設計書

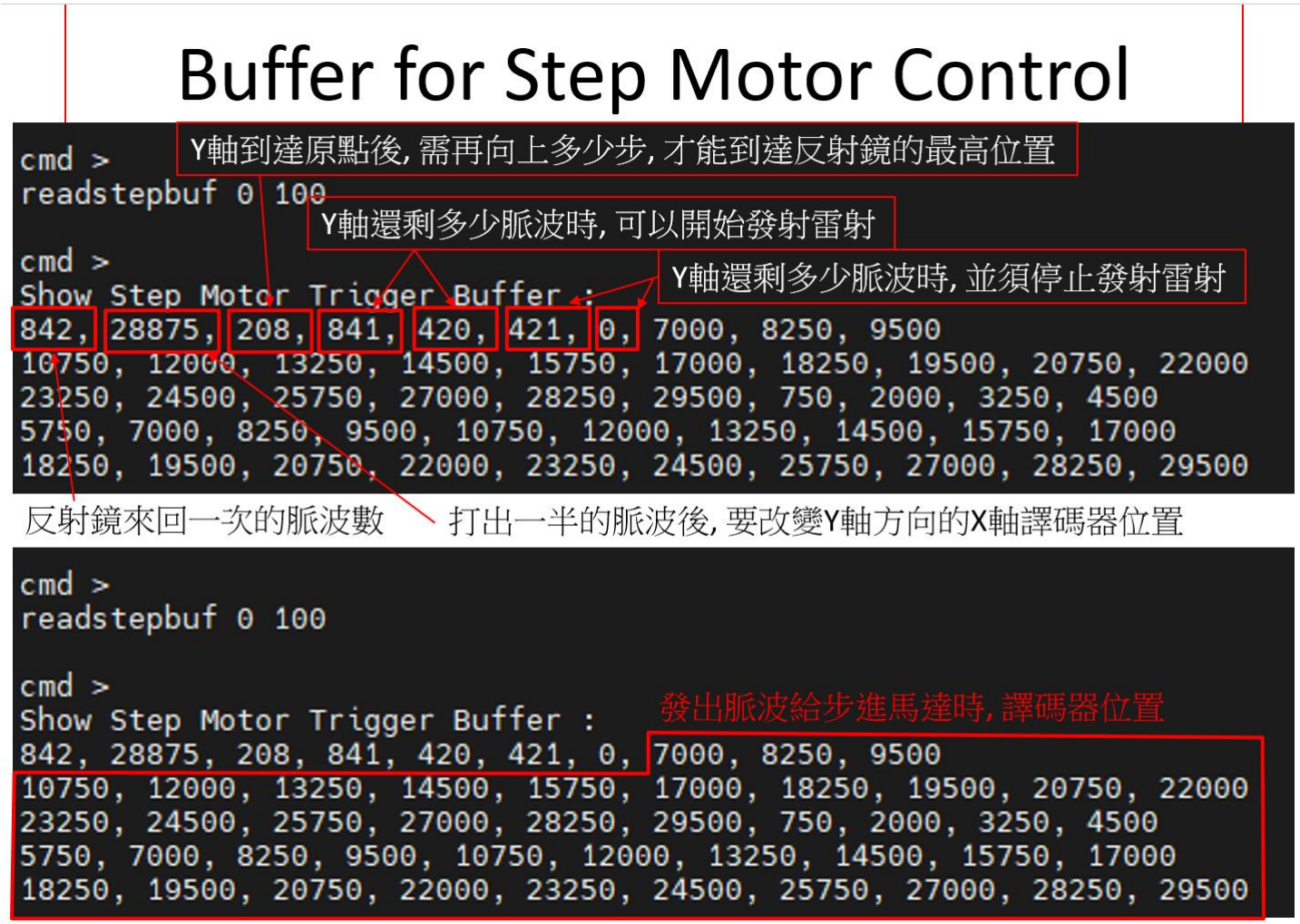
修改

規格 :

1. 水平 600 點, 角解析度 0.1 度
2. 垂直 300 條線, 角解析度 0.1 度
3. 雷射重複率 : 大於 100KHz
4. 距離解析度 : 1 cm
5. 測距最大距離 : > 500M
6. 結束

設計說明 :

1. 水平角解析度為 0.1 度, 雷射重複率為 100KHz, 所以 BLDC 每 10usec 轉動 0.05 度, 因此 BLDC 轉動一圈的時間為 (360/0.05)*10usec = 72000usec = 72msec, 得到 BLDC 每秒轉動 13.89 rps = 833 rpm
2. 垂直 300 條線, 每四條線需要 72msec(每條線 18msec), 所以掃描 300 條線至少須要 5.4sec, 因此 Frame Rate = 0.185 fps
3. 垂直角解析度為 0.1 度, 微步進驅動器的角解析度為 1.8/256 = 0.007031 度(microstepping indexer = 256), 所以 BLDC 轉動 90 度, 步進馬達需要走 14.2222 步, 最大控制角度誤差為 0.003516 度(實際角度誤差可能更大)
4. 步進馬達控制範例 :

