# BAB V Timer pada Nuvoton NUC140

## Tujuan

1. Praktikan mengimplementasikan fungsi timer pada Nuvoton NUC140 untuk menghasilkan *interrupt* secara periodik guna mengendalikan tampilan pada seven segment dan LCD.
2. Praktikan menganalisis pengaruh pengaturan *prescaler* dan nilai TCMPR pada timer untuk menentukan waktu *timeout* yang akurat.
3. Praktikan mengintegrasikan peripheral GPIO, SPI3, dan TIMER0 pada Nuvoton NUC140 untuk mengontrol LED, LCD, dan *seven* *segment* secara bersamaan.
4. Praktikan memverifikasi kemampuan Nuvoton NUC140 dalam menangani sinyal *interrupt* dari *timer* untuk memperbarui tampilan secara *real*-*time.*
5. Praktikan mengevaluasi efektivitas konfigurasi pin menggunakan Coocox CoSmart untuk mempermudah pengembangan aplikasi berbasis mikrokontroler.

## Dasar Teori

### ARM Nuvoton NUC140

Mikrokontroler Nuvoton NUC140 merupakan bagian dari NUC140 Series, yaitu mikrokontroler berbasis ARM Cortex-M0 dengan arsitektur RISC 32-bit yang dikembangkan oleh *Nuvoton Technology Corporation*. Mikrokontroler ini dirancang untuk kontrol industri serta aplikasi yang memerlukan fungsi komunikasi khusus. Cortex-M0 sendiri adalah prosesor ARM terbaru yang menawarkan kinerja 32-bit dengan biaya setara dengan mikrokontroler 8-bit, sehingga memberikan efisiensi tinggi dengan harga yang kompetitif. NuMicro Seri NUC1xx, termasuk NUC140, memiliki kecepatan prosesor hingga 50 MHz, serta dilengkapi dengan memori Flash 32KB/64KB/128KB, SRAM 4KB/8KB/16KB, dan Memori Flash Loader 4KB untuk ISP (*In*-*System Programming*).

Selain itu, NUC140 juga mendukung berbagai *periferal*, seperti GPIO, *Timer*, *Watchdog Timer*, RTC, PDMA, UART, SPI/MICROWIRE, I2C, I2S, PWM, LIN, CAN, PS2, USB 2.0 FS Device, ADC 12-bit, Komparator Analog, *Low Voltage Reset*, dan *Brown Out Detector*. Dengan fitur-fitur tersebut, NUC140 sangat ideal untuk berbagai aplikasi tertanam, termasuk otomasi industri, perangkat IoT, dan sistem kontrol komunikasi. Keunggulan lainnya adalah kemampuannya untuk terintegrasi dengan ASIC (*Application*-*Specific Integrated Circuit*), yang sering digunakan dalam perangkat seperti kontroler printer, mesin cuci, video dekoder, dan *router ethernet*. Kombinasi kinerja tinggi, konsumsi daya rendah, dan dukungan *periferal* yang luas menjadikan Nuvoton NUC140 sebagai solusi optimal untuk sistem tertanam yang membutuhkan efisiensi dan keandalan tinggi.

(Sumber : [*https://latiful.hayat.web.id/mikrokontroler-nuvoton-nuc140-arm/*](https://latiful.hayat.web.id/mikrokontroler-nuvoton-nuc140-arm/) )

### Coocox CoIDE

CooCox CoIDE adalah perangkat lunak pengembangan gratis berbasis *Eclipse* yang dirancang khusus untuk mikrokontroler ARM. *Software* ini telah terintegrasi dengan GNU Compiler (GCC), sehingga memungkinkan pengembangan perangkat lunak yang lebih efisien dan fleksibel. CooCox CoIDE mendukung berbagai arsitektur ARM, termasuk ARM7, ARM9, serta seri Cortex-M0, M3, dan M4. Dengan fitur-fitur yang komprehensif dan antarmuka yang ramah pengguna, CooCox CoIDE menjadi pilihan ideal bagi pengembang sistem tertanam yang ingin mengembangkan aplikasi berbasis ARM dengan lebih mudah dan efisien.

(Sumber : *https://fahmizaleeits.wordpress.com/tag/arm-dengan-coocox-co-ide/*)

### Coocox CoSmart

CooCox CoSmart adalah alat yang dirancang untuk membantu dalam konfigurasi pin dan pembuatan kode secara efisien. Alat ini memungkinkan pengguna untuk mengatur konfigurasi pin *multiplexing* dengan mudah, mendeteksi konflik dalam pengaturan pin, serta memastikan bahwa setiap konfigurasi berjalan dengan optimal. Selain itu, CooCox CoSmart juga mampu menghasilkan kode secara otomatis berdasarkan pengaturan yang telah ditentukan, sehingga mempercepat proses pengembangan perangkat lunak. Fitur lainnya termasuk spesifikasi karakteristik sel I/O, yang memberikan informasi mendetail tentang bagaimana setiap pin berfungsi dalam sistem. Dengan berbagai fitur ini, CooCox CoSmart menjadi alat yang sangat berguna bagi para pengembang yang bekerja dengan mikrokontroler berbasis ARM.

(Sumber : [*https://sourceforge.net/projects/cosmart/*](https://sourceforge.net/projects/cosmart/) *)*

### Seven Segment

Seven Segment adalah jenis display elektronik yang terdiri dari tujuh segmen LED (*Light Emitting Diode*) yang disusun membentuk pola angka 8. Setiap segmen dapat dinyalakan atau dimatikan secara individu untuk menampilkan angka desimal (0-9) atau beberapa karakter huruf sederhana (seperti A, b, C, dll.). *Seven segment* umumnya digunakan dalam aplikasi seperti jam digital, kalkulator, dan sistem *embedded* karena sederhana dan hemat biaya.

