IT WORKS
Everything works

IDWORKS Integrator

Developer's Guideline

V1.10 Update 08 Mar 04

บทนำ

- เป็น library สำหรับประมวลผลลายนิ้วมือ
- ประกอบด้วย 3 ส่วนคือ ส่วนการประมวลผลลายนิ้วมือ (TMWIDW.DLL) ส่วนช่วยจัดเก็บข้อมูลลายนิ้วมือ (FPENROL.DLL) และส่วนข้อมูลเครื่องอ่านลายนิ้วมือ (ITWSSL.DLL)
- ทำการรับลายนิ้วมือ ทำการประมวลผลและบอกค่าให้โดยอัตโนมัติผ่านทาง callback function
- ใช้ได้กับภาษาที่สนับสนุนการเรียก DLL เช่น C, Delphi, VB ฯลฯ

เตรียมตัวใช้งาน

- ติดตั้ง DigitalPersona Driver ลงในเครื่องก่อน
- เมื่อติดตั้งแล้ว ให้เสียบเครื่องอ่านลายนิ้วมือเข้ากับเครื่อง
- ไฟล์ DLL ที่จะใช้งาน ต้องอยู่ใน folder เดียวกับไฟล์ .exe ของโปรแกรมที่พัฒนา หรืออยู่ใน system folder ของ windows (เช่น c:\windows\system32)

เตรียมตัวใช้งาน (ต่อ)

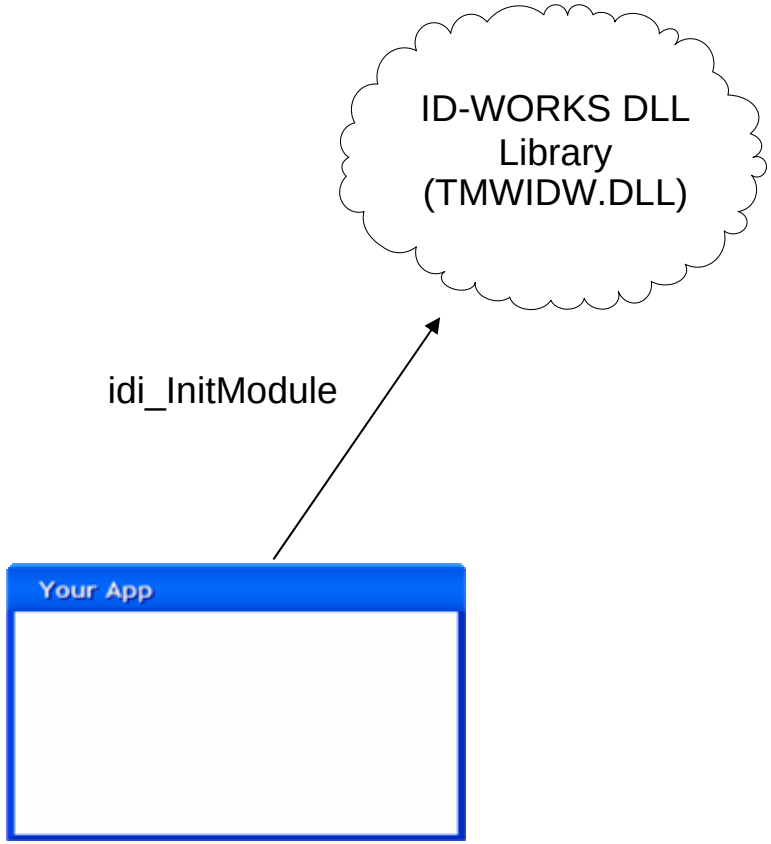
- ศึกษาเรื่องการใช้งาน DLL ในภาษาที่จะพัฒนา
- หากในชุดพัฒนาของเรา ไม่มี header file ที่สนับสนุนในภาษาที่ต้องการ จะต้องมีการแปลง header file ไปยังภาษานั้นๆ
- การแปลง header file ด้วยตนเอง อาจเกิดข้อผิดพลาดได้ง่าย และหาเจอได้ยากในบางครั้ง ดังนั้นในขั้นตอนการแปลงจึงควรระมัดระวังเป็นพิเศษ

IT WORKS
Everything works

ID-WORKS Integrator - ส่วนการประมวลผลลายนิ้วมือ

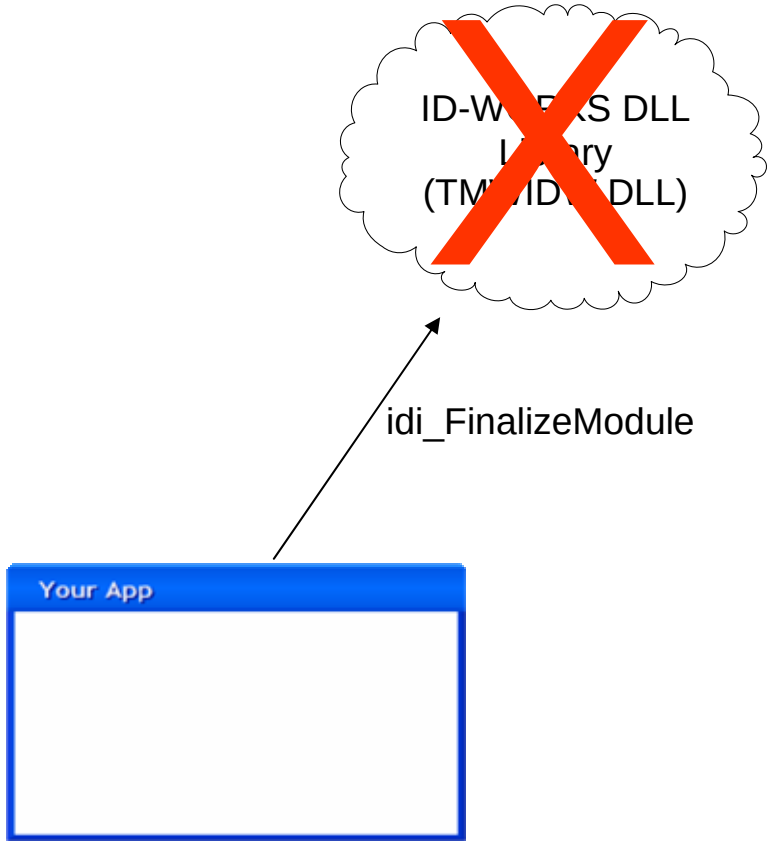
ส่วนการประมวลผลลายนิ้วมือ - เริ่มการใช้งาน

