Catalog

**[Mã đề thi: CĐT – LT 01 1](#_Toc29127)**

**[Mã đề thi: CĐT – LT 02 5](#_Toc10433)**

**[Mã đề thi: CĐT – LT 03 8](#_Toc2504)**

**[Mã đề thi: CĐT – LT 04 11](#_Toc14203)**

**[Mã đề thi: CĐT – LT 05 16](#_Toc16656)**

**[Mã đề thi: CĐT – LT 06 21](#_Toc9775)**

**[Mã đề thi: CĐT – LT 07 27](#_Toc16620)**

**[Mã đề thi: CĐT – LT 08 31](#_Toc16902)**

**[Mã đề thi: CĐT – LT 09 36](#_Toc31582)**

**[Mã đề thi: CĐT – LT 10 40](#_Toc24669)**

**[Mã đề thi: CĐT – LT 11 45](#_Toc2733)**

**[Mã đề thi: CĐT – LT 12 49](#_Toc11418)**

**[Mã đề thi: CĐT – LT 13 52](#_Toc2080)**

**[Mã đề thi: CĐT – LT 14 59](#_Toc12578)**

**[Mã đề thi: CĐT – LT 15 65](#_Toc7308)**

**Mã đề thi: CĐT – LT 01**

|  |  |  |
| --- | --- | --- |
| **Câu** | **Đáp án** | **Điểm** |
| 1 | Phân tích loại mạch và số FF cần dùng.  Modul 16 tương ứng với bộ đếm đầy đủ 4 bit từ 15 lùi về đến 0 rồi lặp lại.  Sơ đồ mạch đếm    Số đếm là DCBA  Giản đồ xung của mạch đếm  CK  A  B  C  D | 0,5  0,5  1 |
| 2 | Lệnh If  Lệnh kiểm tra điều kiện. Khi gặp lệnh này, chương trình sẽ kiểm tra các điều kiện, nếu thỏa mãn thì sẽ thực hiện các lệnh bên trong nó.  Có 3 dạng if.   * Dạng 1 : If ( điều kiện so sánh )   { lệnh 1;  Lệnh 2;  ......  Lệnh n;  }   * Dạng 2 :   If ( điều kiện so sánh )  { lệnh 1;  Lệnh 2;  ......  Lệnh n;  }  elseif ( điều kiện so sánh )  { lệnh 1;  Lệnh 2;  ......  Lệnh n;  }   * Dạng 3 :   If ( điều kiện so sánh )  { lệnh 1;  Lệnh 2;  ......  Lệnh n;  }  else  { lệnh 1;  Lệnh 2;  ......  Lệnh n;  }  Muốn khai báo lệnh if chúng ta phải vào Structure/Avanced/If. Sau đó tùy chỉnh lệnh với tab Command | 0.5  1.0  0.5 |
| 3 | **Counter đếm lên ký hiệu : CTU**    Trong đó:  CU là chân nhận xung đếm lên.  R là chân reset. Chức năng reset giá trị đếm và Cbit của counter.  PV là nơi đặt giá trị đếm của counter.  **Nguyên lý hoạt động của CTU:**  - CTU là bộ đếm sườn lên của xung tín hiệu đầu vào, mỗi lần nhận xung cạnh lên tại CTU thì giá trị counter tăng lên 1. Giá trị đếm này được lưu vào vào thanh ghi 2 byte của bộ đếm gọi là thanh ghi C-word.  - Khi giá trị đếm của counter bằng giá trị đặt thì Cbit = 1 .  - Bộ đếm tiến CTU có chân thực hiện chức năng reset, khi đầu vào chân reset có giá trị logic bằng 1, lúc đó bộ đếm bị reset. Giá trị của C-word và C-bits đều bằng 0.  - Bộ đếm ngừng đếm khi giá trị C-word đạt giá trị 32767.  **Ví dụ về CTU:**  Tương tự là couter đếm xuống CTD ký hiệu:    Trong đó  CD là chân nhận xung đếm xuống.  LD là chân Load. Khi có xung cạnh lên tại đây, giá trị counter bằng giá trị đặt.  PV là nơi đặt giá trị đếm của counter.  **Nguyên lý hoạt động của CTD:**  - CTD là bộ đếm sườn lên của xung tín hiệu đầu vào, mỗi lần nhận xung cạnh lên tại CTU thì giá trị counter giảm xuống 1. Giá trị đếm này được lưu vào vào thanh ghi 2 byte của bộ đếm gọi là thanh ghi C-word.  - Bộ đếm tiến CTD có chân LD, khi đầu vào chân LD có giá trị logic bằng 1, lúc đó bộ đếm có giá trị bằng giá trị đặt (PV) và C-bits bằng 0.  - Khi giá trị đếm của counter bằng 0 giá trị đặt thì Cbit = 1.  - Bộ đếm ngừng đếm khi giá trị C-word giảm xuống còn 0.  Ví dụ về CTD:    Ví dụ theo yêu cầu của đề: | 0,5  1,0  0,5  1,0 |
| 4 | - Hoạt động định thời chế độ 2 của bộ TIMER0:    osc  **/ 12**  Trong chế độ 2, bộ Timer dùng TL0 để chứa giá trị đếm và TH0 để chứa giá trị nạp lại vì vậy chế độ này được gọi là chế độ tự nạp lại 8 bit. Sau khi đếm quá 255 sẽ xảy ra tràn, khi đó TF0 được đặt bằng 1 đồng thời giá trị của Timer tự động được nạp lại bằng nội dung của TH0.  Hình trên mô tả hoạt động của các Timer ở chế độ 2: Nguồn xung clock được đưa tới Timer từ một trong cách phụ thuộc vào bit C-/T0 trong thanh ghi TMOD:  • C/ = 0, xung clock sẽ được lấy từ bộ chia tần trong chip, tần số của xung ở đây là 1/12 tần số của bộ dao động thạch anh (Fosc).  Nguồn xung clock nói trên sẽ được điều khiển để đưa tới các Timer bằng các bit: TR0, GATE và mức logic trên các chân INT0:  • Nếu TR0=0, các Timer sẽ bị cấm mà không cần quan tâm tới GATE và mức logic trên các chân INT0 (thể hiện bằng “cổng AND”).  • Nếu TR0=1, các Timer sẽ hoạt động với một trong 2 điều kiện sau xảy ra (thể hiện bằng cổng ‘OR”): Thứ nhất: bit GATE=1; thứ hai: trên chân INT0 có mức logic 1.  ***-*** Viết chương trình ví dụ tạo xung 10KHz trên chân P1.5:  #include <REGX51.H>  #include<stdio.h>  sbit xung=P1^5;  void main(void)  {  TMOD=0x02;  while(1)  {  TH0= TL0= -50;  TR0=1;  while(!TF0);  TF0=0;  xung=~xung;  }  } | 0,5  0,5  0,5  0,5  1 |

**Mã đề thi: CĐT – LT 02**

|  |  |  |
| --- | --- | --- |
| **Câu** | **Đáp án** | **Điểm** |
| 1 | - Vẽ đúng sơ đồ mạch điện:    - Trình bày đầy đủ nguyên lý hoạt động:   * + Khi 0 ≤ t < t1: Q1 tắt, Q2 dẫn bảo hoà   Lúc này, tụ C2 sẽ được nạp điện từ nguồn Vcc qua Rc1 để đạt đến giá trị Vcc, đồng thời tụ C1 sẽ xả và nạp từ nguồn Vcc qua Rb1 xuống Q2 đang dẫn bảo hoà.  Tụ C1 càng nạp thì điện áp trên tụ càng tăng, làm cho điện thế tại cực nền Q1 tăng lên cho đến khi Vb1 ≥ Vγ thì lúc này Q1 bắt đầu dẫn  ⇒ sụt áp trên Rc1 tăng nên vc/Q1 giảm dẫn đến tiếp giáp JE/Q2 phân cực yếu hơn  ⇒ BJT Q2 chuyển sang chế độ dẫn khuếch đại, làm cho vc/Q2 tăng và thông qua tụ C1 sẽ làm cho BJT Q1 nhanh chóng đi vào dẫn bảo hoà và Q2 nhanh chóng tắt.  Mạch chuyển sang trạng thái không bền thứ hai ứng với Q1 dẫn bảo hoà và Q2 tắt.   * + Khi t1 ≤ t < t2: Q1 dẫn bảo hoà, Q2 tắt   Lúc này, tụ C1 và C2 vẫn tiếp tục luân phiên xả và nạp điện nên tương tự như trên, Q1 sẽ chuyển dần sang dẫn khuếch đại và tắt, còn Q2 sẽ chuyển trạng thái từ tắt sang dẫn khuếch đại và nhanh chóng dẫn bảo hoà.  Quá trình chuyển đổi trạng thái trong mạch cứ tự động tiếp diễn nên mạch luôn luôn tạo độ dài xung ra. | 1  1 |
| 2 | 1. Lệnh wait   Chương trình sẽ dừng lại khi gặp lệnh này sẽ dừng một khoảng thời gian hay đợi một tín hiệu, một hàm,… nếu thỏa điều kiện thì sẽ chương trình sẽ chạy qua lệnh tiếp theo.  Có 3 loại lênh wait.   * Wait .... second. Lệnh chờ thời gian * Wait for Digital input. Lệnh chờ tín hiệu vào dạng số * Wait for analog input. Lệnh chờ tín hiệu vào dạng tương tự.   Muốn khai báo lệnh wait chúng ta phải vào Structure/Basics/wait. Sau đó tùy chỉnh lệnh với tab Command. | 0.5  1.0  0.5 |
| 3 | Trong PLC s7-200 có 2 loại timer không có nhớ là TON và TOF ( 16 bít )  Ký hiệu:    Độ phân giải của timer như sau:    Trong đó:  IN là nguồn nuôi timer.  PT là giá trị đặt của timer. 16 bít.  Timer TON: Còn gọi timer thường hở đóng chậm. Sẽ đếm khi đầu vào IN bằng 1. Giá trị Tbit = 1 khi giá trị đếm của timer lớn hơn hoặc bằng giá trị đặt. Có thể reset timer bằng lệnh reset hay làm cho giá trị ngõ vào IN bằng 0.  Timer TOF: Còn gọi là timer thường đóng mở chậm. Khi IN có điện giá trị Tbit bằng 1. Khi IN mất điện timer bắt đầu đếm thời gian. Khi thời gian Timer bằng thời gian đặt tại PT thì Tbit bằng 0.  Ví dụ:    Ví dụ theo yêu cầu: |  |
| 4 | CODE:  ORG 0000h  CHUC EQU R0  DONVI EQU R1  DEM EQU R2  LL: MOV DPTR,#MA  MOV CHUC,#05  MOV DONVI,#05  TT: MOV A,DONVI  MOVC A,@A+DPTR  MOV P1,A  MOV A,CHUC  MOVC A,@A+DPTR  MOV P2,A  CALL DL  CJNE DONVI,#05,M  CJNE CHUC,#02,M1  INC DEM  CJNE DEM,#02,LL  JMP HT  M: CJNE DONVI,#00,M1  MOV DONVI,#09  DEC CHUC  JMP TT  M1: DEC DONVI  JMP TT  HT: MOV P2,#08EH ; chu f  MOV P1,#0F9H ; chu i  CALL DL  JMP HT  DL: MOV R3,#0FFH  LAP1: MOV R4,#0FFH  LAP: DJNZ R4,LAP  DJNZ R3, LAP1  RET  MA: DB 0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H  END | 0,5  0,5  0,5  0,5  0,5  0,5 |

