|  |
| --- |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  **TRƯỜNG ĐIỆN – ĐIỆN TỬ**  logo_128  ĐỒ ÁN  **TỐT NGHIỆP ĐẠI HỌC**  **Đề tài:**  **THIẾT KẾ HỆ PLC SỬ DỤNG**  **NỀN TẢNG STM32**  Sinh viên thực hiện: LÊ VĂN TRƯỜNG  Mã số sinh viên : 20174306  Lớp KT Điều khiển & TĐH 09 – K62  Giảng viên hướng dẫn: PGS.TS NGUYỄN HỒNG QUANG  Hà Nội, 7-2022 |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  **TRƯỜNG ĐIỆN – ĐIỆN TỬ**  logo_128  ĐỒ ÁN  **TỐT NGHIỆP ĐẠI HỌC**  **Đề tài:**  **THIẾT KẾ HỆ PLC SỬ DỤNG**  **NỀN TẢNG STM32**  Sinh viên thực hiện: LÊ VĂN TRƯỜNG  Mã số sinh viên : 20174306  Lớp KT Điều khiển & TĐH 09 – K62  Giảng viên hướng dẫn: PGS.TS NGUYỄN HỒNG QUANG  Cán bộ phản biện:  Hà Nội, 7-2022 |

**LỜI NÓI ĐẦU**

Tài liệu này được Trường Điện – Điện tử, trường Đại học Bách Khoa Hà Nội soạn thảo và ban hành nhằm mục đích hướng dẫn sinh viên trình bày đồ án tốt nghiệp một cách khoa học và thống nhất. Bản thân tài liệu này được biên soạn phù hợp với các quy định về trình bày một đồ án tốt nghiệp. Vì vậy, sinh viên có thể sử dụng trực tiếp mẫu này như một template khi viết quyển đồ án của mình. Sinh viên cũng được khuyến khích tham khảo hoặc sử dụng mẫu này khi viết báo cáo thực tập, báo cáo kết quả nghiên cứu, và các đồ án môn học khác.

Chi tiết về nội dung của phần Lời nói đầu được trình bày trong Mục 1.3.3.

**LỜI CAM ĐOAN**

Lời cam đoan được viết tại đây. Xem cách viết lời cam đoan tại Mục 1.3.4.

**MỤC LỤC**

[DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT i](#_Toc108859590)

[DANH MỤC HÌNH VẼ ii](#_Toc108859591)

[DANH MỤC BẢNG BIỂU iii](#_Toc108859592)

[TÓM TẮT ĐỒ ÁN iv](#_Toc108859593)

[CHƯƠNG 1. TỔNG QUAN 1](#_Toc108859594)

[1.1 Đặt vấn đề 1](#_Toc108859595)

[1.2 Giải pháp và quy trình thực hiện 1](#_Toc108859596)

[1.2.1 Giải pháp thực hiện 1](#_Toc108859597)

[1.2.2 Quy trình thực hiện 1](#_Toc108859598)

[CHƯƠNG 2. GIỚI THIỆU TẬP LỆNH CƠ BẢN CỦA PLC S7 200 3](#_Toc108859599)

[2.1 Giới thiệu về ngôn ngữ lập trình cho PLC 3](#_Toc108859600)

[2.1.1 Định nghĩa về LAD 3](#_Toc108859601)

[2.1.2 Định nghĩa về STL 3](#_Toc108859602)

[2.1.3 Vòng quét ( thực hiện chương trình) của PLC 4](#_Toc108859603)

[2.2 Các lệnh trong PLC 4](#_Toc108859604)

[2.2.1 Các lệnh logic 4](#_Toc108859605)

[2.2.2 Các lệnh Timer và Counter 6](#_Toc108859606)

[2.2.3 Các lệnh so sánh 10](#_Toc108859607)

[2.2.4 Các câu lệnh Move 12](#_Toc108859608)

[2.2.5 Một số lệnh khác 13](#_Toc108859609)

[CHƯƠNG 3. THIẾT KẾ CHƯƠNG TRÌNH BIÊN DỊCH 16](#_Toc108859610)

[3.1 Phương án thiết kế phần mềm 16](#_Toc108859611)

[3.2 Thuật toán xử lý chung 17](#_Toc108859612)

[3.3 Lập trình LAD và xuất file chứa code STL 17](#_Toc108859613)

[3.3.1 Giới thiệu phần mềm STEP 7 – MicroWIN 17](#_Toc108859614)

[3.3.2 Tạo file chưa code STL 18](#_Toc108859615)

[KẾT LUẬN 24](#_Toc108859616)

[Kết luận chung 24](#_Toc108859617)

[Hướng phát triển 24](#_Toc108859618)

[Kiến nghị và đề xuất 24](#_Toc108859619)

[TÀI LIỆU THAM KHẢO 25](#_Toc108859620)

[PHỤ LỤC 26](#_Toc108859621)

[Phụ lục 1. Mẫu trang bìa chính của đồ án 26](#_Toc108859622)

[Phụ lục 2. Mẫu trang bìa phụ của đồ án 28](#_Toc108859623)

[Phụ lục 3. Mẫu nhận xét đồ án 30](#_Toc108859624)

# DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT

# DANH MỤC HÌNH VẼ

[Hình 2.1 Mô tả ngăn xếp của S7 200 4](#_Toc108946574)

[Hình 2.2 Ví dụ minh họa lệnh LD, NOT, R, EU trong chương trình LAD và STL 6](#_Toc108946575)

[Hình 2.3 Ví dụ minh họa lệnh TON, TONR, TOF EU trong chương trình LAD và STL 8](#_Toc108946576)

[Hình 2.4 Ví dụ minh họa lệnh CTU, CTUD trong chương trình LAD và STL 10](#_Toc108946577)

[Hình 2.5 Ví dụ minh họa các lệnh so sánh trong chương trình LAD và STL 12](#_Toc108946578)

[Hình 2.6 Ví dụ minh họa lệnh MOVB, MOVW, MOVDW trong chương trình LAD và STL 13](file:///C:\Code\DATN\DATN_LVT.docx#_Toc108946579)

[Hình 2.7 Ví dụ minh họa các lệnh OLD, LPS, LDR, LPP trong chương trình LAD và STL 15](#_Toc108946580)

[Hình 3.1 Giao diện cấu hình cho chip trên phần mềm STM32CubeIDE 18](#_Toc108946581)

[Hình 3.2 Vị trí của FreeRTOS trong một firmware stack 19](#_Toc108946582)

[Hình 3.3 Cách hoạt động của hệ điều hành thời gian thực 20](#_Toc108946583)

[Hình 3.4 Hệ điều hành hoạt động ở chế độ Preemptive 21](#_Toc108946584)

[Hình 3.5 Các trạng thái của các Task 22](#_Toc108946585)

[Hình 3.6 Sơ đồ khối các bước thực hiện từ lập trình đến nạp chương trình cho PLC 24](#_Toc108946586)

[Hình 3.7 Giao diện phần mềm STEP7 – MicroWIN 25](#_Toc108946587)

[Hình 3.8 Chương trình LAD trên STEP7 – MicroWIN 26](#_Toc108946588)

[Hình 3.9 Xuất file LAD sang STL 26](#_Toc108946589)

[Hình 3.10 File code STL 27](#_Toc108946590)

[Hình 3.11 Mô hình danh sách liên kết đôi 28](#_Toc108946591)

[Hình 3.12 Sơ đồ khối thuật toán xử lý file, tạo file chứa mã code STL 30](#_Toc108946592)

[Hình 3.13 Sơ đồ khối chuyển file code chương trình vào danh sách liên kết đôi và bảng băm đóng 31](#_Toc108946593)

[Hình 3.14 Sơ đồ khối xử lý chuỗi chuyển đổi phù hợp sang ngôn ngữ vi điều khiển 32](#_Toc108946594)

[Hình 3.15 Sơ đồ khối lưu lệnh và biến theo thứ tựu vào danh sách liên kết đôi 34](#_Toc108946595)

[Hình 3.16 Sơ đồ khối lưu biến chương trình PLC vào bảng băm 35](#_Toc108946596)

# DANH MỤC BẢNG BIỂU

[Bảng 2.1 Các lệnh logic 5](#_Toc108859520)

[Bảng 2.2 Các lệnh Timer 6](#_Toc108859521)

[Bảng 2.3 Timer và độ phân giải 7](#_Toc108859522)

[Bảng 2.4 Các lệnh Counter 8](#_Toc108859523)

[Bảng 2.5 Các lệnh so sánh 10](#_Toc108859524)

[Bảng 2.6 Các lệnh Move 12](#_Toc108859525)

[Bảng 2.7 Các lệnh stack logic 14](#_Toc108859526)

[Bảng 3.1 Các phép logic cơ bản 16](#_Toc108859527)

TÓM TẮT ĐỒ ÁN

# TỔNG QUAN

**Mục đích :**  Nêu lên vấn đề mà đồ án đặt ra và đưa ra phương án giải quyết.

## Đặt vấn đề

Tạo ra một phần mềm có thể chuyển đổi ngôn ngữ ladder logic (LAD), là một trong năm ngôn ngữ lập trình PLC được chỉ định sử dụng theo tiêu chuẩn IEC 61131-31, thành ngôn ngữ C để có thể nạp vào vi điều khiển và hoạt động như một PLC thông thường.

## Giải pháp và quy trình thực hiện

### Giải pháp thực hiện

Hiện nay các hãng PLC nổi tiếng như Siemens, Mitsubishi, Schneider… đều có bộ công cụ phần mềm riêng để có thể lập trình và nạp chương trình của mình xuống PLC.Do giới hạn về thời gian cũng như kinh nghiệm về thiết kế một phần mềm lập trình hoàn hảo như các hãng, trong đồ án này, em sẽ sử dụng sẵn phần mềm của hãng Siemens là STEP 7 – MicroWIN để lập trình chương trình PLC.

Sử dụng chương trình đã được viết bằng ngôn ngữ LAD từ phầm mềm này ta chuyển đổi về dạng ngôn ngữ Statement List (STL). Đây cũng là 1 dạng ngôn ngữ tiêu chuẩn cho PLC được thực hiện bằng cách liệt kê các câu lệnh theo thứ tự.Sau khi đã có file chương trình theo ngôn ngữ STL ta sẽ chuyển đổi tự động về dạng ngôn ngữ C để nạp vảo vi điều khiển bằng một chương trình biên dịch.

Trong phạm vi đồ án này, em chỉ thực hiện việc chuyển đổi code của các tập lệnh vào ra số, các pháp tính cơ bản của một chương trình PLC ( chi tiết các tập lệnh đã được chuyển đổi sẽ nêu chi tiết ở chương 2) và chưa xử lý lệnh đối với tín hiệu tương tự cũng như các câu lệnh phức tạp khác.

### Quy trình thực hiện

* Nghiên cứu và lựa chọn các tập lệnh của PLC s7 200 để lập trình.
* Thiết kế một chương trình để biên dịch ngôn ngữ LAD sang ngôn ngữ C có thể nạp vào vi điều khiển.
* Thiết kế chương trình thu thập, giám sát dữ liệu cho PLC.
* Thiết kế phần cứng đáp ứng các tiêu chuẩn của một PLC trong công nghiệp.
* Chạy thử nghiệm chương trình PLC.

# GIỚI THIỆU TẬP LỆNH CƠ BẢN CỦA PLC S7 200

**Mục đích :** Giới thiệu tổng quan về ngôn ngữ LAD và ngôn ngữ STL,giới thiệu về các tập lệnh được sử dụng và biên dịch trong phạm vi đồ án này.

## Giới thiệu về ngôn ngữ lập trình cho PLC

### Định nghĩa về LAD

LAD là ngôn ngữ lập trình bằng đồ họa. Những thành phần cơ bản dùng trong LAD tương ứng với những thành phần cơ bản dùng trong bảng mạch rơle:

* Tiếp điểm có hai loại: thường hở, thường đóng.
* Cuộn dây (coil).
* Hộp (box): Mô tả các hàm khác nhau, nó làm việc khi có tín hiệu đưa đến hộp. Có các nhóm hộp sau: hộp các bộ định thời, hộp các bộ đếm, hộp di chuyển dữ liệu, hộp các hàm toán học, …
* Mạng LAD: Là mạch nối các phần tử trong thành một mạng hoàn thiện, các phần tử như cuộn dây hoặc các hộp phải được mắc đúng chiều. Nguồn điện có hai đường chính,một đường bên trái thể hiện dây nóng, một đường bên phải là dây trung tính (neutral) nhưng không được thể hiện trên giao diện lập trình. Một mạch làm việc được khi các phần tử được mắc đúng chiều và kín mạch.

### Định nghĩa về STL

Là phương pháp thể hiện chương trình dưới dạng tập hợp các câu lệnh. Để tạo ra một chương trình bằng STL, người lập trình cần phải hiểu rõ phương thức sử dụng 9 bit trong ngăn xếp (stack) logic của S7 200.

Ngăn xếp là một khối 9 bit chồng lên nhau từ S0 đến S8, nhưng tất cả các thuật toán liên quan đến ngăn xếp đều làm việc với bit đầu tiên à bit thứ hai ( S0 và S1 ) của ngăn xếp.

Giá trị logic mới có thể được gởi hoặc nối thêm vào ngăn xếp. Hai bit S0 và S1 phối hợp với nhau thì ngăn xếp sẽ kéo lên một bit.

Ngăn xếp của S7 200 ( logic stack):

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình . Mô tả ngăn xếp của S7 200

### Vòng quét ( thực hiện chương trình) của PLC

PLC thực hiện chương trình theo vòng lặp. Mỗi vòng lặp được gọi là vòng quét

( scan).

Các giai đoạn của vòng quét:

* Đọc các tín hiệu ngõ vào.
* Thực hiện chương trình.
* Thực hiện các quá trình trình truyền thông yêu cầu.
* Ghi các giá trị ngõ ra.

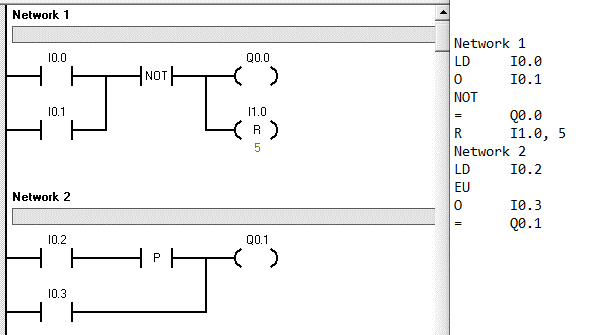
Tuy nhiên, trong đồ án này, em thực hiện một vòng quét cho việc đọc/ ghi cũng như thực hiện chương trình chính riêng so với vòng quét của việc truyền thông dữ liệu ( vòng quét này chậm hơn rất nhiều so với vòng quét chính). Điều này nhằm tăng tốc độ cho chu kỳ quét chương trình giúp cho chương trình thực hiện chính xác hơn.

