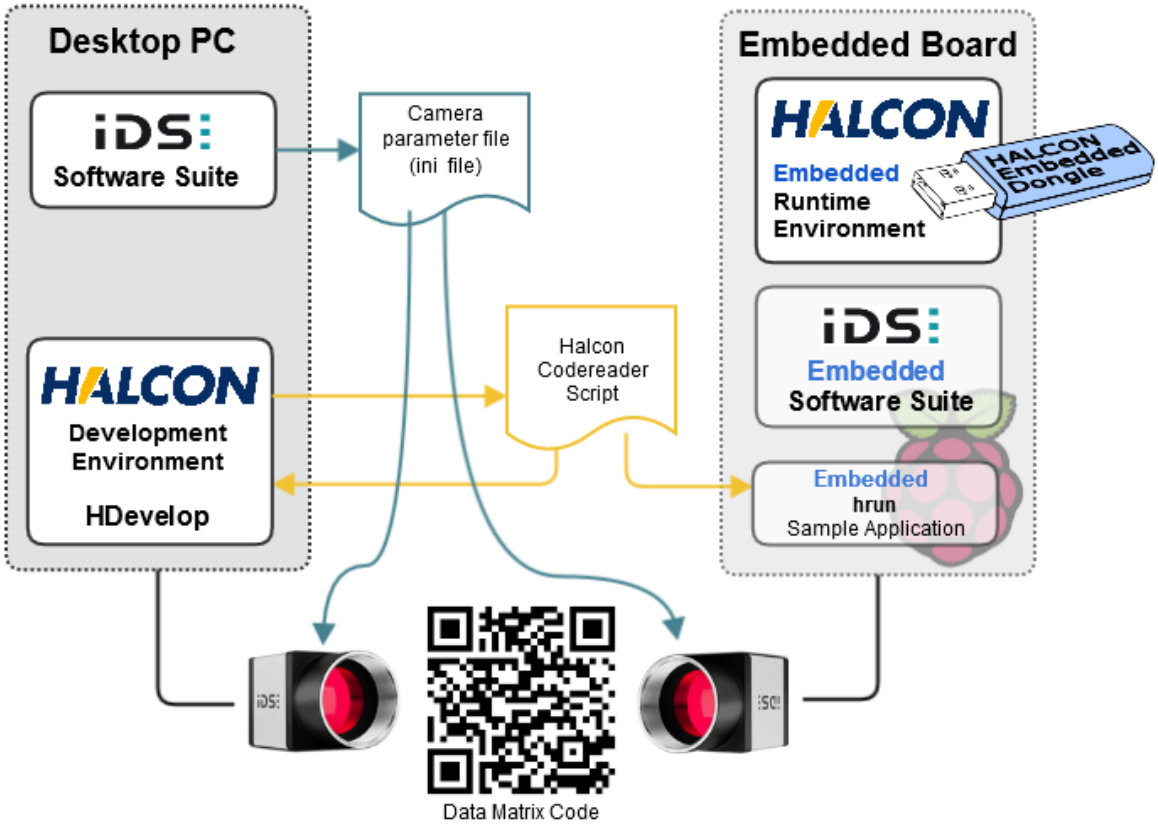
Rapid Prototyping mit HALCON Embedded - Codes lesen mit dem Raspberry Pi und einer IDS Kamera

Mal eben eine Bildverarbeitungsaufgabe auf einem Embedded Board testen? Diese Aufgabe muss nicht am aufwendigen Cross-Compilieren von Sourcecode scheitern. Mit bereits vorhandenen Komponenten der HALCON Embedded Installation und dem IDS Embedded Treiber lassen sich sehr schnell und einfach Machbarkeitsanalysen auf einem Embedded Board wie dem Raspberry Pi erstellen.

Das Lesen von Codes, wie QR, Atztec, ECC200, lässt sich zum Bespiel mit HALCON sehr leicht realisieren. HDevelop beinhaltet dazu einige Beispielskripte. Doch wie bekommen Sie diese Aufgabe auf einem Embedded Board zum Laufen und reicht die Performance der Embedded CPU dafür aus? Dieser Anwendungshinweis zeigt Ihnen Schritt für Schritt, wie Sie vorgehen, um einen einfachen Embedded Codeleser mit bereits vorliegenden Komponenten „zusammenbauen“ und Performancevergleiche mit einem Desktop-PC ziehen können.