**Mã đề thi: CĐT – LT 03**

|  |  |  |
| --- | --- | --- |
| **Câu** | **Đáp án**  **B**i  **C**i-1  Ai  Ci  Si | **Điểm** |
| 1 | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Ngõ vào | | | Ngõ ra | | | Ai | Bi | Ci-1 | Ci | Si | | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 1 | 0 | 1 | | 0 | 1 | 0 | 0 | 1 | | 0 | 1 | 1 | 1 | 0 | | 1 | 0 | 0 | 0 | 1 | | 1 | 0 | 1 | 1 | 0 | | 1 | 1 | 0 | 1 | 0 | | 1 | 1 | 1 | 1 | 1 |     **Hàm logic:**  Ci = Ai Bi + Bi Ci-1 + Ai Ci-1  Si = Ai ⊕ Bi ⊕ Ci-1 | 0,5  0,5  0,5  0,5 |
| 2 | Lệnh set  Đặt một tín hiệu output lên mức Low hoặc High, tín hiệu output bằng hàm số,…  Lệnh này có chức năng là on hoặc off 1 ngõ ra của RB.  Có 2 dạng set.  Dạng 1: Set digital output: thiết lập trạng thái ngõ ra dạng số.  Dạng 2: Set analog output: thiết lập trạng thái ngõ ra dạng tương tự theo áp (0 -10 V) hoặc dòng (0-20mA).  Muốn khai báo lệnh wait chúng ta phải vào Structure/Basics/set. Sau đó tùy chỉnh lệnh với tab Command. | 0.5  1.0  0.5 |
| 3 | Trong PLC s7-200 timer có nhớ gọi là timer TONR  Ký hiệu:    Trong đó IN là nguồn nuôi Timer.  PT giá trị đặt timer ( 16 bít )  Độ phân giải Timer như sau:    Cũng giống như TON, TONR sẽ đếm khi IN có giá trị 1, tuy nhiên giá trị đếm timer này không mất đi khi nguồn nuôi. Khi có điện lại sẽ tiếp tục đếm. Cú pháp sử dụng timer như sau:    Ví dụ theo yêu cầu đề: | 0,5  0,5  0,5  0,5  1,0 |
| 4 | - Hoạt động định thời chế độ 1 của bộ TIMER1:    osc  **/ 12**  Trong chế độ 1, bộ Timer dùng cả 2 thanh ghi TH1 và TL1 để chứa giá trị đếm vì vậy chế độ này còn được gọi là chế độ định thời 16 bit. Bit MSB sẽ là bit D7 của TH1 còn bit LSB là D0 của TL1.  Hình trên mô tả hoạt động của các Timer ở chế độ 1: Nguồn xung clock được đưa tới Timer từ một trong cách phụ thuộc vào bit C/ trong thanh ghi TMOD:   * Nếu C/ = 0, xung clock sẽ được lấy từ bộ chia tần trong chip, tần số của xung ở đây là 1/12 tần số của bộ dao động thạch anh (Fosc).   Nguồn xung clock nói trên sẽ được điều khiển để đưa tới các Timer bằng các bit: TR1, GATE và mức logic trên các chân INT1:   * Nếu TR1=0, các Timer sẽ bị cấm mà không cần quan tâm tới GATE và mức logic trên các chân INT1 (thể hiện bằng “cổng AND”). * Nếu TR1=1, các Timer sẽ hoạt động với một trong 2 điều kiện sau xảy ra (thể hiện bằng cổng ‘OR”): Thứ nhất: bit GATE=1; thứ hai: trên chân INT1 có mức logic 1.   Với chế độ 1, giá trị lớn nhất mà các Timer chứa được là 65535(tương ứng FFFF (H)), khi đếm quá giá trị này sẽ xảy ra tràn, khi cờ tràn TF1 sẽ được đặt bằng 1. Sau khi xảy ra tràn, nếu muốn Timer tiếp tục đếm, chương trình phải có câu lệnh nạp lại giá trị khởi tạo sau khi đã dừng Timer bằng cách xoá bit TR1.  ***-*** Viết chương trình ví dụ tạo xung 1KHz trên chân P1.5:  #include <REGX51.H>  #include<stdio.h>  sbit xung=P1^5;  void main(void)  {  TMOD=0x10;  while(1)  {  TH1=-500/256;  TL1=-500%256;  TR1=1;  while(!TF1);  TF1=0;  TR1=0;  xung=~xung;  }  } | 0,5  0,5  0,5  0,5  1 |

**Mã đề thi: CĐT – LT 04**

|  |  |  |
| --- | --- | --- |
| **Câu** | **Đáp án** | **Điểm** |
| 1 | Bảng trạng thái  00  01  10  11  Hàm kích  J1 = K1 = “ 1”  J2  CP  K2  Q2  \_  Q2  Cp  J1  CP  K1  Q1  \_  Q1  1  J2 = K2 = Q1  Sơ đồ kết nối | 0,5  0,5  1 |
| 2 | Các kiểu Biến  Trong [lập trình](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_m%C3%A1y_t%C3%ADnh" \o "Lập trình máy tính) , một **biến** (*variable*) hay **vô hướng** (*scalar*) là một [vị trí lưu trữ](https://vi.wikipedia.org/w/index.php?title=%C4%90%E1%BA%A1i_ch%E1%BB%89_b%E1%BB%99_nh%E1%BB%9B&action=edit&redlink=1" \o "Đại chỉ bộ nhớ (trang chưa được viết)) gắn liền với một [tên tượng trưng](https://vi.wikipedia.org/wiki/Bi%E1%BB%83u_t%C6%B0%E1%BB%A3ng" \o "Biểu tượng) (*[định danh](https://vi.wikipedia.org/wiki/%C4%90%E1%BB%8Bnh_danh_(l%E1%BA%ADp_tr%C3%ACnh_m%C3%A1y_t%C3%ADnh)" \o "Định danh (lập trình máy tính))*) liên quan, chứa một số lượng thông tin được biết đến hay chưa được biết đến mà gọi là *[giá trị](https://vi.wikipedia.org/wiki/Gi%C3%A1_tr%E1%BB%8B_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)" \o "Giá trị (khoa học máy tính))*. Tên biến là cách thường dùng để [tham chiếu](https://vi.wikipedia.org/wiki/Tham_chi%E1%BA%BFu_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)" \o "Tham chiếu (khoa học máy tính)) đến giá trị được lưu trữ; việc tách rời tên và nội dung cho phép tên được dùng độc lập với thông tin chính xác mà nó đại diện. Định danh trong [mã nguồn](https://vi.wikipedia.org/wiki/M%C3%A3_ngu%E1%BB%93n" \o "Mã nguồn) máy tính có thể [bị ràng buộc](https://vi.wikipedia.org/wiki/R%C3%A0ng_bu%E1%BB%99c_t%C3%AAn" \o "Ràng buộc tên) tới một [giá trị](https://vi.wikipedia.org/wiki/Gi%C3%A1_tr%E1%BB%8B_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)" \o "Giá trị (khoa học máy tính)) trong suốt [thời gian chạy](https://vi.wikipedia.org/w/index.php?title=Th%E1%BB%9Di_gian_ch%E1%BA%A1y_(v%C3%B2ng_%C4%91%E1%BB%9Di_ch%C6%B0%C6%A1ng_tr%C3%ACnh)&action=edit&redlink=1" \o "Thời gian chạy (vòng đời chương trình) (trang chưa được viết)), và giá trị của biến có thể thay đổi trong quá trình [thực thi chương trình](https://vi.wikipedia.org/wiki/Th%E1%BB%B1c_thi_(%C4%91i%E1%BB%87n_to%C3%A1n)" \o "Thực thi (điện toán))  Các dạng biến thường sử dụng:  Bool: giá trị TRUE/ FALSE  - Int: giá trị số nguyên dương 32 bit  - Float: giá trị số thực –  String: giá trị chuỗi  - Pose: biến vị trí  - List: biến mảng  Vào Structure chọn Assignment sau đó chọn tab Command để khai báo biến. Có thể khai báo biến hằng số hoặc biến thay đổi bằng các hàm, giá trị trong mảng,... | 0.5  1.0  0.5 |
| 3 | **Counter đếm lên ký hiệu : CTU**    Trong đó:  CU là chân nhận xung đếm lên.  R là chân reset. Chức năng reset giá trị đếm và Cbit của counter.  PV là nơi đặt giá trị đếm của counter.  **Nguyên lý hoạt động của CTU:**  - CTU là bộ đếm sườn lên của xung tín hiệu đầu vào, mỗi lần nhận xung cạnh lên tại CTU thì giá trị counter tăng lên 1. Giá trị đếm này được lưu vào vào thanh ghi 2 byte của bộ đếm gọi là thanh ghi C-word.  - Khi giá trị đếm của counter bằng giá trị đặt thì Cbit = 1.  - Bộ đếm tiến CTU có chân thực hiện chức năng reset, khi đầu vào chân reset có giá trị logic bằng 1, lúc đó bộ đếm bị reset. Giá trị của C-word và C-bits đều bằng 0.  - Bộ đếm ngừng đếm khi giá trị C-word đạt giá trị 32767.  **Ví dụ về CTU:**  Tương tự là couter đếm xuống CTD ký hiệu:    Trong đó  CD là chân nhận xung đếm xuống.  LD là chân Load. Khi có xung cạnh lên tại đây, giá trị counter bằng giá trị đặt.  PV là nơi đặt giá trị đếm của counter.  **Nguyên lý hoạt động của CTD:**  - CTD là bộ đếm sườn lên của xung tín hiệu đầu vào, mỗi lần nhận xung cạnh lên tại CTU thì giá trị counter giảm xuống 1. Giá trị đếm này được lưu vào vào thanh ghi 2 byte của bộ đếm gọi là thanh ghi C-word.  - Bộ đếm tiến CTD có chân LD, khi đầu vào chân LD có giá trị logic bằng 1, lúc đó bộ đếm có giá trị bằng giá trị đặt (PV) và C-bits bằng 0.  - Khi giá trị đếm của counter bằng 0 giá trị đặt thì Cbit = 1.  - Bộ đếm ngừng đếm khi giá trị C-word giảm xuống còn 0.  Ví dụ về CTD:    Ví dụ theo yêu cầu của đề: | 0,5  1,0  0,5  1,0 |
| 4 | - Hoạt động counter của bộ TIMER0 trong vi điều khiển chế độ 16 bit:    osc  **/ 12**  Trong chế độ 1, bộ Timer dùng cả 2 thanh ghi TH0 và TL0 để chứa giá trị đếm vì vậy chế độ này còn được gọi là chế độ định thời 16 bit. Bit MSB sẽ là bit D7 của TH0 còn bit LSB là D0 của TL0.  Hình trên mô tả hoạt động của các Timer ở chế độ 1: Nguồn xung clock được đưa tới Timer từ một trong cách phụ thuộc vào bit C-/T0 trong thanh ghi TMOD:   * C-/T0 = 1, xung clock sẽ được lấy từ bộ tạo xung bên ngoài qua chân T0.   Nguồn xung clock nói trên sẽ được điều khiển để đưa tới các Timer bằng các bit: TR0, GATE và mức logic trên các chân INT0:   * Nếu TR0=0, các Timer sẽ bị cấm mà không cần quan tâm tới GATE và mức logic trên các chân INT0 (thể hiện bằng “cổng AND”). * Nếu TR0=1, các Timer sẽ hoạt động với một trong 2 điều kiện sau xảy ra (thể hiện bằng cổng ‘OR”): Thứ nhất: bit GATE=1; thứ hai: trên chân INT0 có mức logic 1.   Với chế độ 1, giá trị lớn nhất mà các Timer chứa được là 65535 (tương ứng FFFF (H)), khi đếm quá giá trị này sẽ xảy ra tràn, khi cờ tràn TF0 sẽ được đặt bằng 1. Sau khi xảy ra tràn, nếu muốn Timer tiếp tục đếm, chương trình phải có câu lệnh nạp lại giá trị khởi tạo sau khi đã dừng Timer bằng cách xoá bit TR0.  - Hoạt động counter của bộ TIMER0 trong vi điều khiển chế độ 8 bit:    Trong chế độ 2, bộ Timer dùng TL0 để chứa giá trị đếm và TH0 để chứa giá trị nạp lại vì vậy chế độ này được gọi là chế độ tự nạp lại 8 bit. Sau khi đếm quá 255 sẽ xảy ra tràn, khi đó TF0 được đặt bằng 1 đồng thời giá trị của Timer tự động được nạp lại bằng nội dung của TH0.  Hình trên mô tả hoạt động của các Timer ở chế độ 2: Nguồn xung clock được đưa tới Timer từ một trong cách phụ thuộc vào bit C-/T0 trong thanh ghi TMOD:   * C-/T0 = 1, xung clock sẽ được lấy từ bộ tạo xung bên ngoài qua chân T0.   Nguồn xung clock nói trên sẽ được điều khiển để đưa tới các Timer bằng các bit: TR0, GATE và mức logic trên các chân INT0:   * Nếu TR0=0, các Timer sẽ bị cấm mà không cần quan tâm tới GATE và mức logic trên các chân INT0 (thể hiện bằng “cổng AND”). * Nếu TR0=1, các Timer sẽ hoạt động với một trong 2 điều kiện sau xảy ra (thể hiện bằng cổng ‘OR”): Thứ nhất: bit GATE=1; thứ hai: trên chân INT0 có mức logic 1.   ***-*** Viết chương trình đếm xung chân T0 hiển thị giá trị Port 1 và Port 2:  #include <reg52.h>  void main (void)  {  TMOD =0x05;  TR0 = 1;  while (1)  {  P1=TL0;  P2=TH0;  }  } | 0,25  0,25  0,25  0,25  0,25  0,25  0,25  0,25  1 |