## Các lệnh trong PLC

### Các lệnh logic

Bảng . Các lệnh logic

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STL** | **LAD** | **Mô tả** | **Toán hạng** | **Kiểu dữ liệu** |
| LD | Ảnh có chứa ăng-ten  Mô tả được tạo tự động | Tiếp điểm thường hở sẽ đóng lại khi bit = 1 | **Bit:**  I , Q, M | Bool |
| O |
| A |
| LDN | Ảnh có chứa ăng-ten  Mô tả được tạo tự động | Tiếp điểm thường đóng sẽ được mở khi bit = 1 | **Bit:** I, Q, M | Bool |
| ON |
| AN |
| NOT |  | Đảo giá trị bit đầu tiên trong ngăn xếp | Không | Không |
| EU | Ảnh có chứa ăng-ten  Mô tả được tạo tự động | Bit đầu tiên trong ngăn xếp có giá trị bằng một ( trong khoảng thời gian đúng bằng một chu kỳ quét) khi phát hiện sườn lên tín hiệu đầu vào | **Bit:** I, Q, M | Bool |
| ED | Ảnh có chứa ăng-ten  Mô tả được tạo tự động | Bit đầu tiên trong ngăn xếp có giá trị bằng 1 ( trong khoảng thời gian một chu kỳ quét) khi phát hiện sườn xuống của tín hiệu đầu vào | **Bit:** I, Q, M | Bool |
| S bit, n |  | Set một mảng gồm n tiếp điểm,tính từ tiếp điểm “bit”  (n<=128) | **Bit:** I, Q, M | Bool |
| R bit, n |  | Reset một mảng gồm n tiếp điểm tính từ tiếp điểm “bit” ( n<= 128) | **Bit:** I, Q, M | Bool |



Hình . Ví dụ minh họa lệnh LD, NOT, R, EU trong chương trình LAD và STL

### Các lệnh Timer và Counter

#### Lệnh Timer

Bảng . Các lệnh Timer

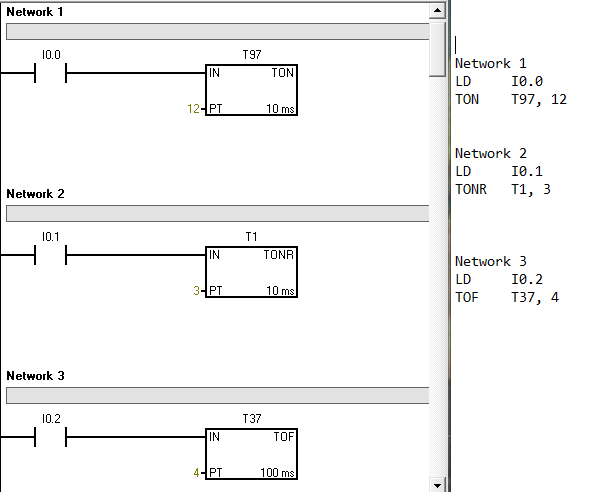
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STL** | **LAD** | **Mô tả** | **Toán hạng** | **Kiểu dữ liệu** |
| TON Txx, PT  TONR Txx, PT |  | Đây là lệnh đếm thời gian hoạt khi tín hiệu EN là ON.Khi giá trị dếm tức thời trong thanh ghi CT >=giá trị đặt trước trong thanh ghi PT thì bit trạng thái Txxx của bộ Timer là ON. Giá trị đếm tức thời trong thanh ghi CT = 0 và bit trạng thái về off khi tín hiệu ở đầu vào là off. Ngược lại với bộ TON, thanh ghi CV và bit trạng thái vẫn giữ nguyên trừ khi có lệnh Reset bộ TONR. Ngoài ra có thể sử dụng lệnh Reset để xoá thanh ghi tức thời cũng như bit trạng thái của bộ TON. Ta có thể sử dụng toán hạng Word (INT) tương ứng với lệnh INT hay toán hạng tương ứng với bit trạng thái. | Txxx  (T0 toT255 ) | Word |
| IN : I , Q , M, T, C, , power flow | Bool |
| PT  ( constant) | Int |
| TOF Txx, PT |  | Khi tín hiệu đầu vào EN = 1 bộ TOF không hoạt động. chỉ hoạt động khi có sườn xuống của tín hiệu đầu vào. Bit trạng thái được bật lên ON khi CV = PT. Reset TOF (cả CV và bit trạng thái) bằng cách cung cấp tín hiệu vào đầu vào EN. | Txxx  (T0 toT255) | Word |
| IN : I , Q , M, T, C, power flow | Bool |
| PT  ( constant) | Int |

Bảng . Timer và độ phân giải

|  |  |  |
| --- | --- | --- |
| **Loại Timer** | **Độ phân giải (ms)** | **Số hiệu Timer** |
| TONR | 1 | T0, T64 |
| 10 | T1 đến T4, T65 đến T68 |
| 100 | T5 đến T31, T69 đến T95 |
| TON,TOF | 1 | T32, T96 |
| 10 | T33 đến T36, T97 đến T100 |
| 100 | T37 đến T63, T101 đến T255 |

Trong PLC S7 200 có tất cả 256 bộ timer khác nhau được nêu ra như trên bảng 2.3

Tuy nhiên, trong một chương trình không thể dùng cùng lúc 2 bộ TON và TOF có cùng một địa chỉ ( ví dụ cả hai cùng dùng chung một địa chỉ T97 thì chương trình sẽ bị lỗi).

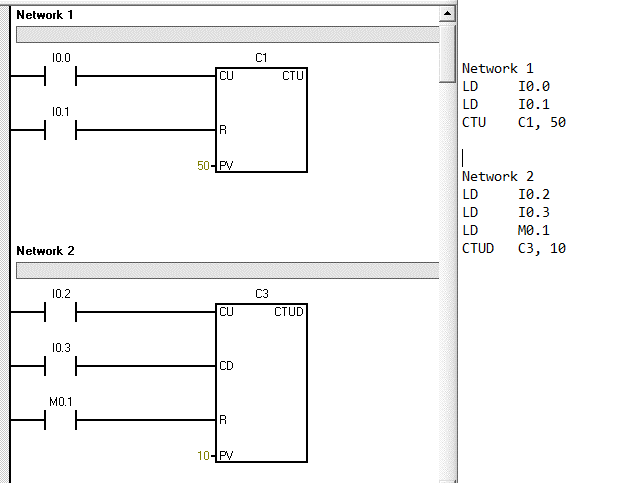


Hình . Ví dụ minh họa lệnh TON, TONR, TOF EU trong chương trình LAD và STL

#### Lệnh Counter

Bảng . Các lệnh Counter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STL** | **LAD** | **Mô tả** | **Toán hạng** | **Kiểu dữ liệu** |
| CTU Cxxx, PV |  | Khai báo bộ đếm tiến theo sườn lên của tín hiệu đầu vào CU.Khi gí trị đếm tức thời C-Word lớn hơn hoặc bằng giá trị đặt trước PV, thì bit trạng thái Cxxx có giá trị bằng 1. Bộ đếm được Reset khi R có giá trị logic bằng 1. Bộ đếm ngừng đếm khi giá trị đếm đạt giá trị cực đại 32767. | Cxxx: Constant  (C0 to C255) | Word |
| CU, R: I , Q, M , T, C, power flow | Bool |
| PV: constant | Int |
| CTD Cxxx, PV |  | Khai báo bộ đếm lùi theo sườn lên của tín hiệu đầu vào CD. Khi giá trị đếm tức thời C-Word lớn hơn hoặc bằng giá trị đặt trước PV, htì bit trạng thái Cxxx có giá trị bằng 1. Bộ đếm được Reset khi R có giá trị logic bằng 1. Bộ đếm ngừng đếm khi giá trị đếm đạt giá trị cực đại 32767 | Cxxx: Constant  (C0 to C255) | Word |
| CD, R: I , Q, M , T, C, power flow | Bool |
| PV: constant | Int |
| CTUD Cxxx,Pv |  | Khai báo bộ đếm tiến/lùi; đếm tiến theo sườn lên của tín hiệu đầu vào CU, đếm lùi theo sườn lên của tín hiệu đầu vào CD. Khi gí trị đếm tức thời C-Word lớn hơn hoặc bằng giá trị đặt trước PV, htì bit trạng thái Cxxx có giá trị bằng 1. Bộ đếm được Reset khi R có giá trị logic bằng 1. Bộ đếm ngừng đếm tiến khi giá trị đếm đạt giá trị cực đại 32767. Bộ đếm ngừng đếm lùi khi giá trị đếm đạt giá trị cực đại -32767. CTUD reset khi đầu vào R có giá trị logic bằng 1 | Cxxx: Constant  (C0 to C255) | Word |
| CU,CD, R: I , Q, M , T, C, power flow | Bool |
| PV: constant | Int |



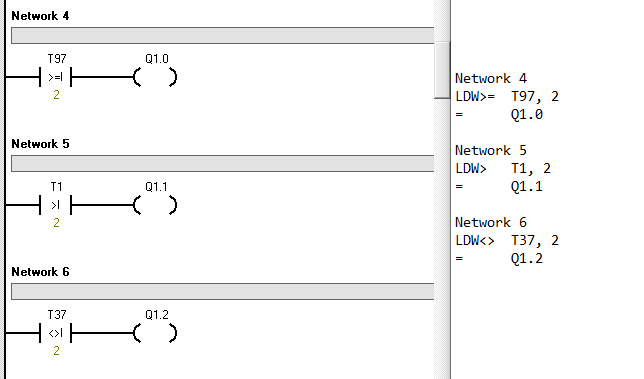
Hình . Ví dụ minh họa lệnh CTU, CTUD trong chương trình LAD và STL

### Các lệnh so sánh

Trong phạm vi đồ án này, em chỉ sử dụng các lệnh so sánh kiểu số nguyên để thao tác với các bộ định thời và bộ đếm.

Bảng . Các lệnh so sánh

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **STL** | **LAD** | | | **Mô tả** | **Toán hạng** | **Kiểu dữ liệu** |
| LDW= | Ảnh có chứa văn bản, ăng-ten  Mô tả được tạo tự động | | | So sánh giá trị 2 World IN1 và IN2.  Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1=IN2 là đúng | T , C,constant | World |
| OW= |
| AW= |
| LDW<> | Ảnh có chứa ăng-ten  Mô tả được tạo tự động | | | So sánh giá trị 2 World IN1 và IN2.  Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1<>IN2 là đúng | T, C, constant | World |
| OW<> |
| AW<> |
| LDW> | Ảnh có chứa văn bản, ăng-ten  Mô tả được tạo tự động | | | So sánh giá trị 2 World IN1 và IN2.  Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1>IN2 là đúng | T, C, constant | World |
| OW> |
| AW> |
| LDW>= | | Ảnh có chứa ăng-ten  Mô tả được tạo tự động | So sánh giá trị 2 World IN1 và IN2.  Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1>=IN2 là đúng | | T, C, constant | World |
| OW>= | |
| AW>= | |
| LDW< | | Ảnh có chứa ăng-ten  Mô tả được tạo tự động | So sánh giá trị 2 World IN1 và IN2.  Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1<IN2 là đúng | | T, C, constant | World |
| OW< | |
| AW< | |
| LDW<= | | Ảnh có chứa ăng-ten  Mô tả được tạo tự động | So sánh giá trị 2 World IN1 và IN2.  Trạng thái tiếp điểm là đóng khi lệnh so sánh IN1<=IN2 là đúng | | T, C, constant | World |
| OW<= | |
| AW<= | |

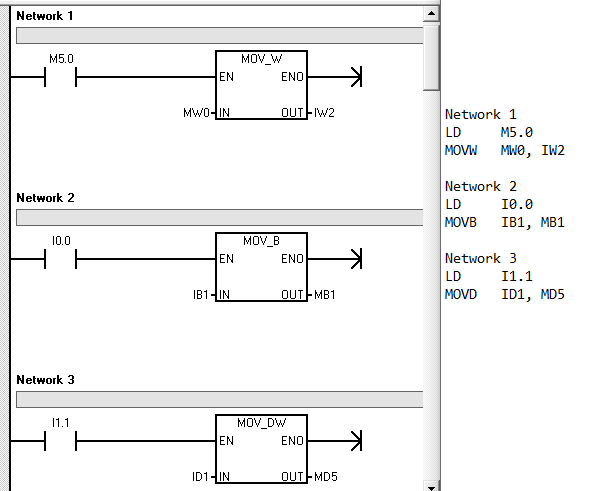


Hình . Ví dụ minh họa các lệnh so sánh trong chương trình LAD và STL

### Các câu lệnh Move

Bảng . Các lệnh Move

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STL** | **LAD** | **Mô tả** | **Toán hạng** | **Kiểu dữ liệu** |
| MOVB IN, OUT |  | Lệnh thực hiện việc chuyển dữ liệu từ byte IN vào byte OUT khi có sườn lên của tín hiệu vào | **IN:**IB,QB,MB  **OUT:**IB,QB,MB | Byte |
| MOVW IN, OUT |  | Lệnh thực hiện việc chuyển dữ liệu từ Word IN vào Word OUT khi có sườn lên của tín hiệu vào | **IN:**IW,QW,MW  **OUT:**IW,QW,MW | Word |
| MOVDW IN, OUT |  | Lệnh thực hiện việc chuyển dữ liệu từ Double Word IN vào Double Word OUT khi có sườn lên của tín hiệu vào | **IN:**IDW,QDW,MDW  **OUT:**IDW,QDW,MDW | Word |



Hình . Ví dụ minh họa lệnh MOVB, MOVW, MOVDW trong chương trình LAD và STL

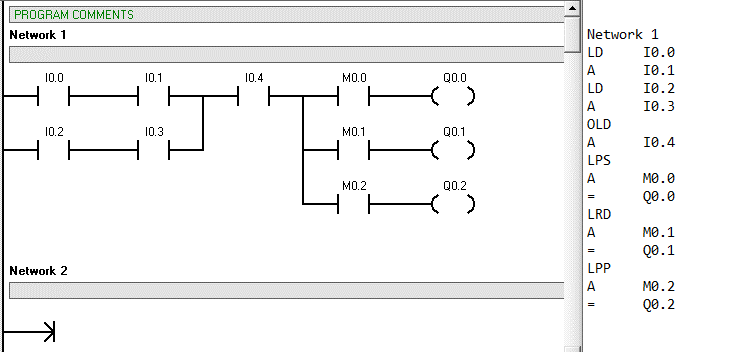
### Một số lệnh khác

Các lệnh tiếp điểm trong đại số Boolean cho phép tạo lập được các mạch logic (không có nhớ). Trong LAD các mạch này biểu diễn thông qua cấu trúc mạch, mắc nối tiếp hay song song các mạch tiếp điểm thường đóng và các tiếp điểm thường mở. STL có thể sử dụng các lệnh A (And) và O (Or) cho các tiếp điểm mắc nối tiếp và song song là thường hở hoặc các lệnh AN (And Not) và ON (Or Not) cho các tiếp điểm mắc nối tiếp và song song là thường đóng. Giá trị của các bit trong ngăn xếp thay đổi tuỳ thuộc vào từng lệnh.

Ngoài những lệnh làm việc trực tiếp với tiếp điểm, S7-200 còn có 5 lệnh đặc biệt biểu diễn các phép tính của đại số Boolean cho các bit trong ngăn xếp, được gọi là các lệnh stack logic. Trong LAD không dùng những lệnh này. STL sử dụng các lệnh này để thực hiện những phép toán của phương trình có nhiều biểu thức con. Sau đây là bảng tóm tắt cú pháp và hướng dẫn cách sử dụng lệnh.

Bảng . Các lệnh stack logic

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STL** | **LAD** | **Mô tả** | **Toán hạng** | **Kiểu dữ liệu** |
| ALD | Không | Lệnh tổ hợp giá trị đầu tiên và giá trị của bit thứ hai trong ngăn xếp bằng phép tính ∧. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp. Giá trị còn lại được kéo lên 1 bit | Không | Không |
| OLD | Không | Lệnh tổ hợp giá trị đầu tiên và giá trị của bit thứ hai trong ngăn xếp bằng phép tính ∨. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp. Gía trị còn lại được kéo lên 1 bit. | Không | Không |
| LPS | Không | Sao chép giá trị của bit đầu tiên vào bit thứ hai trong ngăn xếp. Gía trị còn lại bị đẩy xuống 1 bit. Bit cuối cùng bị đẩy ra ngoài | Không | Không |
| LRD | Không | Lệnh sao chép giá trị của bit thứ hai vào bit đầu tiên của ngăn xếp, các giá trị còn lại của ngăn xếp vẫn giữ nguyên. | Không | Không |
| LPP | Không | Lệnh kéo ngăn xếp lên 1 bit theo nguyên tắc bit sao đè lên bit trước. | Không | Không |



Hình . Ví dụ minh họa các lệnh OLD, LPS, LDR, LPP trong chương trình LAD và STL

# THIẾT KẾ CHƯƠNG TRÌNH BIÊN DỊCH

**Mục tiêu:** Thiết kế một chương trình để chuyển đổi code từ ngôn ngữ ladder sang ngôn ngữ C để nạp vào vi điều khiển.

## Phương án thiết kế phần mềm

Để dịch code từ ngôn ngữ ladder sang C cần chuyển ladder qua STL sau đó xử lý  
chuỗi có dạng ngôn ngữ STL và chuyển sang C.