Pada Nuvoton NUC140, seven segment dihubungkan ke port GPIO, khususnya pada pin GPE 0-7 untuk baris dan GPC 4-7 untuk kolom, yang memungkinkan kontrol melalui multiplexing. Multiplexing dilakukan dengan mengaktifkan kolom secara bergantian untuk menampilkan digit yang berbeda, sehingga mengurangi jumlah pin yang diperlukan. Untuk mengendalikan *seven segment*, mikrokontroler mengatur status logika (HIGH/LOW) pada pin GPIO berdasarkan pola segmen yang diinginkan.

Proses pengendaliannya melibatkan pengaturan *timer* atau *delay* untuk menjaga *refresh rate* yang cukup cepat agar tampilan terlihat stabil bagi mata manusia (biasanya 50-100 Hz per kolom). *Library* seperti "Seven\_Segment.h" pada Coocox CoIDE menyediakan fungsi untuk mempermudah konfigurasi dan pengendalian tampilan seven segment.

(Sumber : *https://artifungsi.com/seven-segment-display/*)

### LED dan LCD

LED (*Light Emitting Diode*) adalah komponen semikonduktor yang menghasilkan cahaya ketika arus listrik mengalir melaluinya. Pada Nuvoton NUC140, LED dihubungkan ke pin GPIO seperti GPA 12-14 (PWM0-PWM2) dan GPC 12-15, yang memungkinkan pengendalian on/off atau modulasi intensitas cahaya melalui PWM (*Pulse Width Modulation*). LED sering digunakan sebagai indikator status atau untuk keperluan debugging pada sistem embedded. Pengendalian LED dilakukan dengan mengatur logika HIGH (menyalakan) atau LOW (mematikan) pada pin GPIO yang sesuai.

LCD (*Liquid Crystal Display*), khususnya LCD Dot Matriks 128x64 yang digunakan dalam percobaan ini, adalah display yang menggunakan kristal cair untuk menampilkan informasi grafis atau teks. LCD ini dihubungkan ke Nuvoton NUC140 melalui antarmuka SPI3 (pin GPD 8-11 untuk SS30, SPCLK, MISO0, MOSI0) dengan pin tambahan GPD 14 untuk mengontrol lampu latar. LCD ini mendukung tampilan teks atau grafis dengan resolusi 128x64 piksel, yang dikendalikan melalui library seperti "LCD\_Driver.h". Pengendalian LCD melibatkan pengiriman perintah dan data melalui protokol SPI untuk mengatur konten yang ditampilkan, seperti teks atau gambar. Baik LED maupun LCD memerlukan konfigurasi pin yang tepat melalui GPIO atau peripheral seperti SPI, serta pengaturan waktu yang akurat untuk memastikan operasi yang stabil. Penggunaan Coocox CoSmart mempermudah pemetaan pin, sementara Coocox CoIDE menyediakan lingkungan untuk mengembangkan kode pengendalian.

(Sumber : *https://proactiveducation.com/5-proyek-layar-lcd-dan-oled-dengan-arduino/*)

## Langkah Kerja

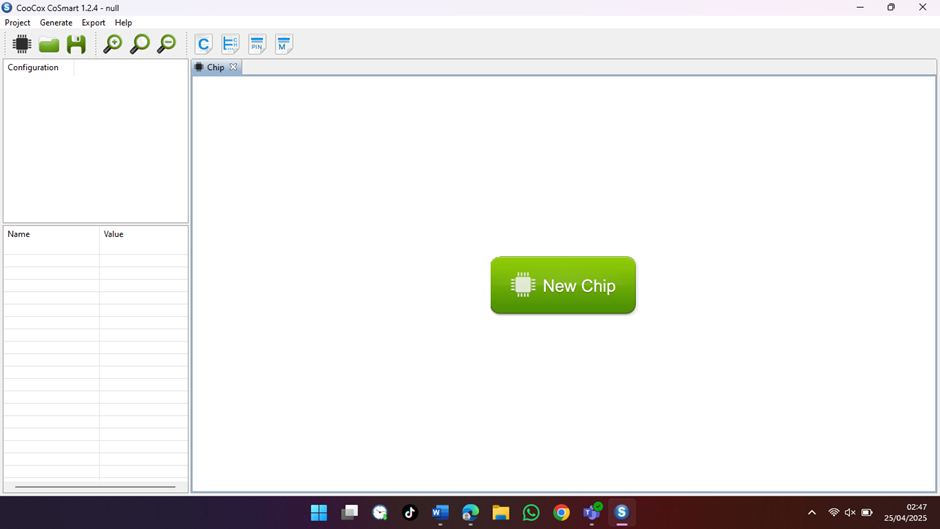
### Percobaan 1

1. Colokan Kabel Nu-Link USB ke bagian ICE Bridge Nu-Link di Nuvoton NUC140, dan Colokkan sisi lain kabelnya ke USB Port pada Laptop Kalian