- เพื่อให้ library พร้อมใช้งาน ก่อนการใช้งานทั้งหมด จะต้องเรียก idi_InitModule เป็นขั้นแรก



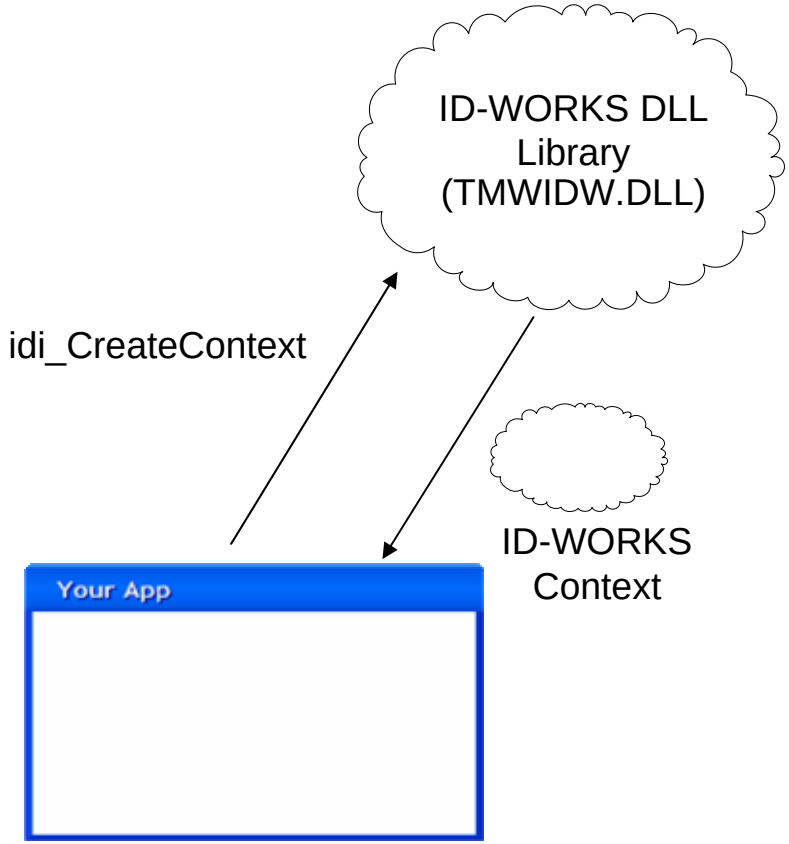
ส่วนการประมวลผลลายนิ้วมือ – จบการใช้งาน

- และหลังจากเลิกใช้งานแล้ว (เช่น ก่อนปิดโปรแกรมของคุณ) จะต้องเรียก idi_FinalizeModule
- หลังเรียกฟังก์ชันดังกล่าวแล้ว จะไม่สามารถเรียกใช้งานฟังก์ชันในโมดูลได้อีก



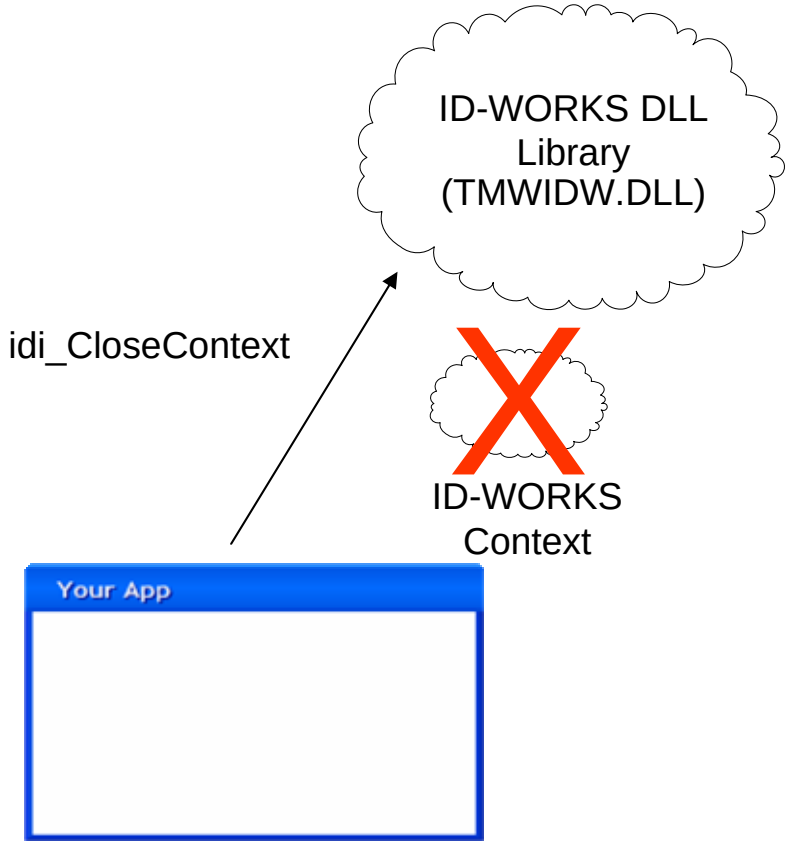
ส่วนการประมวลผลลายนิ้วมือ - Context

- Context เป็นเหมือนโมดูลย่อยที่เราสร้างขึ้นมาเพื่อใช้งาน ID-WORKS Integrator ในโปรแกรม โดยการใช้งาน function ต่างๆ จะต้องอ้างถึง context เสมอ
- การสร้าง context จะใช้ function idi_CreateContext