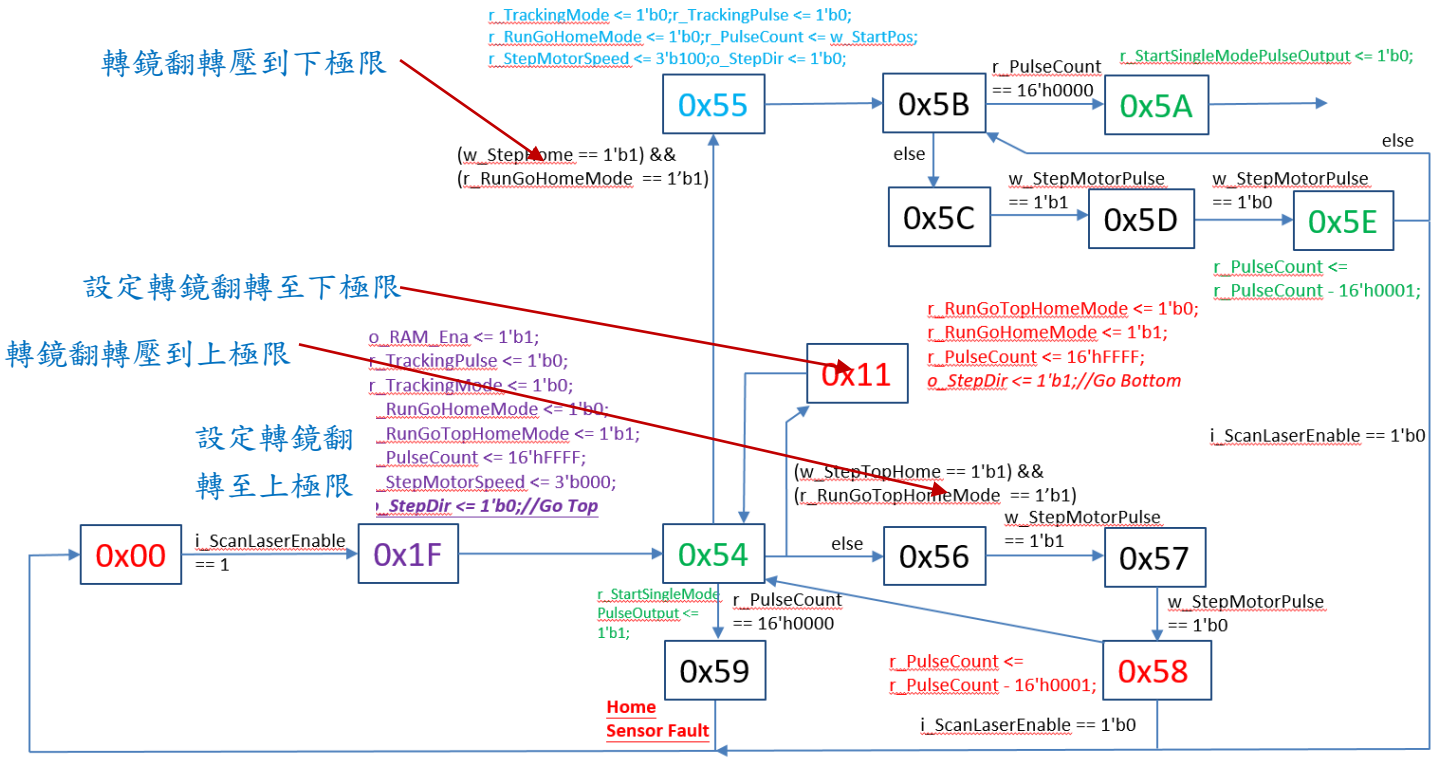


5. 步進馬達運作說明 :
   i. 步進馬達先讓反射鏡向上壓到上方極限, 再讓反射鏡向下壓到下方極限
   ii. 步進馬達讓反射鏡向上達到最高點, 然後再開始讓反射鏡來回掃描
6. 反射鏡控制設計(Buffer for Step Motor Control) :
   i. 反射鏡來回一次的脈波數 : 8534

ii. 打出一半的脈波後(反射鏡掃到下方), 需要改變 Y 軸方向的 X 軸 Encoder 位置
iii. Y 軸到達原點後, 步進馬達需要再向上走多少步, 才能到達反射鏡正常工作的最高位置
iv. Y 軸步進馬達還剩下多少步(脈波)時, 可以開始發射雷射(反射鏡掃到上方)
v. Y 軸步進馬達還剩下多少步(脈波)時, 可以開始發射雷射(反射鏡掃到下方)
vi. Y 軸步進馬達還剩下多少步(脈波)時, 必須停止發射雷射(反射鏡掃到上方)
vii. Y 軸步進馬達還剩下多少步(脈波)時, 必須停止發射雷射(反射鏡掃到下方)

7. 步進馬達控制原理說明 : 步進馬達啟動掃描時, 先讓轉鏡向上翻轉至上極限, 然後再讓轉鏡向下翻轉至下極限, 最後再將轉鏡向上翻轉至掃描的最高點後, 開始進行掃描