**Mã đề thi: CĐT – LT 05**

|  |  |  |
| --- | --- | --- |
| **Câu** | **Đáp án** | **Điểm** |
| 1 | Đồ hình trạng thái:  00  01  10  11  Hàm kích:  J1 = K1 = “ 1”  J2  CP  K2  Q2  \_  Q2  Cp  J1  CP  K1  Q1  \_  Q1  1  J2 = K2 = Q1  Sơ đồ logic: | 0,5  0,5  1 |
| 2 | Lệnh MOVE  Lệnh di chuyển RB.  Có 3 dạng lệnh MOVE.  Dạng 1: MoveL: di chuyển robot theo đường thẳng và dừng tại mỗi điểm đến một khoảng thời gian    Dạng 2: MoveJ: di chuyển robot theo đường cong    Dạng 3: MoveP: di chuyển robot theo cung tròn tại điểm đến    Muốn khai báo lệnh wait chúng ta phải vào Structure/Basics/MOVE. Sau đó tùy chỉnh lệnh với tab Command. | 0.5  1.0  0.5 |
| 3 | Cách lập trình cho S7 – 200 nói riêng và cho các PLC hãng Siemens nói chung dựa trên 3 phương pháp cơ bản: Phương pháp hình thang (Ladder Logic viết tắt là **LAD**) và Phương pháp liệt kê lệnh (Statement List viết tắt là **STL**). Ngoài ra, còn có Phương pháp lập trình theo sơ đồ khối(Funtion Block Diagramm FBD) nhưng chỉ có trong Version 3.0 của phần mềm STEP 7.  Nếu chương trình được viết theo kiểu LAD, thiết bị lập trình sẽ tự tạo ra một chương trình theo kiểu STL tương ứng. Ngược lại không phải mọi chương trình được viết theo kiểu STL cũng có thể chuyển sang được dạng LAD.  Bộ lệnh của phương pháp STL được trình bày đều có một chức năng tương ứng với một tiếp điểm, các cuộn dây và các hộp dùng trong LAD. Những lệnh này phải đọc và phối hợp được trạng thái của các tiếp điểm để đưa ra một quyết định về giá trị trạng thái đầu ra hoặc một giá trị logic cho phép, hoặc không cho phép thực hiện chức năng của một (hay nhiều) hộp.  *\* Phương pháp lập trình LAD*:  LAD là một ngôn ngữ lập trình bằng đồ hoạ. Những thành phần cơ bản dùng trong LAD tương ứng với các thành phần của bảng điều khiển dùng rơle. Trong chương trình LAD các phần tử cơ bản dùng để biểu diễn lệnh logic như sau:  - *Tiếp điểm*:  Là biểu tượng (symbol) mô tả các tiếp điểm của rơle. Các tiếp điểm đó có thể là thường đóng  hay thường.  - *Cuộn dây (coil):*  Là biểu tượng  mô tả rơle được mắc theo chiều dòng điện cung cấp cho rơle.  - *Hộp (box):*  Là biểu tượng mô tả các hàm khác nhau, nó làm việc khi có dòng điện chạy đến hộp. Những dạng hàm thường được biểu diễn bằng hộp là bộ thời gian (Timer), bộ đếm (Counter) và các hàm toán học. Cuộn dây và các hộp phải mắc đúng chiều dòng điện.  - *Mạng LAD:*  Là đường nối các phần tử thành các mạch hoàn thiện, đi từ đường nguồn bên trái đến đường nguồn bên phải. Đường nguồn bên trái là dây nóng, đường nguồn bên phải là dây trung hoà (neutral) hay là đường trở về nguồn cung cấp.  *\* Phương pháp lập trình STL*  Phương pháp liệt kê lệnh (STL) là phương pháp thể hiện chương trình dưới dạng tập hợp các câu lệnh. Mỗi câu lệnh trong chương trình, kể cả những câu lệnh hình thức biểu diễn một chức năng của PLC.  *Định nghĩa về ngăn xếp logic (logic stack):*   |  |  | | --- | --- | | S0  S1  S2  S3  S4  S5  S6  S7  S8 | Stack 0 – bit đầu tiên hay bit trên cùng của ngăn xếp.  Stack 1 – bit thứ hai của ngăn xếp.  Stack 2 – bit thứ ba của ngăn xếp.  Stack 3 – bit thứ tư của ngăn xếp.  Stack 4 – bit thứ năm của ngăn xếp.  Stack 5 – bit thứ sáu của ngăn xếp.  Stack 6 – bit thứ bảy của ngăn xếp.  Stack 7 – bit thứ tám của ngăn xếp.  Stack 8 – bit thứ chín của ngăn xếp. |   **Bảng 2-1 : Bảng định nghĩa về Ngăn xếp lôgic**    Để tạo ra được một chương trình dạng STL, người lập trình cần phải hiểu rõ phương thức sử dụng 9 bit ngăn xếp logic của S7 – 200. Ngăn xếp logic là một khối gồm 9 bit chồng lên nhau. Tất cả các thuật toán liên quan đến ngăn xếp đều chỉ làm việc với bit đầu tiên hoặc với bit đầu và bit thứ hai của ngăn xếp. Giá trị logic mới đều có thể được gởi (hoặc được nối thêm) vào ngăn xếp. Khi phối hợp hai bit đầu tiên của ngăn xếp, thì ngăn xếp sẽ được kéo lên một bit. Ngăn xếp và tên của từng bit trong ngăn xếp được biểu diễn trong hình trên.  *\* Phương pháp lập trình FBD*  Phương pháp sơ đồ khối sử dụng các “Hộp” cho từng chức năng. Ký tự trong hộp cho biết chức năng(thí dụ ký tự & là phép toán logic AND) Ngôn ngữ lập trình này có ưu điểm là 1 người “ không chuyên lập trình “ như 1 kỹ thuật viên công nghệ cũng có thể sử dụng dạng thảo này.  Ví dụ:   |  |  |  | | --- | --- | --- | | **LAD** | **STL** | **FBD** | |  |  |  | | 0,75đ  0,75đ  0,75đ  0,25đ  0,5đ |
| 4 | CODE:  ORG 0000h  CHUC EQU R0  DONVI EQU R1  DEM EQU R2    LL: MOV DPTR,#MA  MOV CHUC,#05  MOV DONVI,#05  TT: MOV A,DONVI  MOVC A,@A+DPTR  MOV P1,A  MOV A,CHUC  MOVC A,@A+DPTR  MOV P2,A  CALL DL  CJNE DONVI,#05,M  CJNE CHUC,#02,M1  INC DEM  CJNE DEM,#02,LL  JMP HT  M: CJNE DONVI,#00,M1  MOV DONVI,#09  DEC CHUC  JMP TT  M1: DEC DONVI  JMP TT  HT: MOV P2,#08EH ; chu f  MOV P1,#0F9H ; chu i  CALL DL  JMP HT  DL: MOV R3,#0FFH  LAP1: MOV R4,#0FFH  LAP: DJNZ R4,LAP  DJNZ R3, LAP1  RET  MA: DB 0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H  END | 0,5  0,5  0,5  0,5  0,5  0,5 |

**Mã đề thi: CĐT – LT 06**

|  |  |  |
| --- | --- | --- |
| **Câu** | **Đáp án** | **Điểm** |
| 1 | Đầu vào dữ liệu d0, d1, d2, d3 và đầu vào địa chỉ s1, s0. hai đầu vào địa chỉ sẽ tạo ra 4 tổ hợp khả dĩ, mỗi đầu vào dữ liệu bị chi phối bởi 1 tổ hợp khác nhau của các mức ở đầu vào địa chỉ.   * Sơ đồ khối      * Bảng trạng thái:  |  |  |  |  | | --- | --- | --- | --- | | S2 | S1 | S0 | D | | 0 | 0 | 0 | F0 | | 0 | 0 | 1 | F1 | | 0 | 1 | 0 | F2 | | 0 | 1 | 1 | F3 | | 1 | 0 | 0 | F4 | | 1 | 0 | 1 | F5 | | 1 | 1 | 0 | F6 | | 1 | 1 | 1 | F7 |  * Phương trình logic:      * Sơ đồ logic:   F0  F1  F2  F3  F4  F5  F6  F7  S2  S1  S0  D | 0,5đ  0,75đ  0,75đ |
| 3 | a/ Timer mở trễ TON *( On-Delay Timer ):*   |  |  |  |  | | --- | --- | --- | --- | | Lệnh | Độ phân giải | Giá trị cực đại | Địa chỉ | | TON | 1 ms | 32,767 s | T0 ÷ T96 | | 10 ms | 327,67 s | T33 ÷ T36, T97 ÷ T100 | | 100 ms | 3276,7 s | T37 ÷ T63, T101 ÷ T127 |  |  |  |  | | --- | --- | --- | | **FBD** | **LAD** | **STL** | |  |  |  |   IN: BOOL: Đầu vào cho phép Timer.  PT: Int: Giá trị đặt trước cho timer ( VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, Constant, \*VD, \*LD, \*AC )  T37: Địa chỉ Timer.  Khi ngõ vào I0.0 =1 Timer T37 được kích , Nếu sau 100 x 100ms =10s, I0.0 vẫn giữ trạng thái thì Bit T37 sẽ lên 1 ( Khi đó Q0.0 lên 1 ).  Nếu I0.0 =1 không đủ thời gian 10S thì bit T37 sẽ không lên mức 1.  I0.0  T-word  T-bít  10s  5s  5s  5s  5s  0  0  0  1  t  t  t  b. Ví dụ minh họa:  Lập trình logic cho PLC điều khiển dây chuyền sản xuất gồm 3 động cơ hoạt động theo yêu cầu sau:  - Nhấn nút khởi động cho động cơ Đ1 chạy, sau 5s cho phép vận hành Đ2.  - Nhấn nút khởi động cho động cơ Đ2 chạy đồng thời lúc này động cơ Đ1 ngừng, sau 10s thì cho phép vận hành động cơ Đ3.  - Nhấn nút khởi động cho động cơ Đ3 chạy đồng thời động cơ Đ2 ngừng.  - Nhấn nút dừng thì bất kỳ động cơ nào chạy cũng phải ngừng.  **Sơ đồ kết nối PLC :** | 0,5  0,5  0,5  1,5 |
| 2 | Lệnh If  Lệnh kiểm tra điều kiện. Khi gặp lệnh này, chương trình sẽ kiểm tra các điều kiện, nếu thỏa mãn thì sẽ thực hiện các lệnh bên trong nó.  Có 3 dạng if.   * Dạng 1 : If ( điều kiện so sánh )   { lệnh 1;  Lệnh 2;  ......  Lệnh n;  }   * Dạng 2 :   If ( điều kiện so sánh )  { lệnh 1;  Lệnh 2;  ......  Lệnh n;  }  elseif ( điều kiện so sánh )  { lệnh 1;  Lệnh 2;  ......  Lệnh n;  }   * Dạng 3 :   If ( điều kiện so sánh )  { lệnh 1;  Lệnh 2;  ......  Lệnh n;  }  else  { lệnh 1;  Lệnh 2;  ......  Lệnh n;  }  Muốn khai báo lệnh if chúng ta phải vào Structure/Avanced/If. Sau đó tùy chỉnh lệnh với tab Command | 0.5  1.0  0.5 |
| 4 | ; Reset Vector  org 0000h  jmp Start  ;============================================  ; CODE SEGMENT  ;============================================  org 0100h  Start:  TD: CALL CT1  CALL CT1  CALL CT2  CALL CT2  CALL CT2  CALL CT3  CALL CT3  CALL CT3  JMP TD  CT1: MOV DPTR,#MA\_LED1 ; 8 led tat dan tu p0.0 den p0.7  TT1: CLR A  MOVC A,@A+DPTR  MOV P0,A  LCALL DELAY  LCALL DELAY  INC DPTR  CJNE A,#0FFH,TT1  RET  CT2: MOV DPTR,#MA\_LED2 ; 8 led tat dan tu p0.7 den p0.0  TT2: CLR A  MOVC A,@A+DPTR  MOV P0,A  LCALL DELAY  LCALL DELAY  INC DPTR  CJNE A,#0FFH,TT2  RET  CT3: MOV DPTR,#MA\_LED3 ; 8LED SANG DAN TU P0.7 -> P0.0  TT3: CLR A  MOVC A,@A+DPTR  MOV P0,A  LCALL DELAY  LCALL DELAY  INC DPTR  CJNE A,#00H,TT3  RET  DELAY: MOV 7EH,#255 ; CHUONG TRINH DELAY  DEL: MOV 7FH,#255  DJNZ 7FH,$  DJNZ 7EH,DEL  RET  MA\_LED1: DB 000H, 01H, 03H, 07H,0FH, 01FH, 03FH, 07FH, 0FFH  MA\_LED2: DB 00H, 080H, 0C0H, 0E0H, 0F0H, 0F8H, 0FCH, 0FEH, 0FFH  MA\_LED3: DB 0FFH, 07FH, 03FH, 01FH, 0FH, 07H, 03H, 01H, 00H    END | 1,0đ  0,5đ  0,25  0,25  0,25  0,25  0,5 |