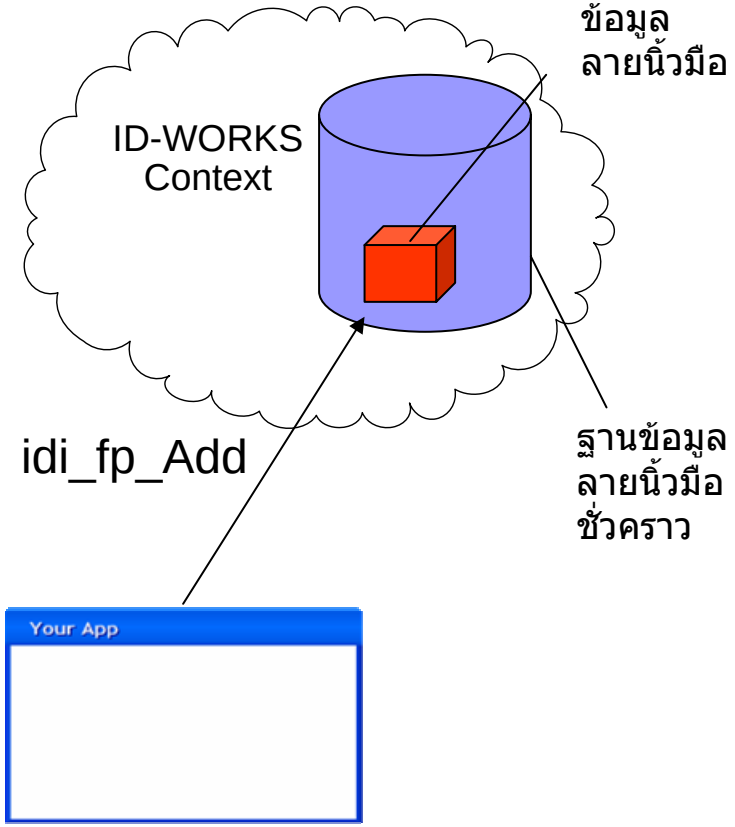
ส่วนการประมวลผลลายนิ้วมือ - Context

- เมื่อไม่ต้องการใช้งาน context แล้ว ให้ปิด context ดังกล่าว โดยการเรียก idi_CloseContext



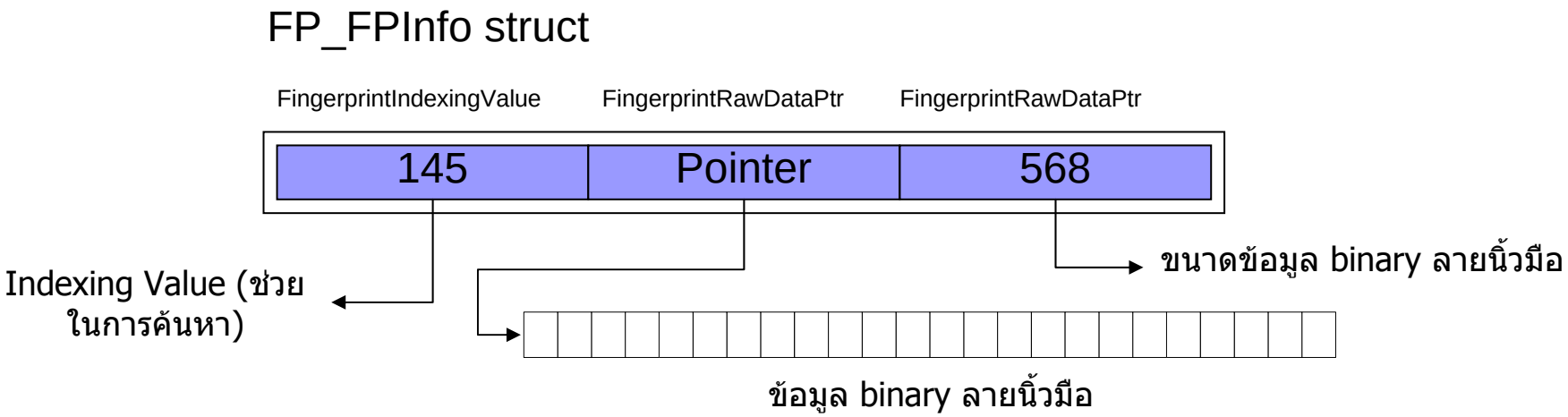
ส่วนการประมวลผลลายนิ้วมือ - ข้อมูลลายนิ้วมือ

- ในแต่ละ Context ของ ID-WORKS จะมีฐานข้อมูลลายนิ้วมือชั่วคราวเป็นของตัวเอง (จะหายไปเมื่อปิด context)
- ฐานข้อมูลจะถูกใช้เมื่อมีการแตะนิ้วมือ ID-WORKS จะทำการประมวลผลและค้นหาลายนิ้วมือจากฐานข้อมูลชั่วคราวนี้
- ดังนั้นผู้พัฒนา จึงต้องเพิ่มข้อมูลเข้าไปในฐานข้อมูลชั่วคราว เพื่อให้ ID-WORKS สามารถทำงานได้
- idi_fp_Add ใช้สำหรับเพิ่มข้อมูลลายนิ้วมือที่จัดเก็บไว้แล้ว เข้าไปในฐานข้อมูล