Die Verwendung der HALCON Skript-Engine (**HDevEngine**) hat den Vorteil, dass Bildverarbeitungs-Skripte auf allen Plattformen verwendbar sind und die Bildverarbeitung damit plattformunabhängig und zudem über HDevelop erweiterbar bleibt. HALCON Skripte werden auf einem Entwicklungs-PC in HDevelop erstellt und auf dem Embedded Board über das bereits vorhandene Tool **hrun** ausgeführt. Dadurch müssen Sie bei diesem Ansatz selbst keinen Anwendungscode cross-kompilieren.

Voraussetzungen

Mit ein paar Vorkehrungen lässt sich diese, wie jede andere Embedded HALCON Anwendung sehr schnell auf einem Raspberry Pi 3 umsetzen. Sie benötigen folgende Dinge:

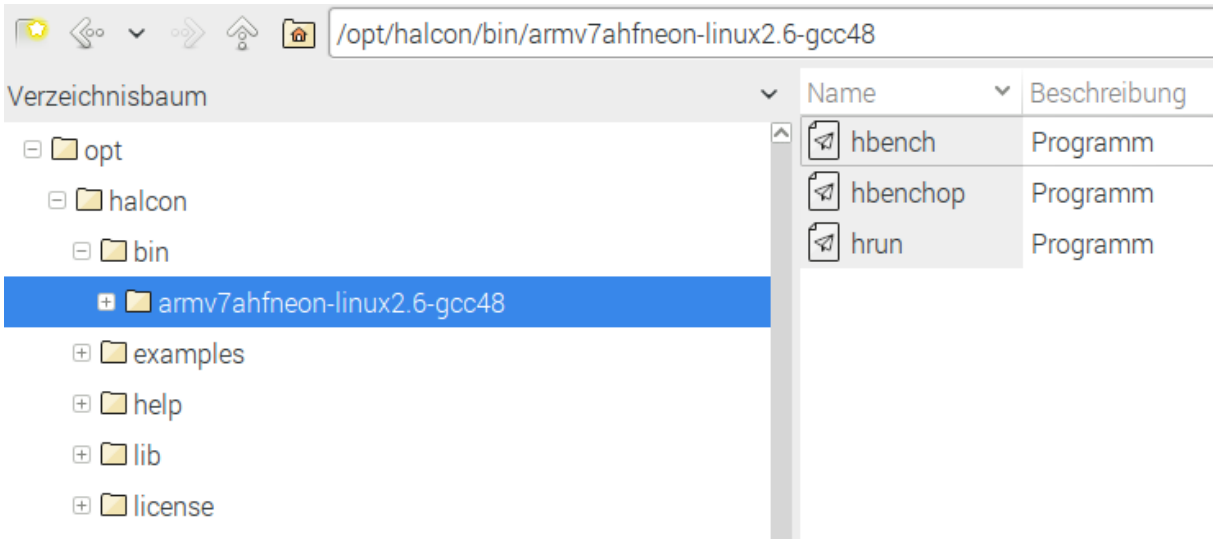
- Raspberry Pi (ab Version 2, ARMv7 Plattform kompatibel) fertig installiert mit Raspian und der Embedded HALCON Runtime Umgebung.
- Desktop PC mit installierter (lizensierter) HALCON Entwicklungsumgebung (Linux oder Windows)
- Für die Nutzung einer IDS Kamera muss der IDS Embedded Treiber auf dem Raspberry Pi installiert sein. (Embedded Treiber für IDS-Industriekameras)

Den HALCON Embedded Treiber bekommen Sie zur Evaluierung mitsamt einer USB-Dongle-Lizenz bei MVTec oder bei IDS.

HALCON Embedded installieren

Kopieren Sie das HALCON Embedded „Runtime“ Paket und entpacken Sie es 1:1 in einen Pfad auf der Zielplattform der als HALCONROOT dient. (z.B. unter `/opt/halcon`)

```
> sudo tar -xvf <paket> -C /opt/halcon
```