Mối liên hệ giữa các ngõ vào và ngõ ra được biểu diễn bằng các biểu thức toán học  
mô tả sự phụ thuộc của biến ngõ ra bởi các biến ngõ vào.

Hàm mô tả sự phụ thuộc biến ngõ ra bởi n biến đầu vào có dạng:

Y = f(x1,x2,..xn)

Việc mô tả sự phụ thuộc biến ra bởi các biến vào dựa trên các phép ^ (AND), v (OR),  
- (NOT), và các phép logic so sánh biến đếm, biến thời gian timer. Ngõ ra sẽ là kết quả  
của phép toán logic các đầu vào.

Bảng . Các phép logic cơ bản

|  |  |
| --- | --- |
| A.1 = 1.A = A | A’’ = A |
| A.0 = 0 | (A.B.C)’ = A’ + B’ + C’ |
| A + 0 = 0 + A = A | (A + B + C)’ = A’.B’.C’ |
| A + 1 = 1 | A.B =B.A |
| A + A.B = A | A + B = B + A |
| A.A’ = 0 | (A.B).C = A.(B.C) = A.B.C |
| A + A’ = 1 | (A + B) + C = A + B + C |
| A.A = A | (A + B).C = A.C + B.C |
| A + A = A | (A + B).(A + C) = A +B.C |

## Lựa chọn vi điều khiển và phương pháp vận hành hệ thống

### Kit STM32F767ZIT Nucleo và phần mềm Stm32CubeIDE

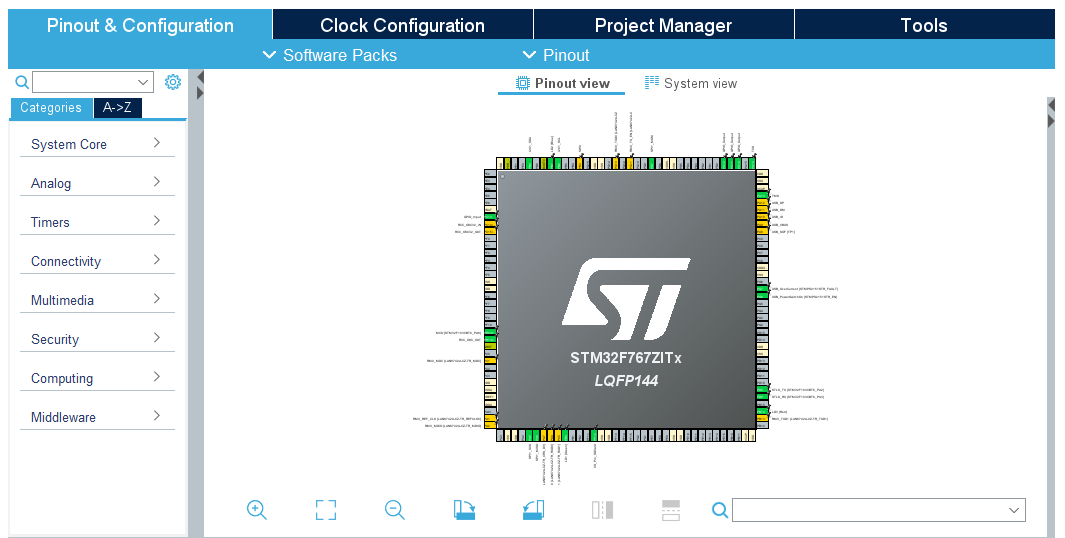
Trong đồ án này, chip STM32F767ZIT được lựa chọn làm vi điều khiển chính của PLC. Chúng ta sẽ tạo ra code C từ chương trình PLC và nạp vào đây để thực hiện các chức năng như một PLC thông thường. Tuy nhiên, hiện tại trong khuôn khổ đồ án này chúng em sẽ dùng luôn kit STM32F767ZIT Nucleo để khảo sát chương trình và chạy mạch thử nghiệm để sau này có thể tích hợp luôn chip vào mạch mà không cần dùng đến kit.

Kit STM32F767ZIT Nucleo được sản xuất với vi điều khiển chính là chip stm32f767, đưa tới những giải pháp với sự linh hoạt, giá cả phải chăng cho người dùng trong việc xây dựng những hệ thống điện tử của mình. Kit STM32F767ZIT rất phù hợp với những ứng dụng, hệ thống vừa và nhỏ đòi hỏi tốc độ xử lý cao và chính xác.

Dưới đây là một vài thông số quan trọng của kit:

* Kích thước bộ nhớ chương trình Flash: 2 Mbytes
* Kích thước bộ nhớ dữ liệu RAM: 532 Kbytes
* Tần số đồng hồ tối đa: 216 MHz
* Số lượng I/O: 114
* Điện áp vận hành: 3.3 V
* ADC: 3x12 bit với số kênh đo lên đến 24 kênh
* DAC: 2x12 bit
* General-purpose DMA: 16-stream DMA controller with FIFOs and burst support
* Lên tới 18 Timers: 13 timer 16-bit, 2 timer 32-bit. 15 timers có thể chạy ở tần số 216 MHz. 2x watchdog và thêm systick timer.
* Kit tích hợp sẵn Debugger ST-Link V2.1, hỗ trợ SWD và JTAG interfaces. Với công cụ mạch nạp ST-Link, người dùng có thể nạp và gỡ rỗi chương trình một cách dễ dàng. Ngoài ra, nhà sản xuất còn tích hợp trên kít các cổng kết nối USB, RJ45 cho kết nối ethernet, nút nhấn reset, nút nhấn cho người dùng, đèn led cho người dùng,…

Đi kèm với các kit như stm32f767zit, nhà sản xuất cũng cung cấp miễn phí cho người dùng bộ công cụ phần mềm là STM32CubeIDE. Phần mềm cho phép người dùng lập trình, nạp và gỡ rối trực tiếp trên vi điều khiển một cách dễ dàng. Ngoài ra phần mềm còn cung cấp giao diện để người dùng có thể cấu hình cho kit một cách chóng, dễ dàng với khả năng sinh code tự động. Người dùng có thể tập trung hoàn toàn năng lượng vào việc sáng tạo cho hệ thống của mình mà không cần bận tâm quá nhiều đến việc viết thư viện driver vì hãng cũng cung cấp sẵn bộ thư viện HAL với sự tiện lợi, dễ sử dụng, ổn định và hiệu suất cao.



Hình . Giao diện cấu hình cho chip trên phần mềm STM32CubeIDE

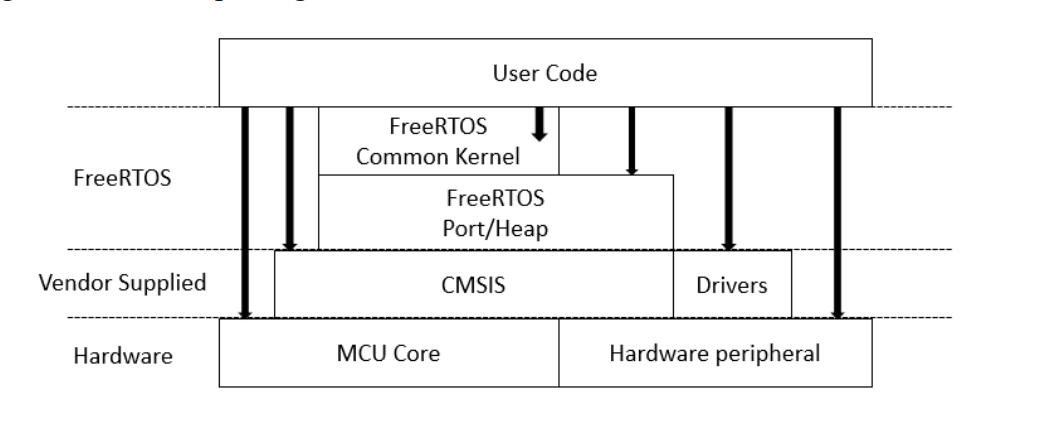
### Hệ điều hành FreeRTOS

Chương trình PLC sau khi được biên dịch ra sẽ được nạp vào vi điều khiển STM32 và vận hành trong hệ điều hành FreeRTOS để có thể đảm bảo được sự đúng đắn về các yếu tố thời gian của một chương trình PLC, đồng thời cũng là có thể điều phối giữa các tác vụ chương trình chính lẫn chương trình thu thập và giám sát dữ liệu.

#### Tìm hiểu chung

Hệ điều hành thời gian thực là hệ điều hành nhằm phục vụ các ứng dụng thời gian thực, với việc xử lý dữ liệu đầu vào mà không có sự chậm trễ của bộ đệm. Tức là hệ điều hành thời gian thực giúp cho các hệ thống thời gian thực hoạt động tin cậy về độ chính xác của kết quả và cả thời điểm đưa ra kết quả đó. Hiện nay có rất nhiều hệ điều hành thời gian thực ( RTOS) dành cho phát triển hệ thống nhúng, việc sử dụng còn phụ thuộc vào khả năng đáp ứng của vi của Vi điều khiển đặc biệt là RAM và tốc độ xung clock. Những hệ điều hành này được phát triển trên các nền ngôn ngữ lập trình khác nhau, phổ biến nhất là ngôn ngữ C, và cũng có thế là JAVA…. Có nhiều hệ điều hành thời gian thực dành cho các dòng vi điều khiển cũng như ứng dụng khác nhau như ChibiOS/RT, FreeOSEK, FreeRTOS, RT-Thread, QNX, VxWorks, Integrity. Trong đó FreeRTOS là một hệ điều hành mã nguồn mở được cộng đồng mạng và các hãng sản xuất như ST hỗ trợ rất nhiều.

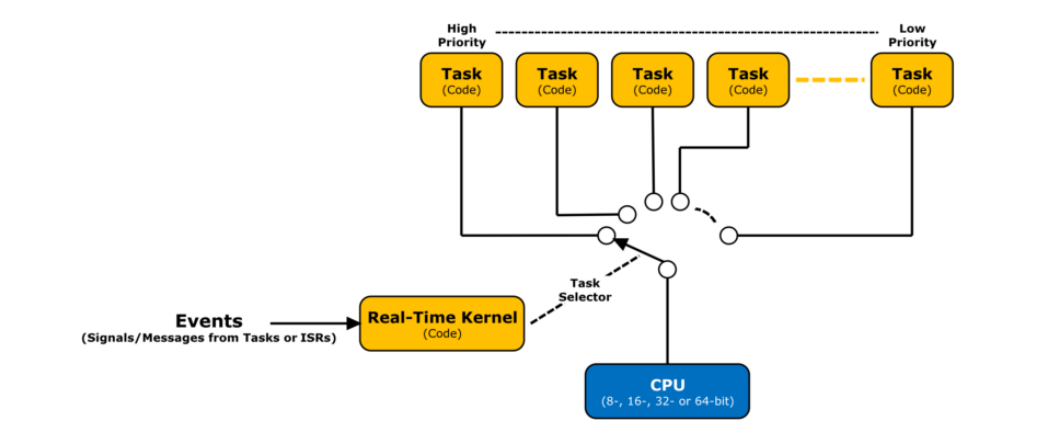
FreeRTOS có tính khả chuyển, kernel có thể download miễn phí và có thể dùng cho các ứng dụng thương mại. Nó phù hợp với các ứng dụng thời gian thực nhỏ. Hầu hết code được viết bằng ngôn ngữ C nên nó có tính phù hợp cao với nhiều nền vi điều khiển khác nhau. Ưu điểm của FreeRTOS là dung lượng nhỏ và có thể chạy trên những nền vi điều khiển mà nhiều hệ điều hành khác không chạy được. Có thể port cho nhiều kiến trúc vi điều khiển và các công cụ phát triển khác nhau. Có cộng đồng mạng đông đảo hỗ trợ miễn phí và cả bản thương mại với các dịch vụ phát triển cũng được cung cấp đầy đủ. Đặc biệt là hệ điều hành FreeRTOS đã được ST hỗ trợ port trên hầu hết các dòng vi điều khiển STM32 và sinh code tự động bằng phần mềm CubeMX nên rất thuận tiện và dễ dàng cho việc học tập và nghiên cứu.



Hình . Vị trí của FreeRTOS trong một firmware stack

#### . Cách hoạt động của FreeRTOS

FreeRTOS cũng giống như những hệ điều hành thời gian thực thông thường, là một phân đoạn hoặc một phần của chương trình nhúng, trong đó nó giải quyết điều phối các task, lập lịch và phân mức ưu tiên cho task, nắm bắt các thông điệp gửi đi từ task. Nhân (kernel) của hệ điều hành giao cho CPU thực hiện việc xử lý các task theo những khoảng thời gian. Nó cũng liên tục kiểm tra mức ưu tiên của các task, sắp xếp các thông điệp từ task và tiến hành lập lịch.

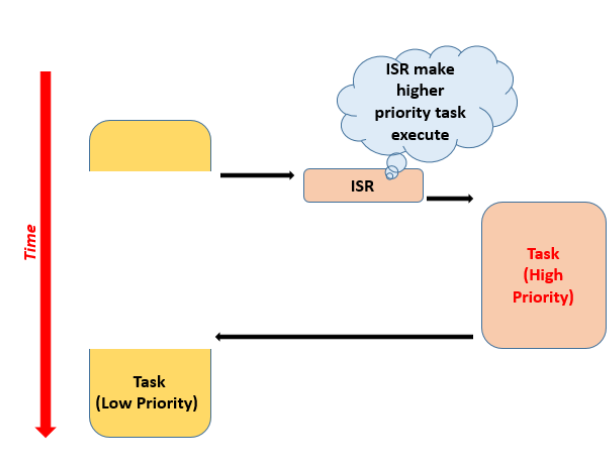


Hình . Cách hoạt động của hệ điều hành thời gian thực

FreeRTOS có hai chế độ lập lịch cho các Task quan trọng là Round-robin và Preemptive Scheduling.

* Round-robin Scheduling: Mỗi task được chia cho một khe thời gian cố định, nếu trong khoảng thời gian được chia đó mà task chưa thực hiện xong thì sẽ bị tạm dừng, chờ đến lượt tiếp theo để thực hiện tiếp công việc sau khi hệ thống xử lý hết một lượt các task.
* Preemptive Scheduling: Chế độ này ưu tiên phân bổ thời gian cho các task có mức ưu tiên cao hơn. Mỗi task được gán 1 mức ưu tiên duy nhất. Có thể có đến 256 mức ưu tiên trong hệ thống, và có thể có nhiều task có cùng mức ưu tiên. Các task có mức ưu tiên cao nhất luôn được kiểm soát bởi CPU, khi phát sinh ISR thì hệ thống sẽ tạm dừng task đang thực thi, hoàn thành ISR sau đó hệ thống thực thi task có mức ưu tiên cao nhất tại thời điểm đó. Sau đó hệ thống mới tiến hành nối lại các task đang bị gián đoạn. Ở chế độ preemptive, hệ thống có thể đáp ứng các công việc khẩn cấp một cách nhanh chóng.