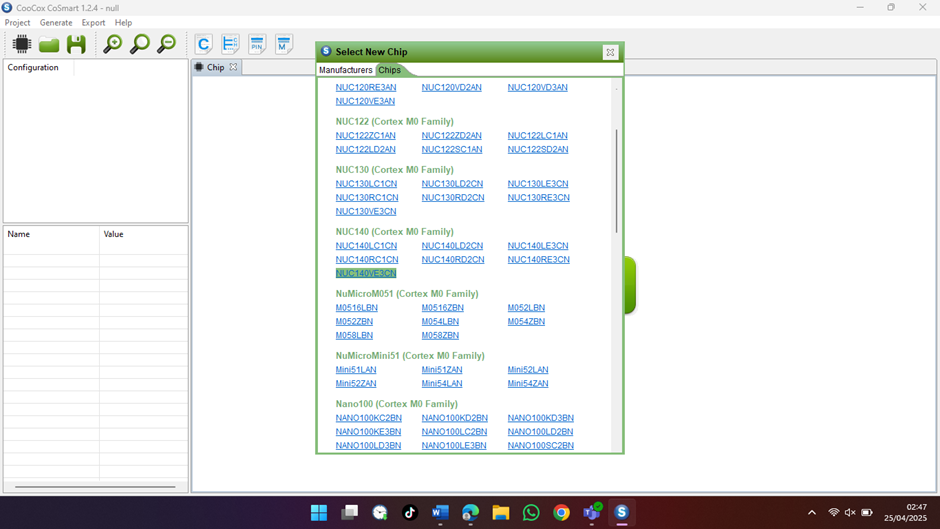
Gambar 1 Colokkan kabel

1. Kemudian bukalah CoSmart, lalu Pilih New Chip



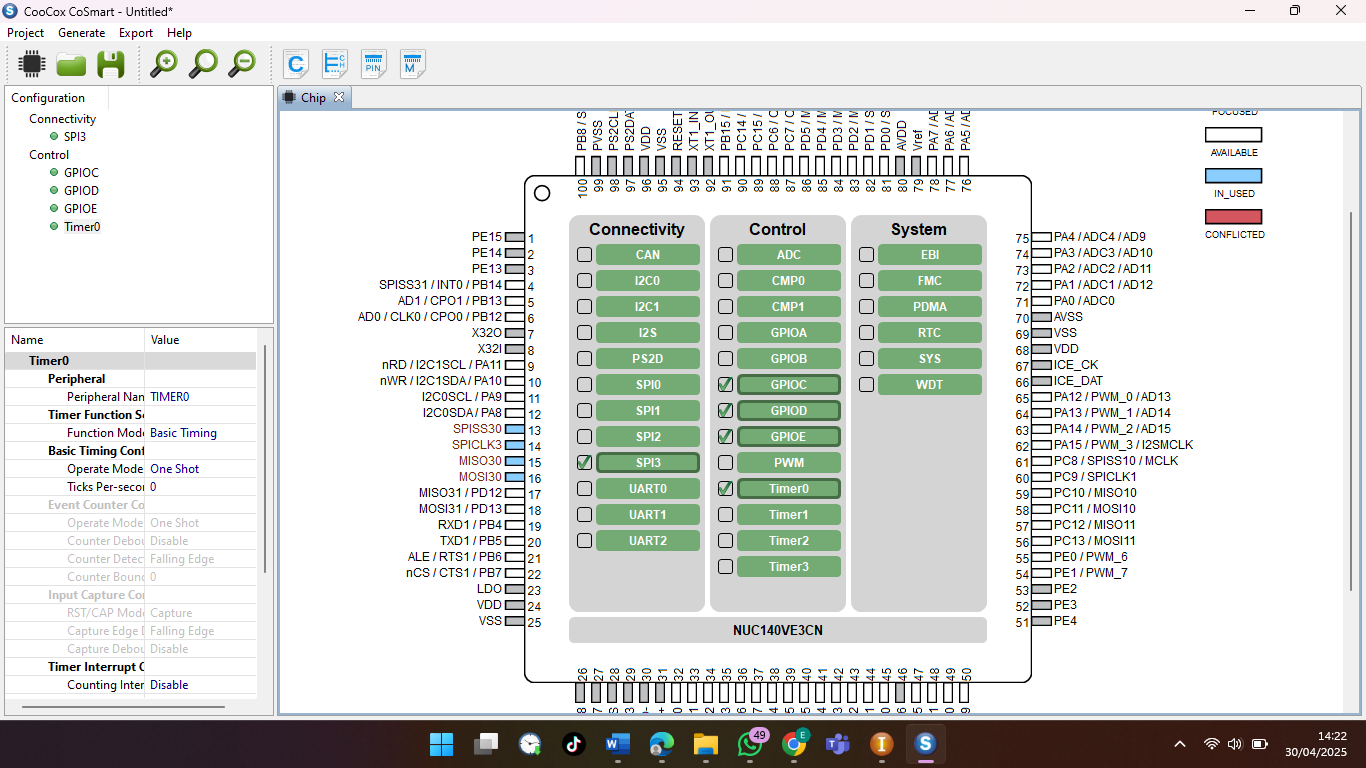
Gambar 2 Membuka CoSmart dan memilih New Chip