步進馬達連動狀態控制(I)微調



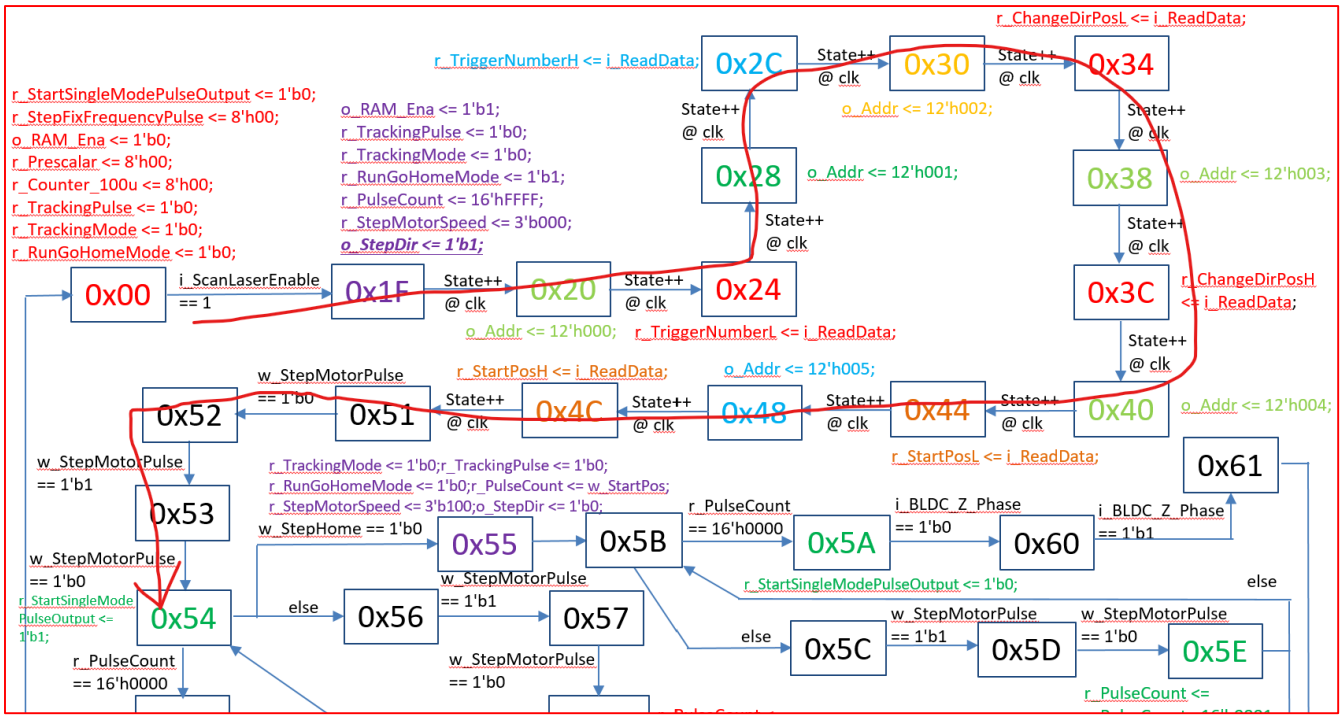
啟動步進馬達追蹤程序前, 必須從 Block Memory 中讀取追蹤控制的相關參數
由狀態 0x54 到 0x5A 的過程中有做微幅修改

```
183  assign w_StepMotorPulse = r_StepFixFrequencyPulse[7-r_StepMotorSpeed];//(r_StepMotorSpeed == 3'b000) ? : ((r_StepMotorSpeed == 3'b000) ? );
184  assign o_StepPulse = (r_StartSingleModePulseOutput == 1'b1) ? w_StepMotorPulse : ((r_TrackingMode == 1'b1) ? r_TrackingPulse : 1'b0);
185  assign w_TriggerNumber = {r_TriggerNumberH, r_TriggerNumberL};
186  assign w_TriggerPoistion = {r_TriggerPositionH, r_TriggerPositionL};
187  assign w_ChangeDirPos = {r_ChangeDirPosH, r_ChangeDirPosL};
188  assign w_StartPos = {r_StartPosH, r_StartPosL};
189  assign w_StepMotorScanLaserlstEnableYpos = {r_StepMotorScanLaserlstEnableYposH, r_StepMotorScanLaserlstEnableYposL};
190  assign w_StepMotorScanLaser2ndEnableYpos = {r_StepMotorScanLaser2ndEnableYposH, r_StepMotorScanLaser2ndEnableYposL};
191  assign w_StepMotorScanLaserlstDisableYpos = {r_StepMotorScanLaserlstDisableYposH, r_StepMotorScanLaserlstDisbleYposL};
192  assign w_StepMotorScanLaser2ndDisableYpos = {r_StepMotorScanLaser2ndDisableYposH, r_StepMotorScanLaser2ndDisableYposL};
```

r_StepFixFrequencyPulse : 是一個 8 bit counter, clock source 的頻率為 10KHz, 因此這個 Counter 的每個 bit 可以產生 5KHz, 2.5KHz, 1.25KHz, 625Hz, 312.5Hz, 156.25Hz, 78.125Hz, 39.063Hz
r_StepMotorSpeed : 可設定 0-7, 選擇上面八個頻率的一個由 w_StepMotorPulse 輸出
實際輸出到步進馬達的 Pulse 是由 o_StepPulse 送出, 當在單動模式下 (r_StartSingleModePulseOutput == 1), o_StepPulse 接至 w_StepMotorPulse, 當在追蹤模式下

(r_StartSingleModePulseOutput == 0 且 r_TrackingMode == 1), o_StepPulse 接至 r_TrackingPulse

r_TrackingPulse ： 根據 Lookup Table 記錄的 Encoder 數值, 產生追蹤 Encoder 位置的脈波



狀態 0x20 到 0x4C 會執行讀取以下參數的程序 ：

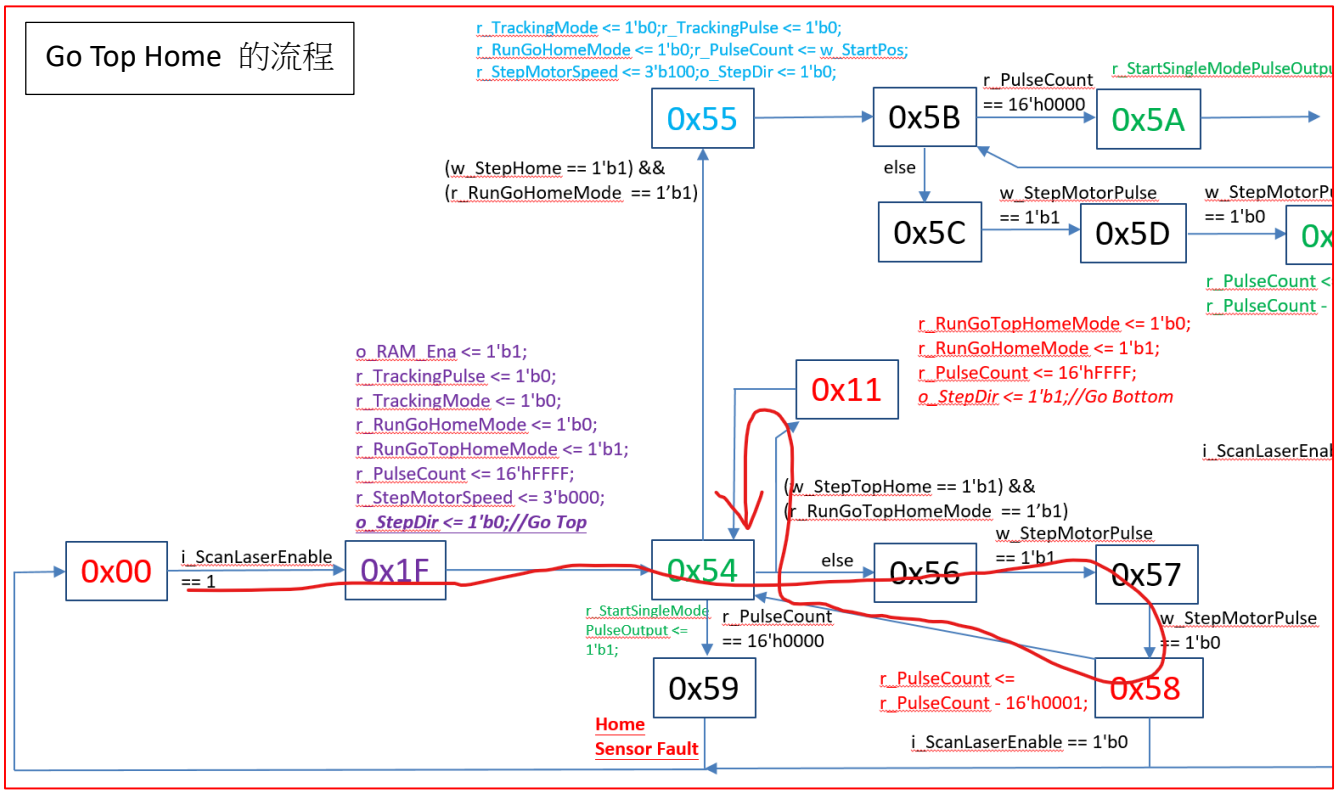
讀取 w_TriggerNumber = {r_ TriggerNumberH, r_ TriggerNumberL}

