Міністерство освіти і науки України

Луцький національний технічний університет

(повне найменування вищого навчального закладу)

Факультет транспорту та механічної інженерії

(повне найменування факультету)

Кафедра прикладної механіки та мехатроніки

(повна найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

РОЗРОБКА УНІВЕРСАЛЬНОЇ СИСТЕМИ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ СКЛАДНИХ ДИНАМІЧНИХ ОБ'ЄКТІВ

спеціальність 131 Прикладна механіка
(шифр і назва спеціальності)
освітня програма «Прикладна механіка»
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи ІМм-21
Шпиняк Андрій Юрійович

_____
(підпис)

Керівник:
д.т.н., професор
Повстяной Олександр Юрійович

_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«___» _____ 20__ р.
Гарант освітньої програми:
к.т.н., доцент
Четвержук Тарас Іванович

_____
(підпис)

Луцьк – 2024 року

АНОТАЦІЯ

Шпиняк А.Ю. Розробка універсальної система імітаційного моделювання складних динамічних об’єктів. Рукопис.

Кваліфікаційна робота магістра ОП «Прикладна механіка» спеціальності 131 Прикладна механіка. Луцький національний технічний університет. Луцьк, 2024.

Кваліфікаційна робота магістра складається із вступу, 4 розділів, висновків, списку використаних джерел.

Метою кваліфікаційної роботи є розробка універсальної системи імітаційного моделювання складних динамічних об’єктів зі зручним інтерфейсом.

Для досягнення мети в кваліфікаційні роботі поставлені такі задачі:

— провести опис програмно-апаратного комплексу та дати характеристику її функціям;

— розробити функціональну схему пристрою для вводу-виводу інформації;

— розробити принципову схему пристрою;

— здійснити огляд систем реального часу;

— розробити програмну частину з розробкою користувацького інтерфейсу

— дослідити роботу програмного забезпечення на надійність.

Організація і планування виробництва має сприяти розвитку економічного підходу до вирішення технічних завдань.

При розробці складних механічних, технічних та мехатронних систем перед розробниками виникає ряд завдань, знайти розв'язання яких аналітично складно чи неможливо. У більшості випадків розробник зацікавлений у наявності моделі системи, що розробляється, працювати з якою йому було б простіше, ніж із реальним об'єктом. Тому перед розробником виникає потреба у моделюванні систем автоматичного управління виробництвом.

Ключові слова: моделювання, динамічний об’єкт, функціональна схема, мехатронна система, технічне завдання.

ABSTRACT

Shpynak A.Yu. Development of a universal simulation modeling system for complex dynamic objects. Manuscript.

Master's qualification work OP "Applied Mechanics" specialty 131 Applied Mechanics. Lutsk National Technical University. Lutsk, 2024.

Master's qualification work consists of an introduction, 4 chapters, conclusions, list of sources used.

The purpose of the qualification work is to develop a universal simulation modeling system for complex dynamic objects with a convenient interface.

To achieve the goal, the following tasks are set in the qualification work:

— to describe the software and hardware complex and characterize its functions;

— to develop a functional diagram of the device for input-output information;

— to develop a schematic diagram of the device;

— to conduct an overview of real-time systems;

— develop the software part with the development of the user interface

— investigate the operation of the software for reliability.

The organization and planning of production should contribute to the development of an economic approach to solving technical problems.

When developing complex mechanical, technical and mechatronic systems, developers face a number of tasks, the solution of which is analytically difficult or impossible. In most cases, the developer is interested in having a model of the system being developed, which would be easier for him to work with than with a real object. Therefore, the developer needs to model automatic production control systems.

Keywords: modeling, dynamic object, functional diagram, mechatronic system, technical task.

ЗМІСТ

ВСТУП

Можливості використання мікропроцесорної техніки розширюються в міру вдосконалення та розробки нових технологій, нових алгоритмів автоматичної обробки цифрової інформації.

У сучасному виробництві особливого значення набуває розвиток автоматизованих систем управління технологічними процесами (АСУ ТП) та новітніх мехатронних систем. Впровадження автоматизованих систем управління в різні сфери господарської діяльності, і в першу чергу в проектування та моделювання, управління обладнанням та технологічними процесами, сприяє прискоренню усьому науково-технічного прогресу. Для вирішення цих завдань необхідно здійснити підготовку кваліфікованих фахівців-інженерів, які здатні створювати та обслуговувати сучасне промислове виробництво, що базується на техніці.