**Mã đề thi: CĐT – LT 07**

|  |  |  |
| --- | --- | --- |
| **Câu** | **Đáp án** | **Điểm** |
| 1 | Bộ ghi nối tiếp có thể dịch phải, dịch trái và cho ra song song hoặc ra nối tiếp. Trên hình dưới giới thiệu sơ đồ bộ ghi nối tiếp dịch phải có các lối ra song song và ra nối tiếp.    **Mach ghi dÞch ph¶i vµo nãi tiÕp ra song song**  Đây là sơ đồ chỉ có lối vào nối tiếp, còn lối cả ra song song và ra nối tiếp.  Khi cho một xung kim âm tác động vào lối vào xoá, các lối ra Q của cả 4 trigơ trong bộ ghi đều ở trạng thái 0.  Muốn ghi ta phải đưa các bit thông tin nối tiếp về thời gian truyền lần lượt vào lối vào nối tiếp theo sự điều khiển đồng bộ của các xung nhịp. Cứ sau mỗi xung nhịp trạng thái của trigơ lại được xác lập theo thông tin lối vào D của nó. Trong sơ đồ hình 3.16 lối ra của trigơ trước lại được nối với vào lối vào D của trigơ sau nên sau mỗi lần có xung nhịp tác động trigơ sau lại nhận giá trị của trigơ đứng trước nó.  Giả sử ta có 4 bit số liệu D1D2D3D4 được truyền liên tiếp tới lối vào của bộ ghi trong đó bit D4 đến trước nhất. Quá trình ghi thông tin diễn ra như sau:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Xung nhịp | Q1 | Q2 | Q3 | Q4 | | 0  1  2  3  4 | 0  **D4**  **D3**  **D2**  **D1** | 0  0  **D4**  **D3**  **D2** | 0  0  0  **D4**  **D3** | 0  0  0  0  **D4** |   Sau 4 xung nhịp thì thông tin được nạp xong, muốn đưa dữ liệu ra ở các lối ra song song ta đặt mức 1 ở lối ‘Điều khiển ra”, lối ra của các cửa AND ở lối ra song song sẽ được xác lập theo trạng thái Q1, Q2, Q3, Q4 của các trigơ trong bộ ghi. Trong cách điều khiển dữ liệu ra song song này thông tin trong bộ ghi vẫn được duy trì.  Để điều khiển dữ liệu ra nối tiếp, ta phải tác động một nhóm 4 xung nhịp ở lối vào CLK (điều khiển ghi). Sau 4 xung nhịp tác động 4 bit dữ liệu lần lượt được đưa ra khỏi bộ ghi.  Như vậy, quá trình điều khiển ghi nối tiếp 4 bit mới cũng là quá trình đưa 4 bit dữ liệu cũ ra khỏi bộ ghi qua lối ra nối tiếp. | 0,75đ  0,5đ  0,75đ |
| 2 | Lệnh wait  Chương trình sẽ dừng lại khi gặp lệnh này sẽ dừng một khoảng thời gian hay đợi một tín hiệu, một hàm,… nếu thỏa điều kiện thì sẽ chương trình sẽ chạy qua lệnh tiếp theo.  Có 3 loại lênh wait.   * Wait .... second. Lệnh chờ thời gian * Wait for Digital input. Lệnh chờ tín hiệu vào dạng số * Wait for analog input. Lệnh chờ tín hiệu vào dạng tương tự.   Muốn khai báo lệnh wait chúng ta phải vào Structure/Basics/wait. Sau đó tùy chỉnh lệnh với tab Command. | 0.5  1.0  0.5 |
| 3 | a, Ký hiệu của lệnh CTU:    b, Nguyên lý hoạt động của CTU:  - CTU là bộ đếm sườn lên của xung tín hiệu đầu vào tức là đếm số lần thay đổi trạng thái logic từ 0 lên 1 của tín hiệu. Số sườn xung đếm được ghi vào thanh ghi 2 byte của bộ đếm gọi là thanh ghi C-word.  - Giá trị của C – word gọi là giá trị tức thời của bộ đếm luôn được so sanh với giá trị đặt của bộ đếm được ký hiệu là PV. Khi giá trị tức thời lớn hơn hoặc bằng giá trị đặt thì bộ đếm báo ra ngoài bằng cách đặt giá trị 1 vào một bits đặc biệt của nó, được gọi là C – bits. Trong trường hợp giá trị tức thời nhỏ hơn giá trị đặt thì C – bits bằng 0.  - Bộ đếm tiến CTU có chân thực hiện chức năng reset, khi đầu vào chân reset có giá trị logic bằng 1, lúc đó bộ đếm bị reset. Giá trị của C-word và C-bits đều bằng 0.  - Bộ đếm ngừng đếm khi giá trị C-word đạt giá trị 32767.  c, Ví dụ: | 0,5  0,5  0,5  0,5  1 |
| 4 | ; Reset Vector  org 0000h  CHUC EQU R0  DONVI EQU R1  DEM EQU R2  jmp Start  ;==================================================  ; CODE SEGMENT  ;==================================================  org 0100h  Start:    LL: MOV DPTR,#MA  MOV CHUC,#00  MOV DONVI,#05  TT: MOV A,DONVI  MOVC A,@A+DPTR  MOV P1,A  MOV A,CHUC  MOVC A,@A+DPTR  MOV P3,A  CALL DL  CJNE DONVI,#05,M  CJNE CHUC,#02,M1  INC DEM  CJNE DEM,#03,LL  JMP HT  M: CJNE DONVI,#09,M1  MOV DONVI,#00  INC CHUC  JMP TT  M1: INC DONVI  JMP TT  HT: MOV P3,#8EH ; chu F HANG CHUC  MOV P1,#08EH ; chu F HANG DV  CALL DL  JMP HT  DL: MOV R3,#0FFH  LAP1:MOV R4,#0FFH  LAP:DJNZ R4,LAP  DJNZ R3, LAP1  RET  MA: DB 0C0H, 0F9H, 0A4H, 0B0H, 99H, 92H, 82H, 0F8H, 80H, 90H  END | 1,0  0,5  0,5  0,5  0,5 |

**Mã đề thi: CĐT – LT 08**

|  |  |  |
| --- | --- | --- |
| **Câu** | **Đáp án** | **Điểm** |
| 1 | Xét mạch mã hóa nhị phân từ 8 sang 3 (8 ngõ vào, 3 ngõ ra).    Trong đó:  X0, X1, ….,X7 là các ngõ vào tín hiệu  A, B, C là các ngõ ra  Mạch mã hóa nhị phân thực hiện biến đổi tín hiệu ngõ vào thành một từ mã nhị phân tương ứng ở ngõ ra, cụ thể như sau   |  |  |  |  | | --- | --- | --- | --- | | 0 → 000 | 2 → 010 | 4 → 100 | 6 → 110 | | 1 → 001 | 3 → 011 | 5 → 101 | 7 → 111 |   Chọn mức tác động (tích cực) ở ngõ vào là mức logic 1, ta có bảng trạng thái mô tả hoạt động của mạch.    *Giải thích bảng trạng thái:*Khi một ngõ vào ở trạng thái tích cực (mức logic 1) và các ngõ vào còn lại không được tích cực (mức logic 0) thì ngõ ra xuất hiện từ mã tương ứng. Cụ thể là: khi ngõ vào x0=1 và các ngõ vào còn lại bằng không thì từ mã ở ngõ ra là 000, khi ngõ vào x1=1 và các ngõ vào còn lại bằng không thì từ mã ở ngõ ra là 001, vv…  Phương trình logic tối giản:    Sơ đồ logic | 0,75đ  0,5đ  0,75đ |
| 2 | Lệnh set  Đặt một tín hiệu output lên mức Low hoặc High, tín hiệu output bằng hàm số,…  Lệnh này có chức năng là on hoặc off 1 ngõ ra của RB.  Có 2 dạng set.  Dạng 1: Set digital output: thiết lập trạng thái ngõ ra dạng số.  Dạng 2: Set analog output: thiết lập trạng thái ngõ ra dạng tương tự theo áp (0 -10 V) hoặc dòng (0-20mA).  Muốn khai báo lệnh wait chúng ta phải vào Structure/Basics/set. Sau đó tùy chỉnh lệnh với tab Command. | 0.5  1.0  0.5 |
| 3 | \* Nguyên lý làm việc:  Các phép so sánh có thể sử dụng là so sánh ==, <>, >, >=, <, <= và chỉ có thể áp dụng cho Byte, số nguyên I, số nguyên kép DI và số thực R.  Dữ liệu tại ngõ vào IN1 được so sánh với dữ liệu tại ngõ vào IN2  Trong soạn thảo LAD thì tiếp điểm sẽ ON khi thoả mãn điều kiện so sánh.  Trong soạn thảo STL các lệnh Load, AND hoặc OR sẽ = 1 khi phép so sánh là True.  + So sánh Byte  IN1=IN2, IN1 > IN2, IN1 >=IN2, IN1<IN2, IN1<=IN2, IN1<>IN2    Giới hạn vùng toán hạng và dạng dữ liệu hợp lệ   |  |  |  | | --- | --- | --- | | **Inputs/Outputs** | **Operands** | **Data Types** | | IN | IB, QB, MB, SMB, VB, SB, LB, AC, Constant, \*VD, \*LD, \*AC | BYTE | | OUT | I, Q, M, SM, T, C, V, S, L, Power Flow | BOOL |   + So sánh số nguyên Integer  IN1=IN2, IN1 > IN2, IN1 >=IN2, IN1<IN2, IN1<=IN2, IN1<>IN2  So sánh số nguyên cần chú ý đến dấu ( 16#7FFF > 16#8000)  Phạm ví so sánh từ – 32768 đến + 32767.    Giới hạn vùng toán hạng và dạng dữ liệu hợp lệ   |  |  |  | | --- | --- | --- | | **Inputs/Outputs** | **Operands** | **Data Types** | | IN | IW, QW, MW, SW, SMW, T, C, VW, LW, AIW, AC, Constant, \*VD, \*LD,\*AC | INT | | OUT | I, Q, M, SM, T, C, V, S, L, Power Flow | BOOL |   + So sánh số nguyên kép Doubleword  IN1=IN2, IN1 > IN2, IN1 >=IN2, IN1<IN2, IN1<=IN2, IN1<>IN2  So sánh số nguyên cần chú ý đến dấu ( 16#7FFFFFF > 16#8000000)  Phạm vi so sánh từ – 2.147.483.647 đến + 2.147.483.647.    + Giới hạn vùng toán hạng và dạng dữ liệu hợp lệ   |  |  |  | | --- | --- | --- | | **Inputs/Outputs** | **Operands** | **Data Types** | | IN | ID, QD, MD, SD, SMD, VD, LD, HC, AC, Constant, \*VD, \*LD, \*AC | DINT | | OUT | I, Q, M, SM, T, C, V, S, L, Power Flow | BOOL |   + So sánh số thực  Dùng để so sánh 2 số thực: IN1 với IN2. Các phép so sánh có thể sử dụng là: IN1=IN2, IN1 > IN2, IN1 >=IN2, IN1<IN2, IN1<=IN2, IN1<>IN2    *+ G*iới hạn vùng toán hạng và dạng dữ liệu hợp lệ   |  |  |  | | --- | --- | --- | | Inputs/Outputs | Operands | Data Types | | IN | ID, QD, MD, SD, SMD, VD, LD, AC, Constant, \*VD, \*LD, \*AC | REAL | | OUT | I, Q, M, SM, T, C, V, S, L, Power Flow | BOOL |   \* Ví dụ so sánh kiểu số nguyên **(**Integer) nhỏ hơn hoặc bằng |  |
| 4 | ; Reset Vector  org 0000h  jmp Start  org 0100h  Start:  CALL CT1  CALL CT1  CALL CT1  CALL CT2  CALL CT2  TD: CALL CT3  JMP TD  CT1: MOV DPTR,#MA\_LED1 ; DIEM TOI CHAY TU P0.0 -> P0.7  TT1: CLR A  MOVC A,@A+DPTR  MOV P0,A  LCALL DELAY  LCALL DELAY  INC DPTR  CJNE A,#0FFH,TT1  RET  CT2: MOV DPTR,#MA\_LED2 ; sang xen ke  TT2: CLR A  MOVC A,@A+DPTR  MOV P0,A  LCALL DELAY  LCALL DELAY  INC DPTR  CJNE A,#0AAH,TT2  RET  CT3: MOV DPTR,#MA\_LED3 ; 8LED SANG DAN TU P0.0 -> P0.7  TT3: CLR A  MOVC A,@A+DPTR  MOV P0,A  LCALL DELAY  LCALL DELAY  INC DPTR  CJNE A,#00H,TT3  RET  DELAY: MOV 7EH,#255 ; CHUONG TRINH DELAY  DEL: MOV 7FH,#255  DJNZ 7FH,$  DJNZ 7EH,DEL  RET  MA\_LED1: DB  000H,01H,02H,04H,08H,010H,020H,040H,080H  MA\_LED2: DB 055H,0AAH  MA\_LED3: DB 0FFH,0FEH,0FCH,0F8H,0F0H,0E0H,0C0H,080H,00H  END | 1,0  0,5  0,25  0,25  0,25  0,25  0,5 |

