ELVITEC Sas
139, rue Philippe de Girard
84120 Pertuis
France

Tél : (33) 04 90 09 25 80
Fax : (33) 04 90 79 34 38
Web : www.elvitec.fr
Email : support@elvitec.fr

Acquisition Multicam 4.xx sous LabWindows™ CVI 7.0

Version du document: 041109_000_001
Article Technique : OUI
Application Démo : OUI
Code Snippet : OUI
Code Source (*) : OUI (* contrat Premium)

Résumé

Cet article présente la mise en place d'un noyau d'acquisition Multicam sous LabWindows™ CVI 7.0.
Cet article est valable pour toute la gamme des cartes Euresys.

Téléchargements

Un exécutable d'exemple est disponible en libre téléchargement.
Le code source de l'application (Projet CVI 7.0) est disponible dans le cadre d'un contrat premium

Article Technique

L'API Multicam présente l'avantage d'avoir une interface C, ce qui la rend compatible avec bon nombre de compilateurs du marché. Dans cet article nous nous pencherons plus particulièrement sur l'un d'entre eux à savoir LabWindows™ CVI, interface de développement de National Instrument. La version de démonstration utilisée pour réaliser les essais est la version 7.0.

Multicam version 4.x présente des mécanismes d' ADVANCED SIGNALING qui permettent de ne pas se contraindre au seul modèle d'acquisition via Call-Back qui, tout efficace et fonctionnel qu'il soit, ne couvre pas l'ensemble des besoins de développement, en particulier lors de la migration des applications depuis des environnements non-Multicam.

En résumé, le développeur dispose de 3 modèles :
1/ SIGNALING avec CALLBACK
2/ SIGNALING avec McWaitSignal
3/ SIGNALING avec l'API Windows classique WaitOn……Object()
1/ et 2/ sont totalement exclusifs l'un de l'autre.

L'implémentation la plus directe sous LabWindows™ repose sur le modèle numéro 2, avec la mise en place d'un Thread contrôlé avec l'API Multicam.

Nous séparerons dans cet article la partie acquisition de la partie affichage. L'acquisition est réalisée via l'API native et permet de récupérer un pointeur sur l'image acquise. L'affichage est quant à lui réalisé via les composants eVision™ (à l'aide des ActiveX).

Note : Nous ne pouvons utiliser l'API native d'EVision sous LabWindows™ parce qu'elle possède, contrairement à Multicam, une interface C++.