Um HALCON im System von Applikationen nutzbar zu machen, müssen ein paar Umgebungsvariablen definiert werden. Legen Sie dazu ein Profile-Script an, setzen Sie die entsprechenden (markierten) Variablen und kopieren Sie es zum Beispiel in Ihren HALCONROOT Ordner. (`/opt/halcon/.profile_halcon`)

```
# Sample shell script for <span class="sp-highlight-term themedarkblue
# sp-cursor-pointer">HALCON</span> environment settings
# (sh syntax)
# If you are using the Bourne shell source this file with the following
# command:
#  source .profile_halcon
HALCONARCH=armv7ahfneon-linux2.6-gcc48; export HALCONARCH
HALCONROOT=/opt/halcon; export HALCONROOT
HALCONEXAMPLES=${HALCONROOT}/examples; export HALCONEXAMPLES
HALCONIMAGES=${HALCONROOT}/examples/images; export HALCONIMAGES
PATH=${HALCONROOT}/bin/${HALCONARCH}:${PATH}; export PATH
if [ ${LD_LIBRARY_PATH} ] ; then
  LD_LIBRARY_PATH=${HALCONROOT}/lib/${HALCONARCH}:${LD_LIBRARY_PATH}; export
LD_LIBRARY_PATH
else
  LD_LIBRARY_PATH=${HALCONROOT}/lib/${HALCONARCH}; export LD_LIBRARY_PATH
fi
```

Um das HALCON Environment direkt beim Anmelden zu aktivieren, setzen Sie folgendes Kommando ans Ende der Profildatei Ihres Benutzerordners: `~/.profile`

```
source /opt/halcon/.profile_halcon
```

HALCON Embedded lizensieren

HALCON Embedded wird über eine Lizenzdatei mit einem zugehörigen USB-Dongle aktiviert. Dieses Lizenzpaar können Sie bei MVTec oder über den IDS Support anfordern. Die Lizensierung können Sie in der Datei */opt/halcon/readme_embedded.txt* unter dem Punkt „Licensing“ nachlesen.

- Kopieren Sie die Lizenzdatei *license.dat* passend zum Dongle in den Ordner `/opt/halcon/license` auf dem Zielsystem.
- Stecken Sie den USB Dongle in einen freien USB Port des Raspberry Pi und prüfen Sie mit dem Kommando *dmesg* auf der Linux Konsole, ob der Dongle mit der VendorID 1547 erkannt wurde.

```
[184728.639356] usb 1-1.4: new low-speed USB device number 5 using dwc_otg
[184728.748339] usb 1-1.4: New USB device found, idVendor=1547, idProduct=1000
[184728.748363] usb 1-1.4: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[184728.748376] usb 1-1.4: Product: SG-Lock USB Key
[184728.748389] usb 1-1.4: Manufacturer: SG-Lock
[184728.758162] hid-generic 0003:1547:1000.0002: hiddev0,hidraw0: USB HID v1.00 Device
.usb-1.4/input0
```

- Danach müssen Sie noch eine UDEV Regel im Linux System anlegen, um allen Prozessen Zugriff auf den Dongle zu gewähren. Legen Sie dazu die Datei /etc/udev/rules.d/**98-sglock.rules** mit folgendem Inhalt an:

```
ATTRS{idVendor}=="1547", ATTRS{idProduct}=="1000", MODE="666"
```

- Der UDEV Daemon muss jetzt neu gestartet werden, um die neuen Regeln zu aktivieren. Starten Sie dazu entweder den Pi einfach neu oder führen Sie folgendes Kommando für den RELOAD des Dienstes aus:

```
> sudo /etc/init.d/udev reload
```

- Mit dem Tool **„hbench“** testen Sie die HALCON Installation. Kommt es zu einem Fehler mit der Nr. 2036 weist dies auf ein Problem mit der Lizensierung hin. Kontrollieren Sie in diesem Fall nochmal die Schritte der Lizensierung.

HALCON Benchmark Tool

HALCON bringt mit dem Tool **hbench** ein eigenes Benchmark Tool mit, das Prozedur-Analysen auf dem Testsystem durchlaufen und bewerten kann. Es ist in jeder HALCON Installation (`<HALCON-ROOT>/bin/<HALCONARCH>/hbench`) für die entsprechende Plattform enthalten.

```
pi@raspberrypi:~ $ $HALCONROOT/bin/$HALCONARCH/hbench -operator
find_data_code_2d -reentrant
HALCON 12.0 Benchmark (v3.1)
============================
(computing on images of size 640x480)

OPERATOR                                reentrant
                                         time[ms]
Data code reading, ecc 200 (byte) ......    25.481
```

Abbildung 1 - HALCON Benchmark für Operator find_data_code_2d auf dem Raspberry Pi 3

```
C:\Program Files\MVTec\HALCON-12.0\bin\x64-win64\hbench.exe -operator
find_data_code_2d -reentrant
HALCON 12.0.2 Benchmark (v3.1)
==============================
(computing on images of size 640x480)

OPERATOR                                reentrant
                                         time[ms]
Data code reading, ecc 200 (byte) ......     4.811
```

Abbildung 2 - HALCON Benchmark für Operator find_data_code_2d auf einem Core i7 Windows PC

Der Benchmark des HALCON Operators zum Lesen von 2D Codes (ECC200) zeigt deutliche Unterschiede in der Laufzeit auf verschiedenen Plattformen.