1. Kemudian pilih Nuvoton
2. Lalu pada NUC140 pilih NUC140VE3CN



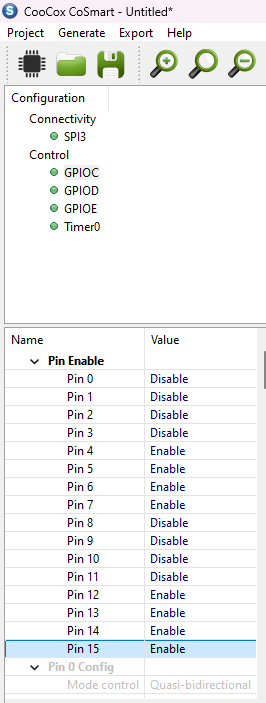
Gambar 3 Pilih NUC140VE3CN

1. Lalu akan muncul tampilan Chip di Panel Utama, kemudian centang GPIOC, GPIOD, GPIOE, SPI3, dan TIMER0



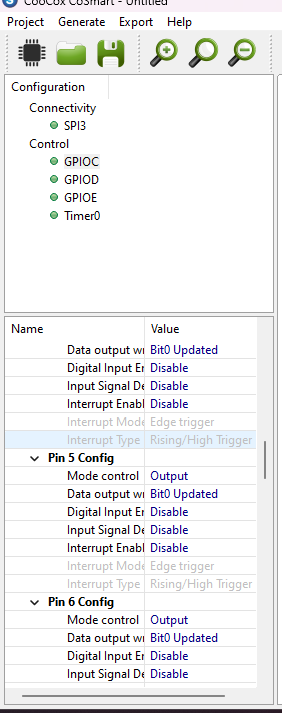
Gambar 4 Tampilan chip di panel utama

1. Lalu pada bagian Panel Kiri, ubah Pin Enable dari Pin 12 – 15 dan Pin 4 - 7 pada GPIOC menjadi Enable



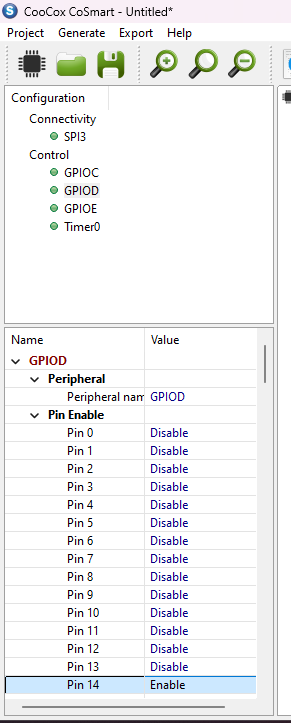
Gambar 5 Ubah pin menjadi enable

1. Lalu Scroll ke bawah dan ubah Mode Pin 12 – 15 dan Pin 4 – 7 menjadi Output



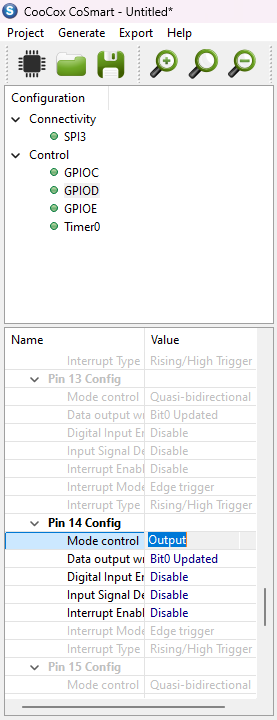
Gambar 6 Ubah mode pin menjadi Output

1. Kemudian di Panel Kiri (*Configuration)*, ganti dari GPIOC menjadi GPIOD, lalu ubah Pin Enable dari Pin 14 pada GPIOD menjadi Enable