Організація і планування виробництва має сприяти розвитку економічного підходу до вирішення технічних завдань. Суть прискорення росту – всіляка інтенсифікація виробництва на основі НТП, знаходження ефективних форм управління, організації та стимулювання праці.

При розробці складних механічних, технічних та мехатронних систем перед розробниками виникає ряд завдань, знайти розв'язання яких аналітично складно чи неможливо. У більшості випадків розробник зацікавлений у наявності моделі системи, що розробляється, працювати з якою йому було б простіше, ніж із реальним об'єктом. Тому перед розробником виникає потреба у моделюванні систем автоматичного управління виробництвом.

Метою кваліфікаційної роботи є розробка універсальної системи імітаційного моделювання складних динамічних об’єктів зі зручним інтерфейсом.

Для досягнення мети в кваліфікаційні роботі поставлені такі *задачі*:

— провести опис програмно-апаратного комплексу та дати характеристику її функціям;

— розробити функціональну схему пристрою для вводу-виводу інформації;
— розробити принципову схему пристрою;
— здійснити огляд систем реального часу;
— розробити програмну частину з розробкою користувацького інтерфейсу
— дослідити роботу програмного забезпечення на надійність.

Об'єкт дослідження – математичні моделі та програмно-апаратного комплекс.

Предмет дослідження – універсальна система імітаційного моделювання складних динамічних об’єктів

Особистий внесок магістранта. Основні положення, результати, рекомендації, які є в наявності в кваліфікаційній роботі, отримані автором самостійно з консультуванням наукового керівника.

1 ТЕХНОЛОГІЧНА ЧАСТИНА

1.1 Опис програмно-апаратного комплексу

Враховуючи сучасні тенденції у конструюванні інтегрованих систем управління, що широко використовують мікроконтролери виникає необхідність використання програмних та апаратних засобів для синтезування, налагодження та тестування сучасних керуючих програм.

Отримані результати дозволяють здійснити швидку перевірку працездатності схеми управління та збільшити ефективність використання мікроконтролерів, до переваг яких належить гнучкість, потужність, різноманітність функцій роботи з пристроями, додаткові апаратні можливості.

Для забезпечення всієї повноти можливостей роботи з мікроконтролерами можна описати набір функцій комплексу налагодження:

- написання керуючих програм мовами високого рівня, а також спільне використання різних програмних модулів в одному проекті;
- компіляція програми в коди та покрокове налагодження з контролем стану портів;
- програмування пам'яті програм і даних мікроконтролерів різних типів;
- забезпечення номінальних параметрів функціонування мікроконтролерів у тестовій схемі;
- можливість швидко та надійно комутувати лінії всіх портів введення-виведення мікроконтролера;
- управління потоком виконання програми, використовуючи імітатори зовнішніх сигналів у вигляді датчиків;
- контроль виведення сигналів за допомогою світлових індикаторів;
- високошвидкісний інформаційний зв'язок між комп'ютером та мікроконтролером;

— програмне налаштування виведення сигналів з комп'ютера в мікроконтролер реальному часі;
— використання послідовного інтерфейсу зв'язку між мікроконтролером та мікропроцесорною системою;

Виходячи з перерахованих функцій апаратний склад комплексу налагодження має включати:

— схему програматорів з різним способом програмування;
— персональний комп'ютер із двома послідовними паралельними портами;
— процесорний модуль, який оснащений панеллю з нульовим зусиллям, схемою скидання, частотними елементами, роз'ємами;
— плату кнопкових датчиків;
— плату індикації зі світлодіодами та сегментними індикаторами;
— плату для динамічної індикації;
— плату інтерфейсу введення-виводу для зв'язку комп'ютера з модулем мікроконтролера і високошвидкісний виведення цифрових сигналів, трьох аналогових сигналів і введення цифрових сигналів.

1.1.1 Структура програмно-апаратного комплексу. Структурний блок – це сукупність елементів, які виконують одну або подібну функцію. Виходячи з цього склад системи, що розробляється, може бути представлений у вигляді наступних блоків:

ЦПМ – центральний процесорний модуль.

ПК – плата клавіатури.

ПІ - плата індикації.

КДПІ - плата динамічної індикації.

ІППВВ – плата цифрового та аналогового сигналу.

БПр – блок програматора.

БК – блок комутації.

БЖ – блок живлення.