(Xem tiếp trang sau)



Hình . Hệ điều hành hoạt động ở chế độ Preemptive

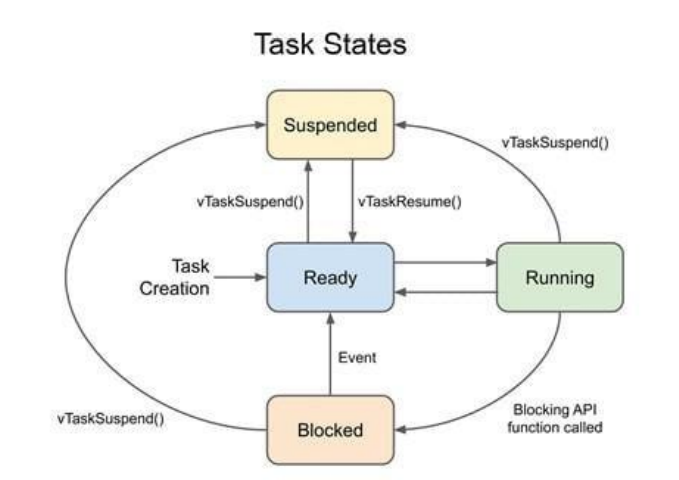
Thông thường hệ điều hành FreeRTOS đều hoạt động ở chế độ preemptive. Ở chế độ này mỗi Task khi được tạo ra sẽ được đi liền với hai thông số:

* Vùng nhớ Stack: mỗi task sẽ được cấp phát một vùng nhớ stack riêng khi được người dùng tạo ra, do đó các task không thể tùy tiện truy cập vào vùng nhớ mà không phải của mình.
* Mức ưu tiên (priority): Cho phép bộ lập lịch quyết định task nào sẽ được chạy với mục tiêu là đảm bảo task có mức ưu tiên cao nhất luôn được chạy trước.

#### Các trạng thái của các Task

Trong quá trình hệ điều hành thời gian thực hoạt động để điều phối các Task, mỗi Task sẽ có bốn trạng thái khác nhau như hình Hình 3.5. Các trạng thái này không được thể hiện trên Task mà nó được tương tác với nhau bởi nhân hệ điều hành.

(Xem tiếp trang sau)



Hình . Các trạng thái của các Task

Trạng thái Running:

Là trạng thái task đang làm việc. Với vi điều khiển chỉ có một nhân thì trong một ngữ cảnh chỉ có một task có trạng thái Running và cũng có thể không có task nào đang ở trong trạng thái Running cả. Task này sẽ chạy cho đến khi nó gọi ra một API và đưa task vào trạng thái blocked hoặc sẽ bị chuyển trạng thái qua một task khác có mức ưu tiên cao hơn (hoặc do hết thời gian mà task được chạy và phải chuyển cho task khác dù có cùng mức ưu tiên).

Trạng thái Ready:

Các task đang ở trong trạng thái sẵn sàng để hoạt động, tức là đang đợi để được bộ lập lịch trao cho tài nguyên của CPU để chạy. Ví dụ có một task A đi vào trạng thái Blocked để đợi một sự kiện nào đó xảy ra từ ngắt. Khi sự kiện đó xảy ra thì đồng nghĩa với việc task A sẽ chuyển sang trạng thái Ready và nếu nó là task có mức ưu tiên cao nhất đang ở trạng thái sẵn sàng thì sẽ được trao quyền để chạy ngay lập tức.

Trạng thái Blocked:

Một task ở trạng thái Blocked là task đang đợi một sự kiện nào đó xảy ra. Có hai tình huống để một task thoát khỏi trạng thái Blocked:

* Một sự kiện xảy ra và kích hoạt task chuyển đổi từ trạng thái Blocked sang trạng thái Ready.
* Trong quá trình hoạt động, task chủ động gọi API để đưa vào trạng thái blocked trong một khoảng thời gian nhất định. Khi hết thời gian thì task sẽ tự động được nhân hệ điều hành đưa về trạng thái Ready.
* Khi một task đang ở trạng thái Blocked, nó không sử dụng bất kì tài nguyên nào của CPU. Khi thoát khỏi trạng thái này thì task chuyển sang trạng thái Ready và được cho phép chạy khi nó trở thành task có mức ưu tiên cao nhất trong hệ thống.

Trạng thái Suspended

Một task đi vào trạng thái Suspended khi nó chủ động gọi API vTaskSuspend(). Task sẽ ngừng hoạt động và không bị ảnh hưởng bởi bộ lập lịch cho đến khi vTaskRusme() được gọi lại và chuyển task sang trạng thái Ready. Cũng giống như trạng thái Blocked, trạng thái Suspended không sử dụng thời gian hoạt động của CPU.

#### . Các dịch vụ mà FreeRTOS cung cấp

Trong một hệ thống nhúng có nhiều tác vụ khác nhau thì giữa các task cũng luôn cần phải có sự kết nối, giao tiếp và trao đổi dữ liệu với nhau. FreeRTOS đưa ra gần như đầy đủ các dịch vụ mà một hệ điều hành cần có như:

* Với Inter-task Communication:
  + Signal events: Đồng bộ các task
  + Message queue: Trao đổi tin nhắn giữa các task trong hoạt động bằng việc sử dụng cấu trúc giữ liệu hàng đợi FIFO
  + Stream buffer, message buffer: Trao đổi một lượng giữ liệu theo số lượng byte cố định hoặc tùy ý giữa một chương trình ngắt với task hoặc giữa hai task với nhau.

- Với Resource sharing (chia sẻ tài nguyên):

* Semaphore: có thể truy xuất, quản lý tài nguyên của chương trình từ các task khác nhau hoặc dùng để đếm sự kiện.
* Mutex: đồng bộ hóa và dựa vào cơ chế token để đảm bảo tài nguyên được truy cập một cách có tổ chức.

Tất cả các dịch vụ trên đều được quản lý và điều khiển bởi nhân hệ điều hành. Trong đồ án này tuy chưa sử dụng hết các dịch vụ của FreeRTOS nhưng sẽ là tiền đề sau này để phát triển các phần phụ trợ liên quan về mặt truyền thông và trao đổi dữ liệu của PLC để chương trình chính luôn hoạt động đúng mà vẫn đảm bảo việc truyền thông dữ liệu.

## Thuật toán xử lý chung

Các bước tổng quát từ viết chương trình đến chạy chương trình hoạt động:



Hình . Sơ đồ khối các bước thực hiện từ lập trình đến nạp chương trình cho PLC

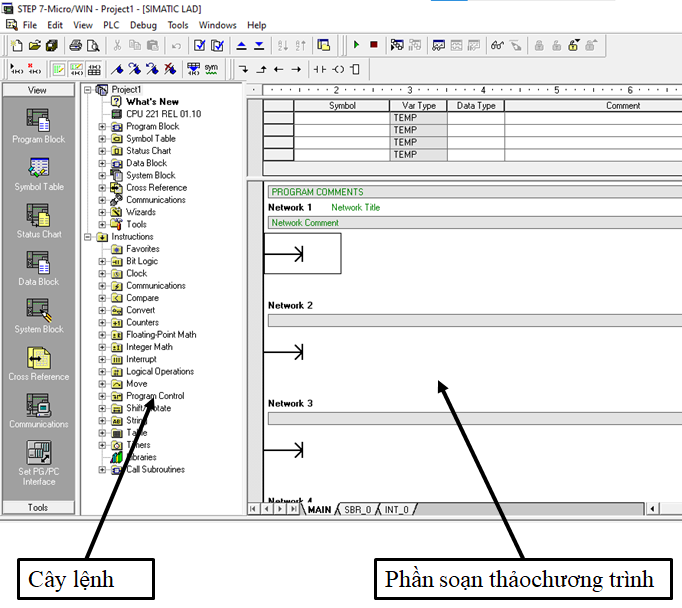
Chương trình cuối cùng được biên dịch ra sẽ gồm một file.c và một file.h . Các thuật toán trong chương trình sẽ được thiết kế kết hợp với các thư viện của hệ điều hành RTOS

## Lập trình LAD và xuất file chứa code STL

### Giới thiệu phần mềm STEP 7 – MicroWIN

Phần mềm STEP 7 – MicroWIN được dùng để lập trình cho họ PLC S7 – 200 của Siemens và thiết lập điều khiển giữa họ PLC này và các module khác như: module truyền thông EM241, module điều khiển vị trí EM253, màn hình TD 200,…Phần mềm cho phép thiết lập truyền thông giữa máy tính PC, lập trình PLC, thực hiện chức năng gán ngõ vào/ra, giám sát bộ nhớ của PLC… Tuy nhiên, trong phạm vi đồ án này, phần mềm chỉ sử dụng với mục đích lập trình để tạo ra file có thể biên dịch sang ngôn ngữ C để nạp vào vi điều khiển.

Phần mềm này tương đối nhẹ và yêu cầu cấu hình thấp chỉ cần Microsoft Windows 2000 trở lên là có thể cài đặt được. Ngoài ra nó có thể soạn thảo ở chế độ offline và kiểm tra hoạt động của chương trình với tính năng mô phỏng. Những điều này rất phù hợp với yêu cầu đề ra là chỉ dùng để lập trình chương trình PLC và sử dụng để mô phỏng kiểm nghiệm.

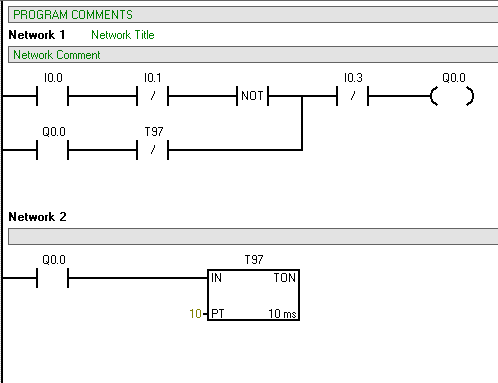


Hình . Giao diện phần mềm STEP7 – MicroWIN

Hình 3.2 là giao diện của phần mềm STEP7 – MicroWin. Trên giao diện này, ta quan tâm hai phần chính là khu vực soạn thảo chương trình và khu vực cây lệnh để lựa chọn các lệnh cần thiết.

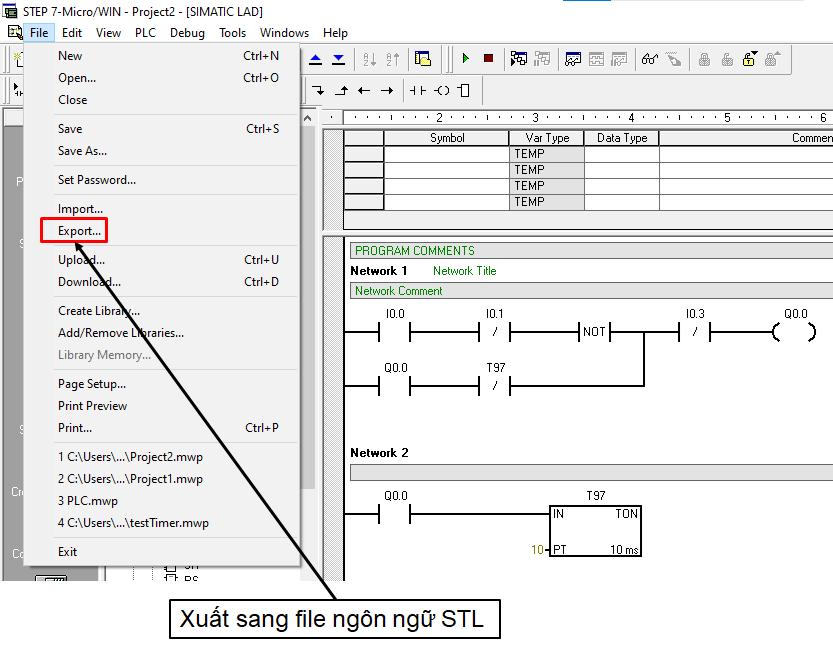
### Tạo file chưa code STL

Ban đầu ta cần lập trình một chương trình bằng ngôn ngữ LAD trên phần mềm STEP7 – MicroWin như hình 3.3 sau:



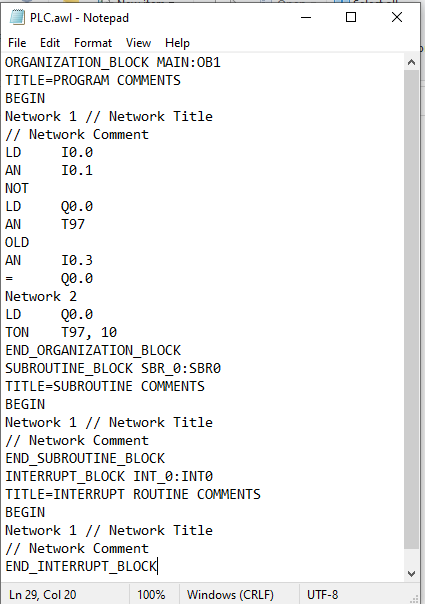
Hình . Chương trình LAD trên STEP7 – MicroWIN

Sau đó ta xuất file LAD này sang một file mới là dạng STL:



Hình . Xuất file LAD sang STL

Sau đó ta sẽ nhận được một file .awl chứa chương trình PLC dưới dạng ngôn ngữ STL như trên hình 3.10:



Hình . File code STL

Trong file này còn chưa nhiều dòng khân cần thiết nên nhiệm vụ đầu tiên là phải loại bỏ các dòng không cần thiết trước rồi sẽ thược hiện các thao tác biên dịch.

## Chương trình dịch code

Các bước viết trong chương trình dịch code:

Bước 1: Loại bỏ các dòng không cần thiết trong file code STL.

Bước 2: Chuyển file chứa code STL thành danh sách các ký tự, xử lý danh sách sao cho thuận tiện bước 3 nhất.

Bước 3:

- Tạo file .c

- Viết chương trình khai báo tự động các biến cần thiết trong chương trình.

- Viết chương trình đọc biểu thức ngõ vào .Sau đó viết biểu thức ngõ ra dựa trên các biến ngõ vào từ danh sách đã được xử lý vào file. Cuối cùng là chương trình ghi các giá trị ngõ ra. Tất cả những điều này đều nằm chung trong một hàm để sau này xử lý như một tác vụ trong hệ điều hành FreeRTOS.

- Viết các hàm khai báo timer sử dụng, hàm cho phép chạy timer , hàm xử lý timer. Nếu như trong chương trình có timer. Các timer này là các software timer ( timer mềm ), là một tính năng của hệ điều hành FreeRTOS không hoạt động dựa trên việc đếm tăng các thanh ghi của vi điều khiển ( như các timer cứng, bị giới hạn bởi giá trị tối đa mà thanh ghi có thể thao tác đến ) mà dựa vào việc đếm số chu kỳ lập lịch của hệ điều hành nên có thể hoạt động linh hoạt hơn.

Bước 4: Tạo file .h để định nghĩa ( define ) các vùng nhớ, khai báo các nguyên hàm cần sử dụng như hàm chương trình PLC, hàm khai báo timer, hàm khởi tạo timer, hàm xử lý timer.

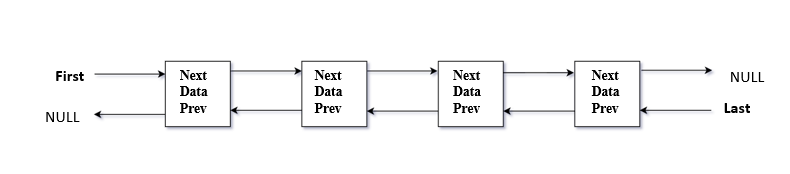
Để có thể thao tác được với chuỗi kí tự ( lưu trữ tên các biến, tên mã lệnh STL và xử lý chương trình theo kiểu ngăn xếp ( ngôn ngữ STL hoạt động theo kiểu ngăn xếp như đã đề cập ở mục 2.1.2) ta sẽ phải sử dụng các cấu trúc dữ liệu đặc biệt sau đây:

* Danh sách liên kết đôi (Doubly Linked List)
* Bảng băm đóng ( Hash table)

### Cấu trúc dữ liệu hỗ trợ chương trình biên dịch

#### Danh sách liên kết đôi

Danh sách liên kết đôi là danh sách mà mỗi phần tử của nó gồm 3 thàn phần: phần dữ liệu, con trỏ **next** chứa địa chỉ phần tử tiếp theo, con trỏ **prev**  chứa địa chỉ của phần tử liền trước, con trỏ **next**  của phần tử cuối cùng của danh sách sẽ bằng NULL, con trỏ **prev** của phần tử đầu tiên cũng bằng NULL. Danh sách liên kết đôi hoàn toàn được xác định bởi hai con trỏ: con trỏ **First**  chứa địa chỉ của phần tử đầu tiên, con trỏ **Last** chứa địa chỉ của phần tử cuối cùng.



Hình . Mô hình danh sách liên kết đôi

Bằng cách thay đổi địa chỉ trỏ đến của con trỏ **next**  và **prev** của các phần tử ta có thể tạo ra vô số phần tử.

