Lab7 STM32 Clock and Timer

0510532 楊上萱

# Lab objectives 實驗目的

瞭解STM32的各種clock source使用與修改

瞭解STM32的timer使用原理

瞭解STM32的PWM使用原理與應用

# Steps 實驗步驟

1. **Modify system initial clock**

驅動部分

先把GPIO\_init()和delay\_1s的function放入.s檔中並宣告為.global。

硬體部分

PA5為LED output pin，PC13為user button input pin。

設定system clock:

關掉PLL並等它穩定。

改變PLLN、PLLM來設定RCC->PLLCFGR。

重新開啟PLL並等它穩定並把PLL作為SYSTEM CLK。

最後改變RCC->CFGR來設定AHB prescaler。

如何更改頻率:

照著公式推算:

The PLL clock frequency is calculated as f(VCO clock) = f(PLL clock input) × (PLLN / PLLM)

The final output to the system clock frequency is f(PLL\_R) = f(VCO clock) / PLLR

1. **Timer 計時器**

驅動部分:

將max7219\_init()、GPIO\_init()、max7219send的function放入.s檔中並宣告為.global。

硬體部分:

DIN->PA5，CS->PA6，CLK->PA7，VCC->3v3，GND->GND

程式解釋:

MSI的default frequency是4MHz, TIM2的prescaler為39999，auto-reload register是99。

這樣子設定的結果，timer frequency變為100Hz，counter每0.01秒加一次，然後把counter/100的結果印在max7219上，一直數到TIME\_SEC\*100為止。

1. **Music keypad**

蜂鳴器頻率：

Freq = HCLK / (prescalar + 1) / (AutoReload\_reg + 1);

* Prescalar = HCLK / Freq / (AutoReload\_reg + 1)-1;

PWM：

CCR\_reg初始為dyty\_cycle=50，我們用

改變CCR\_reg(duty\_cycle)的值，每按一次”#”，duty\_cycle+1，每按一次”\*”，duty\_cycle-1。

# Results and analysis 實驗結果與分析

7\_1:

每次按user\_button，LED閃爍的頻率會依序變成1MHz – 6MHz – 10MHz – 16MHz – 40MHz – 1MHz …。

7\_2:

設定TIME\_SEC，程式會一邊數秒並秀在max7219上，數到TIME\_SEC為止。

7\_3:

用keypad控制蜂鳴器的輸出頻率，keypad上的1~8分別代表標準Do、Re、Mi、Fa、…、高音Do，按#會把duty\_cycle+5，按\*會把duty\_cycle-5。

1. **Conclusions and ideas 心得討論與應用聯想**

這次Lab比之前都難，因此也花了更多時間在做，不過過程中學到了許多，包括如何對system clock、timer做設定，還有用timer設定蜂鳴器的頻率也非常有趣。

1. **Code**

**main7\_1.c**

|  |
| --- |
| 1. #include "stm32l476xx.h" 2. #include "util.h" 3. int plln = 16, pllm = 7, prescaler = 9; 4. enum {S1MHZ, S6MHz, S10MHZ, S16MHZ, S40MHZ} state = S40MHZ; 5. int prev\_btn = 1, curr\_btn = 1; 6. void SystemClock\_Config(); 7. int main() 8. { 9. SystemClock\_Config(); 10. gpio\_init(); 11. int flag=0; 12. int j=1; 13. while (1) 14. { 15. //if (!prev\_btn && curr\_btn) 16. j=1; 17. if(flag) 18. { 19. switch (state) 20. { 21. case S1MHZ: 22. plln = 16; // 16 7 9 23. pllm = 7; 24. prescaler = 9; 25. j=1; 26. break; 27. case S6MHz: 28. plln = 24;//24 7 0 29. pllm = 7; 30. prescaler = 0; 31. j=1000; 32. break; 33. case S10MHZ: 34. plln = 40;//40 7 0 35. pllm = 7; 36. prescaler = 0; 37. j=1400; 38. break; 39. case S16MHZ: 40. plln = 64;// 64 7 0 41. pllm = 7; 42. prescaler = 0; 43. j=1800; 44. break; 45. case S40MHZ: 46. plln = 20;// 20 0 0 47. pllm = 0; 48. prescaler = 0; 49. j=3000; 50. break; 51. default: 52. break; 53. } 54. SystemClock\_Config(); 55. state = state == S40MHZ ? S1MHZ : state + 1; 56. } 57. GPIOA->BSRR = (1 << 5); 58. //prev\_btn = curr\_btn; 59. flag=0; 60. //int test = GPIO\_ReadInputDataBit(GPIOC, GPIO\_Pin\_13); 61. for(int i=0;i<1000;i++){ 62. //          if(flag==0) 63. //              curr\_btn = GPIO\_ReadInputDataBit(GPIOC, GPIO\_Pin\_13); 64. if(!GPIO\_ReadInputDataBit(GPIOC, GPIO\_Pin\_13)&&flag==0){ 65. for(int k=1;k<j;k++) 66. delay\_1s(); 67. flag=1; 68. } 69. delay\_1s(); 70. } 71. GPIOA->BRR = (1 << 5); 72. for(int i=0;i<1000;i++){ 73. //          if(flag ==0) 74. //              curr\_btn = GPIO\_ReadInputDataBit(GPIOC, GPIO\_Pin\_13); 75. if(!GPIO\_ReadInputDataBit(GPIOC, GPIO\_Pin\_13)&&flag==0){ 76. for(int k=1;k<j;k++) 77. delay\_1s(); 78. flag=1; 79. } 80. delay\_1s(); 81. } 82. //prev\_btn = curr\_btn; 83. //curr\_btn = GPIO\_ReadInputDataBit(GPIOC, GPIO\_Pin\_13); 84. } 85. } 86. void SystemClock\_Config() 87. { 88. RCC->CFGR = 0x00000000; 89. // CFGR reset value 90. RCC->CR &= 0xFEFFFFFF; 91. // main PLL enable: PLL off 92. while (RCC->CR & 0x02000000); 93. // main PLL clock ready flag: PLL locked 94. RCC->PLLCFGR = 0x01000001; 95. // main PLL PLLCLK output enable: PLLCLK output enable 96. // main PLL entry clock source: MSI clock selected as PLL clock entry 97. RCC->PLLCFGR |= plln << 8; 98. // main PLL multiplication factor for VCO 99. RCC->PLLCFGR |= pllm << 4; 100. // division factor for the main PLL input clock 101. // f(VCO clock) = f(PLL clock input) × (PLLN / PLLM) 102. // f(PLL\_R) = f(VCO clock) / PLLR 103. RCC->CR |= 0x01000000; 104. // main PLL enable: PLL on 105. while (!(RCC->CR & 0x02000000)); 106. // main PLL clock ready flag: PLL locked 107. RCC->CFGR = 0x00000003; 108. // system clock switch: PLL selected as system clock 109. RCC->CFGR |= prescaler << 4; 110. // AHB prescaler: SYSCLK divided by N 111. } |