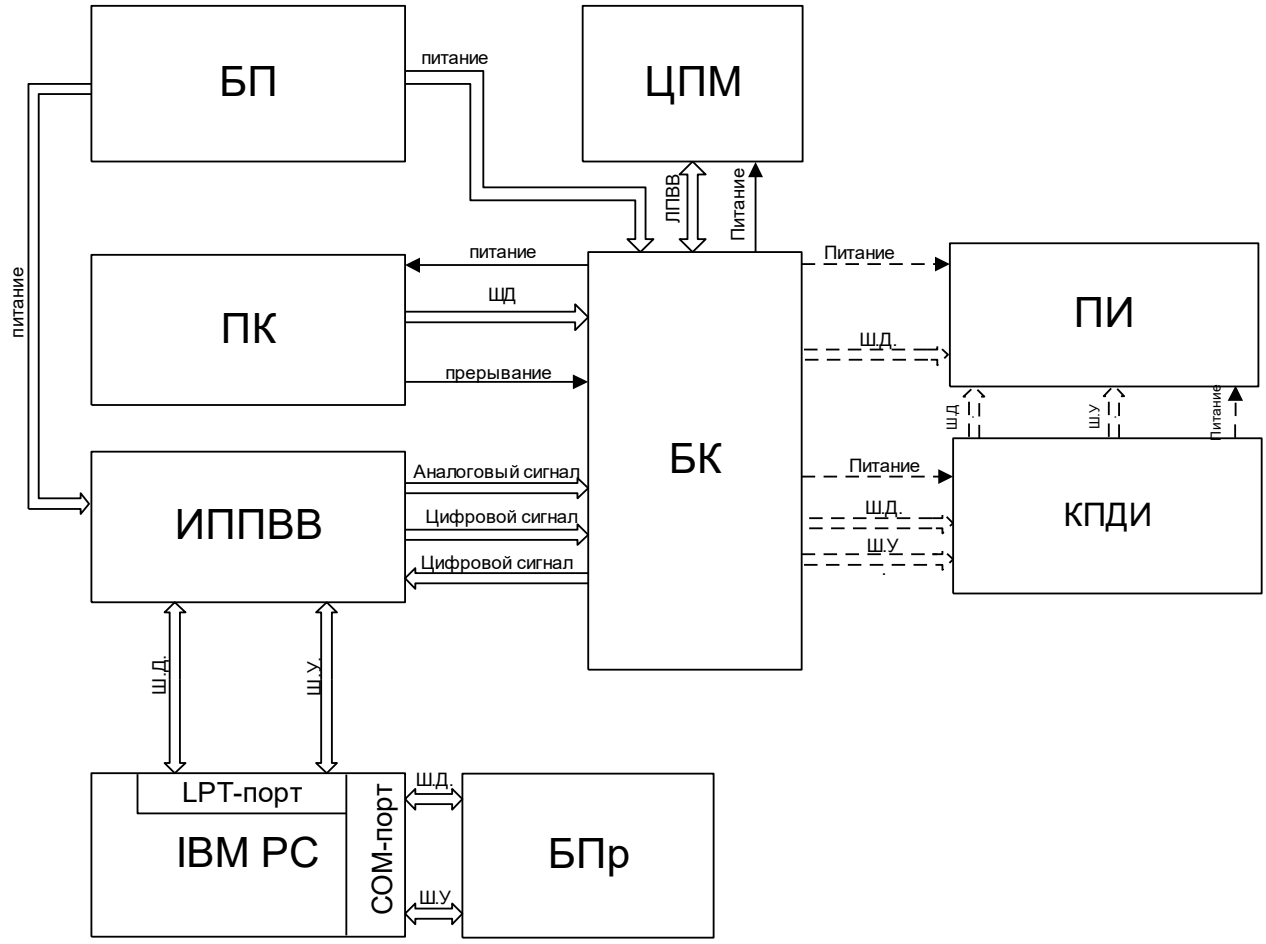

Рисунок 1.1 – Структурна схема програмно-апаратного комплексу

ЦПМ – центральний процесорний модуль, який складається з мікроконтролера, генератора тактових імпульсів, панелі під PIC – контролер з нульовим підсилювачем та пристрої «скидання».

До функцій цього блоку входить:

— забезпечення функціонування програми управління,

— подачі живлення на всі інші модулі

— формування синхросигналів.

ПК – плата клавіатури, складається з блоку кнопкових датчиків, пристрою гасіння брязкоту контактів та схеми формування сигналу переривання.

