีขั้นตอนสำคัญอยู่ 2 ส่วนคือ ส่วนกำหนดค่าเริ่มต้นการทำงานของ ET-PCI8255 V3 และส่วนของการอ่านและนำออกไปยังพอร์ต 8255 ตัวที่ 1 และ 3

รูปที่ 1 หน้าต่างของโปรแกรมตัวอย่าง

อุปกรณ์ประกอบการทดลอง

1. เครื่องคอมพิวเตอร์ส่วนบุคคลที่มีช่องเสียบการ์ดแบบ PCI เหลืออยู่จำนวน 1 ช่อง
2. การ์ด ET-PCI8255 V3
3. บอร์ด ET-Test I/O

การกำหนดค่าเริ่มต้นของ 8255

สิ่งที่ต้องทำคือกำหนดให้ 8255 ตัวแรกเป็นส่วนแสดงผล และ 8255 ตัวที่ 3 เป็นส่วนนำข้อมูลเข้า โดยอาศัยตารางที่ 1

จากตารางจะได้ว่า การกำหนดให้ 8255 ตัวแรกเป็นส่วนนำออกทั้งหมดนั้นเราจะต้องกำหนดค่าควบคุมเป็น 0x80 และการกำหนดให้ 8255 ตัวที่ 3 เป็นส่วนนำเข้าทั้งหมดจะต้องกำหนดค่าควบคุมเป็น 0x9B

ตารางที่ 1 ค่าควบคุมของ 8255

ค่าควบคุม	Port A	Port B	Port C^{HI}	Port C^{LOW}
0x80	output	output	output	output
0x81	output	output	output	input
0x82	output	input	output	output
0x83	output	input	output	input
0x88	output	output	input	output
0x89	output	output	input	input
0x8A	output	input	input	output
0x8B	output	input	input	input
0x90	input	output	output	output
0x91	input	output	output	input
0x92	input	input	output	output
0x93	input	input	output	input
0x98	input	output	input	output
0x99	input	output	input	input
0x9A	input	input	input	output
0x9B	input	input	input	input

การรับและส่งออก

ขั้นตอนของการรับและส่งออกเขียนเป็นโค้ดโปรแกรมได้ดังต่อไปนี้

```
unsigned char p3a, p3b, p3c;
p3a = pio32_read(PA3);
p3b = pio32_read(PB3);
p3c = pio32_read(PC3);
//...
pio32_write(PA1,p3a);
pio32_write(PB1,p3b);
pio32_write(PC1,p3c);
```

โปรแกรมตัวอย่าง

```
// etPCI8255demo4Dlg.h : header file
//

#pragma once
#include "afxwin.h"


// CetPCI8255demo4Dlg dialog
class CetPCI8255demo4Dlg : public CDialog
{
// Construction
public:
	CetPCI8255demo4Dlg(CWnd* pParent = NULL);
// standard constructor
```

```
// Dialog Data
	enum { IDD = IDD_ETPCI8255DEMO4_DIALOG };

	protected:
	virtual void DoDataExchange(CDataExchange* pDX);
// DDX/DDV support


// Implementation
protected:
	HICON m_hIcon;

	// Generated message map functions
	virtual BOOL OnInitDialog();
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	DECLARE_MESSAGE_MAP()
public:
	afx_msg void OnBnClickedButton1();
	CButton btnStart;
	int m_Timer;
	afx_msg void OnClose();
	afx_msg void OnTimer(UINT nIDEvent);
};
```

```
// etPCI8255demo4Dlg.cpp : implementation file
//

#include "stdafx.h"
#include "etPCI8255demo4.h"
#include "etPCI8255demo4Dlg.h"
#include ".\etpci8255demo4dlg.h"
#include "pio32.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CetPCI8255demo4Dlg dialog



CetPCI8255demo4Dlg::CetPCI8255demo4Dlg(CWnd* pParent /*=NULL*/)
	: CDialog(CetPCI8255demo4Dlg::IDD, pParent)
	, m_Timer(0)
{
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CetPCI8255demo4Dlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	DDX_Control(pDX, IDC_BUTTON1, btnStart);
}
```

```
BEGIN_MESSAGE_MAP(CetPCI8255demo4Dlg, CDialog)
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	//}}AFX_MSG_MAP
	ON_BN_CLICKED(IDC_BUTTON1, OnBnClickedButton1)
	ON_WM_CLOSE()
	ON_WM_TIMER()
END_MESSAGE_MAP()


// CetPCI8255demo4Dlg message handlers

BOOL CetPCI8255demo4Dlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Set the icon for this dialog.  The framework does this
automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here

	return TRUE;
// return TRUE  unless you set the focus to a control
}

// If you add a minimize button to your dialog, you will need the
// code below to draw the icon.  For MFC applications using the
// document/view model, this is automatically done for you by the
// framework.

void CetPCI8255demo4Dlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND,
reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}
```

```
// The system calls this function to obtain the cursor to display
// while the user drags the minimized window.
HCURSOR CetPCI8255demo4Dlg::OnQueryDragIcon()
{
	return static_cast<HCURSOR>(m_hIcon);
}

void CetPCI8255demo4Dlg::OnBnClickedButton1()
{
	// TODO: Add your control notification handler code here
	btnStart.EnableWindow(0);
	m_Timer = 0;
	m_Timer = SetTimer(1,100,0);
	if (m_Timer) {
		pio32_init(0xE400);
		pio32_write(PCC1,0x80);
		pio32_write(PCC3,0x9B);
	}
}

void CetPCI8255demo4Dlg::OnClose()
{
	// TODO: Add your message handler code here and/or call default
	if (m_Timer) {
		KillTimer(m_Timer);
		m_Timer = 0;
		pio32_write(PCC1,0x80);
		pio32_write(PCC2,0x80);
		pio32_write(PCC3,0x80);
		pio32_write(PA1,0x00);
		pio32_write(PB1,0x00);
		pio32_write(PC1,0x00);
		pio32_write(PA2,0x00);
		pio32_write(PB2,0x00);
		pio32_write(PC2,0x00);
		pio32_write(PA3,0x00);
		pio32_write(PB3,0x00);
		pio32_write(PC3,0x00);
	}

	CDialog::OnClose();
}

void CetPCI8255demo4Dlg::OnTimer(UINT nIDEvent)
{
	// TODO: Add your message handler code here and/or call default
	if (m_Timer) {
		unsigned char p3a, p3b, p3c;
		p3a = pio32_read(PA3);
		p3b = pio32_read(PB3);
		p3c = pio32_read(PC3);
		//...
		pio32_write(PA1,p3a);
		pio32_write(PB1,p3b);
		pio32_write(PC1,p3c);
	}
	CDialog::OnTimer(nIDEvent);
}
```

สรุป

จากบทความนี้ผู้เขียนคาดว่าคงเพียงพอต่อการนำโปรแกรมไปประยุกต์ใช้ต่อไปในอนาคต เพราะได้ครอบคลุมหลักการทำงานครบทั้งในส่วนของการรับเข้า ส่งออก และการทำงานโดยใช้ตัวตั้งเวลา ซึ่งล้วนเป็นต้นแบบสำคัญในการนำไปใช้

และเช่นเดิม บทความนี้สำเร็จลุล่วงได้ด้วยดีเนื่องจากการสนับสนุนของครอบครัวและทีมงานอีทีที โดยเฉพาะคุณกอบกิจ เติมผาติ ที่ได้ให้กำลังใจและส่งเสริมการเขียนบทความด้วยดีเสมอมา

ขอให้สนุกกับการเขียนโปรแกรมครับ