**Mã đề thi: CĐT – LT 09**

|  |  |  |
| --- | --- | --- |
| **Câu** | **Đáp án** | **Điểm** |
| 1 | **Bảng trạng thái**  00  01  10  11  Hàm kích  J1 = K1 = “ 1”  J2  CP  K2  Q2  \_  Q2  Cp  J1  CP  K1  Q1  \_  Q1  1  J2 = K2 = Q1  Sơ đồ mạch logic | 0,5  0,5  1 |
| 2 | Các kiểu Biến  Trong [lập trình](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_m%C3%A1y_t%C3%ADnh" \o "Lập trình máy tính) , một **biến** (*variable*) hay **vô hướng** (*scalar*) là một [vị trí lưu trữ](https://vi.wikipedia.org/w/index.php?title=%C4%90%E1%BA%A1i_ch%E1%BB%89_b%E1%BB%99_nh%E1%BB%9B&action=edit&redlink=1" \o "Đại chỉ bộ nhớ (trang chưa được viết)) gắn liền với một [tên tượng trưng](https://vi.wikipedia.org/wiki/Bi%E1%BB%83u_t%C6%B0%E1%BB%A3ng" \o "Biểu tượng) (*[định danh](https://vi.wikipedia.org/wiki/%C4%90%E1%BB%8Bnh_danh_(l%E1%BA%ADp_tr%C3%ACnh_m%C3%A1y_t%C3%ADnh)" \o "Định danh (lập trình máy tính))*) liên quan, chứa một số lượng thông tin được biết đến hay chưa được biết đến mà gọi là *[giá trị](https://vi.wikipedia.org/wiki/Gi%C3%A1_tr%E1%BB%8B_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)" \o "Giá trị (khoa học máy tính))*. Tên biến là cách thường dùng để [tham chiếu](https://vi.wikipedia.org/wiki/Tham_chi%E1%BA%BFu_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)" \o "Tham chiếu (khoa học máy tính)) đến giá trị được lưu trữ; việc tách rời tên và nội dung cho phép tên được dùng độc lập với thông tin chính xác mà nó đại diện. Định danh trong [mã nguồn](https://vi.wikipedia.org/wiki/M%C3%A3_ngu%E1%BB%93n" \o "Mã nguồn) máy tính có thể [bị ràng buộc](https://vi.wikipedia.org/wiki/R%C3%A0ng_bu%E1%BB%99c_t%C3%AAn" \o "Ràng buộc tên) tới một [giá trị](https://vi.wikipedia.org/wiki/Gi%C3%A1_tr%E1%BB%8B_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)" \o "Giá trị (khoa học máy tính)) trong suốt [thời gian chạy](https://vi.wikipedia.org/w/index.php?title=Th%E1%BB%9Di_gian_ch%E1%BA%A1y_(v%C3%B2ng_%C4%91%E1%BB%9Di_ch%C6%B0%C6%A1ng_tr%C3%ACnh)&action=edit&redlink=1" \o "Thời gian chạy (vòng đời chương trình) (trang chưa được viết)), và giá trị của biến có thể thay đổi trong quá trình [thực thi chương trình](https://vi.wikipedia.org/wiki/Th%E1%BB%B1c_thi_(%C4%91i%E1%BB%87n_to%C3%A1n)" \o "Thực thi (điện toán))  Các dạng biến thường sử dụng:  Bool: giá trị TRUE/ FALSE  - Int: giá trị số nguyên dương 32 bit  - Float: giá trị số thực –  String: giá trị chuỗi  - Pose: biến vị trí  - List: biến mảng  Vào Structure chọn Assignment sau đó chọn tab Command để khai báo biến. Có thể khai báo biến hằng số hoặc biến thay đổi bằng các hàm, giá trị trong mảng,... | 0.5  1.0  0.5 |
| 3 | **Counter đếm lên ký hiệu : CTU**    Trong đó:  CU là chân nhận xung đếm lên.  R là chân reset. Chức năng reset giá trị đếm và Cbit của counter.  PV là nơi đặt giá trị đếm của counter.  **Nguyên lý hoạt động của CTU:**  - CTU là bộ đếm sườn lên của xung tín hiệu đầu vào, mỗi lần nhận xung cạnh lên tại CTU thì giá trị counter tăng lên 1. Giá trị đếm này được lưu vào vào thanh ghi 2 byte của bộ đếm gọi là thanh ghi C-word.  - Khi giá trị đếm của counter bằng giá trị đặt thì Cbit =1.  - Bộ đếm tiến CTU có chân thực hiện chức năng reset, khi đầu vào chân reset có giá trị logic bằng 1, lúc đó bộ đếm bị reset. Giá trị của C-word và C-bits đều bằng 0.  - Bộ đếm ngừng đếm khi giá trị C-word đạt giá trị 32767.  **Ví dụ về CTU:**  Tương tự là couter đếm xuống CTD ký hiệu:    Trong đó  CD là chân nhận xung đếm xuống.  LD là chân Load. Khi có xung cạnh lên tại đây, giá trị counter bằng giá trị đặt.  PV là nơi đặt giá trị đếm của counter.  **Nguyên lý hoạt động của CTD:**  - CTD là bộ đếm sườn lên của xung tín hiệu đầu vào, mỗi lần nhận xung cạnh lên tại CTU thì giá trị counter giảm xuống 1. Giá trị đếm này được lưu vào vào thanh ghi 2 byte của bộ đếm gọi là thanh ghi C-word.  - Bộ đếm tiến CTD có chân LD, khi đầu vào chân LD có giá trị logic bằng 1, lúc đó bộ đếm có giá trị bằng giá trị đặt (PV) và C-bits bằng 0.  - Khi giá trị đếm của counter bằng 0 giá trị đặt thì Cbit =1.  - Bộ đếm ngừng đếm khi giá trị C-word giảm xuống còn 0.  Ví dụ về CTD:    Ví dụ theo yêu cầu của đề: |  |
| 4 | CODE:  org 0000h  CHUC EQU R0  DONVI EQU R1  DEM EQU R2  jmp Start  ;====================================================  ; CODE SEGMENT  ;====================================================  org 0100h  Start:    LL: MOV DPTR,#MA  MOV CHUC,#09  MOV DONVI,#09  TT: MOV A,DONVI  MOVC A,@A+DPTR  MOV P1,A  MOV A,CHUC  MOVC A,@A+DPTR  MOV P2,A  CALL DL  CJNE DONVI,#00,M  MOV DONVI,09  CJNE CHUC,#00,M1  INC DEM  CJNE DEM,#02,LL  JMP HT  M: DEC DONVI  JMP TT  M1: DEC CHUC  JMP TT    HT: MOV P2,#08EH ; chu f  MOV P1,#08EH ; chu i  CALL DL  JMP HT  DL: MOV R3,#0FFH  LAP1:MOV R4,#0FFH  LAP:DJNZ R4,LAP  DJNZ R3, LAP1  RET  MA: DB 0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H  END | 0,5  0,5  0,5  0,5  0,5  0,5 |

**Mã đề thi: CĐT – LT 10**

|  |  |  |
| --- | --- | --- |
| **Câu** | **Đáp án** | **Điểm** |
| 1 | + Clk = Clk1 = Clk2  JB=KB= TB = 1  JA=KA= TA = A  + An+1 = T ⊕ An  + Bảng trạng thái   |  |  |  |  | | --- | --- | --- | --- | | tn | | tn+1 | | | A | B | A | B | | 1 | 1 | 1 | 0 | | 1 | 0 | 0 | 1 | | 0 | 1 | 0 | 0 | | 0 | 0 | 1 | 1 | | 0,5  0,5  0,5  0,5 |
| 2 | Lệnh MOVE  Lệnh di chuyển RB.  Có 3 dạng lệnh MOVE.  Dạng 1: MoveL: di chuyển robot theo đường thẳng và dừng tại mỗi điểm đến một khoảng thời gian    Dạng 2: MoveJ: di chuyển robot theo đường cong    Dạng 3: MoveP: di chuyển robot theo cung tròn tại điểm đến    Muốn khai báo lệnh wait chúng ta phải vào Structure/Basics/MOVE. Sau đó tùy chỉnh lệnh với tab Command. | 0.5  1.0  0.5 |
| 3 | Trong PLC s7-200 có 2 loại timer không có nhớ là TON và TOF ( 16 bít )  Ký hiệu:    Độ phân giải của timer như sau:    Trong đó:  IN là nguồn nuôi timer.  PT là giá trị đặt của timer. 16 bít.  Timer TON: Còn gọi timer thường hở đóng chậm. Sẽ đếm khi đầu vào IN bằng 1. Giá trị Tbit = 1 khi giá trị đếm của timer lớn hơn hoặc bằng giá trị đặt. Có thể reset timer bằng lệnh reset hay làm cho giá trị ngõ vào IN bằng 0.  Timer TOF: Còn gọi là timer thường đóng mở chậm. Khi IN có điện giá trị Tbit bằng 1. Khi IN mất điện timer bắt đầu đếm thời gian. Khi thời gian Timer bằng thời gian đặt tại PT thì Tbit bằng 0.  Ví dụ:    Ví dụ theo yêu cầu: | 0,5  0,5  0,5  0,5  1,0 |
| 4 | ; Reset Vector  org 0000h  CHUC EQU R0  DONVI EQU R1  DEM EQU R2  jmp Start  ;====================================================  ; CODE SEGMENT  ;====================================================  org 0100h  Start:    LL: MOV DPTR,#MA  MOV CHUC,#00  MOV DONVI,#09  TT: MOV A,DONVI  MOVC A,@A+DPTR  MOV P2,A  MOV A,CHUC  MOVC A,@A+DPTR  MOV P3,A  CALL DL  CJNE DONVI,#08,M  CJNE CHUC,#08,M1  INC DEM  CJNE DEM,#02,LL  JMP HT  M: CJNE DONVI,#09,M1  MOV DONVI,#00  INC CHUC  JMP TT  M1: INC DONVI  JMP TT    HT: MOV P3,#08EH ; chu f  MOV P2,#08EH ; chu U  CALL DL  JMP HT  DL: MOV R3,#0FFH  LAP1: MOV R4,#0FFH  LAP: DJNZ R4,LAP  DJNZ R3, LAP1  RET  MA: DB 0C0H, 0F9H, 0A4H, 0B0H, 99H, 92H, 82H, 0F8H, 80H, 90H  END | 0,5  0,5  0,5  0,5  0,5  0,5 |