До функцій, що виконуються цим блоком, входять:

— введення необхідної інформації,

— захист від брязкоту контакту

— формування сигналу переривання.

ПІ – плата індикації складається з блоку сегментних індикаторів та набору світлодіодних індикаторів.

До функцій, що виконуються цим блоком, входить: відображення виведеної інформації. Реалізовано на трьох сегментних індикаторах HG1HG3, восьми світлодіодних індикаторів VD1VD8 та струмообмежувальних резисторів R1R32. Кожен індикатор підключений до окремого входу, через який на нього приходять сигнали керування.

КПДІ – плата динамічної індикації. Функції даного блоку полягають у перемиканні плати статичної індикації в динамічний режим шляхом почергового включення сегментних індикаторів. Він складається з двох вхідних і трьох вихідних роз'ємів, які з'єднанні провідниками, що розгалужуються.

ІППВВ - інтерфейсна плата паралельного введення / виводу. Вона складається з: 2-х шинних формувачів, дешифратора адреси, регістра адреси, логічної схеми управління записом/читання, регістрів введення та виведення, регістрів коду аналогового сигналу, цифро-аналогових перетворювачів та операційних підсилювачів.

1.1.2 Функції програмно-апаратного комплексу. У функції, які виконує блок входить: восьмиканальний виведення цифрового сигналу і трьохканальний виведення аналогового сигналу зі зміною його як за часом, так і по амплітуді з підвищенням його здатності навантаження, а також можливість зчитування цифрового сигналу по восьми каналах з PIC - контролера з подальшою передачею його на IBM PC через LPT – порт.

БК – блок комутації. Складається з набору роз'ємів, які з'єднані у певній послідовності для оптимальної комутації модулів.

Функція цього блоку полягає у забезпеченні різних варіантів комутації всіх модулів даного комплексу.

БПр – блок програматора. Він складається зі схеми управління та контролю процесу програмування та блоку адаптера.

У функції, які виконує цей блок входять: програмування широкого спектра PIC – контролерів, зчитування даних із пам'яті мікроконтролерів і їх порівняння з еталонними.

БЖ – блок живлення. Складається з тороїдального трансформатора, двох випрямляючих мостів, набором ємностей, які згладжують, і захисних запобіжників.

У функції, що виконуються блоком, входить – забезпечення модулів всього комплексу напругою живлення: +5В 2А, +15В 0,15А.

1.2 Пакети розширень Matlab, які використовуються при розробці проекту

Побачити в ряді додатків таких як, управління системою в реальному часі; проект DSP Real-Time Workshop генерує код мов C++ для блокових діаграм Simulink, що складаються з систем дискретного часу, безперервного часу, і гібридних систем. Real–Time Workshop – найкраща версія цього продукту.

Характеристики даної системи:

— значно швидший Target Language Compiler (TLC) генератор кодового процесу;

— профайлер TLC для налагодження програм TLC;

— нова ефективність у згенерованому коді включає покращену сигнальну пам'ять, що багаторазово використовується в процесі, усунений блок констант і параметр pooling.

Нова версія Real-Time Workshop Embedded Coder додана до версії RTW і значно підвищує ефективність Embedded Real-Time для виконання її роботи:

— інтерфейс користувача Real-Time Workshop покращено, включаючи також сторінкову конфігурацію параметра моделі;
— підтримка для окремих Simulink-блоків, також включено таблицю Look-Up з дуже ефективним згенерованим кодом;
— підтримка мети S-Функції для змінних налаштування кроку та регулювання параметрів;
— підтримка для матричних операцій є у більшості блоків;
— підтримка для фреймової обробки моделей Simulink для блоків зовнішньої підтримки режиму DSP;
— множина додаткових блоків типів для різного типу сигналу;
— автоматичне генерування S-функцій для вкладеного коду, що враховує підтвердження згенерованого коду Simulink;
— підтримка для генерації коду та модулів програм;
— підтримка підсистем для об'єктів даних Simulink;
— підтримка генерації файлів певного типу даних ASAP2.

Real-Time Workshop генерує два стилі коду. Один – придатний для швидкого макетрування та пов'язану з цими двома стилями коду.

1.2.1 Побудова математичної моделі. Real-Time Workshop генерує код алгоритму програми, як визначено нашою моделлю. Вони можуть вагатися від високорівневих алгоритмів обробки сигналів до драйверів пристрою низького рівня.