( Xem chi tiết Phụ lục 1)

#### Bảng băm đóng

Bảng băm đóng là một mô hình dữ liệu tập hợp chỉ giữ lại 3 phép toán: chèn một phần tử vào tập hợp, xóa một phần tử khỏi tập hợp, tìm một phần tử trong tập hợp.

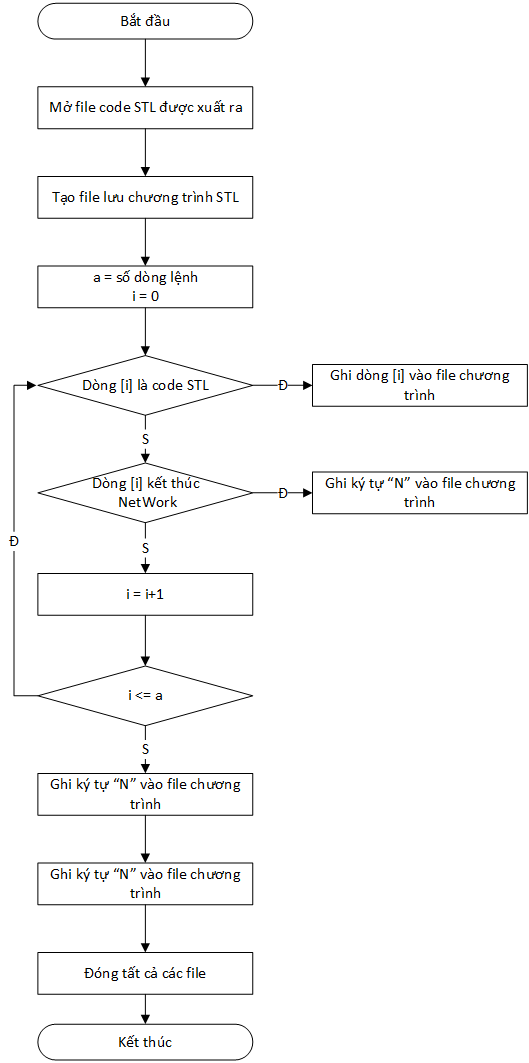
Trong bảng băm đóng ta chưa các phần tử trong một mảng cố định t[n] với n là một số nguyên tố quy định số phần tử của tập hợp.

Cách cài đặt một bảng băm:

* Ban đầu ta khai báo tất cả các phần tử là “ vacant” mang ý nghĩa là chưa có phần tử nào được ghi tại đây. Nếu một phần tử đã được thêm vào mà sau đó xóa đi thì sẽ được thế vào chỗ đó là “delete”.
* Xác định hàm băm h(x) với x là phần tử cần chèn vào. Để xác định vị trí cần chèn x vào thì ta xác định i = h(x), nếu t[i] =” vacant” hoặc t[i] = “delete” thì có thể chèn vào.Nếu t[i] đang chứa một phần tử khác thì ta dùng hàm băm lại h1(x)= h(x)+1 ; h2(x) = h(x)+2 ;…(h(x)+i)%n.
* Thiết lập một hàm thăm dò F(x,k,j) nhằm thăm dò xem phần tử x tại vị trí t[i] với i = h(x), nếu t[i] = x hoặc t[i] = “ vacant” thì k = i = j và dừng thăm dò. Trái lại, ta tiếp tục thăm dò tiếp các vị trí t[k] = h1(x),h2(x),… Quá trình thăm dò sẽ dừng lại khi xảy ra một trong ba trường hợp sau: t[k] = “ vacant” ( phần tử x không có trong bảng) ; t[k] = x ( tìm thấy phần tử x trong bảng) ; k = i ( chứng tỏ ta đã xuất phát từ vị trí i và đi hết một vòng toàn bộ bảng và quay về đây, bảng đã đầy không thể chèn thêm x vào). Tham số k trong hàm thăm dò nêu lên vị trí mà tại đó quá trình thăm dò dừng còn giá trị j là giá trị mà tại đó quá trình thăm dò lần đầu tiên gặp “vacant” hoặc “delete” , nếu cần có thể chèn x vào đây.
* Từ hàm thăm dò xây dựng các hàm xóa phần tử, tìm phần tử theo mục đích yêu cầu.
* ( Xem chi tiết Phụ lục 1)

### Xử lý và tạo ra file chỉ chứa code STL

Thuật toán để đọc và loại bỏ những hàng không phải code STl được đề trình bày dưới dạng lưu đồ thuật toán như trên hình 3.12 sau:



Hình . Sơ đồ khối thuật toán xử lý file, tạo file chứa mã code STL

Sau khi thực hiện ta có kết quả như sau:

Ảnh có chứa bàn

Mô tả được tạo tự động

Hình . Kết quả sau khi thực hiện lưu đồ hình 3.12

### Lưu code STL vào danh sách liên kết đôi và xử lý

Sau khi có file chứa code STL, chuyển file code STL sang danh sách liên kết đôi, sau đó thay thế các ký tự cho phù hợp với ngôn ngữ C, như dấu “.” chuyển thành dấu “\_’. Dấu “,” thì bỏ qua và tiếp điểm thường đóng thêm ký tự “!” đằng trước, thêm ký tự “sl” hoặc “sx” đằng sau nếu gặp các lệnh sườn.

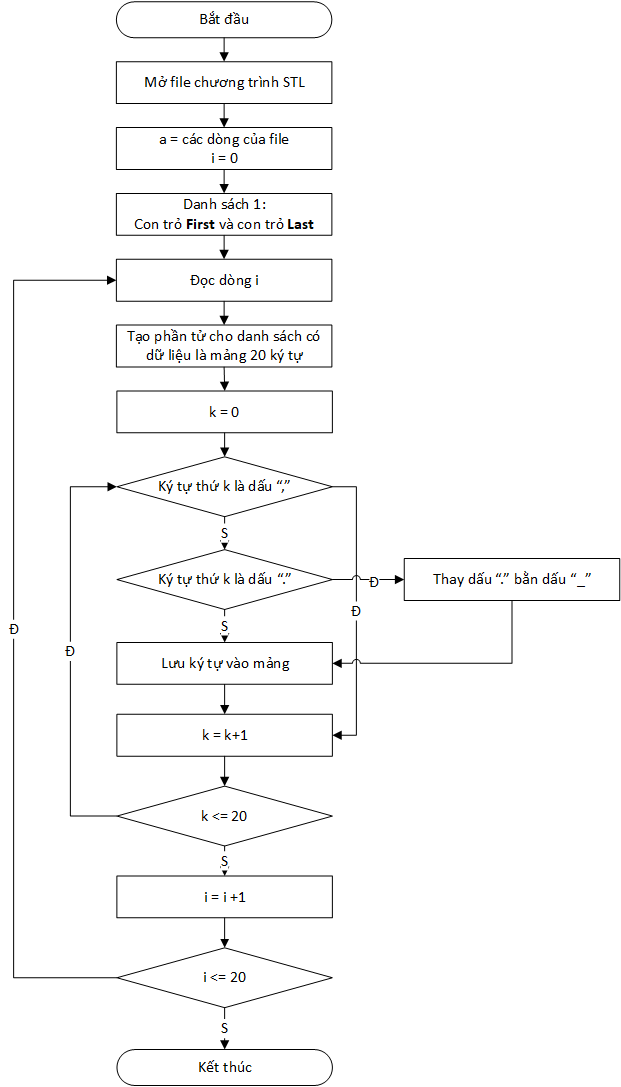
Thuật toán lưu code STL vào trong sanh sách liên kết đôi và lưu danh sách các biến vào bảng băm đóng:

Ảnh có chứa văn bản

Mô tả được tạo tự động

Hình . Sơ đồ khối chuyển file code chương trình vào danh sách liên kết đôi và bảng băm đóng

Thuật toán lưu các dòng code STL vào danh sách liên kết đôi thứ nhất đồng thời xóa “,” và thay “.” bằng dấu “\_” để phù hợp lập trình vi điều khiển.

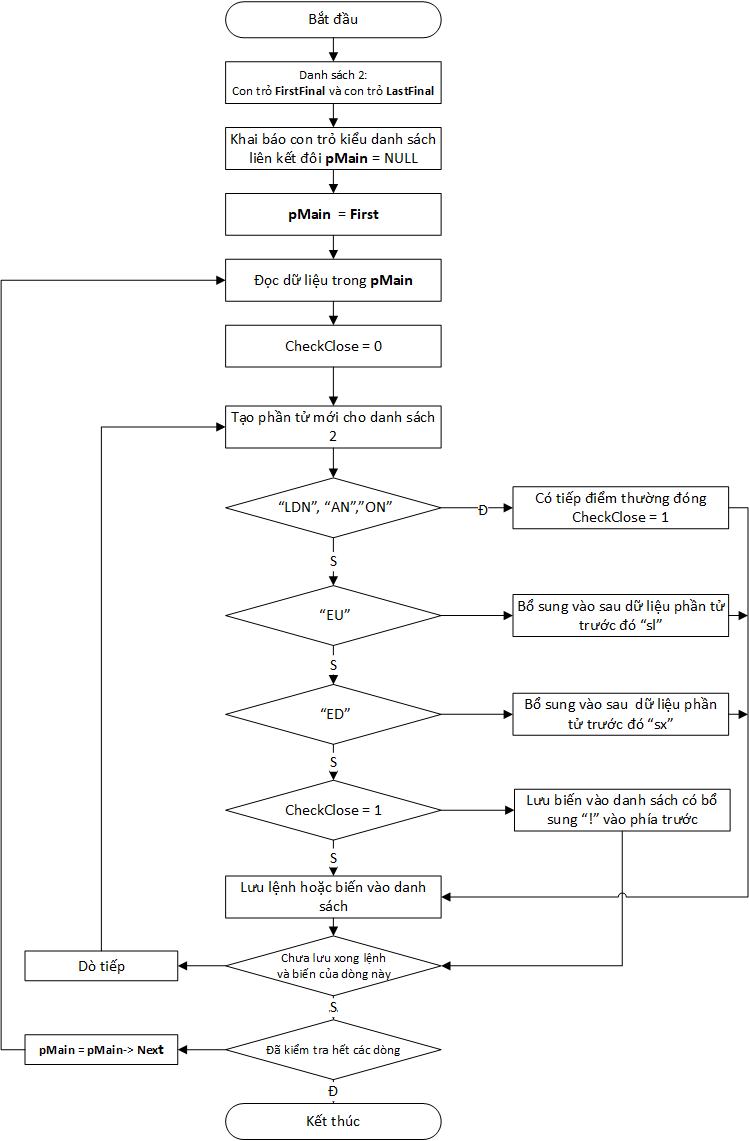


Hình . Sơ đồ khối xử lý chuỗi chuyển đổi phù hợp sang ngôn ngữ vi điều khiển

Lưu ý: Đây sẽ là danh sách liên kết đôi đầu tiên ( sử dụng con trỏ **First** và con trỏ **Last**  ) và các phần tử của danh sách là các dòng code STL đã được chỉnh sửa cho phù hợp. Mỗi phần tử của danh sách này có dữ liệu là một mảng gồm 20 kì tự do các dòng lệnh hầu như đều không có quá 20 kí tự, ta sẽ kiểm tra từng kí tự của dòng code trước khi lưu vào mảng này.

Thuật toán lưu các mã lệnh STL và các biến theo thứ tự vào trong danh sách liên kết đôi thứ 2 đồng thời bổ sung các ký tự “!” , “sl”, “sx” nếu cần.

( Xem trang sau )



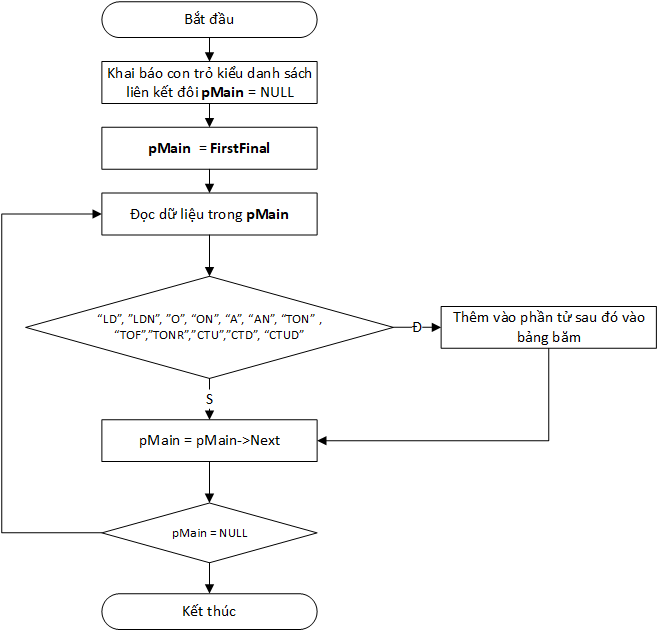
Hình . Sơ đồ khối lưu lệnh và biến theo thứ tự vào danh sách liên kết đôi

Sau khi thực hiện lưu đồ như trên hình 3.16 ta sẽ được một danh sách liên kết đôi như sau:

LD ; I0\_0 ; AN ; (!I0\_1) ; NOT ; LD ; Q0\_0 ; AN ; (!T97) ; OLD ; AN ; (!I0\_3) ; = ; Q0\_0 ; N ; LD ; Q0\_0 ; TON ; T97 ; 10 ;

( Dấu “;” ở đây chỉ dùng để phân biệt các phần tử trong danh sách liên kết đôi )

Tiếp theo là thuật toán lưu trữ tên các biến của chương trình vào trong bảng băm. Mục đích của việc lưu vào bảng băm là các biến chỉ lưu một lần, nếu đã lưu tên rồi sẽ không cần lưu nữa, đồng thời sau này nếu có tìm kiếm cũng rất dẽ dàng.