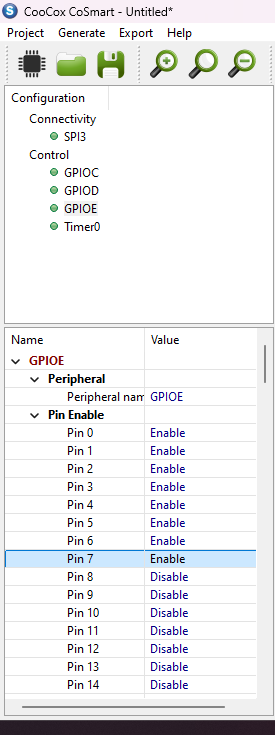
Gambar 7 Ubah pin menjadi enable

1. Lalu *Scroll* ke bawah dan Ubah Mode menjadi *Output*



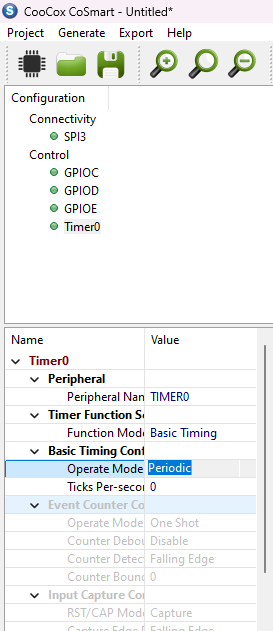
Gambar 8 Ubah mode pin menjadi output

1. Lalu ganti dari GPIOD menjadi GPIOE, lalu ubah Pin 0 – 7 menjadi *Enable*, lalu Ubah Mode menjadi *Output*



Gambar 9 Ubah Pin menjadi enable

1. Lalu



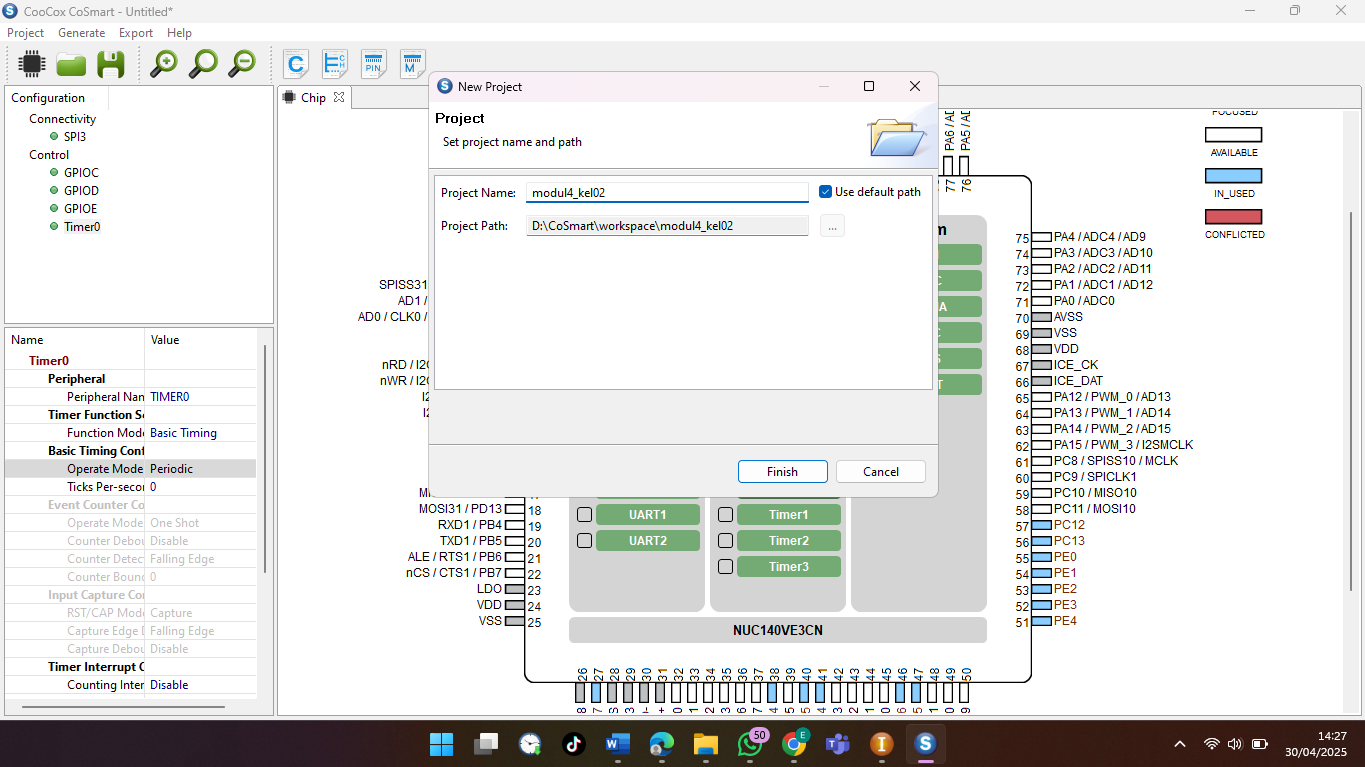
Gambar 10 Ubah mode menjadi Output

1. Setelah itu, Klik *Generate*, lalu pilih Generate CoIDE Project