ส่วนการประมวลผลลายนิ้วมือ – ข้อมูลลายนิ้วมือ (ต่อ)

- ข้อมูลลายนิ้วมือ สามารถเก็บได้โดยใช้ส่วนช่วยการจัดเก็บลายนิ้วมือ
- ในข้อมูลลายนิ้วมือ จะประกอบด้วย 2 ส่วนหลักคือ ข้อมูลทั่วไป และข้อมูล binary ที่เป็นรายละเอียดของลายนิ้วมือ ซึ่งสามารถนิยามได้โดยใช้ FP_FPInfo

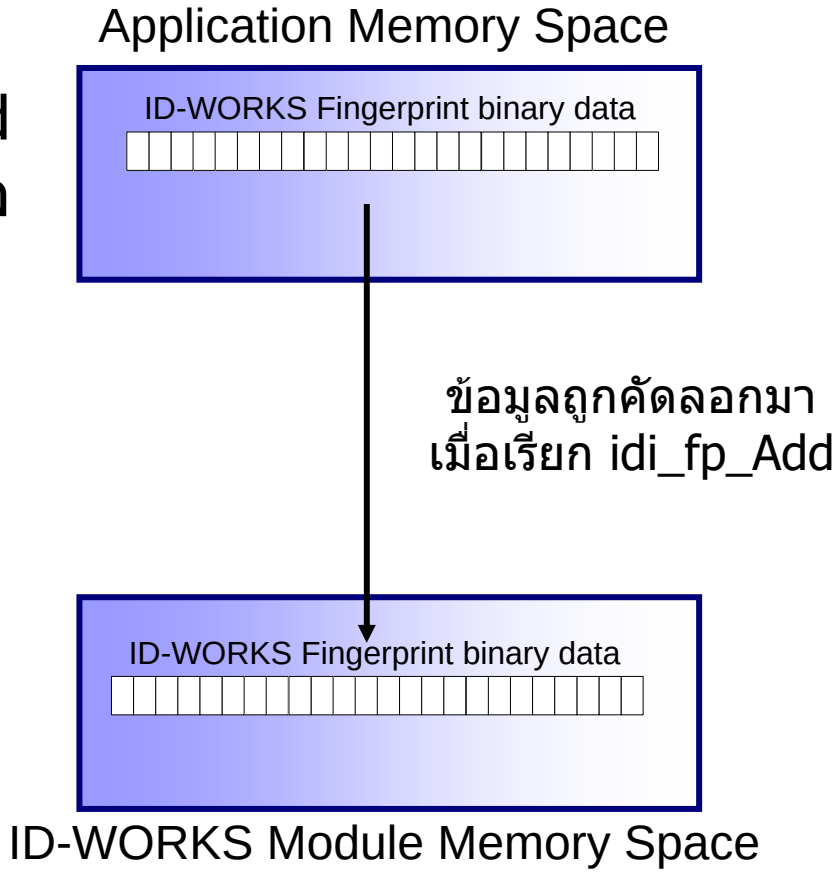


ส่วนการประมวลผลลายนิ้วมือ – ข้อมูลลายนิ้วมือ (ต่อ)

- ข้อมูลลายนิ้วมือ จะสามารถอ้างถึงได้ผ่านทางข้อมูลประเภท FP_FPINFO
- FP_FPINFO.FingerprintRawDataPtr จะต้องชี้ไปยังข้อมูล binary ของลายนิ้วมือ
- FP_FPINFO.FingerprintRawDataSize จะต้องเป็นขนาดของข้อมูล binary ของลายนิ้วมือ
- หากต้องการล้างฐานข้อมูลชั่วคราว ให้ใช้ฟังก์ชัน idi_fp_ClearAll

ส่วนการประมวลผลลายนิ้วมือ – ข้อมูลลายนิ้วมือ (ต่อ)

- เมื่อเรียกฟังก์ชัน idi_fp_Add โมดูลจะทำการคัดลอกข้อมูลลายนิ้วมือ ไปเก็บไว้ในฐานข้อมูลชั่วคราว ซึ่งเมื่อคัดลอกแล้ว ก็จะไม่เกี่ยวกับ FP_FPInfo ที่ส่งเข้าไปกับฟังก์ชันอีก ดังนั้น memory ในส่วนที่เก็บข้อมูล binary ของลายนิ้วมือ จึงสามารถ deallocate ได้