Real-Time також забезпечує час прогону інтерфейсу, який виконує згенерований код моделі. Час прогону інтерфейсу та коду моделі компілюються разом для створення програму моделі.

Діаграма (рисунок 1.2) показує високорівневий об'єктно-орієнтований вид програми.

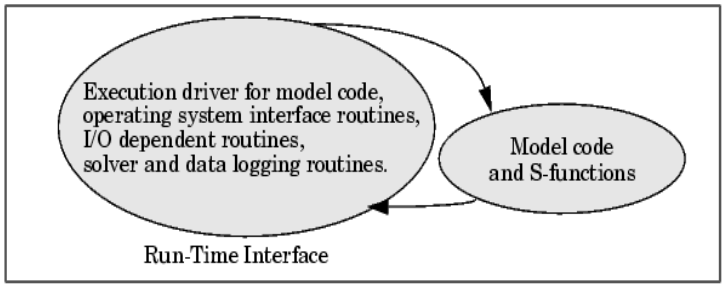

Рисунок 1.2 – Вид об’єктно-орієнтованої програми

Загалом концептуальне проектування драйвера виконання моделі не змінюється між швидким макетуванням і вкладеним стилем згенерованого коду. Наступні секції описують побудову моделі до singletasking та multitasking середовища для моделювання у реальному часі. Для більшості прикладів, multitasking середовище забезпечить найбільш ефективне виконання моделі (тобто, найшвидша норма відбору).

Поняття, які корисні для опису виконання моделі:

Ініціалізація - ініціалізувати час прогону інтерфейсу коду та коду моделі.

ModelOutputs – виклик всіх блоків у моделі, щоб було зупинено час у поточній точці в певному місці. ModelOutputs може бути зупинений в основних або незначних кроках часу. Протягом основних кроків часу вихід є даним імітаційним кроком часу.

ModelUpdate – виклик всіх блоків у вашій моделі, що був зупинений у поточній точці часу.

ModelDerivatives – виклик всіх блоків у вашій моделі, що мати безперервні стани і вміти коригувати їх похідні та викликається лише незначними кроками часу.

Псевдокод нижче показує виконання моделі для одногорозділу моделювання (nonreal-час).

```
main()
{
  Initialization
  While (time < final time)
    ModelOutputs     -- Major time step.
    LogTXY           -- Log time, states and root

 outports.
    ModelUpdate      -- Major time step.
    Integrate:       -- Integration in minor time step for
models
                     -- with continuous states.
      ModelDerivatives
      Do 0 or more:
        ModelOutputs
        ModelDerivatives
      EndDo (Number of iterations depends upon the solver.)
      Integrate derivatives to update continuous states.
    EndIntegrate
  EndWhile
  Shutdown
}
```

Спочатку відбувається фаза ініціалізації процесу. Вона складається з станів моделі, що ініціалізують, та установки двигуна для виконання. Модель потім виконує один крок за часу. Спочатку ModelOutputs виконується за час t, потім дані вводу / виводу реєструються в робочій області, а потім коригує дискретні стани.

Якщо ваша модель має безперервні стани, ModelDerivatives впроваджує безперервні похідні станів, щоб генерувати стан протягом часу.

Вони починають працювати неправильно, якщо використовуються макрос (ssIsSampleHit, або ssIsSpecialSampleHit), який перевіряє зупинку процесу.

Псевдокод нижче показує виконання моделі для multitasking моделювання (nonreal-час).

```
main()
{
  Initialization
  While (time < final time)
    ModelOutputs(tid=0)   -- Major time step.
    LogTXY                -- Log time, states, and root
outports.
    ModelUpdate(tid=1)    -- Major time step.
    For i=1:NumTids
      ModelOutputs(tid=i) -- Major time step.
      ModelUpdate(tid=i)  -- Major time step.
    EndFor
    Integrate       -- Integration in minor time step for
models
                    -- with continuous states.
      ModelDerivatives
      Do 0 or more:
        ModelOutputs(tid=0)
        ModelDerivatives
      EndDo (Number of iterations depends upon the solver.)
      Integrate derivatives to update continuous states.
    EndIntegrate
```

```
  EndWhile
  Shutdown
     }
```

Multitasking - дія є більш комплексною в порівнянні з singletasking виконанням, оскільки вихідні дані та функції корекції підрозділені ідентифікатором задачі (tid), який прописаний в ці функції, що звичайно ж і відбувається якщо не було ніякого права допуску в системі реального часу

Multitasking припускає, що всі завдання – це безліч базових показників.