**Mã đề thi: CĐT – LT 11**

|  |  |  |
| --- | --- | --- |
| **Câu** | **Đáp án** | **Điểm** |
| 1 | - Vẽ đúng sơ đồ mạch điện:  C1  C2  Vo2  Rc2  Rb1  Q2  1  2  3  Vcc  Q1  1  2  Rc1  Rb2  3  Vo1  - Trình bày đầy đủ nguyên lý hoạt động:   * + Khi 0 ≤ t < t1: Q1 tắt, Q2 dẫn bảo hoà   Lúc này, tụ C2 sẽ được nạp điện từ nguồn Vcc qua Rc1 để đạt đến giá trị Vcc, đồng thời tụ C1 sẽ xả và nạp từ nguồn Vcc qua Rb1 xuống Q2 đang dẫn bảo hoà.  Tụ C1 càng nạp thì điện áp trên tụ càng tăng, làm cho điện thế tại cực nền Q1 tăng lên cho đến khi Vb1 ≥ Vγ thì lúc này Q1 bắt đầu dẫn  ⇒ sụt áp trên Rc1 tăng nên vc/Q1 giảm dẫn đến tiếp giáp JE/Q2 phân cực yếu hơn  ⇒ BJT Q2 chuyển sang chế độ dẫn khuếch đại, làm cho vc/Q2 tăng và thông qua tụ C1 sẽ làm cho BJT Q1 nhanh chóng đi vào dẫn bảo hoà và Q2 nhanh chóng tắt.  Mạch chuyển sang trạng thái không bền thứ hai ứng với Q1 dẫn bảo hoà và Q2 tắt.   * + Khi t1 ≤ t < t2: Q1 dẫn bảo hoà, Q2 tắt   Lúc này, tụ C1 và C2 vẫn tiếp tục luân phiên xả và nạp điện nên tương tự như trên, Q1 sẽ chuyển dần sang dẫn khuếch đại và tắt, còn Q2 sẽ chuyển trạng thái từ tắt sang dẫn khuếch đại và nhanh chóng dẫn bảo hoà.  Quá trình chuyển đổi trạng thái trong mạch cứ tự động tiếp diễn nên mạch luôn luôn tạo độ dài xung ra. | 1  1 |
| 2 | 1. Lệnh wait   Chương trình sẽ dừng lại khi gặp lệnh này sẽ dừng một khoảng thời gian hay đợi một tín hiệu, một hàm,… nếu thỏa điều kiện thì sẽ chương trình sẽ chạy qua lệnh tiếp theo.  Có 3 loại lênh wait.   * Wait .... second. Lệnh chờ thời gian * Wait for Digital input. Lệnh chờ tín hiệu vào dạng số * Wait for analog input. Lệnh chờ tín hiệu vào dạng tương tự.   Muốn khai báo lệnh wait chúng ta phải vào Structure/Basics/wait. Sau đó tùy chỉnh lệnh với tab Command. | 0.5  1.0  0.5 |
| 3 | **Counter đếm lên ký hiệu : CTU**    Trong đó:  CU là chân nhận xung đếm lên.  R là chân reset. Chức năng reset giá trị đếm và Cbit của counter.  PV là nơi đặt giá trị đếm của counter.  **Nguyên lý hoạt động của CTU:**  - CTU là bộ đếm sườn lên của xung tín hiệu đầu vào, mỗi lần nhận xung cạnh lên tại CTU thì giá trị counter tăng lên 1. Giá trị đếm này được lưu vào vào thanh ghi 2 byte của bộ đếm gọi là thanh ghi C-word.  - Khi giá trị đếm của counter bằng giá trị đặt thì Cbit = 1 .  - Bộ đếm tiến CTU có chân thực hiện chức năng reset, khi đầu vào chân reset có giá trị logic bằng 1, lúc đó bộ đếm bị reset. Giá trị của C-word và C-bits đều bằng 0.  - Bộ đếm ngừng đếm khi giá trị C-word đạt giá trị 32767.  **Ví dụ về CTU:**  Tương tự là couter đếm xuống CTD ký hiệu:    Trong đó  CD là chân nhận xung đếm xuống.  LD là chân Load. Khi có xung cạnh lên tại đây, giá trị counter bằng giá trị đặt.  PV là nơi đặt giá trị đếm của counter.  **Nguyên lý hoạt động của CTD:**  - CTD là bộ đếm sườn lên của xung tín hiệu đầu vào, mỗi lần nhận xung cạnh lên tại CTU thì giá trị counter giảm xuống 1. Giá trị đếm này được lưu vào vào thanh ghi 2 byte của bộ đếm gọi là thanh ghi C-word.  - Bộ đếm tiến CTD có chân LD, khi đầu vào chân LD có giá trị logic bằng 1, lúc đó bộ đếm có giá trị bằng giá trị đặt (PV) và C-bits bằng 0.  - Khi giá trị đếm của counter bằng 0 giá trị đặt thì Cbit = 1.  - Bộ đếm ngừng đếm khi giá trị C-word giảm xuống còn 0.  Ví dụ về CTD:    Ví dụ theo yêu cầu của đề: | 0,5  1,0  0,5  1,0 |
| 4 | - Hoạt động định thời chế độ 2 của bộ TIMER0:    osc  **/ 12**  Trong chế độ 2, bộ Timer dùng TL0 để chứa giá trị đếm và TH0 để chứa giá trị nạp lại vì vậy chế độ này được gọi là chế độ tự nạp lại 8 bit. Sau khi đếm quá 255 sẽ xảy ra tràn, khi đó TF0 được đặt bằng 1 đồng thời giá trị của Timer tự động được nạp lại bằng nội dung của TH0.  Hình trên mô tả hoạt động của các Timer ở chế độ 2: Nguồn xung clock được đưa tới Timer từ một trong cách phụ thuộc vào bit C-/T0 trong thanh ghi TMOD:  • C/ = 0, xung clock sẽ được lấy từ bộ chia tần trong chip, tần số của xung ở đây là 1/12 tần số của bộ dao động thạch anh (Fosc).  Nguồn xung clock nói trên sẽ được điều khiển để đưa tới các Timer bằng các bit: TR0, GATE và mức logic trên các chân INT0:  • Nếu TR0=0, các Timer sẽ bị cấm mà không cần quan tâm tới GATE và mức logic trên các chân INT0 (thể hiện bằng “cổng AND”).  • Nếu TR0=1, các Timer sẽ hoạt động với một trong 2 điều kiện sau xảy ra (thể hiện bằng cổng ‘OR”): Thứ nhất: bit GATE=1; thứ hai: trên chân INT0 có mức logic 1.  ***-*** Viết chương trình ví dụ tạo xung 10KHz trên chân P1.5:  #include <REGX51.H>  #include<stdio.h>  sbit xung=P1^5;  void main(void)  {  TMOD=0x02;  while(1)  {  TH0= TL0= -50;  TR0=1;  while(!TF0);  TF0=0;  xung=~xung;  }  } | 0,5  0,5  0,5  0,5  1 |