ส่วนการประมวลผลลายนิ้วมือ – เครื่องอ่านลายนิ้วมือ

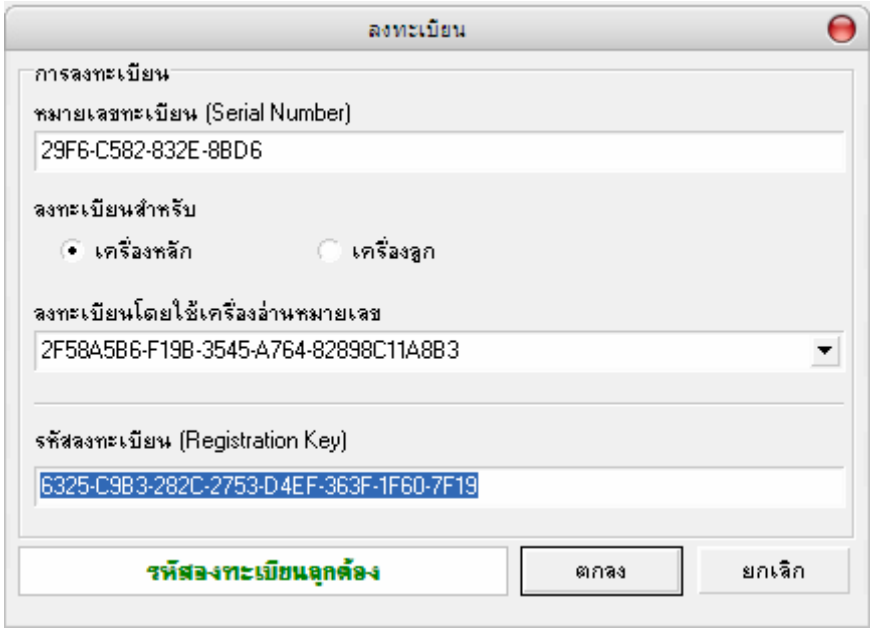
- ก่อนการใช้งานเครื่องอ่านลายนิ้วมือทุกครั้ง จะต้องเรียกฟังก์ชัน idi_ActivateSensor
- ฟังก์ชันนี้จะต้องถูกเรียกในโปรแกรมในช่วงที่เริ่มมีการใช้เครื่องอ่าน
- การเรียกฟังก์ชันนี้ จะต้องมี Serial และ Key ของ ID-WORKS Integrator ซึ่งจะเป็นรหัสเฉพาะของเครื่องอ่านลายนิ้วมือแต่ละเครื่อง ซึ่งจะไม่ซ้ำกัน

ส่วนการประมวลผลลายนิ้วมือ – เครื่องอ่านลายนิ้วมือ (ต่อ)

- ข้อมูลที่จำเป็นสำหรับการเรียก idi_ActivateSensor
  - Serial ของเครื่องอ่าน สำหรับ U.are.U 4000 จะอยู่ในรูปของ GUID เช่น {2F58A5B6-F18B-3545-A764-82898C11A8B3}
  - ID-WORKS Integrator Serial จะเป็นรหัส 16 หลัก เช่น 29F6-C582-832E-8B56
  - ID-WORKS Integrator Key จะเป็นรหัส 32 หลัก เช่น 6325-C9B3-282C-2763-D4EF-363F-1F60-7F19

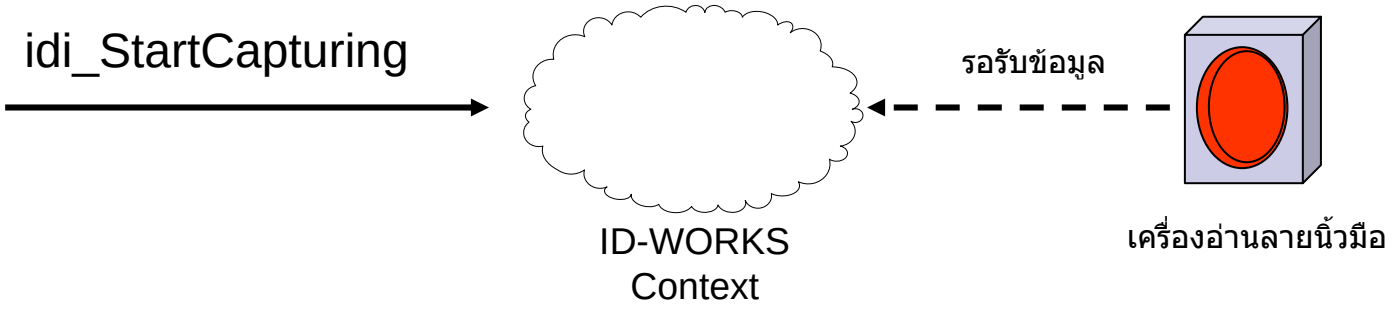
ส่วนการประมวลผลลายนิ้วมือ – เครื่องอ่านลายนิ้วมือ (ต่อ)

- ในการใช้งานจริง ควรจัดทำหน้าต่างสำหรับผู้ใช้เพื่อให้สามารถกรอก Serial และ Key ได้เอง และจัดเก็บใน Registry หรือ INI file ดังตัวอย่างจากโปรแกรม TimeWORKS ในรูป



ส่วนการประมวลผลลายนิ้วมือ – เครื่องอ่านลายนิ้วมือ (ต่อ)