讀取 w_ChangeDirPos = {r_ ChangeDirPosH, r_ ChangeDirPosL}

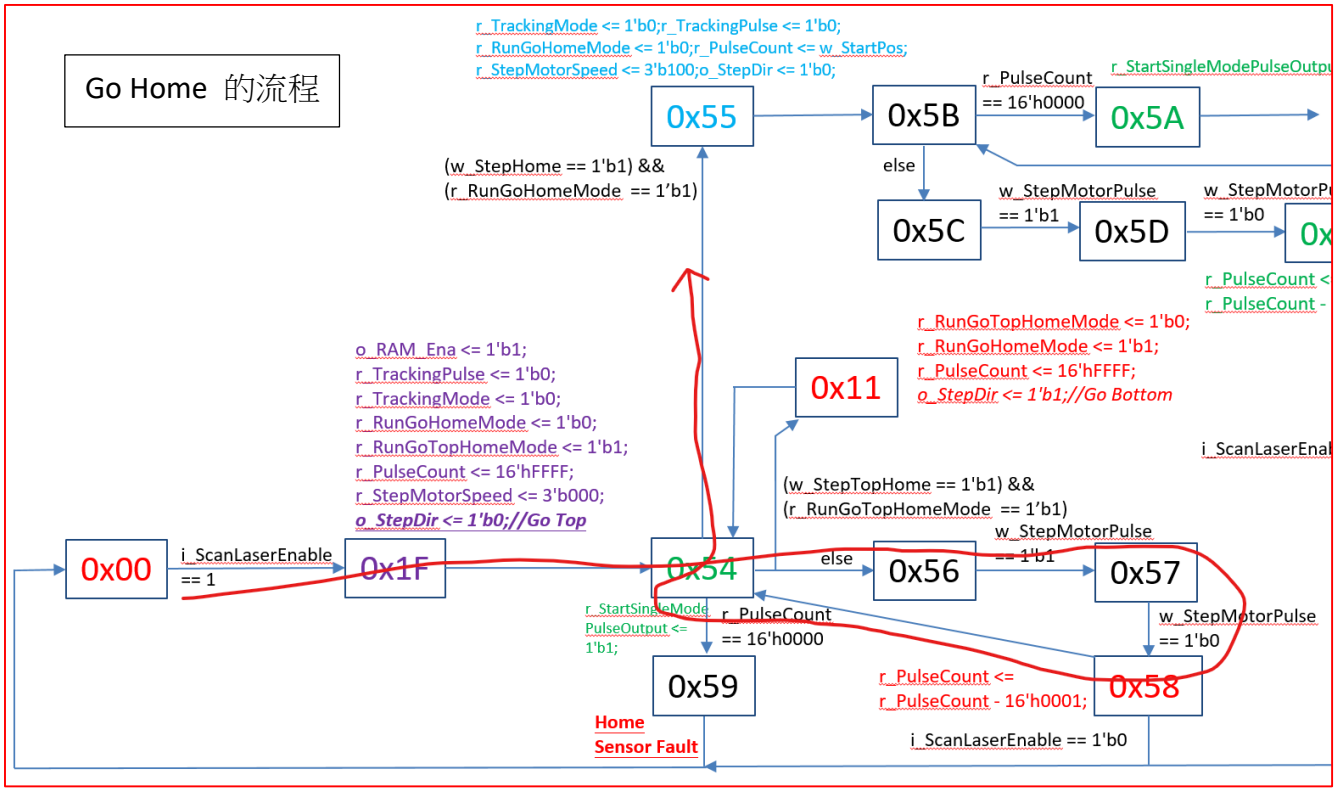
讀取 w_StartPos = {r_ StartPosH, r_ StartPosL}