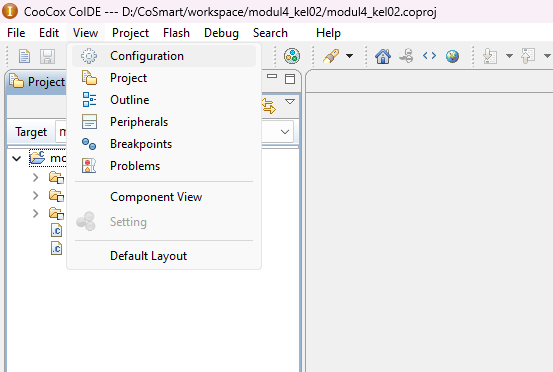
Gambar 11 Klik Generate CoIDE Project

1. Lalu masukkan *Project Name*, dan pilih lokasi untuk menyimpan *Project*, kemudian Klik *Finish*



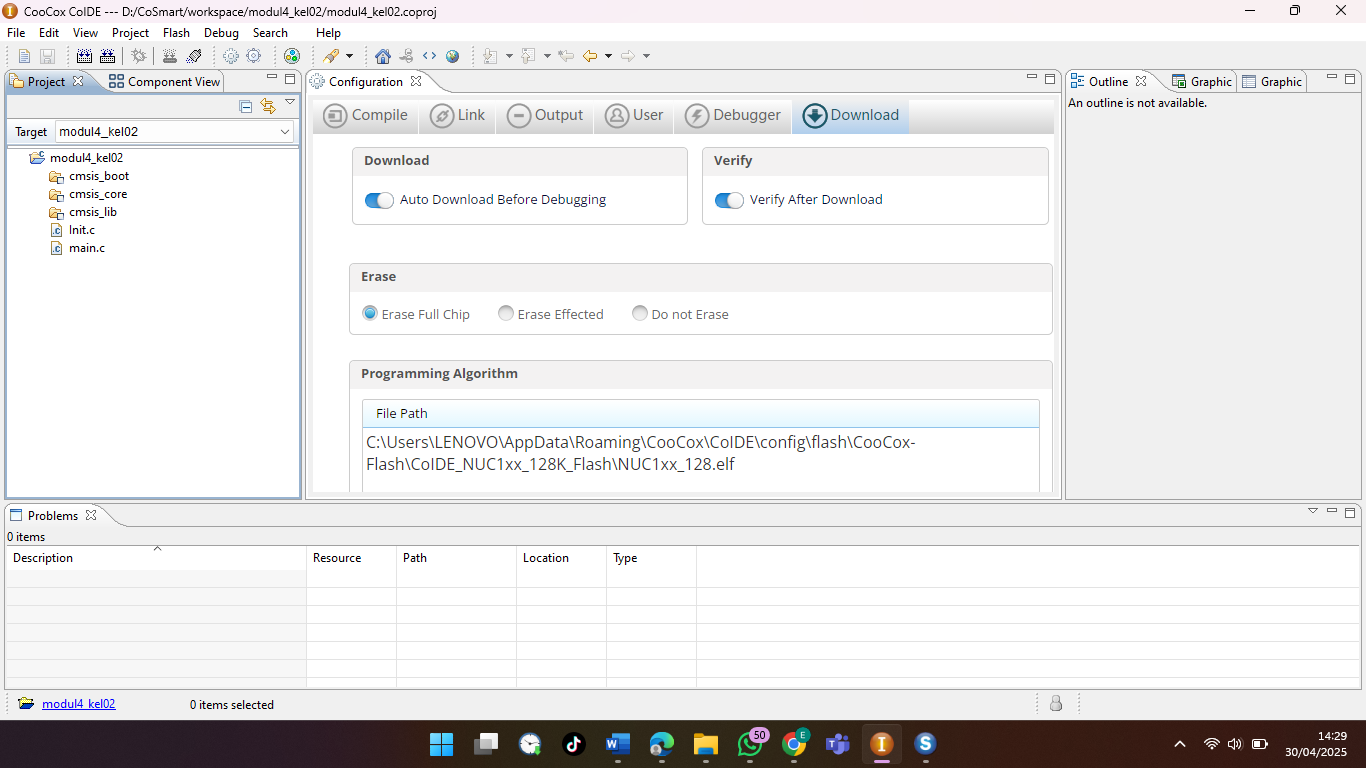
Gambar 12 Buat project

1. Kemudain buka Coocox CoIDE, pilih Project, lalu *Open Project* dan cari *Project* yang tadi telah dibuat melalui CoSmart
2. Kemudian setelah itu, klik *Open*. Lalu klik pada bagian *View* pilih *Configuration*