Hình . Sơ đồ khối lưu biến chương trình PLC vào bảng băm

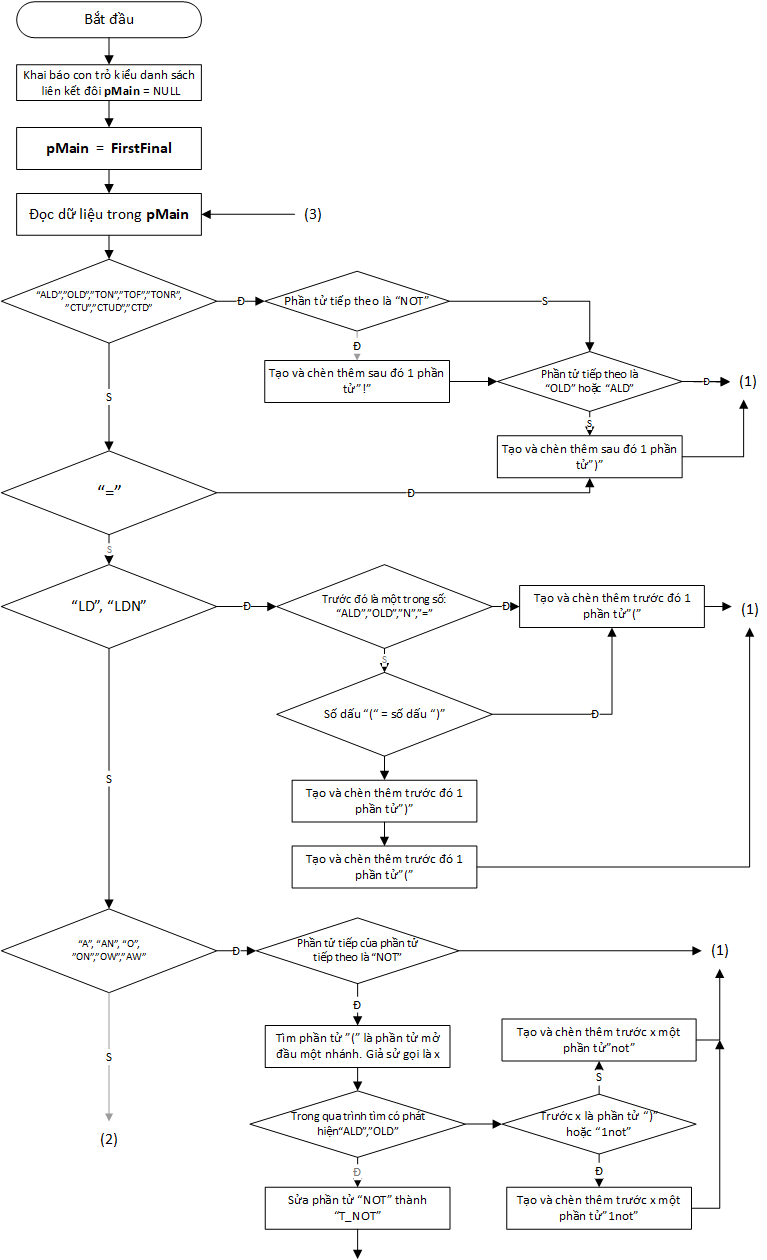
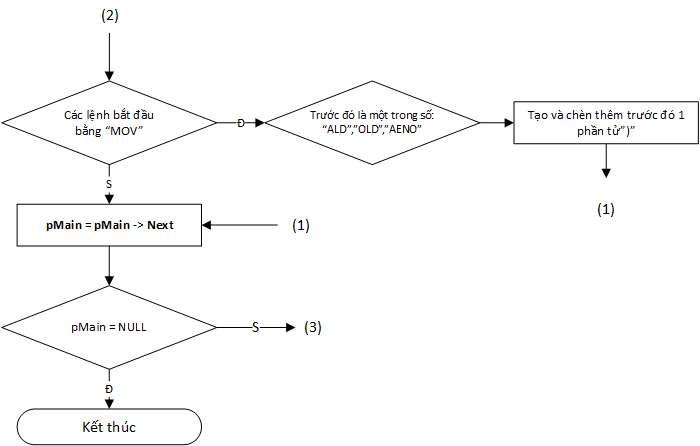
Sau khi thực hiện thì ta sẽ được các giá trị sau trong bảng băm;

T97 ; I0\_0 ; I0\_1 ; I0\_2 ; Q0\_0 ;

( Dấu “;” ở đây chỉ dùng để phân biệt các phần tử trong bảng băm, các biến chỉ xuất hiện có một lần duy nhất)

Cuối cùng là thuật toán tách nhánh. Nhánh là chuỗi các câu lệnh thẳng không rẽ nhánh trong LAD. Tách các nhánh có nhiệm vụ quan trọng để thực hiện các biểu thức ở bước 3 dễ dàng hơn. Tách nhánh bằng cách thực hiện kiểm tra điều kiện mở đầu nhánh, mở đầu network, kết thúc nhánh, kết thúc network để đặt dấu “(“ hoặc “)” cho phù hợp .

( xem lưu đồ trang sau )

Hình . Sơ đồ khối tách các nhánh và kiểm tra lệnh NOT trong code STL

Trên lưu đồ hình 3.17 có bổ sung thêm các phần tử “not” và “1not” cũng như sửa đổi lệnh thành “T\_NOT” vào danh sách liên kiết thì những phần tử này sẽ hỗ trợ việc giải quyết các lệnh phủ định được trình bày ở các lưu đồ phần sau của đồ án.

Sau khi thực hiện lưu đồ ta có kết quả sau:

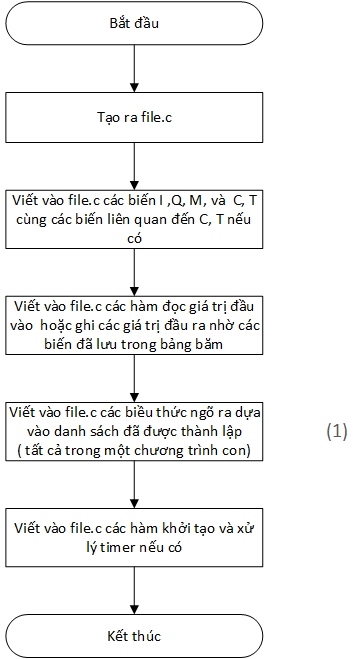
not ; ( ; LD ; I0\_0 ; AN ; (!I0\_1) ; NOT ; ) ; ( ; LD ; Q0\_0 ; AN ; (!T97) ; ) ; OLD ; AN ; (!I0\_3) ; = ; Q0\_0 ; N ; ( ; LD ; Q0\_0 ; ) ; TON ; T97 ; 10 ;

( Dấu “;” ở đây chỉ dùng để phân biệt các phần tử trong danh sách liên kết đôi )

### Tạo tệp nguồn

( Xem chi tiết Phụ lục 2)

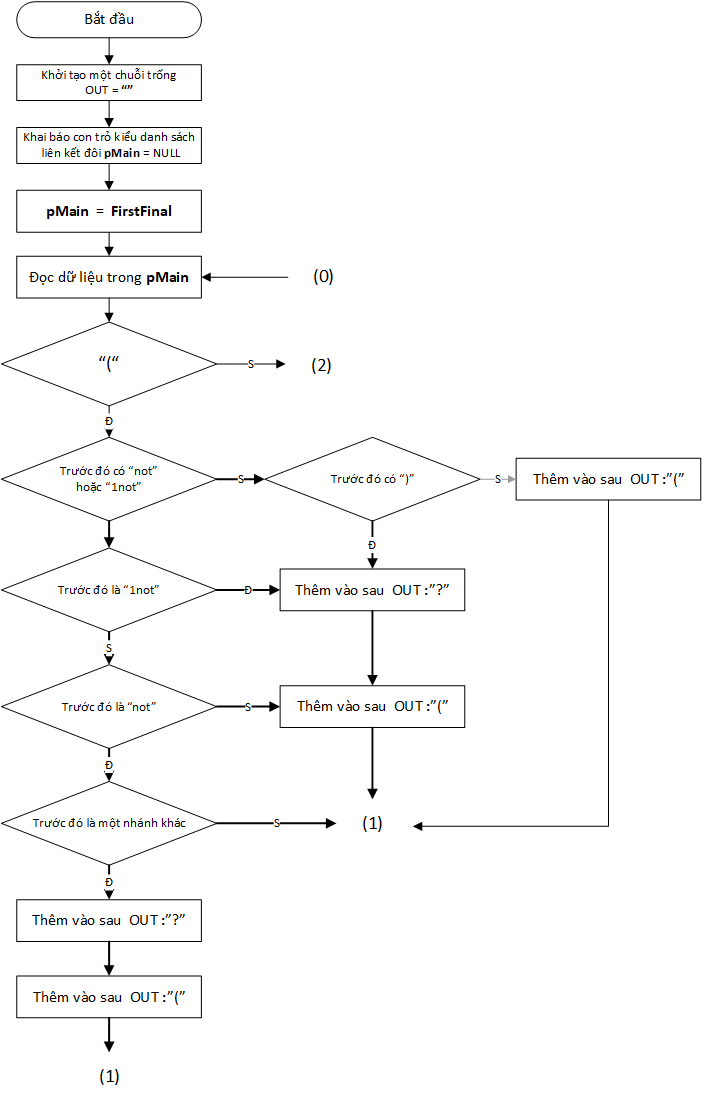
Lưu đồ thuật toán chung cho việc khởi tạo tệp nguồn ( file .c ) chứa khai báo các biến sử dụng, các hàm như: hàm chương trình PLC, các hàm liên quan đến tỉmer ( nếu có).



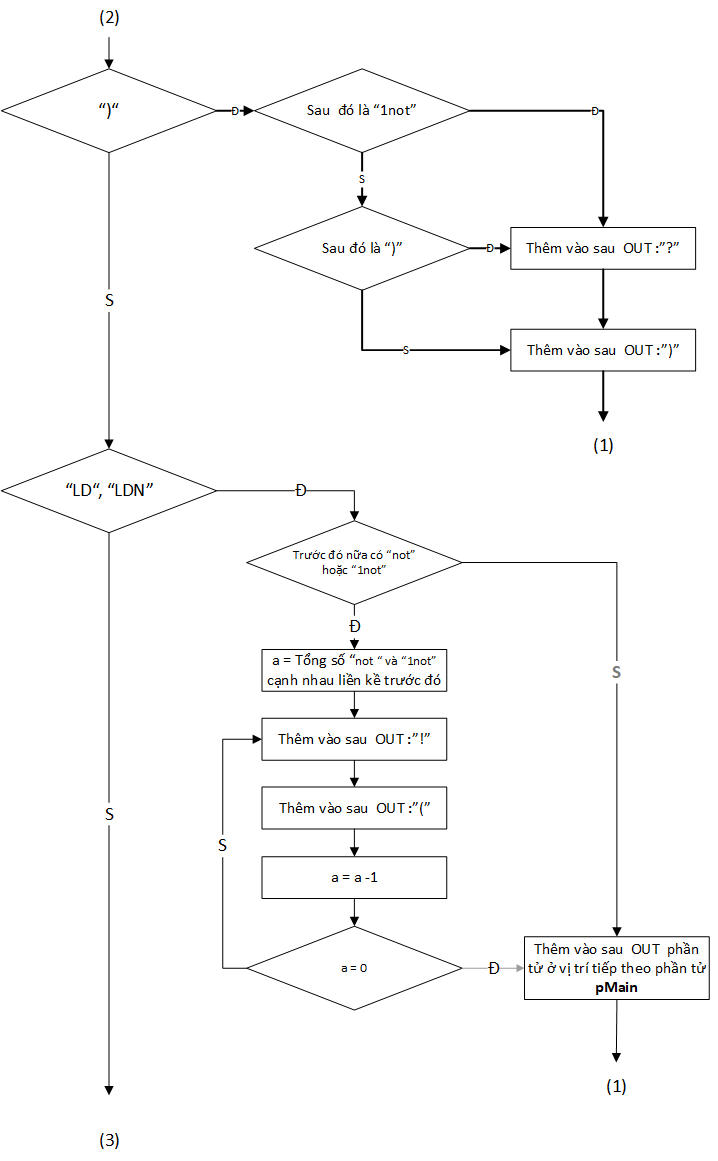
Hình . Sơ đồ khối thuật toán chung tạo ra tệp nguồn

Thuật toán chung tạo ra tệp nguồn nêu trên Hình 3.18 ta sẽ tập trung vào mục (1). Các phần còn lại sẽ được đưa ra ở phần phụ lục.

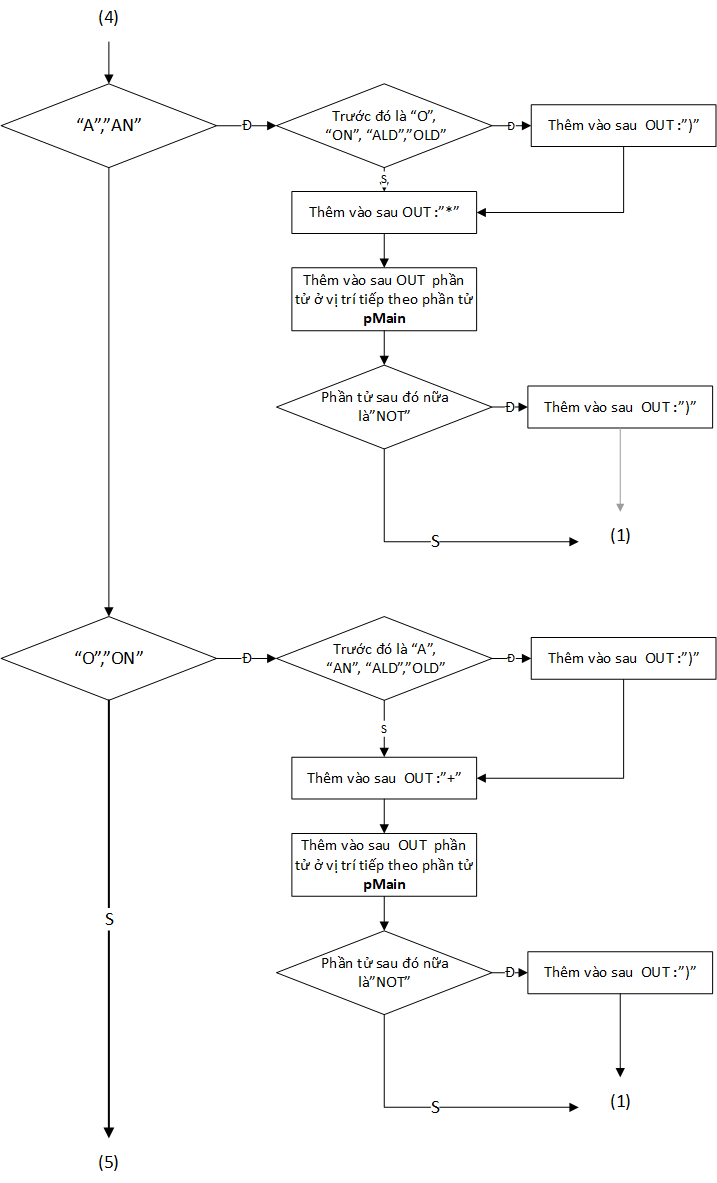
Sau đây là thuật toán xử lý các biểu thức ngõ ra:



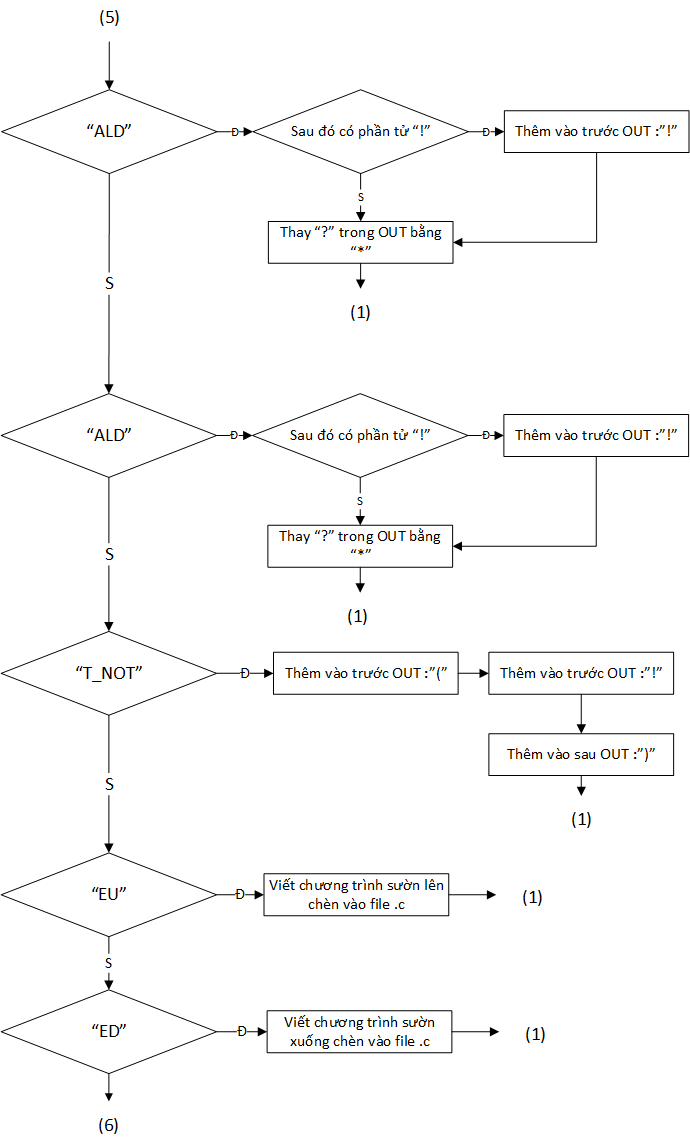
Hình . Sơ đồi khối thuật toán xử lý các biểu thức ngõ ra phần 1