r_TriggerNumberL ： 存放在 address = 0

r_StartPosH ： 存放在 address = 5

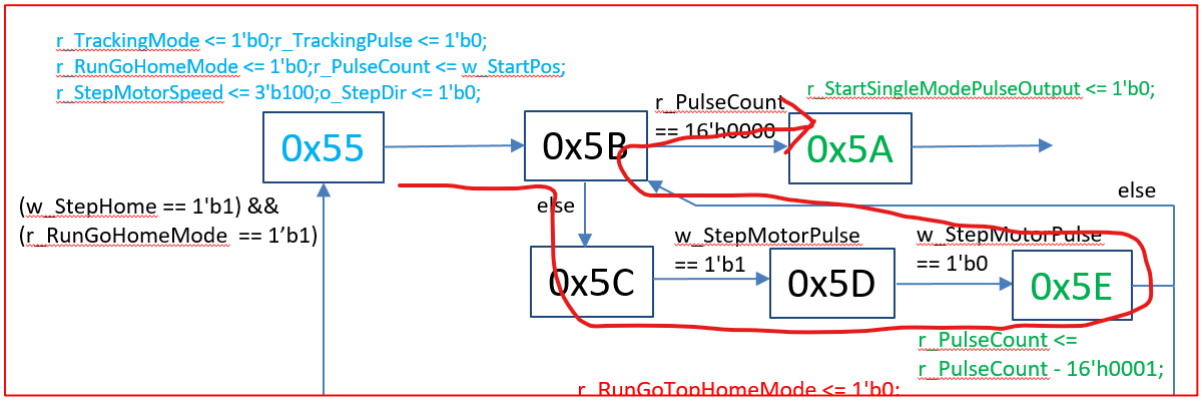


當 w_StepTopHome == 1 且在 r_RunGoTopHomeMode == 1 時, 會進入 0x11 的狀態, 將 r_RunGoTopHomeMode 設為 0 並設定 r_RunGoHomeMode = 1, 停止 Go Top Home Mode 並啟動 Go Home Mode



當 w_StepHome == 1 且在 r_RunGoHomeMode == 1 時, 會進入 0x55 的狀態, 將 r_RunGoHomeMode 設為 0, 停止 Go Home Mode, 同時並設定前往起點需要的脈波數(r_PulseCount <= w_StartPos), 然後進入 0x5B 狀態, 開始進行前往掃描起點的程序

步進馬達將轉鏡轉至由上往下掃描起點的流程 : 當轉鏡轉至由上往下掃描起點後, 就會開始進行垂直方向的掃描(步進馬達讓轉鏡上下掃描), 到達起點後, 要結束固定頻率的脈波輸出, 因此將 r_StartSingleModePulseOutput 設為 0, 所以, 將 r_StartSingleModePulseOutput 設為 1'b0, 此後, 步進馬達要根據無刷馬達的位置, 決定轉鏡是否需要轉動(轉鏡需要轉動)



狀態 0xC0 到 0xFF 會執行讀取以下參數的程序 :

讀取 w_StepMotorScanLaser1stEnableYpos = {r_ StepMotorScanLaser1stEnableYposH, r_ StepMotorScanLaser1stEnableYposL}

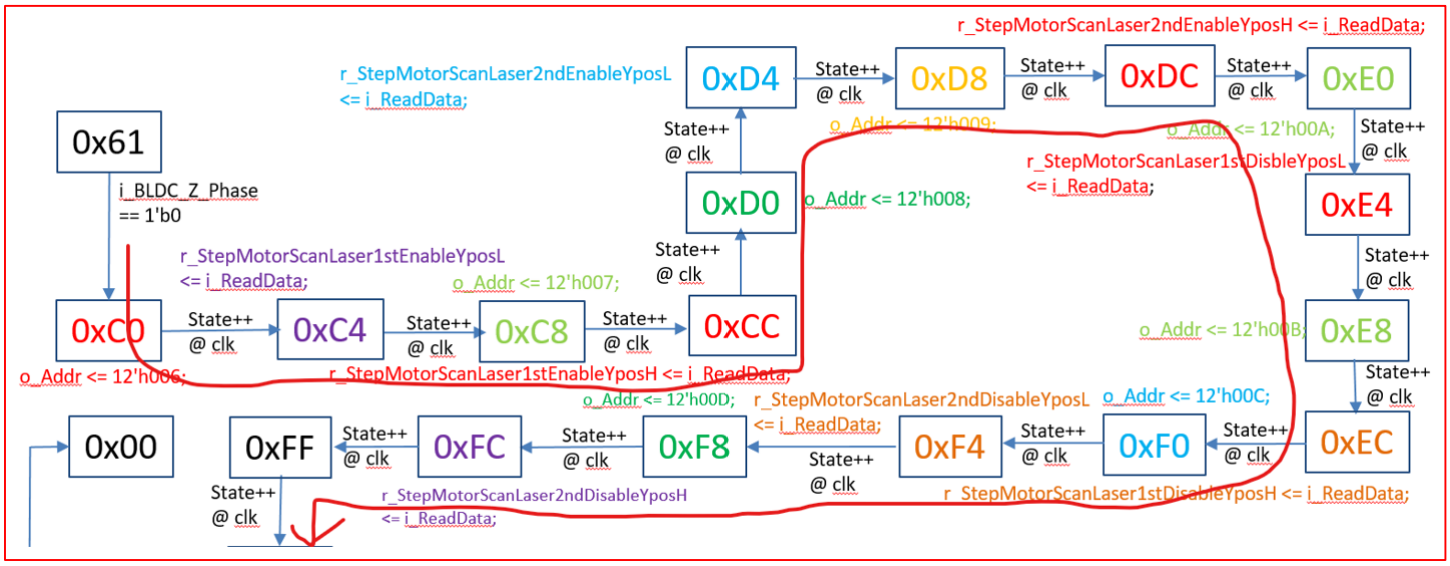
讀取 w_StepMotorScanLaser2ndEnableYpos = {r_StepMotorScanLaser2ndEnableYposH, r_StepMotorScanLaser2ndEnableYposL}