/***/
Initialisation de la voie d'acquisition
/***/

Le modèle d'initialisation est le même que sous un environnement « classique ».

```
MCHANDLE m_ChannelInstance;
MCHANDLE m_SurfaceInstance;


BOOL MultiInitialize ()
{
	MCSTATUS Status ;
	char pText[128];
	INT32 imageSize ;
	HRESULT hEvisionResult;

	// Establish communication with the driver
	Status = McOpenDriver (NULL);
	if (Status != MC_OK)return 0 ;

	// Create a channel
	Status = McCreateNm (EURESYS_SAMPLE_PROGRAM_CREATION_MODEL,
	&m_ChannelInstance);
	if (Status != MC_OK) return 0;

	// Associate a camera file to the channel
	Status = McSetParamStr (m_ChannelInstance, MC_CamFile,
	EURESYS_SAMPLE_PROGRAM_CAM_FILENAME);
	if (Status != MC_OK) return 0;

	// Associate a Multi board with the channel
	Status = McSetParamInt (m_ChannelInstance, MC_DriverIndex, 2);
	if (Status != MC_OK) return 0;

	// Enable event
	Status = McSetParamInt(m_ChannelInstance, MC_SignalEnable +
	MC_SIG_SURFACE_FILLED, MC_SignalEnable_ON);
	if (Status != MC_OK) return 0;

	// Retrieve image parameters
```

```
	Status = McGetParamInt(m_ChannelInstance, MC_ImageSizeY, &m_ImageSizeY);
	if (Status != MC_OK) return 0;

	Status = McGetParamInt(m_ChannelInstance, MC_ImageSizeX, &m_ImageSizeX);
	if (Status != MC_OK) return 0;

	// Check the pixel-size
	Status = McGetParamInt(m_ChannelInstance, MC_ImagePixelSize, &m_PixelSize);
	if (Status != MC_OK) return 0;

	// Create an image buffer
	imageSize = m_ImageSizeY * m_ImageSizeX * m_PixelSize;
	m_ImageBuffer = (PUINT32)malloc( imageSize);
	if (m_ImageBuffer == NULL) return 0;

	// Create a surface
	Status = McCreate (MC_DEFAULT_SURFACE_HANDLE, &m_SurfaceInstance);
	if (Status != MC_OK) return 0;

	// Set surface parameters
	Status = McSetParamInt(m_SurfaceInstance, MC_SurfaceSize, imageSize);
	if (Status != MC_OK) return 0;

	Status = McSetParamInt(m_SurfaceInstance, MC_SurfacePitch, m_ImageSizeX *
	m_PixelSize);
	if (Status != MC_OK) return 0;

	Status = McSetParamInt(m_SurfaceInstance, MC_SurfaceAddr, (INT32)m_ImageBuffer);
	if (Status != MC_OK) return 0;

	// Surface Context Information
	Status = McSetParamInt(m_SurfaceInstance, MC_SurfaceContext,
	(INT32)m_ImageBuffer);
	if (Status != MC_OK) return 0;

	// Associate surface with channel
	Status = McSetParamInst(m_ChannelInstance, MC_Cluster, m_SurfaceInstance);
	if (Status != MC_OK) return 0;

	// Set Channel to Idle state
	Status = McSetParamInt (m_ChannelInstance, MC_ChannelState,
	MC_ChannelState_IDLE);
	if (Status != MC_OK) return 0;

	// Enable image acquisitions
	Status = McStartAcq();
	if (Status != MC_OK) return 0;

	status= GetObjHandleFromActiveXCtrl (gPanel, PANEL_EC24IMAGE,
	&EvisionEC24ImageOCXHdle);

	return 1 ;
}
```

Note : ne pas registrer la fonction de callback puisque nous mettons en place un système d'acquisition à base de thread.

/***/
Mise en place du modèle
/***/

L'API LabWindows™ propose la mise en place de plusieurs modèles de thread. Nous ne détaillerons pas ici chacun d'entre eux et vous invitons à parcourir les documents relatifs à ce sujet sur le site de National Instrument (cf notamment « Easy Multithreading Programming in LabWindows™/CVI)

Le Thread utilisé dans notre exemple est un Thread pool.

Sa création se fait comme suit :

```
  CmtScheduleThreadPoolFunction (DEFAULT_THREAD_POOL_HANDLE,
                                 ThreadFunction1, NULL,
                                 &threadFunctionId1);

 /* Wait for the thread functions to finish executing */
   CmtWaitForThreadPoolFunctionCompletion (DEFAULT_THREAD_POOL_HANDLE,
                                           threadFunctionId1,
                                           OPT_TP_PROCESS_EVENTS_WHILE_WAITING);
```

CmtScheduleThreadPoolFunction fait référence à la fonction ThreadFunction1 dans laquelle nous assurerons les fonctionnalités d'acquisition

```
static int CVICALLBACK ThreadFunction1 (void *functionData)
{
MCCALLBACKINFO SignalInformation;
unsigned char* pData=NULL;
MCHANDLE hSurface;

   while (!gExiting)
   {
      SetCtrlVal (gPanel, PANEL_NUM1, i++);

        if (McWaitSignal (m_ChannelInstance,
                                    MC_SIG_SURFACE_FILLED,
                                    (UINT32)400,
                                    &SignalInformation)== MC_OK)
              {

              hSurface = (MCHANDLE)SignalInformation.SignalInfo;
              (void) McGetParamInt(hSurface,MC_SurfaceContext,(PINT32)&pData);

              // TRAITEMENT DES DONNEES ICI

              McSetParamInt(hSurface,MC_SurfaceState,MC_SurfaceState_FREE);
```

```
            }
        }

    return 0;
}
```

(void) McGetParamInt(hSurface,MC_SurfaceContext,(PINT32)&pData); permet de récupérer un pointeur sur l'image acquise, représenté ici par pData.