Einfaches bildbasiertes Code-Lese Skript

Schon mit einem einfachen **QR-Code** HALCON-Skript, das Beispielbilder einliest und auf beiden Systemen mittels der **HDevEngine** ausgeführt wird, können Sie die Unterschiede in der **Lese-Performance** nachweisen.

Beginnen Sie z.B. mit dem HALCON Beispielskript **qrcode_simple.hdev**. Sie finden es im Ordner `/examples/hdevelop/Applications/Data_Codes/`.
Erweitern Sie es um eine einfache Zeitmessung des Lesevorgangs und einer Ausgabenachricht.

```
* Step 2: Read the data codes
* ------------------------------------------------
* Search and read the data codes in each image and
* display the decoded string for each found data code
for Index := 1 to ImageNum by 1
    read_image (Image, ImageFiles + Index$'.2d')
    count_seconds(Start)
    find_data_code_2d (Image, SymbolXLDs, DataCodeHandle, [], [], ResultHandles, DecodedDataStrings)
    count_seconds(End)
    Seconds := End - Start
    *
    * Display the results
    dev_display (Image)
    dev_display (SymbolXLDs)
    disp_message (WindowHandle, 'Image ' + Index + ' of ' + ImageNum, 'window', 12, 12, 'black', 'true')
    disp_message (WindowHandle, DecodedDataStrings, 'window', 40, 12, 'black', 'true')
    disp_message (WindowHandle, 'Zeit: '+(Seconds*1000) $ '.2f'+' msec', 'window', 68, 12, 'black', 'true')
```

Nachdem Sie das Skript mit HDevelop getestet haben kopieren Sie es auf den Embedded PC und führen es direkt mit dem Tool **hrun** aus. Dieses wird mit HALCON Embedded vorkompiliert ausgeliefert. Sie finden es im Ordner `<HALCONROOT>/bin/<HALCONARCH>/hRun`

```
pi@raspberrypi:~ $ hrun qrcode_simple.hdev
```

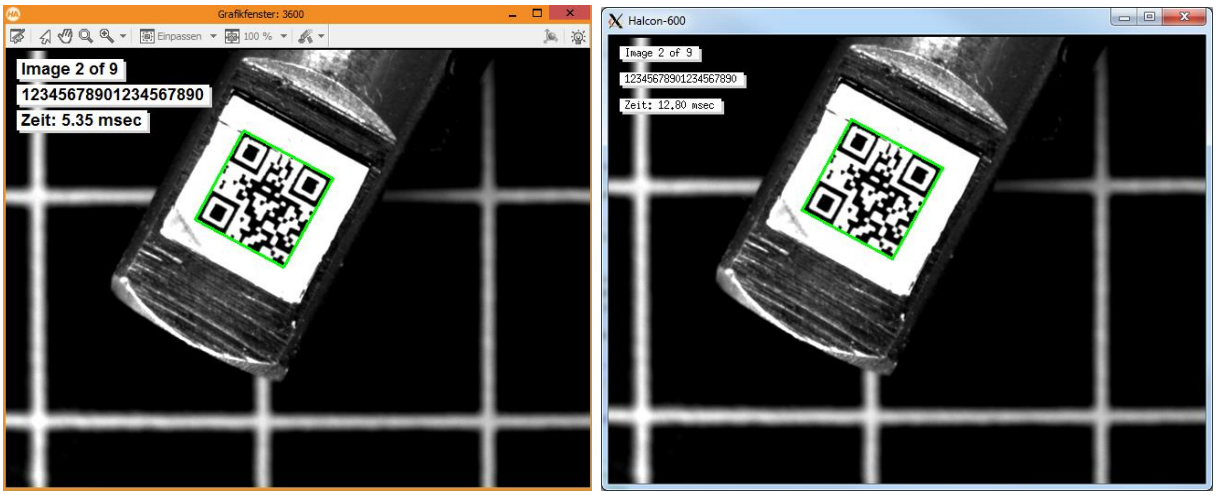


Abbildung 3 - Die Zeitmessung bestätigt den Performanceunterschied der beiden PC Systeme (links:Windows Desktop Core i7, rechts: Raspberry Pi 3)