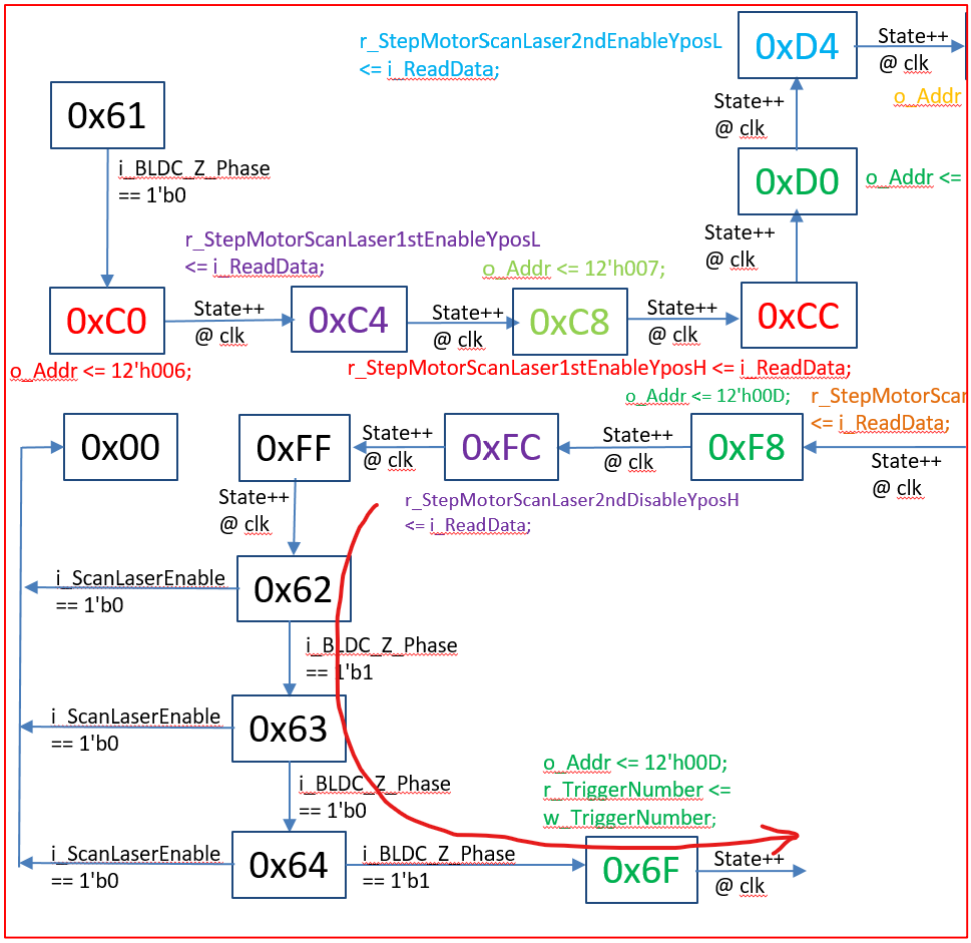
讀取 w_StepMotorScanLaser1stDisableYpos = {r_StepMotorScanLaser1stDisableYposH, r_StepMotorScanLaser1stDisbleYposL};

讀取 w_StepMotorScanLaser2ndDisableYpos = {r_StepMotorScanLaser2ndDisableYposH, r_StepMotorScanLaser2ndDisableYposL};

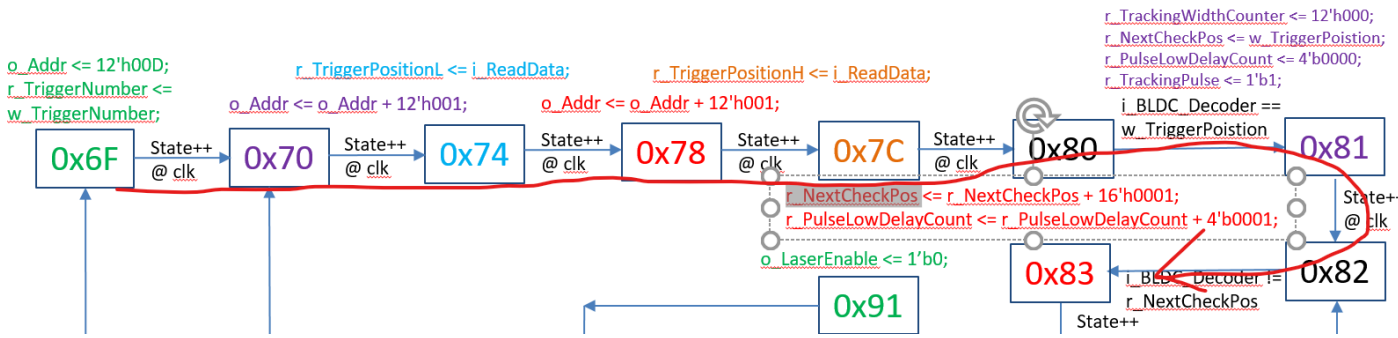
r_ StepMotorScanLaser1stEnableYposL : 存放在 Block Memory 的 address = 6

r_StepMotorScanLaser2ndDisableYposH ： address = 存放在 Block Memory 的 13(0x0D)



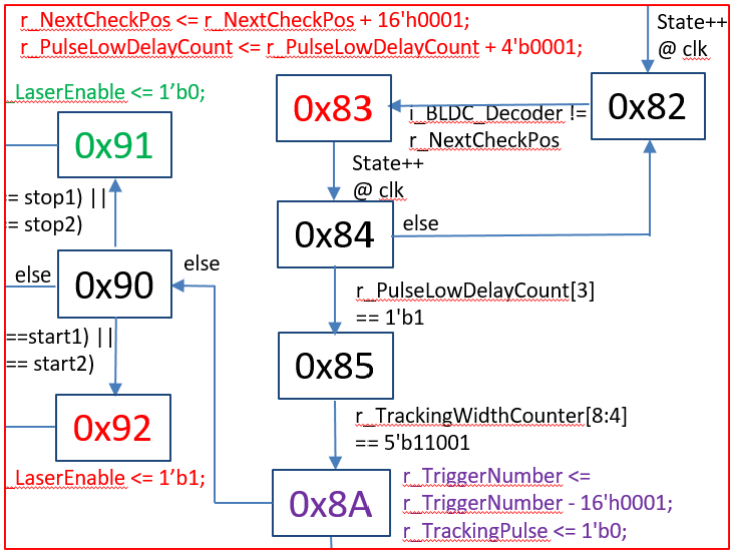


所有目前需要的參數都載入後，會等候兩次 Z-phase 後(Z-phase = 1 ➔ 0 ➔ 1)，進入追蹤掃描程序(由狀態 0x6F 進入)