**Mã đề thi: CĐT – LT 12**

|  |  |  |
| --- | --- | --- |
| **Câu** | **Đáp án**  **B**i  **C**i-1  Ai  Ci  Si | **Điểm** |
| 1 | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Ngõ vào | | | Ngõ ra | | | Ai | Bi | Ci-1 | Ci | Si | | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 1 | 0 | 1 | | 0 | 1 | 0 | 0 | 1 | | 0 | 1 | 1 | 1 | 0 | | 1 | 0 | 0 | 0 | 1 | | 1 | 0 | 1 | 1 | 0 | | 1 | 1 | 0 | 1 | 0 | | 1 | 1 | 1 | 1 | 1 |     **Hàm logic:**  Ci = Ai Bi + Bi Ci-1 + Ai Ci-1  Si = Ai ⊕ Bi ⊕ Ci-1 | 0,5  0,5  0,5  0,5 |
| 2 | Các kiểu Biến  Trong [lập trình](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_m%C3%A1y_t%C3%ADnh" \o "Lập trình máy tính) , một **biến** (*variable*) hay **vô hướng** (*scalar*) là một [vị trí lưu trữ](https://vi.wikipedia.org/w/index.php?title=%C4%90%E1%BA%A1i_ch%E1%BB%89_b%E1%BB%99_nh%E1%BB%9B&action=edit&redlink=1" \o "Đại chỉ bộ nhớ (trang chưa được viết)) gắn liền với một [tên tượng trưng](https://vi.wikipedia.org/wiki/Bi%E1%BB%83u_t%C6%B0%E1%BB%A3ng" \o "Biểu tượng) (*[định danh](https://vi.wikipedia.org/wiki/%C4%90%E1%BB%8Bnh_danh_(l%E1%BA%ADp_tr%C3%ACnh_m%C3%A1y_t%C3%ADnh)" \o "Định danh (lập trình máy tính))*) liên quan, chứa một số lượng thông tin được biết đến hay chưa được biết đến mà gọi là *[giá trị](https://vi.wikipedia.org/wiki/Gi%C3%A1_tr%E1%BB%8B_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)" \o "Giá trị (khoa học máy tính))*. Tên biến là cách thường dùng để [tham chiếu](https://vi.wikipedia.org/wiki/Tham_chi%E1%BA%BFu_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)" \o "Tham chiếu (khoa học máy tính)) đến giá trị được lưu trữ; việc tách rời tên và nội dung cho phép tên được dùng độc lập với thông tin chính xác mà nó đại diện. Định danh trong [mã nguồn](https://vi.wikipedia.org/wiki/M%C3%A3_ngu%E1%BB%93n" \o "Mã nguồn) máy tính có thể [bị ràng buộc](https://vi.wikipedia.org/wiki/R%C3%A0ng_bu%E1%BB%99c_t%C3%AAn" \o "Ràng buộc tên) tới một [giá trị](https://vi.wikipedia.org/wiki/Gi%C3%A1_tr%E1%BB%8B_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh)" \o "Giá trị (khoa học máy tính)) trong suốt [thời gian chạy](https://vi.wikipedia.org/w/index.php?title=Th%E1%BB%9Di_gian_ch%E1%BA%A1y_(v%C3%B2ng_%C4%91%E1%BB%9Di_ch%C6%B0%C6%A1ng_tr%C3%ACnh)&action=edit&redlink=1" \o "Thời gian chạy (vòng đời chương trình) (trang chưa được viết)), và giá trị của biến có thể thay đổi trong quá trình [thực thi chương trình](https://vi.wikipedia.org/wiki/Th%E1%BB%B1c_thi_(%C4%91i%E1%BB%87n_to%C3%A1n)" \o "Thực thi (điện toán))  Các dạng biến thường sử dụng:  Bool: giá trị TRUE/ FALSE  - Int: giá trị số nguyên dương 32 bit  - Float: giá trị số thực –  String: giá trị chuỗi  - Pose: biến vị trí  - List: biến mảng  Vào Structure chọn Assignment sau đó chọn tab Command để khai báo biến. Có thể khai báo biến hằng số hoặc biến thay đổi bằng các hàm, giá trị trong mảng,... | 0,5  1,0  0,5 |
| 3 | Trong PLC s7-200 timer có nhớ gọi là timer TONR  Ký hiệu:    Trong đó IN là nguồn nuôi Timer.  PT giá trị đặt timer ( 16 bít )  Độ phân giải Timer như sau:  Cũng giống như TON, TONR sẽ đếm khi IN có giá trị 1, tuy nhiên giá trị đếm timer này không mất đi khi nguồn nuôi. Khi có điện lại sẽ tiếp tục đếm. Cú pháp sử dụng timer như sau:    Ví dụ theo yêu cầu đề: | 0,5  0,5  0,5  0,5  1,0 |
| 4 | - Hoạt động counter của bộ TIMER0 trong vi điều khiển chế độ 16 bit:    osc  **/ 12**  Trong chế độ 1, bộ Timer dùng cả 2 thanh ghi TH0 và TL0 để chứa giá trị đếm vì vậy chế độ này còn được gọi là chế độ định thời 16 bit. Bit MSB sẽ là bit D7 của TH0 còn bit LSB là D0 của TL0.  Hình trên mô tả hoạt động của các Timer ở chế độ 1: Nguồn xung clock được đưa tới Timer từ một trong cách phụ thuộc vào bit C-/T0 trong thanh ghi TMOD:   * C-/T0 = 1, xung clock sẽ được lấy từ bộ tạo xung bên ngoài qua chân T0.   Nguồn xung clock nói trên sẽ được điều khiển để đưa tới các Timer bằng các bit: TR0, GATE và mức logic trên các chân INT0:   * Nếu TR0=0, các Timer sẽ bị cấm mà không cần quan tâm tới GATE và mức logic trên các chân INT0 (thể hiện bằng “cổng AND”). * Nếu TR0=1, các Timer sẽ hoạt động với một trong 2 điều kiện sau xảy ra (thể hiện bằng cổng ‘OR”): Thứ nhất: bit GATE=1; thứ hai: trên chân INT0 có mức logic 1.   Với chế độ 1, giá trị lớn nhất mà các Timer chứa được là 65535 (tương ứng FFFF (H)), khi đếm quá giá trị này sẽ xảy ra tràn, khi cờ tràn TF0 sẽ được đặt bằng 1. Sau khi xảy ra tràn, nếu muốn Timer tiếp tục đếm, chương trình phải có câu lệnh nạp lại giá trị khởi tạo sau khi đã dừng Timer bằng cách xoá bit TR0.  - Hoạt động counter của bộ TIMER0 trong vi điều khiển chế độ 8 bit:    Trong chế độ 2, bộ Timer dùng TL0 để chứa giá trị đếm và TH0 để chứa giá trị nạp lại vì vậy chế độ này được gọi là chế độ tự nạp lại 8 bit. Sau khi đếm quá 255 sẽ xảy ra tràn, khi đó TF0 được đặt bằng 1 đồng thời giá trị của Timer tự động được nạp lại bằng nội dung của TH0.  Hình trên mô tả hoạt động của các Timer ở chế độ 2: Nguồn xung clock được đưa tới Timer từ một trong cách phụ thuộc vào bit C-/T0 trong thanh ghi TMOD:   * C-/T0 = 1, xung clock sẽ được lấy từ bộ tạo xung bên ngoài qua chân T0.   Nguồn xung clock nói trên sẽ được điều khiển để đưa tới các Timer bằng các bit: TR0, GATE và mức logic trên các chân INT0:   * Nếu TR0=0, các Timer sẽ bị cấm mà không cần quan tâm tới GATE và mức logic trên các chân INT0 (thể hiện bằng “cổng AND”). * Nếu TR0=1, các Timer sẽ hoạt động với một trong 2 điều kiện sau xảy ra (thể hiện bằng cổng ‘OR”): Thứ nhất: bit GATE=1; thứ hai: trên chân INT0 có mức logic 1.   ***-*** Viết chương trình đếm xung chân T0 hiển thị giá trị Port 1 và Port 2:  #include <reg52.h>  void main (void)  {  TMOD =0x05;  TR0 = 1;  while (1)  {  P1=TL0;  P2=TH0;  }  } | 0,25  0,25  0,25  0,25  0,25  0,25  0,25  0,25  1 |

**Mã đề thi: CĐT – LT 13**

|  |  |  |
| --- | --- | --- |
| **Câu** | **Đáp án** | **Điểm** |
| 1 | Đầu vào dữ liệu d0, d1, d2, d3 và đầu vào địa chỉ s1, s0. hai đầu vào địa chỉ sẽ tạo ra 4 tổ hợp khả dĩ, mỗi đầu vào dữ liệu bị chi phối bởi 1 tổ hợp khác nhau của các mức ở đầu vào địa chỉ.   * Sơ đồ khối      * Bảng trạng thái:  |  |  |  |  | | --- | --- | --- | --- | | S2 | S1 | S0 | D | | 0 | 0 | 0 | F0 | | 0 | 0 | 1 | F1 | | 0 | 1 | 0 | F2 | | 0 | 1 | 1 | F3 | | 1 | 0 | 0 | F4 | | 1 | 0 | 1 | F5 | | 1 | 1 | 0 | F6 | | 1 | 1 | 1 | F7 |  * Phương trình logic:      * Sơ đồ logic:   F0  F1  F2  F3  F4  F5  F6  F7  S2  S1  S0  D | 0,5  0,75  0,75 |
| 2 | Lệnh MOVE  Lệnh di chuyển RB.  Có 3 dạng lệnh MOVE.  Dạng 1: MoveL: di chuyển robot theo đường thẳng và dừng tại mỗi điểm đến một khoảng thời gian    Dạng 2: MoveJ: di chuyển robot theo đường cong    Dạng 3: MoveP: di chuyển robot theo cung tròn tại điểm đến    Muốn khai báo lệnh wait chúng ta phải vào Structure/Basics/MOVE. Sau đó tùy chỉnh lệnh với tab Command. | 0.5  1.0  0.5 |
| 3 | Cách lập trình cho S7 – 200 nói riêng và cho các PLC hãng Siemens nói chung dựa trên 3 phương pháp cơ bản: Phương pháp hình thang (Ladder Logic viết tắt là **LAD**) và Phương pháp liệt kê lệnh (Statement List viết tắt là **STL**). Ngoài ra, còn có Phương pháp lập trình theo sơ đồ khối(Funtion Block Diagramm FBD) nhưng chỉ có trong Version 3.0 của phần mềm STEP 7.  Nếu chương trình được viết theo kiểu LAD, thiết bị lập trình sẽ tự tạo ra một chương trình theo kiểu STL tương ứng. Ngược lại không phải mọi chương trình được viết theo kiểu STL cũng có thể chuyển sang được dạng LAD.  Bộ lệnh của phương pháp STL được trình bày đều có một chức năng tương ứng với một tiếp điểm, các cuộn dây và các hộp dùng trong LAD. Những lệnh này phải đọc và phối hợp được trạng thái của các tiếp điểm để đưa ra một quyết định về giá trị trạng thái đầu ra hoặc một giá trị logic cho phép, hoặc không cho phép thực hiện chức năng của một (hay nhiều) hộp.  *\* Phương pháp lập trình LAD*:  LAD là một ngôn ngữ lập trình bằng đồ hoạ. Những thành phần cơ bản dùng trong LAD tương ứng với các thành phần của bảng điều khiển dùng rơle. Trong chương trình LAD các phần tử cơ bản dùng để biểu diễn lệnh logic như sau:  - *Tiếp điểm*:  Là biểu tượng (symbol) mô tả các tiếp điểm của rơle. Các tiếp điểm đó có thể là thường đóng  hay thường.  - *Cuộn dây (coil):*  Là biểu tượng  mô tả rơle được mắc theo chiều dòng điện cung cấp cho rơle.  - *Hộp (box):*  Là biểu tượng mô tả các hàm khác nhau, nó làm việc khi có dòng điện chạy đến hộp. Những dạng hàm thường được biểu diễn bằng hộp là bộ thời gian (Timer), bộ đếm (Counter) và các hàm toán học. Cuộn dây và các hộp phải mắc đúng chiều dòng điện.  - *Mạng LAD:*  Là đường nối các phần tử thành các mạch hoàn thiện, đi từ đường nguồn bên trái đến đường nguồn bên phải. Đường nguồn bên trái là dây nóng, đường nguồn bên phải là dây trung hoà (neutral) hay là đường trở về nguồn cung cấp.  *\* Phương pháp lập trình STL*  Phương pháp liệt kê lệnh (STL) là phương pháp thể hiện chương trình dưới dạng tập hợp các câu lệnh. Mỗi câu lệnh trong chương trình, kể cả những câu lệnh hình thức biểu diễn một chức năng của PLC.  *Định nghĩa về ngăn xếp logic (logic stack):*   |  |  | | --- | --- | | S0  S1  S2  S3  S4  S5  S6  S7  S8 | Stack 0 – bit đầu tiên hay bit trên cùng của ngăn xếp.  Stack 1 – bit thứ hai của ngăn xếp.  Stack 2 – bit thứ ba của ngăn xếp.  Stack 3 – bit thứ tư của ngăn xếp.  Stack 4 – bit thứ năm của ngăn xếp.  Stack 5 – bit thứ sáu của ngăn xếp.  Stack 6 – bit thứ bảy của ngăn xếp.  Stack 7 – bit thứ tám của ngăn xếp.  Stack 8 – bit thứ chín của ngăn xếp. |   **Bảng 2-1 : Bảng định nghĩa về Ngăn xếp lôgic**    Để tạo ra được một chương trình dạng STL, người lập trình cần phải hiểu rõ phương thức sử dụng 9 bit ngăn xếp logic của S7 – 200. Ngăn xếp logic là một khối gồm 9 bit chồng lên nhau. Tất cả các thuật toán liên quan đến ngăn xếp đều chỉ làm việc với bit đầu tiên hoặc với bit đầu và bit thứ hai của ngăn xếp. Giá trị logic mới đều có thể được gởi (hoặc được nối thêm) vào ngăn xếp. Khi phối hợp hai bit đầu tiên của ngăn xếp, thì ngăn xếp sẽ được kéo lên một bit. Ngăn xếp và tên của từng bit trong ngăn xếp được biểu diễn trong hình trên.  *\* Phương pháp lập trình FBD*  Phương pháp sơ đồ khối sử dụng các “Hộp” cho từng chức năng. Ký tự trong hộp cho biết chức năng(thí dụ ký tự & là phép toán logic AND) Ngôn ngữ lập trình này có ưu điểm là 1 người “ không chuyên lập trình “ như 1 kỹ thuật viên công nghệ cũng có thể sử dụng dạng thảo này.  Ví dụ:   |  |  |  | | --- | --- | --- | | **LAD** | **STL** | **FBD** | |  |  |  | | 0,75  0,75  0,75  0,25  0,5 |
| 4 | ; Reset Vector  org 0000h  jmp Start  ;============================================  ; CODE SEGMENT  ;============================================  org 0100h  Start:  TD: CALL CT1  CALL CT1  CALL CT2  CALL CT2  CALL CT2  CALL CT3  CALL CT3  CALL CT3  JMP TD  CT1: MOV DPTR,#MA\_LED1 ; 8 led tat dan tu p0.0 den p0.7  TT1: CLR A  MOVC A,@A+DPTR  MOV P0,A  LCALL DELAY  LCALL DELAY  INC DPTR  CJNE A,#0FFH,TT1  RET  CT2: MOV DPTR,#MA\_LED2 ; 8 led tat dan tu p0.7 den p0.0  TT2: CLR A  MOVC A,@A+DPTR  MOV P0,A  LCALL DELAY  LCALL DELAY  INC DPTR  CJNE A,#0FFH,TT2  RET  CT3: MOV DPTR,#MA\_LED3 ; 8LED SANG DAN TU P0.7 -> P0.0  TT3: CLR A  MOVC A,@A+DPTR  MOV P0,A  LCALL DELAY  LCALL DELAY  INC DPTR  CJNE A,#00H,TT3  RET  DELAY: MOV 7EH,#255 ; CHUONG TRINH DELAY  DEL: MOV 7FH,#255  DJNZ 7FH,$  DJNZ 7EH,DEL  RET  MA\_LED1: DB 000H, 01H, 03H, 07H,0FH, 01FH, 03FH, 07FH, 0FFH  MA\_LED2: DB 00H, 080H, 0C0H, 0E0H, 0F0H, 0F8H, 0FCH, 0FEH, 0FFH  MA\_LED3: DB 0FFH, 07FH, 03FH, 01FH, 0FH, 07H, 03H, 01H, 00H    END | 1,0  0,5  0,25  0,25  0,25  0,25  0,5 |