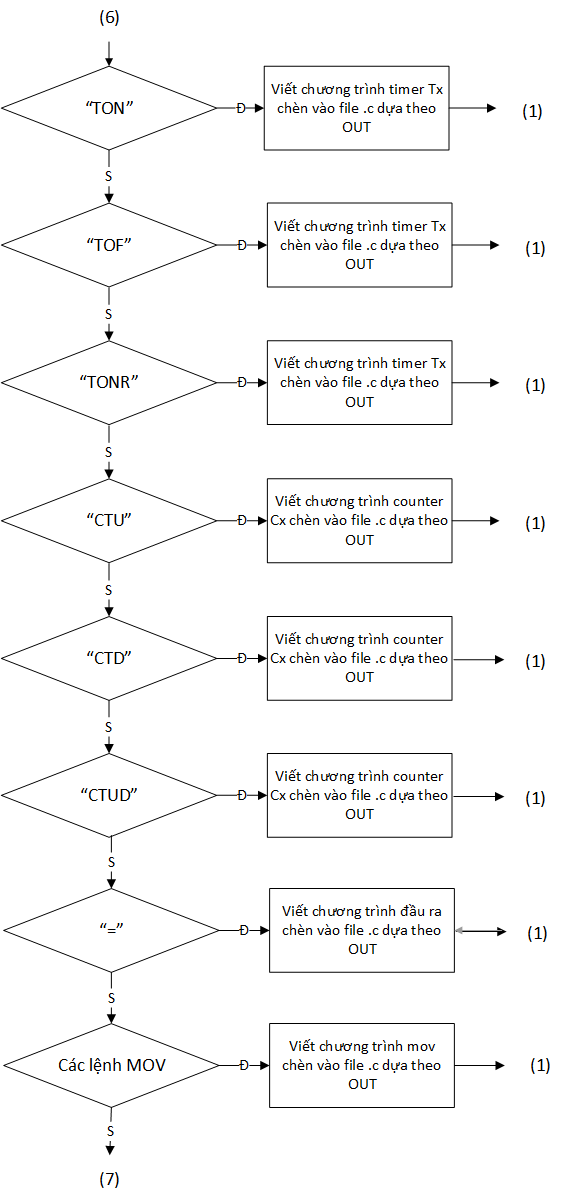
Hình . Sơ đồi khối thuật toán xử lý các biểu thức ngõ ra phần 2



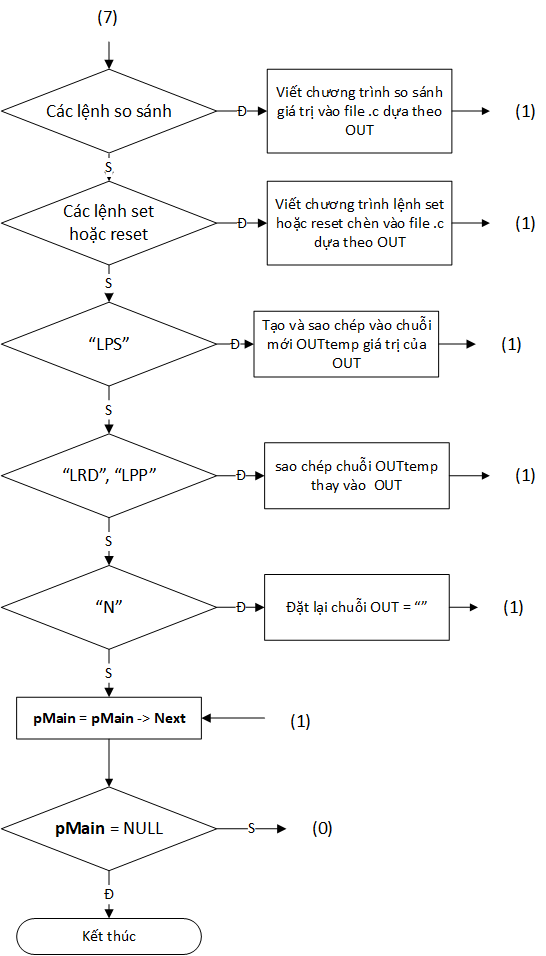
Hình . Sơ đồi khối thuật toán xử lý các biểu thức ngõ ra phần 3



Hình . Sơ đồi khối thuật toán xử lý các biểu thức ngõ ra phần 4



Hình . Sơ đồi khối thuật toán xử lý các biểu thức ngõ ra phần 5



Hình . Sơ đồi khối thuật toán xử lý các biểu thức ngõ ra phần 6

Sau khi thực hiện xong bược này ta được CÁc biểu thức ngõ ra như sau:

/\*--------------NetWork 1 -----------\*/

Q0\_0 =(((!(I0\_0\*(!I0\_1)))+(Q0\_0\*(!T97)))\*(!I0\_3));

if (Q0\_0 > 0)

{

Q0\_0 = 1;

}

else

{

Q0\_0 = 0;

}

/\*--------------NetWork 2 -----------\*/

if (!T97reset)

{

vaoT97 = (Q0\_0);

T97;

if (vaoT97)

{

if (countT97 >= datT97)

T97 = 1;

if ((T97 == 0) && (checkT97 == 0))

{ xTimerStart(handle\_timerPLC[0],portMAX\_DELAY);

checkT97 = 1;

}

}

else

{

if (checkT97 == 1)

{

xTimerStop(handle\_timerPLC[0],portMAX\_DELAY);

checkT97 = 0;

}

T97 = 0;

countT97 = 0;

}

}

else

{

xTimerStop(handle\_timerPLC[0],portMAX\_DELAY);

countT97 = 0;

T97 = 0;

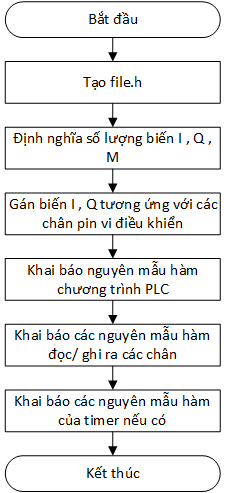
checkT97 = 0;

}

### Tạo tệp thư viện

( Xem chi tiết Phụ lục 3)

Tạo file .h để định nghĩa ( define ) các vùng nhớ, khai báo các nguyên hàm cần sử dụng như hàm chương trình PLC, hàm khai báo timer, hàm khởi tạo timer, hàm xử lý timer. Thuật toán chung được sử dụng như sau:



Để có thể thực hiện chính xác việc xác định số lượng đầu vào ra cần dùng thì ta chỉ việc đọc các giá trị trong bảng băm đóng mà ta đã thêm vào trước. Các giá trị trong bảng băm là duy nhất nên chúng ta sẽ không bị khai báo nhầm lẫn.

# KẾT LUẬN

## Kết luận chung

Xem Mục 1.3.12

## Hướng phát triển

(Nếu có)

## Kiến nghị và đề xuất

(Nếu có)

# TÀI LIỆU THAM KHẢO

1. T. H. Cormen, C. E. Leiserson, and R. L. Rivet, *Introduction to Algorithm*. MIT Press, McGraw-Hill, 1990.
2. J. W. DuBois, S. Schuetze-Coburn, S. Cumming, and D. Paolino, “Outline of discourse transcription,” in *Talking Data: Transcription and Coding in Discourse Research*, J. A. Edwards and M. D. Lampert, Ed. Hillsdale, NJ: Lawrence Erlbaum Associates, 1993, pp. 45-89.
3. J. M. Airey, J. H. Rohfl, F. Brooks Jr., “Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments,” *Comptuer Graphics*, Vol. 24, No. 2, pp. 41-50, 1990.
4. S. Brandt, G. Nutt, T. Berk, M. Humphrey, “Soft Real time Application Execution with Dynamic Quality of Service Assurance,” in *Proceedings of the Sixth IEEE/IFIP International Workshop on Quality of Service*, Hawaii, USA, May 1998, pp. 154-163.
5. K. Riley, “Language theory: Applications versus practice,” presented at the Conf. of the Modern Language Association, Boston, MA, December 27-30, 1990.
6. J. Jones. (1991). *Networks* (2nd ed.) [Online]. Available: http://www.atm.com.

# PHỤ LỤC

## Phụ lục 1. Chương trình cài đặt các cấu trúc dữ liệu

1. Danh sách liên kết đôi

Khai báo cấu trúc:

struct Link

{

char \*data;

struct Link \*next;

struct Link \*prev;

}; // Cấu trúc 1 phần tử

typedef struct Link LinkList;

Các thuật thoán với danh sách liên kết đôi:

Thuật toán chèn vào sau phần tử:

// CurrentElement : Vị trí phần tử xét

// Insert : Phần tử chèn

// NextElement : Phần tử tiếp theo của phần tử đang xét

Void InsertNextElement(LinkList\*\*CurrentElement,LinkList\*\*Insert,LinkList\*\*NextElement)

{

(\*CurrentElement)->next = \*Insert;

(\*Insert)->prev = \*CurrentElement;

(\*Insert)->next = \*NextElement;

(\*NextElement)->prev = \*Insert;

}

Thuật toán chèn vào trước phần tử:

// CurrentElement : Phần tử đang xét

// Insert : Phần tử chèn

void InsertPrevElement(LinkList \*\*CurrentElement,LinkList \*\*Insert)

{

LinkList \*PreElement;

PreElement = (\*CurrentElement)->prev;

PreElement->next = \*Insert;

(\*Insert)->prev = PreElement;

(\*Insert)->next = \*CurrentElement;

(\*CurrentElement)->prev = \*Insert;

}

Thuật toán xóa một phần tử:

// pMain: Vị trí phần tử cần xóa

// pPREV: Phần tử trước của phần tử đang xét

// pNEXT: Phần tử tiếp theo của phần tử đang xét

void DeleteLinkList(LinkList \*\*pMain,LinkList \*\*pPREV,LinkList \*\*pNEXT)

{

(\*pPREV)->next = (\*pNEXT);

(\*pNEXT)->prev = (\*pPREV);

free((\*pMain));

}

B. Các thuật toán với bảng băm đóng

#define sizeOfHash ((int)15) // Kích thước bảng băm

#define PrimeNumber ((int)151) // Số N

typedef char stringHashTable[sizeOfHash] ; // Kiểu bảng

stringHashTable SaveIO[PrimeNumber]; // Khai báo một bảng băm

int H\_HashFunction(stringHashTable node) // Hàm băm

{

int sumChar, sizeOfNode;

sizeOfNode = strlen(node);

sumChar = 0;

for (int i = 0; i < sizeOfNode; i++)

sumChar += (int)node[i];

return (sumChar % PrimeNumber);

}

void H\_ProbeFunction(stringHashTable node, int \*StopPosition, int \*FirstInsert) // Hàm thăm dò

{

int tempStopPosition, tempFirstInsert;

int check = H\_HashFunction(node);

tempFirstInsert = check;

if ((strcmp(SaveIO[check], node) == 0) || strcmp(SaveIO[check], "vacant") == 0)

tempStopPosition = check;

else

{

tempStopPosition = (check + 1) % PrimeNumber;

while ((tempStopPosition != check) && (strcmp(SaveIO[tempStopPosition], node) != 0) && strcmp(SaveIO[tempStopPosition], "vacant") != 0)

{

if((strcmp(SaveIO[tempStopPosition],"delete")==0)&&(strcmp(SaveIO[tempFirstInsert], "delete") != 0))

tempFirstInsert = tempStopPosition;

tempStopPosition = (tempStopPosition + 1) % PrimeNumber;

}

if (strcmp(SaveIO[tempStopPosition],"vacant") == 0 && (strcmp(SaveIO[tempFirstInsert], "delete") != 0))

tempFirstInsert = tempStopPosition;

}

\*StopPosition = tempStopPosition;

\*FirstInsert = tempFirstInsert;

}

void H\_InsertFunction(stringHashTable node) // Hàm chèn thêm phần tử vào bảng

{

int StopPosition, FirstInsert;

H\_ProbeFunction(node, &StopPosition, &FirstInsert);

if (strcmp(SaveIO[StopPosition], node) != 0)

{

if (strcmp(SaveIO[FirstInsert], "vacant") == 0 || (strcmp(SaveIO[FirstInsert], "delete") == 0))

{

strcpy(SaveIO[FirstInsert], node);

}

else

{

assert(1); // Bảng bị đầy

}

}

}

int H\_FindFunction(stringHashTable node) // Hàm tìm một phần tử

{

int StopPosition, FirstInsert;

H\_ProbeFunction(node, &StopPosition, &FirstInsert);

if (strcmp(SaveIO[StopPosition], node) == 0)

{

return StopPosition;

}

return -1;

}

## Phụ lục 2. Chương trình ví dụ hình 3.8 cho file.c

#include "DataPLC.h"

volatile static uint8\_t I[2][8] = {};

volatile static uint8\_t Q[2][8] = {};

volatile static uint8\_t M[10][2] = {};

volatile static uint8\_t T97 = 0;

volatile static uint32\_t countT97 = 0;

volatile static uint32\_t T97reset = 0;

volatile static uint8\_t vaoT97 = 0;

volatile static const uint32\_t datT97 = 10;

volatile static uint8\_t checkT97 = 0;

void read\_Pin\_Input()

{

I0\_0 = HAL\_GPIO\_ReadPin(I0\_0\_PORT, I0\_0\_PIN);

I0\_1 = HAL\_GPIO\_ReadPin(I0\_1\_PORT, I0\_1\_PIN);

I0\_3 = HAL\_GPIO\_ReadPin(I0\_3\_PORT, I0\_3\_PIN);

}

void write\_Pin\_Output()

{

if (Q0\_0 >= 1)

{

HAL\_GPIO\_WritePin(Q0\_0\_PORT, Q0\_0\_PIN, ENABLE);

}

else

{

HAL\_GPIO\_WritePin(Q0\_0\_PORT, Q0\_0\_PIN, DISABLE);

}

}

void Main\_task(void \*param)

{

while (1)

{

read\_Pin\_Input();

/\*--------------NetWork 1 -----------\*/

Q0\_0 = (((!(I0\_0 \* (!I0\_1))) + (Q0\_0 \* (!T97))) \* (!I0\_3));

if (Q0\_0 > 0)

{

Q0\_0 = 1;

}

else

{

Q0\_0 = 0;

}

/\*--------------NetWork 2 -----------\*/

if (!T97reset)

{

vaoT97 = (Q0\_0);

T97;

if (vaoT97)

{

if (countT97 >= datT97)

T97 = 1;

if ((T97 == 0) && (checkT97 == 0))

{

xTimerStart(handle\_timerPLC[0],portMAX\_DELAY);

checkT97 = 1;

}

}

else

{

if (checkT97 == 1)

{

xTimerStop(handle\_timerPLC[0],portMAX\_DELAY);

checkT97 = 0;

}

T97 = 0;

countT97 = 0;

}

}

else

{

xTimerStop(handle\_timerPLC[0], portMAX\_DELAY);

countT97 = 0;

T97 = 0;

checkT97 = 0;

}

write\_Pin\_Output();

}

}

void TimerCallBack(TimerHandle\_t xTimer)

{

int id;

id = (uint32\_t)pvTimerGetTimerID(xTimer);

switch (id)

{

case 1:

countT97++;

break;

}

}

void initTimer(void)

{

handle\_timerPLC[0]=xTimerCreate("timerTONT97",pdMS\_TO\_TICKS(10),pdTRUE, (void \*)(0+1), TimerCallBack);

}

## Phụ lục 3. Chương trình ví dụ hình 3.8 cho file.h

#ifndef INC\_CSDL\_H\_

#define INC\_CSDL\_H\_

#include "main.h"

// Define I

#define IB0 I[0][0]

#define IDW0 I[0][0]

#define IW0 I[0][0]

#define I0\_0 I[0][0]

#define I0\_1 I[0][1]

#define I0\_2 I[0][2]

#define I0\_3 I[0][3]

#define I0\_4 I[0][4]

#define I0\_5 I[0][5]

#define I0\_6 I[0][6]

#define I0\_7 I[0][7]

#define IB1 I[1][0]

#define IDW1 I[1][0]

#define IW1 I[1][0]

#define I1\_0 I[1][0]

#define I1\_1 I[1][1]

#define I1\_2 I[1][2]

#define I1\_3 I[1][3]

#define I1\_4 I[1][4]

#define I1\_5 I[1][5]