main7\_2.c

|  |
| --- |
| 1. #include "stm32l476xx.h" 2. #include "util.h" 3. #define TIME\_SEC 12.7 4. int cal\_len(int a) 5. { 6. int sum = 0; 7. while (a > 0) 8. { 9. a /= 10; 10. sum++; 11. } 12. return sum; 13. } 14. void timer\_init() 15. { 16. RCC->APB1ENR1 |= 0b1; 17. TIM2->ARR = (uint32\_t) (TIME\_SEC \* (4000000 / 40000)); // reload value 18. TIM2->PSC = (uint32\_t) 39999; // prescaler 19. TIM2->EGR = TIM\_EGR\_UG; // reinitialize the counter 20. } 21. void timer\_start() 22. { 23. TIM2->CR1 |= TIM\_CR1\_CEN; 24. display(0, -1003); 25. if (TIME\_SEC <= 0 || TIME\_SEC > 10000) 26. { 27. TIM2->CR1 &= ~TIM\_CR1\_CEN; 28. return; 29. } 30. int pre\_val = 0; 31. while (1) 32. { 33. int now\_val = TIM2->CNT; 34. if (pre\_val > now\_val) 35. { 36. TIM2->CR1 &= ~TIM\_CR1\_CEN; 37. return; 38. } 39. pre\_val = now\_val; 40. int len = cal\_len(now\_val); 41. if (now\_val < 100) 42. len = 3; 43. display(now\_val, -1000 - len); 44. } 45. } 46. int main() 47. { 48. gpio\_init(); 49. max7219\_init(); 50. timer\_init(); 51. timer\_start(); 52. } |