**Mã đề thi: CĐT – LT 14**

|  |  |  |
| --- | --- | --- |
| **Câu** | **Đáp án** | **Điểm** |
| 1 | Bộ ghi nối tiếp có thể dịch phải, dịch trái và cho ra song song hoặc ra nối tiếp. Trên hình dưới giới thiệu sơ đồ bộ ghi nối tiếp dịch phải có các lối ra song song và ra nối tiếp.    **Mach ghi dÞch ph¶i vµo nãi tiÕp ra song song**  Đây là sơ đồ chỉ có lối vào nối tiếp, còn lối cả ra song song và ra nối tiếp.  Khi cho một xung kim âm tác động vào lối vào xoá, các lối ra Q của cả 4 trigơ trong bộ ghi đều ở trạng thái 0.  Muốn ghi ta phải đưa các bit thông tin nối tiếp về thời gian truyền lần lượt vào lối vào nối tiếp theo sự điều khiển đồng bộ của các xung nhịp. Cứ sau mỗi xung nhịp trạng thái của trigơ lại được xác lập theo thông tin lối vào D của nó. Trong sơ đồ hình 3.16 lối ra của trigơ trước lại được nối với vào lối vào D của trigơ sau nên sau mỗi lần có xung nhịp tác động trigơ sau lại nhận giá trị của trigơ đứng trước nó.  Giả sử ta có 4 bit số liệu D1D2D3D4 được truyền liên tiếp tới lối vào của bộ ghi trong đó bit D4 đến trước nhất. Quá trình ghi thông tin diễn ra như sau:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Xung nhịp | Q1 | Q2 | Q3 | Q4 | | 0  1  2  3  4 | 0  **D4**  **D3**  **D2**  **D1** | 0  0  **D4**  **D3**  **D2** | 0  0  0  **D4**  **D3** | 0  0  0  0  **D4** |   Sau 4 xung nhịp thì thông tin được nạp xong, muốn đưa dữ liệu ra ở các lối ra song song ta đặt mức 1 ở lối ‘Điều khiển ra”, lối ra của các cửa AND ở lối ra song song sẽ được xác lập theo trạng thái Q1, Q2, Q3, Q4 của các trigơ trong bộ ghi. Trong cách điều khiển dữ liệu ra song song này thông tin trong bộ ghi vẫn được duy trì.  Để điều khiển dữ liệu ra nối tiếp, ta phải tác động một nhóm 4 xung nhịp ở lối vào CLK (điều khiển ghi). Sau 4 xung nhịp tác động 4 bit dữ liệu lần lượt được đưa ra khỏi bộ ghi.  Như vậy, quá trình điều khiển ghi nối tiếp 4 bit mới cũng là quá trình đưa 4 bit dữ liệu cũ ra khỏi bộ ghi qua lối ra nối tiếp. | 0,75đ  0,5đ  0,75đ |
| 2 | Lệnh wait  Chương trình sẽ dừng lại khi gặp lệnh này sẽ dừng một khoảng thời gian hay đợi một tín hiệu, một hàm,… nếu thỏa điều kiện thì sẽ chương trình sẽ chạy qua lệnh tiếp theo.  Có 3 loại lênh wait.   * Wait .... second. Lệnh chờ thời gian * Wait for Digital input. Lệnh chờ tín hiệu vào dạng số * Wait for analog input. Lệnh chờ tín hiệu vào dạng tương tự.   Muốn khai báo lệnh wait chúng ta phải vào Structure/Basics/wait. Sau đó tùy chỉnh lệnh với tab Command. | 0.5  1.0  0.5 |
| 3 | \* Nguyên lý làm việc:  Các phép so sánh có thể sử dụng là so sánh ==, <>, >, >=, <, <= và chỉ có thể áp dụng cho Byte, số nguyên I, số nguyên kép DI và số thực R.  Dữ liệu tại ngõ vào IN1 được so sánh với dữ liệu tại ngõ vào IN2  Trong soạn thảo LAD thì tiếp điểm sẽ ON khi thoả mãn điều kiện so sánh.  Trong soạn thảo STL các lệnh Load, AND hoặc OR sẽ = 1 khi phép so sánh là True.  + So sánh Byte  IN1=IN2, IN1 > IN2, IN1 >=IN2, IN1<IN2, IN1<=IN2, IN1<>IN2    Giới hạn vùng toán hạng và dạng dữ liệu hợp lệ   |  |  |  | | --- | --- | --- | | **Inputs/Outputs** | **Operands** | **Data Types** | | IN | IB, QB, MB, SMB, VB, SB, LB, AC, Constant, \*VD, \*LD, \*AC | BYTE | | OUT | I, Q, M, SM, T, C, V, S, L, Power Flow | BOOL |   + So sánh số nguyên Integer  IN1=IN2, IN1 > IN2, IN1 >=IN2, IN1<IN2, IN1<=IN2, IN1<>IN2  So sánh số nguyên cần chú ý đến dấu ( 16#7FFF > 16#8000)  Phạm ví so sánh từ – 32768 đến + 32767.    Giới hạn vùng toán hạng và dạng dữ liệu hợp lệ   |  |  |  | | --- | --- | --- | | **Inputs/Outputs** | **Operands** | **Data Types** | | IN | IW, QW, MW, SW, SMW, T, C, VW, LW, AIW, AC, Constant, \*VD, \*LD,\*AC | INT | | OUT | I, Q, M, SM, T, C, V, S, L, Power Flow | BOOL |   + So sánh số nguyên kép Doubleword  IN1=IN2, IN1 > IN2, IN1 >=IN2, IN1<IN2, IN1<=IN2, IN1<>IN2  So sánh số nguyên cần chú ý đến dấu ( 16#7FFFFFF > 16#8000000)  Phạm vi so sánh từ – 2.147.483.647 đến + 2.147.483.647.    + Giới hạn vùng toán hạng và dạng dữ liệu hợp lệ   |  |  |  | | --- | --- | --- | | **Inputs/Outputs** | **Operands** | **Data Types** | | IN | ID, QD, MD, SD, SMD, VD, LD, HC, AC, Constant, \*VD, \*LD, \*AC | DINT | | OUT | I, Q, M, SM, T, C, V, S, L, Power Flow | BOOL |   + So sánh số thực  Dùng để so sánh 2 số thực: IN1 với IN2. Các phép so sánh có thể sử dụng là: IN1=IN2, IN1 > IN2, IN1 >=IN2, IN1<IN2, IN1<=IN2, IN1<>IN2    *+ G*iới hạn vùng toán hạng và dạng dữ liệu hợp lệ   |  |  |  | | --- | --- | --- | | Inputs/Outputs | Operands | Data Types | | IN | ID, QD, MD, SD, SMD, VD, LD, AC, Constant, \*VD, \*LD, \*AC | REAL | | OUT | I, Q, M, SM, T, C, V, S, L, Power Flow | BOOL |   \* Ví dụ so sánh kiểu số nguyên **(**Integer) nhỏ hơn hoặc bằng |  |
| 4 | ; Reset Vector  org 0000h  jmp Start  org 0100h  Start:  CALL CT1  CALL CT1  CALL CT1  CALL CT2  CALL CT2  TD: CALL CT3  JMP TD  CT1: MOV DPTR,#MA\_LED1 ; DIEM TOI CHAY TU P0.0 -> P0.7  TT1: CLR A  MOVC A,@A+DPTR  MOV P0,A  LCALL DELAY  LCALL DELAY  INC DPTR  CJNE A,#0FFH,TT1  RET  CT2: MOV DPTR,#MA\_LED2 ; sang xen ke  TT2: CLR A  MOVC A,@A+DPTR  MOV P0,A  LCALL DELAY  LCALL DELAY  INC DPTR  CJNE A,#0AAH,TT2  RET  CT3: MOV DPTR,#MA\_LED3 ; 8LED SANG DAN TU P0.0 -> P0.7  TT3: CLR A  MOVC A,@A+DPTR  MOV P0,A  LCALL DELAY  LCALL DELAY  INC DPTR  CJNE A,#00H,TT3  RET  DELAY: MOV 7EH,#255 ; CHUONG TRINH DELAY  DEL: MOV 7FH,#255  DJNZ 7FH,$  DJNZ 7EH,DEL  RET  MA\_LED1: DB  000H,01H,02H,04H,08H,010H,020H,040H,080H  MA\_LED2: DB 055H,0AAH  MA\_LED3: DB 0FFH,0FEH,0FCH,0F8H,0F0H,0E0H,0C0H,080H,00H  END | 1,0  0,5  0,25  0,25  0,25  0,25  0,5 |

**Mã đề thi: CĐT – LT 15**

|  |  |  |
| --- | --- | --- |
| **Câu** | **Đáp án** | **Điểm** |
| 1 | **Bảng trạng thái**  00  01  10  11  Hàm kích  J1 = K1 = “ 1”  J2  CP  K2  Q2  \_  Q2  Cp  J1  CP  K1  Q1  \_  Q1  1  J2 = K2 = Q1  Sơ đồ mạch logic | 0,5  0,5  1 |
| 2 | Lệnh MOVE  Lệnh di chuyển RB.  Có 3 dạng lệnh MOVE.  Dạng 1: MoveL: di chuyển robot theo đường thẳng và dừng tại mỗi điểm đến một khoảng thời gian    Dạng 2: MoveJ: di chuyển robot theo đường cong    Dạng 3: MoveP: di chuyển robot theo cung tròn tại điểm đến    Muốn khai báo lệnh wait chúng ta phải vào Structure/Basics/MOVE. Sau đó tùy chỉnh lệnh với tab Command. | 0.5  1.0  0.5 |
| 3 | Trong PLC s7-200 có 2 loại timer không có nhớ là TON và TOF ( 16 bít )  Ký hiệu:    Độ phân giải của timer như sau:    Trong đó:  IN là nguồn nuôi timer.  PT là giá trị đặt của timer. 16 bít.  Timer TON: Còn gọi timer thường hở đóng chậm. Sẽ đếm khi đầu vào IN bằng 1. Giá trị Tbit = 1 khi giá trị đếm của timer lớn hơn hoặc bằng giá trị đặt. Có thể reset timer bằng lệnh reset hay làm cho giá trị ngõ vào IN bằng 0.  Timer TOF: Còn gọi là timer thường đóng mở chậm. Khi IN có điện giá trị Tbit bằng 1. Khi IN mất điện timer bắt đầu đếm thời gian. Khi thời gian Timer bằng thời gian đặt tại PT thì Tbit bằng 0.  Ví dụ:    Ví dụ theo yêu cầu: | 0,5  0,5  0,5  0,5  1,0 |
| 4 | CODE:  org 0000h  CHUC EQU R0  DONVI EQU R1  DEM EQU R2  jmp Start  ;====================================================  ; CODE SEGMENT  ;====================================================  org 0100h  Start:    LL: MOV DPTR,#MA  MOV CHUC,#09  MOV DONVI,#09  TT: MOV A,DONVI  MOVC A,@A+DPTR  MOV P1,A  MOV A,CHUC  MOVC A,@A+DPTR  MOV P2,A  CALL DL  CJNE DONVI,#00,M  MOV DONVI,09  CJNE CHUC,#00,M1  INC DEM  CJNE DEM,#02,LL  JMP HT  M: DEC DONVI  JMP TT  M1: DEC CHUC  JMP TT    HT: MOV P2,#08EH ; chu f  MOV P1,#08EH ; chu i  CALL DL  JMP HT  DL: MOV R3,#0FFH  LAP1:MOV R4,#0FFH  LAP:DJNZ R4,LAP  DJNZ R3, LAP1  RET  MA: DB 0C0H,0F9H,0A4H,0B0H,99H,92H,82H,0F8H,80H,90H  END | 0,5  0,5  0,5  0,5  0,5  0,5 |