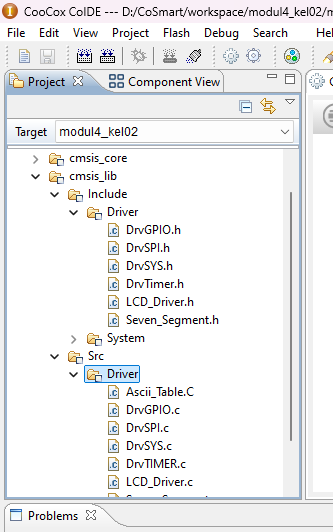
Gambar 13 Pilih View lalu Configuration

1. Pada bagian *Download*, pada Erase ganti yang awalnya Erase Effected menjadi Erase Full Chip



Gambar 14 Ubah Erase Effected menjadi Erase Full Chip

1. Lalu masukkan *Library* untuk Seven Segment dan LCD ke dalam *Project* baik yang berformat C maupun yang berformat H

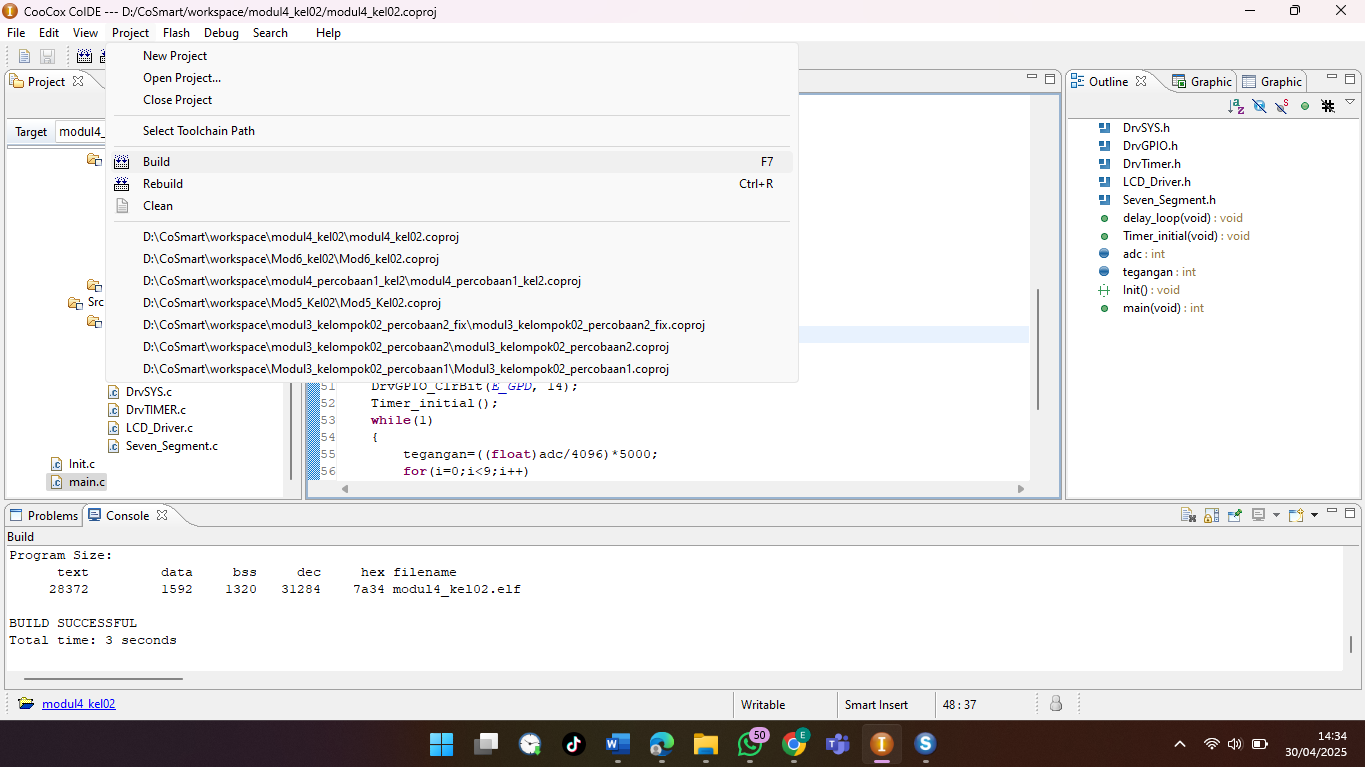


Gambar 15 Masukkan Library seven segment dan LCD

1. Lalu selanjutnya pada Panel Kiri (*Project*) pilih Main.c, kemudian masukkan Source Code yang telah disediakan

|  |
| --- |
| #include "DrvSYS.h"  #include "DrvGPIO.h"  #include "DrvTimer.h"  #include "LCD\_Driver.h"  #include "Seven\_Segment.h"  void delay\_loop(void)  {  uint32\_t i,j;  for(i=0;i<2;i++)  {  for(j=0;j<10000;j++);  }  }  void Timer\_initial(void)  {  SYSCLK->CLKSEL1.TMR0\_S = 0;  SYSCLK->APBCLK.TMR0\_EN = 1;  TIMER0->TCSR.MODE=1;  TIMER0->TCSR.PRESCALE=0;  TIMER0->TCMPR = 2400000;  TIMER0->TCSR.IE = 1;  TIMER0->TISR.TIF = 1;  NVIC\_EnableIRQ(TMR0\_IRQn);  TIMER0->TCSR.CRST = 1;  TIMER0->TCSR.CEN = 1;  TIMER0->TCSR.TDR\_EN = 1;  }  int adc;  int tegangan;  void Init();  int main(void)  {  Init();  int i=0, j=0;  UNLOCKREG();  DrvSYS\_SetOscCtrl(E\_SYS\_XTL12M, 1);  SysTimerDelay(5000);  DrvSYS\_SelectHCLKSource(0);  LOCKREG();  DrvSYS\_SetClockDivider(E\_SYS\_HCLK\_DIV, 0);  Initial\_pannel();  clr\_all\_pannal();  print\_lcd(0, “Florencia Irena”);  print\_lcd(1, “Timer Interrupt”);  print\_lcd(2, “Kelompok XX”);  print\_lcd(3, “Sistem Komputer”);  DrvGPIO\_ClrBit(E\_GPD, 14);  Timer\_initial();  while(1)  {  tegangan=((float)adc/4096)\*5000;  for(i=0;i<9;i++)  {  for(j=0;j<4;j++)  {  show\_seven\_segment(j, i);  delay\_loop();  }  }  }  } |

1. Selanjutnya pilih *Project*, lalu *Build.*



Gambar 16 Pilih Project lalu Build

1. Kemudian setelah *Build Success,* maka pilih Flash kemudian Program Download



Gambar 17 Lakukan Flash Program dan Program Download

1. Lalu, amatilah apa yang terjadi bagian *Board* Nuvoton NUC140 dengan menekan tombol *Switch Button*

## Hasil Percobaan dan Analisis Percobaan

### Percobaan 1

Percobaan ini bertujuan untuk mengkonfigurasi board Nuvoton NUC140VE3CN menggunakan CoSmart dan CoIDE untuk mengimplementasikan timer interrupt yang mengontrol tampilan seven segment dan LCD. Langkah-langkah dimulai dengan menghubungkan board ke laptop via Nu-Link USB, mengatur pin GPIO (GPIOC, GPIOD, GPIOE) dan komponen SPI3 serta TIMER0 pada CoSmart, dan menghasilkan proyek CoIDE. Pin GPIOC (4-7, 12-15), GPIOD (14), dan GPIOE (0-7) dikonfigurasi sebagai output, library seven segment dan LCD ditambahkan, dan kode main.c diimplementasikan untuk menginisialisasi timer interrupt (frekuensi 1 Hz dengan clock 12 MHz dan prescaler 0), menampilkan teks statis pada LCD, serta menampilkan angka 0-8 secara berurutan pada four-digit seven segment. Proses build dan pengunduhan berhasil, dan pengamatan dilakukan dengan menekan tombol switch pada board.

Hasil percobaan menunjukkan bahwa sistem berfungsi sebagaimana mestinya. LCD menampilkan teks statis seperti "Florencia Irena", "Timer Interrupt", "Kelompok XX", dan "Sistem Komputer" dengan jelas. Seven segment secara berulang menampilkan angka 0 hingga 8 pada masing-masing digit (0-3) dengan jeda yang diatur oleh fungsi delay\_loop(), yang memastikan transisi angka terlihat jelas. Timer interrupt yang diatur dengan TCMPR = 2400000 menghasilkan interupsi setiap 1 detik (berdasarkan clock 12 MHz), meskipun dalam kode ini interupsi tidak secara eksplisit digunakan untuk mengubah tampilan, melainkan hanya menginisialisasi timer. Ketika tombol switch ditekan, tidak ada perubahan spesifik yang diamati pada LCD atau seven segment, karena kode tidak menyertakan fungsi untuk mendeteksi input tombol.