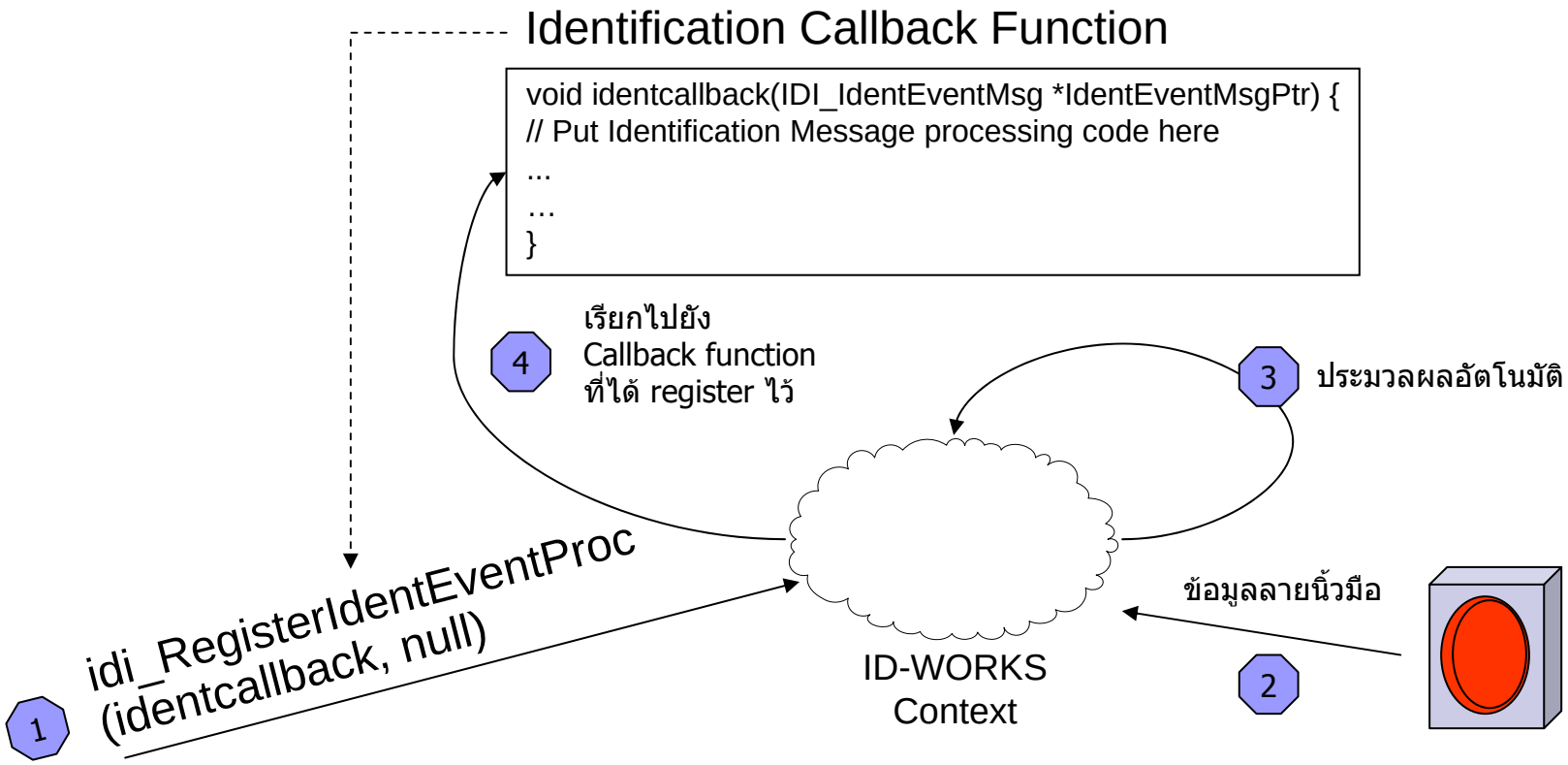
- เมื่อสั่ง idi_Activate แล้ว สามารถเริ่มการทำงานของเครื่องอ่านลายนิ้วมือได้ โดยใช้ฟังก์ชัน idi_StartCapturing
- หลังจากเรียกแล้ว โมดูล ID-WORKS จะรอรับข้อมูลจากเครื่องอ่านลายนิ้วมือ



ส่วนการประมวลผลลายนิ้วมือ – การประมวลผลลายนิ้วมือ

- เมื่อสั่งเริ่มการทำงานของเครื่องอ่านลายนิ้วมือแล้ว โมดูล ID-WORKS จะส่ง Message กลับมาหา Application ผ่านทาง callback function ที่ register ไว้
- สามารถ register callback function ได้โดยใช้ฟังก์ชัน idi_RegisterIdentEventProc

ส่วนการประมวลผลลายนิ้วมือ – การประมวลผลลายนิ้วมือ (ต่อ)



ส่วนการประมวลผลลายนิ้วมือ – การจัดการ Message

- เมื่อมีผู้ใช้แตะลายนิ้วมือ และเครื่องอ่านลายนิ้วมือสามารถจับภาพลายนิ้วมือได้ ฟังก์ชัน callback ที่ register ไว้จะถูกเรียก ไม่ว่าจะรู้จักลายนิ้วมือนั้นหรือไม่
- IDI_IdentEventMsg.MsgID บ่งบอกประเภทของ Message เช่น รู้จักลายนิ้วมือ หรือไม่รู้จัก
- การใช้งานค่า MsgParam ใน IDI_IdentEventMsg จะขึ้นกับค่า MsgID

ส่วนการประมวลผลลายนิ้วมือ – การจัดการ Message (ต่อ)

- กรณี IDI_IdentEventMsg.MsgID เป็น IDI_ID_FINGERPRINTMATCHED
  - ค่า MsgParam จะเป็น Pointer ชี้ไปยัง IDI_IdentEventMatchedStruct
  - ใน IDI_IdentEventMatchedStruct จะมี Pointer ที่ชี้ไปยัง FP_FPInfo อีกทีหนึ่ง

ส่วนการประมวลผลลายนิ้วมือ – การจัดการ Message (ต่อ)

- ในการใช้ฟังก์ชัน idi_RegisterIdentEventProc นั้น สามารถที่จะส่งค่า RefParam ซึ่งเป็น Pointer เข้าไปด้วยได้ ซึ่งจะถูกใส่กลับมาใน IDI_IdentEventMsg.RefParam เมื่อมีการเรียกกลับมายัง callback function
- เนื่องจาก idi_RegisterIdentEventProc จะรับ callback function เป็น function ธรรมดา ดังนั้นผู้พัฒนาบางท่านอาจมีปัญหาในการเรียกกลับไปยัง method ของ object ซึ่งสามารถแก้ปัญหาได้โดยการใช้ประโยชน์ RefParam ในการอ้างถึง object ที่ทำการ Register Event ในกรณีที่เขียนโปรแกรมในรูปแบบของ object-oriented เช่นอาจใช้ว่า