Codes lesen mit einer IDS Kamera

Als nächstes „scannen“ Sie die 2D Codes mit einer uEye Kamera. Dazu muss der IDS Embedded Treiber auf dem Raspberry Pi installiert sein. (Embedded Treiber für IDS-Industriekameras)

Sie starten mit dem einfachen Skript *„datacode.hdev“*, das ebenfalls in der Embedded Installation von HALCON mitgeliefert wird. Sie finden es in dem Teil, der für den Entwicklungs-PC bestimmt ist.

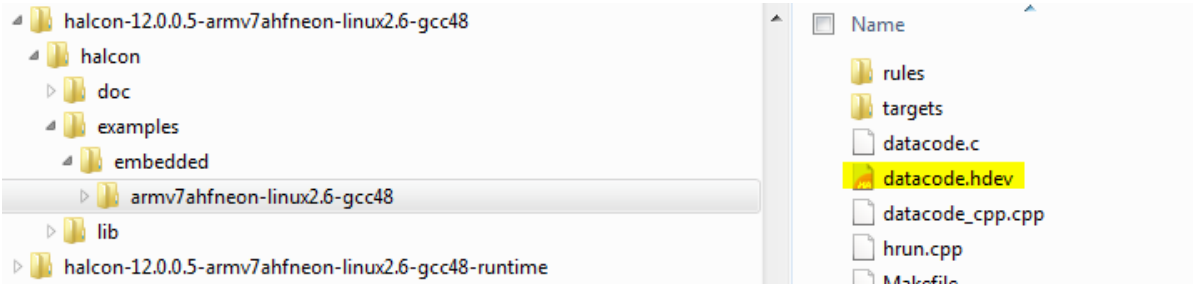


Abbildung 4 - DIe HALCON Embedded Installation beinhaltet das Skript datacode.hdev zum Lesen von 2D Codes

Um eine uEye Kamera damit einzusetzen, modifizieren Sie lediglich ein paar Zeilen Code. Sie benötigen nur einen einzigen „framegrabber“-Aufruf für alle Plattformen, anstatt hier zu unterscheiden, da sich die Kameras auf allen Plattformen identisch ansprechen lassen.

```
* get_system ('halcon_arch', HalconArch)
* if (HalconArch = 'armv7ahfneon-linux2.6-gcc46' or HalconArch = 'armv7aneon-linux2.6-gcc46' or HalconArch = 'armv7aneo
*     open_framegrabber ('Video4Linux2', 640, 480, 0, 0, 0, 0, 'default', -1, 'default', 'capture_format=YUYV', 'defaul
* else
*     open_framegrabber ('File', -1, -1, -1, -1, -1, -1, 'default', -1, 'default', -1, 'default', 'datacode/ecc200/ecc2
* endif
*
open_framegrabber ('uEye', 1, 1, 0, 0, 0, 0, 'default', 8, 'default', -1, 'false', '/cam/set1', '1', 0, -1, AcqHandle)

```

Abbildung 5 - uEye Kameras inkl. Treiber sind für beide Plattformen verfügbar und identisch anzusprechen

Sie können die Skriptzeilen für den Bildeinzug mit einer uEye Kamera auch über den HDevelop Assistenten „Image Acquisition“ einfügen. Wählen Sie dabei als Schnittstelle „uEye“ aus.

Danach ist das Skript sowohl auf Ihrem Desktop-PC als auch auf dem Raspberry Pi lauffähig. Kopieren Sie es auf den Embedded PC und führen es wieder mit dem Tool **hrun** aus.

```
pi@raspberrypi:~ $ hrun datacode.hdev
```

Als Ergebnis erhalten Sie ein Livebild der uEye Kamera. Wenn Sie Datamatrix Codes des Typs ECC 200 vor die Kamera halten, wird HALCON die Codes für Sie dekodieren und das Ergebnis anzeigen.

Ihr einfacher Embedded Codeleser ist damit funktionsbereit.