#define I1\_6 I[1][6]

#define I1\_7 I[1][7]

// Define Q

#define QB0 Q[0][0]

#define QDW0 Q[0][0]

#define QW0 Q[0][0]

#define Q0\_0 Q[0][0]

#define Q0\_1 Q[0][1]

#define Q0\_2 Q[0][2]

#define Q0\_3 Q[0][3]

#define Q0\_4 Q[0][4]

#define Q0\_5 Q[0][5]

#define Q0\_6 Q[0][6]

#define Q0\_7 Q[0][7]

#define QB1 Q[1][0]

#define QDW1 Q[1][0]

#define QW1 Q[1][0]

#define Q1\_0 Q[1][0]

#define Q1\_1 Q[1][1]

#define Q1\_2 Q[1][2]

#define Q1\_3 Q[1][3]

#define Q1\_4 Q[1][4]

#define Q1\_5 Q[1][5]

#define Q1\_6 Q[1][6]

#define Q1\_7 Q[1][7]

// Define M

#define MB0 M[0][0]

#define MDW0 M[0][0]

#define MW0 M[0][0]

#define M0\_0 M[0][0]

#define M0\_1 M[0][1]

#define M0\_2 M[0][2]

#define M0\_3 M[0][3]

#define M0\_4 M[0][4]

#define M0\_5 M[0][5]

#define M0\_6 M[0][6]

#define M0\_7 M[0][7]

#define MB1 M[1][0]

#define MDW1 M[1][0]

#define MW1 M[1][0]

#define M1\_0 M[1][0]

#define M1\_1 M[1][1]

#define M1\_2 M[1][2]

#define M1\_3 M[1][3]

#define M1\_4 M[1][4]

#define M1\_5 M[1][5]

#define M1\_6 M[1][6]

#define M1\_7 M[1][7]

#define MB2 M[2][0]

#define MDW2 M[2][0]

#define MW2 M[2][0]

#define M2\_0 M[2][0]

#define M2\_1 M[2][1]

#define M2\_2 M[2][2]

#define M2\_3 M[2][3]

#define M2\_4 M[2][4]

#define M2\_5 M[2][5]

#define M2\_6 M[2][6]

#define M2\_7 M[2][7]

// Define Input Pin

#define I0\_0\_PIN GPIO\_PIN\_

#define I0\_1\_PIN GPIO\_PIN\_

#define I0\_3\_PIN GPIO\_PIN\_

// Define Input Port

#define I0\_0\_PORT GPIO

#define I0\_1\_PORT GPIO

#define I0\_3\_PORT GPIO

//efine Q0\_0\_PIN GPIO\_PIN\_

// Define Output pin

#dDefine Output Port

#define Q0\_0\_PORT GPIO

void read\_Pin\_Input(void);

void write\_Pin\_Output(void);

void Main\_task(void \*param);

void TimerCallBack(TimerHandle\_t xTimer);

void initTimer(void);

extern TimerHandle\_t handle\_timerPLC[1];

#endif /\*INC\_CSDL\_H\_\*/

## Phụ lục 4 : Các chương trình các lệnh trong PLC

1. Timer TON

if (!T97reset)

{

vaoT97 = (I0\_0);

T97;

if (vaoT97)

{

if (countT97 >= datT97)

T97 = 1;

if ((T97 == 0) && (checkT97 == 0))

{

xTimerStart(handle\_timerPLC[0],portMAX\_DELAY);

checkT97 = 1;

}

}

else

{

if (checkT97 == 1)

{

xTimerStop(handle\_timerPLC[0],portMAX\_DELAY);

checkT97 = 0;

}

T97 = 0;

countT97 = 0;

}

}

else

{

xTimerStop(handle\_timerPLC[0], portMAX\_DELAY);

countT97 = 0;

T97 = 0;

checkT97 = 0;

}

1. Timer TOF

if (!T37reset)

{

vaoT37 = (I0\_1);

T37;

if (vaoT37)

{

countT37 = 0;

checkT37On = 1;

T37 = 1;

if (checkT37Off == 1 || startT37 == 1)

{

xTimerStop(handle\_timerPLC[1],portMAX\_DELAY);

checkT37Off = 0;

startT37 = 0

}

}

else

{

checkT37Off = 1;

if (checkT37On == 1)

{

xTimerStart(handle\_timerPLC[1],portMAX\_DELAY);

checkT37On = 0;

}

if (countT37 >= datT37)

{

T37 = 0;

}

}

}

else

{

xTimerStop(handle\_timerPLC[1], portMAX\_DELAY);

countT37 = 0;

T37 = 0;

}

1. Timer TONR

if (!T1reset)

{

vaoT1 = (I0\_2);

T1;

if (vaoT1)

{

checkT1On = 1;

if (checkT1Off == 1 || startT1 == 1)

{

xTimerStart(handle\_timerPLC[2],portMAX\_DELAY);

checkT1Off = 0;

startT1 = 0;

}

if ((countT1 - checkCountT1) >= datT1)

{

T1 = 1;

checkCountT1 = 0;

xTimerStop(handle\_timer[2], portMAX\_DELAY);

countT1 = 0;

}

}

else

{

checkT1Off = 1;

if ((!countT1) && (checkT1On == 1))

checkCountT1 = countT1;

checkT1On = 0;

}

}

else

{

xTimerStop(handle\_timerPLC[2], portMAX\_DELAY);

countT1 = 0;

T1 = 0;

checkT1Off = 1

}

1. Counter CTU

vaoC1 = (I0\_3);

resetC1 = (I0\_4);

if (resetC1)

{

countC1 = 0;

C1 = 0;

startC1 = 1;

}

else

{

if (vaoC1)

{

if (((checkC1 == 1) || (startC1 == 1)) && (countC1 <=4294967295))

{

countC1++;

startC1 = 0;

}

if (countC1 >= datC1)

{

C1 = 1;

}

}

else

{

checkC1 = 1;

}

}

1. Counter CTD

vaoC2 = (I0\_5);

resetC2 = (I0\_6);

if (resetC2)

{

nhoC2 = datC2;

countC2 = datC2;

C2 = 0;

}

else

{

if (vaoC2)

{

if (((checkC2 == 1) || (startC2 == 1)) && (nhoC2 > 0))

{

nhoC2--;

countC2 = nhoC2;

startC2 = 0;

}

if (countC2 <= 0)

{

C2 = 1;

}

}

else

{

checkC2 = 1;

}

}

1. Counter CTUD

tangC3 = (I0\_7);

giamC3 = (I1\_0);

resetC3 = (I1\_1);

if (resetC3)

{

nhoC3 = 0;

countC3 = 0;

C3 = 0;

}

else

{

if (tangC3)

{

if (((checkC3tang == 1) || (startC3tang == 1)) && (countC3 <= 4294967295))

{

nhoC3++;

countC3 = nhoC3;

startC3tang = 0;

}

}

else

{

checkC3tang = 1;

}

if (giamC3)

{

if (((checkC3giam == 1) || (startC3giam == 1)) && (nhoC3 > 0))

{

nhoC3--;

countC3 = nhoC3;

startC3giam = 0;

}

}

else

{

checkC3giam = 1;

}

if (countC3 >= datC3)

{

C3 = 1;

}

else

{

C3 = 0;

}

}

1. Lệnh so sánh

volatile uint8\_t tempT97\_0 = 0;

if (countT97 >= 2)

{

tempT97\_0 = 1;

}

Q0\_0 = (tempT97\_0);

if (Q0\_0 > 0)

{

Q0\_0 = 1;

}

else

{

Q0\_0 = 0;

}

1. Lệnh MOV

( ví dụ mov 1 byte)

if ((I1\_2))

(memcpy(&MB0, &IB0, 1));

1. Lệnh sườn lên EU

volatile uint8\_t I1\_2sl0 = I1\_2;

volatile static uint8\_t checkEU0 = 0;

if (!(I1\_2))

{

checkEU0 = 0;

}

if ((!checkEU0) && (I1\_2))

{

checkEU0 = 0;

I1\_2sl0 = 1;

}

Q0\_2 = (I1\_2sl0);

if (Q0\_2 > 0)

{

Q0\_2 = 1;

}

else

{

Q0\_2 = 0;

}

1. Lệnh sườn xuống

volatile uint8\_t I1\_3sx0 = I1\_3;

volatile static uint8\_t checkEU0 = 1;

if (I1\_3)

{

checkEU0 = 0;

}

if ((!checkEU0) && (!(I1\_3)))

{

checkEU0 = 0;

I1\_3sx0 = 1;

}

Q0\_3 = (I1\_3sx0);

if (Q0\_3 > 0)

{

Q0\_3 = 1;

}

else

{

Q0\_3 = 0;

}

**ĐÁNH GIÁ QUYỂN ĐỒ ÁN TỐT NGHIỆP**

(Dùng cho giảng viên hướng dẫn)

Tên giảng viên đánh giá:

Họ và tên sinh viên: MSSV:

Tên đồ án:

**Chọn các mức điểm phù hợp cho sinh viên trình bày theo các tiêu chí dưới đây:**

Rất kém (1); Kém (2); Đạt (3); Giỏi (4); Xuất sắc (5)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Có sự kết hợp giữa lý thuyết và thực hành (20)** | | | | | | |
| 1 | Nêu rõ tính cấp thiết và quan trọng của đề tài, các vấn đề và các giả thuyết (bao gồm mục đích và tính phù hợp) cũng như phạm vi ứng dụng của đồ án | 1 | 2 | 3 | 4 | 5 |
| 2 | Cập nhật kết quả nghiên cứu gần đây nhất (trong nước/quốc tế) | 1 | 2 | 3 | 4 | 5 |
| 3 | Nêu rõ và chi tiết phương pháp nghiên cứu/giải quyết vấn đề | 1 | 2 | 3 | 4 | 5 |
| 4 | Có kết quả mô phỏng/thực nghiệm và trình bày rõ ràng kết quả đạt được | 1 | 2 | 3 | 4 | 5 |
| **Có khả năng phân tích và đánh giá kết quả (15)** | | | | | | |
| 5 | Kế hoạch làm việc rõ ràng bao gồm mục tiêu và phương pháp thực hiện dựa trên kết quả nghiên cứu lý thuyết một cách có hệ thống | 1 | 2 | 3 | 4 | 5 |
| 6 | Kết quả được trình bày một cách logic và dễ hiểu, tất cả kết quả đều được phân tích và đánh giá thỏa đáng | 1 | 2 | 3 | 4 | 5 |
| 7 | Trong phần kết luận, tác giả chỉ rõ sự khác biệt (nếu có) giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai | 1 | 2 | 3 | 4 | 5 |
| **Kỹ năng viết quyển đồ án (10)** | | | | | | |
| 8 | Đồ án trình bày đúng mẫu quy định với cấu trúc các chương logic và đẹp mắt (bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến; căn lề thống nhất, có dấu cách sau dấu chấm, dấu phảy v.v.), có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn đúng quy định | 1 | 2 | 3 | 4 | 5 |
| 9 | Kỹ năng viết xuất sắc (cấu trúc câu chuẩn, văn phong khoa học, lập luận logic và có cơ sở, từ vựng sử dụng phù hợp v.v.) | 1 | 2 | 3 | 4 | 5 |
| **Thành tựu nghiên cứu khoa học (5)** *(chọn 1 trong 3 trường hợp)* | | | | | | |
| 10a | Có bài báo khoa học được đăng hoặc chấp nhận đăng/Đạt giải SVNCKH giải 3 cấp Viện trở lên/Có giải thưởng khoa học (quốc tế hoặc trong nước) từ giải 3 trở lên/Có đăng ký bằng phát minh, sáng chế | 5 | | | | |
| 10b | Được báo cáo tại hội đồng cấp Viện trong hội nghị SVNCKH nhưng không đạt giải từ giải 3 trở lên/Đạt giải khuyến khích trong các kỳ thi quốc gia và quốc tế khác về chuyên ngành (VD: TI contest) | 2 | | | | |
| 10c | Không có thành tích về nghiên cứu khoa học | 0 | | | | |
| **Điểm tổng** | | **/50** | | | | |
| **Điểm tổng quy đổi về thang 10** | |  | | | | |

***Nhận xét khác*** *(về thái độ và tinh thần làm việc của sinh viên)*

|  |  |
| --- | --- |
|  | Ngày: … / … / 20…  **Người nhận xét**  (Ký và ghi rõ họ tên) |

**ĐÁNH GIÁ QUYỂN ĐỒ ÁN TỐT NGHIỆP**

(Dùng cho cán bộ phản biện)

Giảng viên đánh giá:

Họ và tên sinh viên: MSSV:

Tên đồ án:

**Chọn các mức điểm phù hợp cho sinh viên trình bày theo các tiêu chí dưới đây:**

Rất kém (1); Kém (2); Đạt (3); Giỏi (4); Xuất sắc (5)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Có sự kết hợp giữa lý thuyết và thực hành (20)** | | | | | | |
| 1 | Nêu rõ tính cấp thiết và quan trọng của đề tài, các vấn đề và các giả thuyết (bao gồm mục đích và tính phù hợp) cũng như phạm vi ứng dụng của đồ án | 1 | 2 | 3 | 4 | 5 |
| 2 | Cập nhật kết quả nghiên cứu gần đây nhất (trong nước/quốc tế) | 1 | 2 | 3 | 4 | 5 |
| 3 | Nêu rõ và chi tiết phương pháp nghiên cứu/giải quyết vấn đề | 1 | 2 | 3 | 4 | 5 |
| 4 | Có kết quả mô phỏng/thực nghiệm và trình bày rõ ràng kết quả đạt được | 1 | 2 | 3 | 4 | 5 |
| **Có khả năng phân tích và đánh giá kết quả (15)** | | | | | | |
| 5 | Kế hoạch làm việc rõ ràng bao gồm mục tiêu và phương pháp thực hiện dựa trên kết quả nghiên cứu lý thuyết một cách có hệ thống | 1 | 2 | 3 | 4 | 5 |
| 6 | Kết quả được trình bày một cách logic và dễ hiểu, tất cả kết quả đều được phân tích và đánh giá thỏa đáng | 1 | 2 | 3 | 4 | 5 |
| 7 | Trong phần kết luận, tác giả chỉ rõ sự khác biệt (nếu có) giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai | 1 | 2 | 3 | 4 | 5 |
| **Kỹ năng viết quyển đồ án (10)** | | | | | | |
| 8 | Đồ án trình bày đúng mẫu quy định với cấu trúc các chương logic và đẹp mắt (bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến; căn lề thống nhất, có dấu cách sau dấu chấm, dấu phảy v.v.), có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn đúng quy định | 1 | 2 | 3 | 4 | 5 |
| 9 | Kỹ năng viết xuất sắc (cấu trúc câu chuẩn, văn phong khoa học, lập luận logic và có cơ sở, từ vựng sử dụng phù hợp v.v.) | 1 | 2 | 3 | 4 | 5 |
| **Thành tựu nghiên cứu khoa học (5)** *(chọn 1 trong 3 trường hợp)* | | | | | | |
| 10a | Có bài báo khoa học được đăng hoặc chấp nhận đăng/Đạt giải SVNCKH giải 3 cấp Viện trở lên/Có giải thưởng khoa học (quốc tế hoặc trong nước) từ giải 3 trở lên/Có đăng ký bằng phát minh, sáng chế | 5 | | | | |
| 10b | Được báo cáo tại hội đồng cấp Viện trong hội nghị SVNCKH nhưng không đạt giải từ giải 3 trở lên/Đạt giải khuyến khích trong các kỳ thi quốc gia và quốc tế khác về chuyên ngành (VD: TI contest) | 2 | | | | |
| 10c | Không có thành tích về nghiên cứu khoa học | 0 | | | | |
| **Điểm tổng** | | **/50** | | | | |
| **Điểm tổng quy đổi về thang 10** | |  | | | | |

***Nhận xét khác của cán bộ phản biện***

|  |  |
| --- | --- |
|  | Ngày: … / … / 20…  **Người nhận xét**  (Ký và ghi rõ họ tên) |