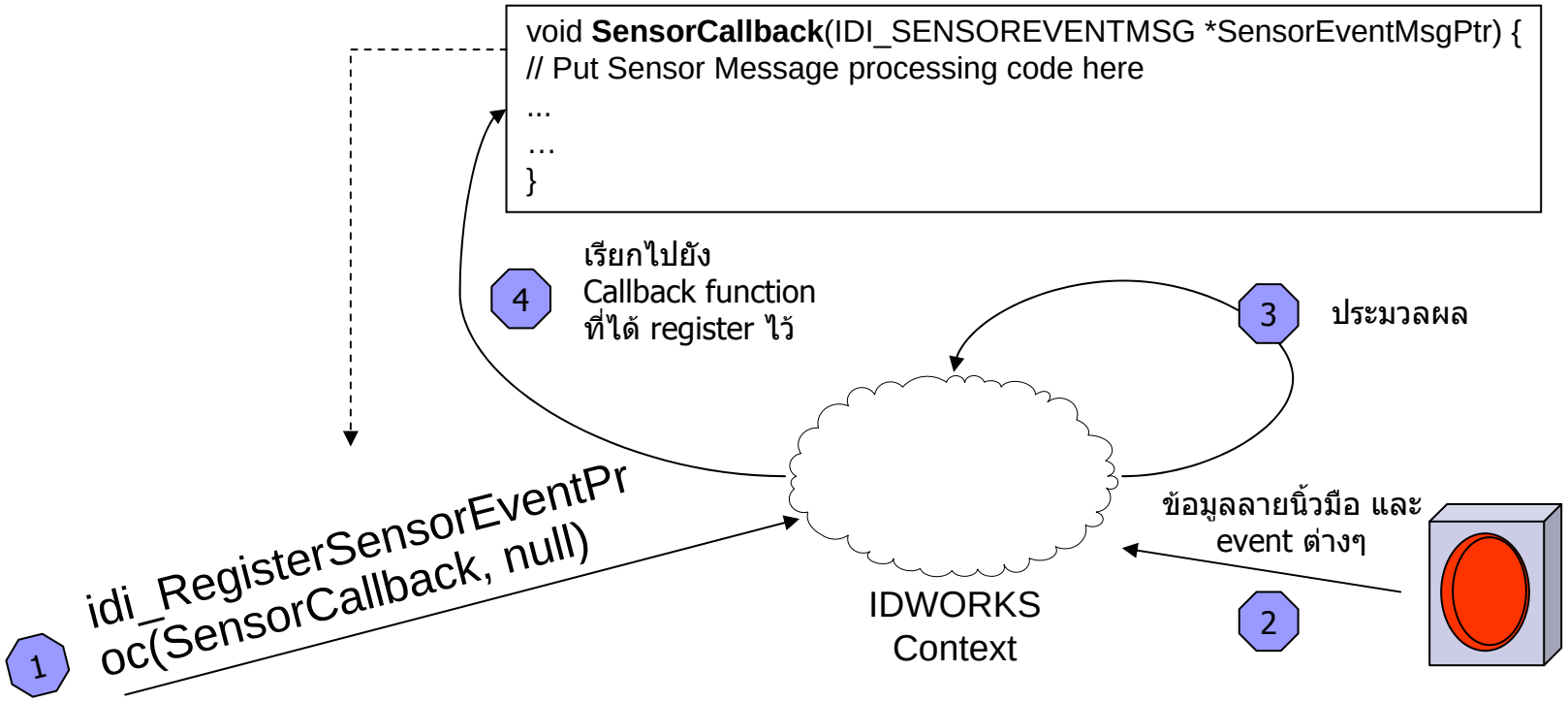
  ```
  idi_RegisterEventProc(identcallback, &this);
  ```

  เพื่อให้ callback function สามารถเรียกกลับมายัง object ได้

ส่วนการประมวลผลลายนิ้วมือ - การรับรูปภาพ

- ผู้พัฒนาที่ต้องการแสดงผลรูปลายนิ้วมือขึ้นมาในโปรแกรม สามารถทำได้โดยใช้ฟังก์ชัน idi_RegisterSensorEventProc ซึ่งจะเป็นการ register callback function ในรูปแบบเดียวกันกับ idi_RegisterIdentEventProc แต่จะเป็นการ register เพื่อรับ event เกี่ยวกับเครื่องอ่านลายนิ้วมือโดยเฉพาะ เช่น การรับรูปภาพ
- เมื่อโมดูลรับรูปภาพมาจากเครื่องอ่านลายนิ้วมือ ในขั้นแรกจะส่ง message ชื่อ IDI_SE_IMAGEACQUIRED ให้กับฟังก์ชันที่ได้ทำการ register ไว้ และเมื่อประมวลผลลายนิ้วมือเสร็จสิ้น ก็จะส่ง IDI_SE_IMAGEPROCESSED อีกครั้งหนึ่ง

ส่วนการประมวลผลลายนิ้วมือ – การรับรูปภาพ (ต่อ)



ส่วนการประมวลผลลายนิ้วมือ – การรับรูปภาพ (ต่อ)

- ใน IDI_SENSOREVENTMSG จะมี member ชื่อ MsgParam
- ถ้าหากว่า MsgID เป็น IDI_SE_IMAGEACQUIRED หรือ IDI_SE_IMAGEPROCESSED ค่าใน MsgParam จะเป็น pointer ที่ชี้ไปยัง structure IDI_SENSOREVENTIMAGEACQUIREDSTRUCT
- ใน IDI_SENSOREVENTIMAGEACQUIREDSTRUCT จะประกอบด้วยค่าความกว้าง (ImageWidth) และความสูง (ImageHeight)ของรูป และ pointer ที่ชี้ไปยัง 8-bit grayscale bitmap ของลายนิ้วมือ (ImageBitmapPtr)