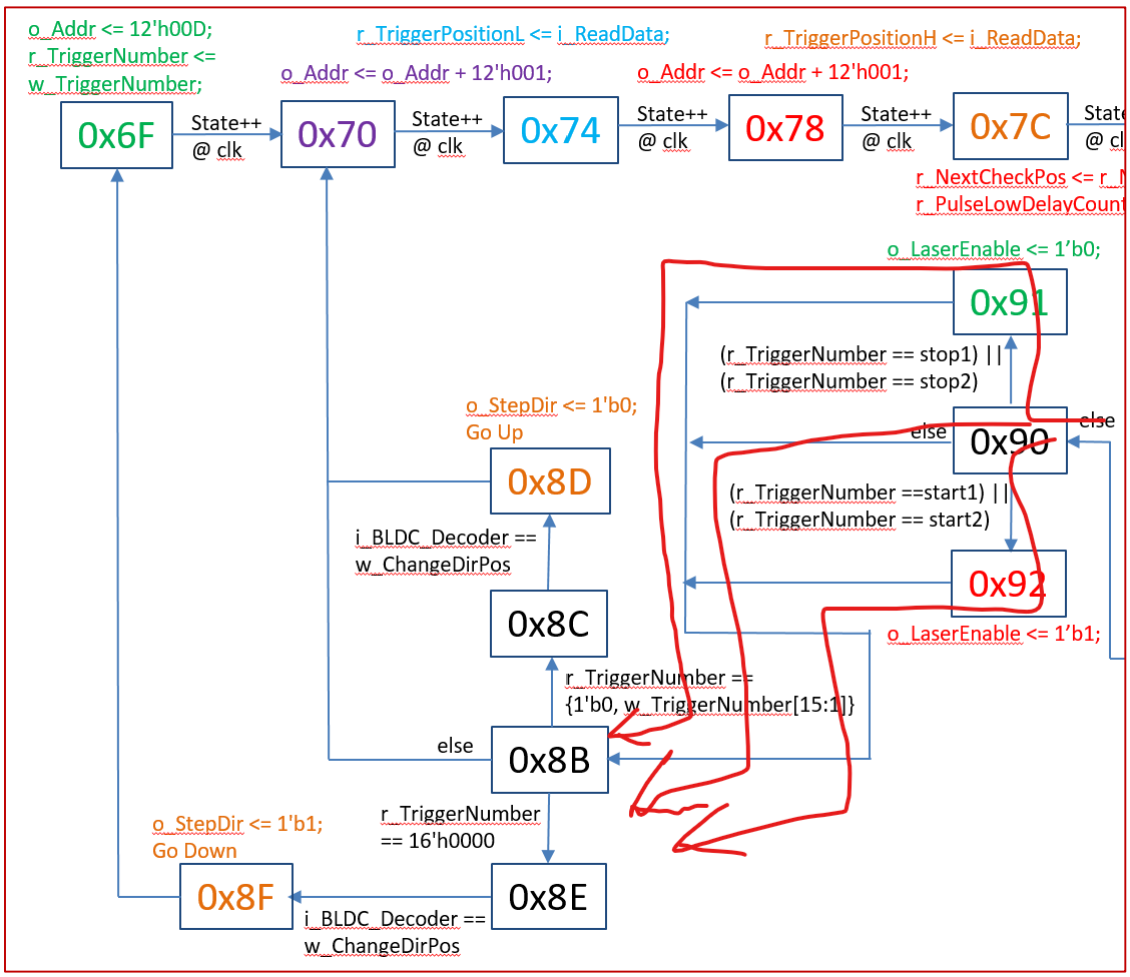


狀態 0x74 時會載入 r_TriggerPositionL，狀代 0x7C 時會載入 r_ TriggerPositionH

(w_TriggerPosition = {r_ TriggerPositionH , r_ TriggerPositionL})
當無刷馬達的 Encoder 到達 w_ TriggerPosition 位置時, r_TrackingPulse 設 Hi, 同時將 r_NextCheckPos 設為 r_TriggerPosition 位置



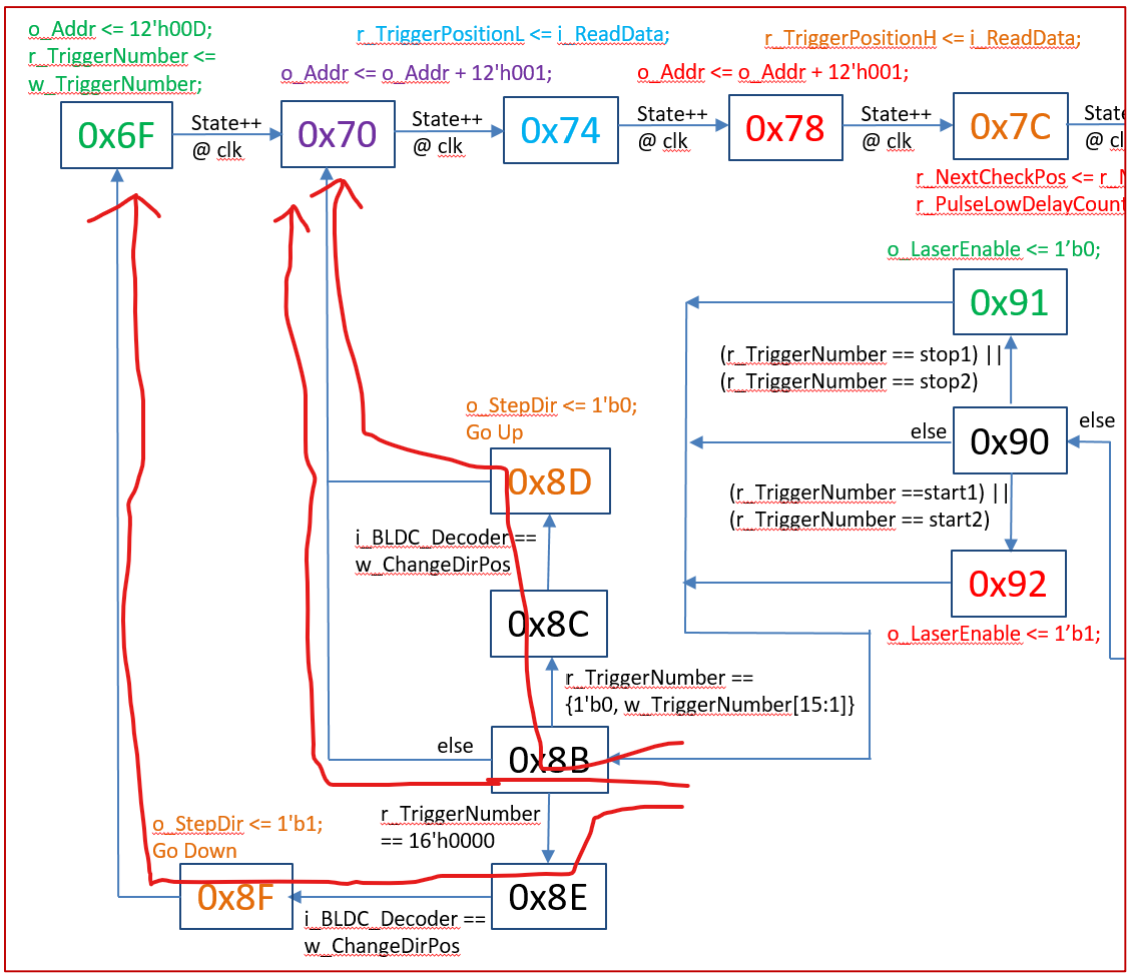
並等候無刷馬達的 Encoder 離開 r_ NextCheckPos 後, 才往下一個狀態(0x83), 每個 Encoder 數值變化時, r_PulseLowDelayCount 都會加 1, 直到 r_PulseLowDelayCount >= 8 後, 再去確認 r_TrackingWidthCounter >= 400(0x19x)是否成立, 條件確立後, 進入狀態 0x8A, 將 r_TriggerNumber 減 1 及 r_TrackingPulse 設為 0, 結束一個步進馬達的脈波控制