Tipps zum Bildeinzug

Da die **Konfiguration der Kamera** auf dem Embedded PC bzw. unter HALCON nicht besonders einfach oder komfortabel ist, bietet sich die Vorkonfiguration der Kamera auf dem Windows PC mittels des **uEye Cockpits** an. Hier haben Sie alle Möglichkeiten die Kamera bestmöglich einzustellen. Speichern Sie dann die Konfiguration am besten im Parameterset der Kamera. Mit HALCON können Sie diese Konfiguration dann verwenden. Das geht sehr einfach beim Öffnen des Frame Grabbers, indem Sie **‚/cam/set1'** für den Parameter **„CameraType"** angeben. Wenn Sie mehr über die Möglichkeiten der Kamerakonfiguration wissen wollen lesen Sie unseren TechTipp „Konfigurieren statt Programmieren: Der schnellere Weg zur Kameraeinstellung".

Da der Embedded- PC nicht die Leistung eines Desktop-PC aufweist, sollten Sie für Ihren ersten Test die Framerate der Kamera reduzieren. Versuchen Sie erst außerhalb von HALCON mit der Kamera ein stabiles Livebild zu erzielen, bevor Sie die Konfiguration mit der Bildverarbeitung einsetzen.

Tipps zum Aufbau der Embedded Entwicklungsumgebung

Die Einrichtung des Embedded Boards vom Aufspielen des Betriebssystem-Image über das Hin-und-her-Kopieren von Dateien mit dem Desktop-PC bis zur Ausführung von Programmen können Sie auf ganz unterschiedliche Arten erledigen. Der Raspberry Pi lässt sich mit dem aktuellen Raspbian-Image mittels SSH und VNC Server auch komplett „Headless" (ohne Bildschirm, Tastatur und Maus angeschlossen) über das Netzwerk betreiben. Ein paar Hürden sind dafür aber zu nehmen. Gerade in einem Firmennetzwerk kann schon die Netzwerkverbindung zwischen Raspberry Pi und Desktop-PC zu einer Herausforderung werden.

Der einfachste Weg zu beginnen, ist der klassische Weg den Raspberry Pi mittels eigenem Monitor, Maus und Tastatur einzurichten und die Daten (Programme, Skripte, HALCON Installation) über einen USB-Stick auf den Raspberry Pi zu kopieren. Für aufwendigere Entwicklungen ist das nicht sehr komfortabel aber für einen ersten Test, ist das schnell und einfach realisiert.

Weitere Anwendungshinweise und TechTipps finden Sie auf unserer Webseite unter https://de.ids-imaging.com/good-to-know.html

Autor

Heiko Seitz, Technischer Redakteur

Kontakt

IDS Imaging Development Systems GmbH
Dimbacher Straße 6-8
74182 Obersulm
Deutschland

Tel.: +49 7134 96196-0
E-Mail: marketing@ids-imaging.com
Web: www.ids-imaging.com

Anhang

Sourcecode „datacode.hdev“

Modifizierter Code des Skripts **datacode.hdev**, um die Codes mit einer uEye Kamera zu lesen. Die wichtigste Änderung ist eingefärbt. Um das Skript zu verwenden, kopieren Sie es in HDevelop und speichern es als *„datacode.hdev“.*

```
dev_update_off()
dev_close_window()

Width  := -1
Height := -1

close_all_framegrabbers ()
open_framegrabber ('uEye', 1, 1, 0, 0, 0, 0, 'default', 8, 'default', -1, \
                   'false', '/cam/set1', '1', 0, -1, AcqHandle)

create_data_code_2d_model ('Data Matrix ECC 200', [], [], DataCodeHandle)
grab_image_start (AcqHandle, -1)
while (1)
    grab_image_async (Image, AcqHandle, -1)
    if (Width = -1)
        get_image_size (Image, Width, Height)
        open_window (0, 0, Width, Height, 0, 'visible', '', WindowHandle)
    endif
    disp_obj (Image, WindowHandle)
    find_data_code_2d (Image, SymbolXLDs, DataCodeHandle, [], [], ResultHandles,\
                       DecodedDataStrings)
    if (|DecodedDataStrings|)
        set_color (WindowHandle, 'green')
        disp_obj (SymbolXLDs, WindowHandle)
        disp_message (WindowHandle, DecodedDataStrings, 'window', 12, 12, \
                      'black', 'true')
    else
        disp_message (WindowHandle, 'No ECC200 datacode found !', 'window', \
                      12, 12, 'red', 'true')
    endif
endwhile

close_window (WindowHandle)
close_framegrabber (AcqHandle)
```