IT WORKS
Everything works

ส่วนช่วยจัดเก็บลายนิ้วมือ

ส่วนช่วยจัดเก็บลายนิ้วมือ

- เป็นไลบรารี ชื่อ FPENROL.DLL
- ใช้ลงทะเบียนลายนิ้วมือผู้ใช้
- ข้อมูลที่ได้จะประกอบด้วย ข้อมูล binary ของลายนิ้วมือ และ IndexingValue สำหรับการ optimize การค้นหาลายนิ้วมือ

ส่วนช่วยจัดเก็บลายนิ้วมือ (ต่อ)

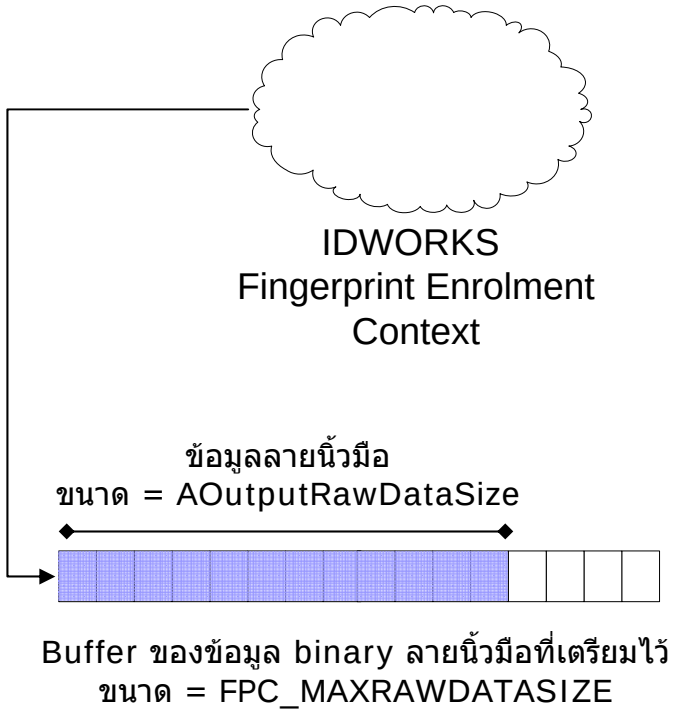
- ก่อนการเริ่มใช้งาน จะต้องทำการเรียก fpe_InitModule จากนั้นสร้าง context โดยใช้ fpe_CreateContext และทำการ ActivateSensor โดยใช้ fpe_ActivateSensor
- การเริ่มใช้งาน และหลักการของ context จะมีรูปแบบเหมือนกับโมดูลประมวลผลลายนิ้วมือ

ส่วนช่วยจัดเก็บลายนิ้วมือ (ต่อ)

- เมื่อเรียกฟังก์ชัน fpe_StartEnrolment จะปรากฏหน้าต่างดังรูป
- เมื่อผู้ใช้แตะนิ้วครบ 4 ครั้ง จะปรากฏผลการลงทะเบียน ซึ่งถ้าสำเร็จผู้ใช้จะสามารถกดปุ่ม “ตกลง” ได้

ส่วนช่วยจัดเก็บลายนิ้วมือ (ต่อ)

- การเรียกฟังก์ชัน fpe_StartEnrolment จะต้องมีการเตรียม buffer สำหรับเก็บข้อมูลลายนิ้วมือก่อน โดยจะต้องมีขนาดอย่างน้อยเท่ากับ FPC_MAXRAWDATASIZE (4096 bytes)
- หลังจากนั้นส่งตำแหน่ง (Pointer) ของ buffer นั้นเข้าไปในการเรียกฟังก์ชัน
- เมื่อฟังก์ชันเก็บตัวอย่างลายนิ้วมือ และได้ข้อมูลที่ต้องการแล้ว จะเขียนข้อมูลกลับมายัง buffer ที่เตรียมไว้
- ขนาดของข้อมูลลายนิ้วมือที่เขียนลงใน buffer จะถูกส่งคืนมาในตัวแปร AOutputRawDataSize ที่ส่งเข้าไป



ส่วนช่วยจัดเก็บลายนิ้วมือ (ต่อ)

- ข้อมูลลายนิ้วมือที่ได้ สามารถจัดเก็บลงในไฟล์หรือฐานข้อมูลได้
- เมื่อนำมาใช้งาน จะใช้กับฟังก์ชัน idi_fp_Add โดยให้ FP_FPINFO.FingerprintRawDataPtr ชี้ไปยังข้อมูลลายนิ้วมือ ใส่ค่า FP_FPINFO.FingerprintRawDataSize ตามขนาดจริงของข้อมูล และใส่ค่า FP_FPINFO.FingerprintIndexingValue ตามข้อมูลที่ได้เก็บไว้

IT WORKS
Everything works

ส่วนข้อมูลเครื่องอ่านลายนิ้วมือ

ส่วนข้อมูลเครื่องอ่านลายนิ้วมือ

- โมดูล ITWSSL.DLL
- ใช้ตรวจสอบว่ามีเครื่องอ่านลายนิ้วมือใดบ้างที่ต่ออยู่กับเครื่อง
- ในโมดูลนี้สามารถใช้ได้โดยไม่ต้องสร้าง Context และไม่จำเป็นต้อง Activate

ส่วนข้อมูลเครื่องอ่านลายนิ้วมือ - การใช้งาน

- ปัจจุบันมี 2 ฟังก์ชัน คือ GetDeviceCount และ GetDeviceList
- ก่อนใช้ฟังก์ชัน GetDeviceList จะต้องหาก่อนว่ามีเครื่องอ่านกี่เครื่องโดย GetDeviceCount ก่อน
- เมื่อได้จำนวนเครื่องอ่านแล้ว จะต้องจอง array ของ ISL_SENSORINFO ตามจำนวนเครื่องอ่าน เพื่อส่งเข้าไปกับ GetDeviceList