Note : Dans notre cas, il est de la responsabilité du développeur de fixer manuellement l'ETAT DE LA SURFACE à MC_SurfaceState_FREE , après le traitement des données d'acquisition.

/**/

Lancement de l'acquisition en mode « live »

/**/

```
BOOL MultiStartLive ()
{
        MCSTATUS Status ;

        // Set Channel to Idle state
        Status = McSetParamInt (m_ChannelInstance, MC_ChannelState,
        MC_ChannelState_IDLE);
        if (Status != MC_OK) return 0;

        // Set GrabCount to MC_INFINITE -> infinite acquisitions
        Status = McSetParamInt (m_ChannelInstance, MC_GrabCount, MC_INFINITE);
        if (Status != MC_OK) return 0;

        // Set Channel to Active state
        Status = McSetParamInt (m_ChannelInstance, MC_ChannelState,
        MC_ChannelState_ACTIVE);
        if (Status != MC_OK) return 0;


        return 1 ;
}
```

/**/

Quant à l'affichage…

/**/

Nous utilisons dans notre exemple les ActiveX eVision™ pour le display.

Initialisation des composants (notre exemple travaille sur une acquisition d'image couleur)

```
static  CAObjHandle EvisionEC24ImageOCXHdle;
```

Paramétrage dynamique de la taille du composant

```
status= GetObjHandleFromActiveXCtrl (gPanel, PANEL_EC24IMAGE,
&EvisionEC24ImageOCXHdle);

hEvisionResult = eVision__DEC24ImageSetImageWidth (EvisionEC24ImageOCXHdle, NULL,
m_ImageSizeX);

hEvisionResult = eVision__DEC24ImageSetImageHeight (EvisionEC24ImageOCXHdle,
NULL, m_ImageSizeY);
```

Récupération du pointeur image et rafraîchissement display dans le thread d'acquisition

```
if (McWaitSignal (m_ChannelInstance,
                  MC_SIG_SURFACE_FILLED,
                  (UINT32)400,
                  &SignalInformation)== MC_OK)
    {

    hSurface = (MCHANDLE)SignalInformation.SignalInfo;
    (void) McGetParamInt(hSurface,MC_SurfaceContext,(PINT32)&pData);

    // TRAITEMENT DES DONNEES ICI
    eVision__DEC24ImageSetImagePointer(EvisionEC24ImageOCXHdle,
    NULL,(long)pData);
    eVision__DEC24ImageRefresh(EvisionEC24ImageOCXHdle,NULL);

    McSetParamInt(hSurface,MC_SurfaceState,MC_SurfaceState_FREE);

    }
```

/**/

Arrêt de l'acquisition et destruction de thread

/**/

Arrêt de l'acquisition

```
BOOL MultiStopLive ()
{
    MCSTATUS Status ;

    // Set Channel to Idle state
    Status = McSetParamInt (m_ChannelInstance, MC_ChannelState,
    MC_ChannelState_IDLE);
    if (Status != MC_OK) return 0;

    return 1 ;
}
```

Destruction du channel

```
BOOL MultiTerminate ()
{
	MCSTATUS Status ;

	// Disable image acquisitions
	Status = McStopAcq();

	// Delete Channel Instance
	if (m_ChannelInstance)
	{
		Status = McDelete (m_ChannelInstance);

	}

	// Delete Surface Instance
	if (m_SurfaceInstance)
	{
		Status = McDelete (m_SurfaceInstance);

	}

	// Free allocated memory
	if (m_ImageBuffer)
		free(m_ImageBuffer);

	// Terminate communication with the driver
	Status = McCloseDriver ();

	return 1 ;
}
```

Destruction du Thread

```
CmtReleaseThreadPoolFunctionID (DEFAULT_THREAD_POOL_HANDLE, threadFunctionId1);
```

Commentaires :
Ce document est complété régulièrement, n'hésitez pas à nous envoyer vos commentaires, suggestions à l'adresse e-mail support@elvitec.fr . Merci.