main7\_3.c

|  |
| --- |
| 1. #include "stm32l476xx.h" 2. #include "keypad.h" 3. #include "7-seg.h" 4. int intlen (int n); 5. void timer\_init (TIM\_TypeDef \*timer); 6. void timer\_start (TIM\_TypeDef \*timer); 7. void timer\_stop (TIM\_TypeDef \*timer); 8. void C4 (TIM\_TypeDef \*timer); 9. void D4 (TIM\_TypeDef \*timer); 10. void E4 (TIM\_TypeDef \*timer); 11. void F4 (TIM\_TypeDef \*timer); 12. void G4 (TIM\_TypeDef \*timer); 13. void A4 (TIM\_TypeDef \*timer); 14. void B4 (TIM\_TypeDef \*timer); 15. void C5 (TIM\_TypeDef \*timer); 16. void timer\_init (TIM\_TypeDef \*timer) 17. { 18. // Sound freq = 4 MHz / (pres + 1) / 100 19. // pres = 4 MHz / Sound freq / 100 - 1 20. timer->PSC = (uint32\_t) 152; 21. timer->ARR = (uint32\_t) 99; 22. /\* CH1 \*/ 23. timer->CCR1 = 50; 24. timer->CCMR1 |= TIM\_CCMR1\_OC1M\_2 | TIM\_CCMR1\_OC1M\_1; 25. timer->CR1 |= TIM\_CR1\_ARPE; 26. timer->EGR = TIM\_EGR\_UG; 27. timer->CCER |= TIM\_CCER\_CC1E;   /\* CH1 \*/ 28. } 29. void timer\_start (TIM\_TypeDef \*timer) 30. { 31. timer->CR1 |= TIM\_CR1\_CEN; 32. } 33. void timer\_stop (TIM\_TypeDef \*timer) 34. { 35. timer->CR1 &= ~TIM\_CR1\_CEN; 36. } 37. int main (void) 38. { 39. int     t = 0, key = 0, duty = 50; 40. TIM\_TypeDef \*timer = TIM3; 41. max7219\_init (); 42. display (duty, intlen (duty)); 43. keypad\_init (); 44. /\* GPIO: set PB4 as alternate function \*/ 45. RCC->AHB2ENR |= 0x1 << 1;   /\* enable AHB2 clock for port B \*/ 46. GPIOB->MODER |= GPIO\_MODER\_MODE4\_1; 47. GPIOB->AFR[0] |= GPIO\_AFRL\_AFSEL4\_1;    /\* PB4: AF2 (TIM3\_CH1) \*/ 48. RCC->APB1ENR1 |= RCC\_APB1ENR1\_TIM3EN; 49. timer\_init (timer); 50. while (1) { 51. if (!row4 () && !row3 () && !row2 () && !row1 ()) { 52. t = 0; 53. key = 0; 54. timer\_stop (timer); 55. } else { 56. if (key == 0) 57. key = keypad\_scan (); 58. if (t == 700) { 59. if (key & 0x1 << 1) { 60. C4 (timer); 61. timer\_start (timer); 62. } else if (key & 0x1 << 2) { 63. D4 (timer); 64. timer\_start (timer); 65. } else if (key & 0x1 << 3) { 66. E4 (timer); 67. timer\_start (timer); 68. } else if (key & 0x1 << 4) { 69. F4 (timer); 70. timer\_start (timer); 71. } else if (key & 0x1 << 5) { 72. G4 (timer); 73. timer\_start (timer); 74. } else if (key & 0x1 << 6) { 75. A4 (timer); 76. timer\_start (timer); 77. } else if (key & 0x1 << 7) { 78. B4 (timer); 79. timer\_start (timer); 80. } else if (key & 0x1 << 8) { 81. C5 (timer); 82. timer\_start (timer); 83. } else if (key & 0x1 << 14) { 84. duty += 500; 85. if (duty > 100000) 86. duty = 90; 87. display (duty, intlen (duty)); 88. timer->CCR1 = duty; 89. } else if (key & 0x1 << 15) { 90. duty -= 5; 91. if (duty < 10) 92. duty = 10; 93. display (duty, intlen (duty)); 94. timer->CCR1 = duty; 95. } 96. } 97. allhigh (); 98. ++t; 99. } 100. } 101. return 0; 102. } 103. void C4 (TIM\_TypeDef \*timer) 104. { 105. timer->PSC = (uint32\_t) 152;    // 4 MHz / 261.6 Hz / 100 - 1 = 151.90 = 152; 106. } 107. void D4 (TIM\_TypeDef \*timer) 108. { 109. timer->PSC = (uint32\_t) 135;    // 4 MHz / 293.7 Hz / 100 - 1 = 135.19 = 135 110. } 111. void E4 (TIM\_TypeDef \*timer) 112. { 113. timer->PSC = (uint32\_t) 120;    // 4 MHz / 329.6 Hz / 100 - 1 = 120.36 = 120 114. } 115. void F4 (TIM\_TypeDef \*timer) 116. { 117. timer->PSC = (uint32\_t) 114;    // 4 MHz / 349.2 Hz / 100 - 1 = 113.55 = 114 118. } 119. void G4 (TIM\_TypeDef \*timer) 120. { 121. timer->PSC = (uint32\_t) 101;    // 4 MHz / 392.0 Hz / 100 - 1 = 101.04 = 101 122. } 123. void A4 (TIM\_TypeDef \*timer) 124. { 125. timer->PSC = (uint32\_t) 90; // 4 MHz / 440.0 Hz / 100 - 1 = 89.91 = 90 126. } 127. void B4 (TIM\_TypeDef \*timer) 128. { 129. timer->PSC = (uint32\_t) 80; // 4 MHz / 493.9 Hz / 100 - 1 = 79.99 = 80 130. } 131. void C5 (TIM\_TypeDef \*timer) 132. { 133. timer->PSC = (uint32\_t) 75; // 4 MHz / 523.3 Hz / 100 - 1 = 75.44 = 75 134. } 135. int intlen (int n) 136. { 137. int len = 1; 138. while (n > 9) { 139. n /= 10; 140. ++len; 141. } 142. return len; 143. } |