每完成一個步進馬達的脈波, r_TriggerNumber 會減一, 若 r_TriggerNumber == stop1 或 stop2, 則 o_LaserEnable <= 0, 若 r_TriggerNumber == start1 或 start2, 則 o_LaserEnable <= 1, 否則 o_LaserEnable 狀態不變
當無刷馬達轉到發射角度時,只有 o_LaserEnable == 1 時, Laser 才能發射

start1 和 start2 分別為 : w_StepMotorScanLaser1stEnableYpos 和 w_StepMotorScanLaser2ndEnableYpos

stop1 和 stop2 分別為 : w_StepMotorScanLaser1stDisableYpos 和 w_StepMotorScanLaser2ndDisableYpos

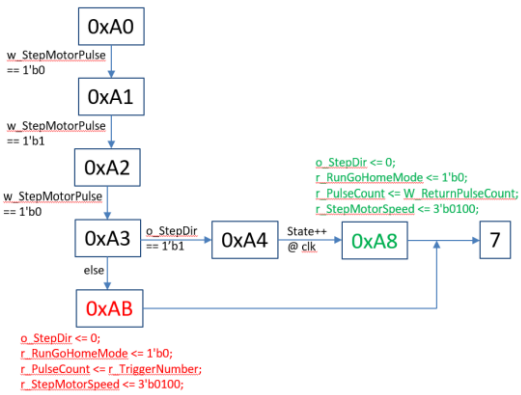


當 r_TriggerNumber == w_TriggerNumber(存放在 Block Memory 的 0x00 和 0x01 的位置) 的一半時, 或 r_TriggerNumber == 0 時, 在無刷馬達 Encoder 為 w_ChangeDirPos(存放在 Block Memory 的 0x02 和 0x03 的位置)時, o_StepDir 的狀態才能改變,

當 r_TriggerNumber == w_TriggerNumber 時, o_StepDir <= 0, 步進馬達讓轉鏡向上, 步進馬達控制狀態 0x70, 繼續讀取下一個 r_TriggerPosition

當 r_TriggerNumber == 0 時, o_StepDir <= 1, 步進馬達讓轉鏡向下, 步進馬達控制狀態 0x6F, 將 r_TriggerNumber 設為 w_TriggerNumber, o_Addr = 0x0D, 從頭開始讀取 r_TriggerPosition(第一個 r_TriggerPosition)

```
Full_Adder #(.N(15)) U1 (.A(w_TriggerNumber), .B((~r_TriggerNumber)), .Cin(1'b1), .Sum(W_ReturnPulseCount ));
```



步進馬達控制狀態 0xA0 這部分的流程(如上圖), 目前沒有設計進 Verilog 中

這部分的控制是, 等 w_StepMotorPulse == 0 時, 無論步進馬達是正在讓轉鏡向下轉至低點或向上轉至高點, 都是設為向上轉至高點(o_StepDir 均設為 0)

8. 測距最大距離為 500M = 50000cm, 所以可使用 2 Bytes 來表示距離, 則水平 600 點須使用 1200 Bytes 的空間來存放距離資訊, 此長度低於 1458 Bytes(一個 UDP 封包的最大資料容量), 可使用一個封包傳送
9. 若最大距離超過 655.35M = 65535cm, 則需使用 4 Bytes 來表示距離, 因此水平 600 點需要使用 2400 Bytes 的空間來存放距離資訊, 此長度高於 1458 Bytes(一個 UDP 封包的最大資料容量), 須拆成兩個封包傳送
10. 封包格式 :
    i. Header(2 Bytes) : 0xAA55
    ii. Echo 訊息(2 Bytes) :
        i. 掃描線號碼(bit0 - bit8): 0-299
        ii. Echo ID(bit9 - bit12) : 強度, 掃描線的前半部, 掃描線的後半部, 整條掃描線
        iii. Echo 位置(bit13 - bit15) : 0 - 4 表示 1st Echo - 5th Echo
    iii. 額外訊息(2 Bytes) : Frame ID
11. 結束