## Tugas Praktikum

*Source Code*

|  |
| --- |
| #include "DrvSYS.h"  #include "DrvGPIO.h"  #include "DrvTimer.h"  #include "LCD\_Driver.h"  #include "Seven\_Segment.h"  void delay\_ms(uint32\_t ms)  {  DrvSYS\_Delay(ms \* 1000);  }  void myInit(void)  {  UNLOCKREG();  DrvSYS\_SetOscCtrl(E\_SYS\_XTL12M, 1);  SysTimerDelay(5000);  DrvSYS\_SelectHCLKSource(0);  LOCKREG();  DrvSYS\_SetClockDivider(E\_SYS\_HCLK\_DIV, 0);    Initial\_pannel();  clr\_all\_pannal();    print\_lcd(0, "Prak TMA Mod4");  print\_lcd(1, "Edi, Anis, Naufal");  print\_lcd(2, "Kelompok 02");  print\_lcd(3, "Teknik Komputer");    DrvGPIO\_Open(E\_GPD, 14, E\_IO\_OUTPUT)  DrvGPIO\_ClrBit(E\_GPD, 14);  }  void refresh\_display(uint16\_t value)  {  uint8\_t d3 = (value / 1000) % 10;  uint8\_t d2 = (value / 100) % 10;  uint8\_t d1 = (value / 10) % 10;  uint8\_t d0 = value % 10;    close\_seven\_segment();  show\_seven\_segment(3, d3); delay\_ms(2);    close\_seven\_segment();  show\_seven\_segment(2, d2); delay\_ms(2);    close\_seven\_segment();  show\_seven\_segment(1, d1); delay\_ms(2);    close\_seven\_segment();  show\_seven\_segment(0, d0); delay\_ms(2);    close\_seven\_segment();  }  uint16\_t detikToMMSS(uint16\_t totalSeconds)  {  uint8\_t minutes = totalSeconds / 60;  uint8\_t seconds = totalSeconds % 60;  return (minutes \* 100) + seconds;  }  int main(void)  {  myInit();    uint16\_t countdown = 4 \* 60 + 2;  uint8\_t interval = 2;    while (countdown > 0)  {   uint16\_t mmss = detikToMMSS(countdown);     for (int i = 0; i < 250; i++)   {   refresh\_display(mmss);   }     delay\_ms(interval \* 1000);   if (countdown >= interval)   countdown -= interval;   else   countdown = 0;  }      DrvGPIO\_SetBit(E\_GPD, 14);  for (int i = 0; i < 250; i++)   refresh\_display(0);  delay\_ms(3000);  DrvGPIO\_ClrBit(E\_GPD, 14);    while (1)  {   refresh\_display(0);  } |

Analisis Kode:

Kode di atas merupakan program *countdown timer* berbasis mikrokontroler dengan tampilan 7-segment dan kendali LED menggunakan pustaka DrvSYS, DrvGPIO, dan lainnya dari sistem tertanam seperti Nuvoton. *Timer* dimulai dari waktu 04:02 (4 menit 2 detik) dan akan berkurang setiap 2 detik (interval = 2) seperti yang terlihat pada gambar dibawah. Fungsi detikToMMSS mengubah format detik menjadi format MMSS agar bisa ditampilkan dengan benar di 7-segment melalui refresh\_display(), yang bekerja dengan sistem multiplexing (menyalakan digit satu per satu dengan delay pendek).

Ketika waktu mencapai 00:00, LED yang terhubung ke pin GPD14 akan menyala selama 3 detik sebagai indikator waktu habis, lalu tampilan 7-segment menunjukkan 0000 secara terus menerus. Kode ini juga mengatur inisialisasi sistem, LCD untuk informasi kelompok, serta pengaturan clock dan GPIO.



Gambar 18 Hasil board percobaan yang dilakukan



Gambar 19 Foto kelompok hasil percobaan

## Kesimpulan

1. Pengaturan timer dengan prescaler = 0 dan TCMPR = 2.400.000 menghasilkan interrupt periodik yang konsisten, memungkinkan pembaruan tampilan *seven segment* dan LCD secara teratur. Hal ini dibuktikan dengan perubahan nilai yang ditampilkan pada *seven segment* secara berurutan.
2. Penggunaan GPIO (GPIOC, GPIOD, GPIOE) dan SPI3 untuk mengendalikan seven segment, LCD, dan LED berjalan dengan baik, menunjukkan bahwa konfigurasi pin melalui CoSmart akurat dan mendukung komunikasi antarperipheral.
3. Fungsi delay\_loop yang digunakan untuk mengatur waktu tampilan *seven* *segment* menyebabkan sedikit penundaan dalam respons sistem, menunjukkan bahwa pengaturan waktu berbasis *timer* lebih efisien dibandingkan *delay* berbasis *loop*.
4. Informasi seperti nama, judul, dan kelompok yang ditampilkan pada LCD tetap stabil, sementara *seven segment* menampilkan angka secara berurutan, mengindikasikan bahwa *interrupt timer* tidak mengganggu operasi LCD.
5. Pengaturan *clock* eksternal 12 MHz (XTL12M) dan *enabling interrupt* pada NVIC memastikan timer bekerja dengan akurasi tinggi, yang terlihat dari konsistensi siklus pembaruan tampilan.