Цикл виконання multitasking дуже подібний до того ж самого singletasking, за винятком використання ідентифікатора задачі (tid) аргументу на ModelOutputs і ModelUpdate.

Псевдокод показує виконання в одному розділі в реальному часі системи, де модель може прокомпілюватися на рівні переривання.

```
rtOneStep()
{
  Check for interrupt overflow
  Enable "rtOneStep" interrupt
  ModelOutputs    -- Major time step.
  LogTXY          -- Log time, states and root outports.
  ModelUpdate     -- Major time step.
  Integrate       --  Integration  in  minor  time  step  for
models
                  -- with continuous states.
     ModelDerivatives
     Do 0 or more

       ModelOutputs
       ModelDerivatives
     EndDo (Number of iterations depends upon the solver.)
```

```
    Integrate derivatives to update continuous states.
  EndIntegrate
}

main()
{
  Initialization (including installation of rtOneStep as an
    interrupt service routine, ISR, for a real-time clock).
  While(time < final time)
    Background task.
  EndWhile
  Mask interrupts (Disable rtOneStep from executing.)
  Complete any background tasks.
  Shutdown
}
```

Виконання задачі одного розділу в реальному часі дуже подібне до управління виконанням задачами в не реальному часі.

В інтервалі визначеному програмною базовою нормою відбору, переривання (ISR) захоплює завдання, показане на цьому тлі, щоб виконувати код моделі. Базова норма відбору є найшвидшим показником моделі. Якщо модель має блоки з безперервними значеннями, тоді розмір кроку інтеграції визначає базову норму відбору. Протягом цього переривання, диспетчер читає свої введення та обчислює виконання завдання. Потім контроль програми повертається у вихідний стан. Усі ці кроки мають відбутися перед наступним перериванням.

Наступний псевдокод показує, як модель працює у реальному часі системи.

```
rtOneStep()
{
  Check for interrupt overflow
  Enable "rtOneStep" interrupt
```

```
  ModelOutputs(tid=0)     -- Major time step.
  LogTXY                  -- Log time, states and root
outports.
  ModelUpdate(tid=0)      -- Major time step.
  Integrate               -- Integration in minor time step
for
                          -- models with continuous states.
     ModelDerivatives
     Do 0 or more:
       ModelOutputs(tid=0)
       ModelDerivatives
     EndDo (Number of iterations depends upon the solver.)
     Integrate derivatives and update continuous states.
  EndIntegrate
  For i=1:NumTasks
    If (hit in task i)

      ModelOutputs(tid=i)
      ModelUpdate(tid=i)
    EndIf
  EndFor
}
main()
{
  Initialization (including installation of rtOneStep as an
    interrupt service routine, ISR, for a real-time clock).
  While(time < final time)
    Background task.
  EndWhile
```

```
  Mask interrupts (Disable rtOneStep from executing.)
  Complete any background tasks.
  Shutdown
```

Робота моделі на рівні переривання в реальному часі дуже схожа на роботу моделі в попередньому singletasking середовищі.

Робота моделі в singletasking або multitasking середовищі при примітивному використанні операційної системи в реальному часі для виконання завдань дуже схожі на приклади розглянуті вище.

```
tSingleRate()
{

  MainLoop:
    If clockSem already "given", then error out due to
overflow.
    Wait on clockSem
    ModelOutputs            -- Major time step.
    LogTXY                  -- Log time, states and root
outports
    ModelUpdate             -- Major time step
    Integrate               --  Integration  in  minor  time
step
for                             -- models with continuous
states.
      ModelDeriviatives
      Do 0 or more:
        ModelOutputs
        ModelDerivatives
```

```
      EndDo (Number of iterations depends upon the solver.)
      Integrate derivatives to update continuous states.
    EndIntegrate
  EndMainLoop
}
main()
{
  Initialization
  Start/spawn task "tSingleRate".
  Start clock that does a "semGive" on a clockSem semaphore.
  Wait on "model-running" semaphore.
  Shutdown
```

У цьому середовищі модель працює, використовуючи операційну систему реального часу для управління з простими функціями. У цьому середовищі необхідно створити кілька функцій моделі tBaseRate і кілька функцій tSubRate, щоб запустити код моделі.

Функції (tBaseRate) має більший пріоритет. Функція subrate для tid = 1 має більш високий пріоритет ніж функція subrate для tid = 2 тощо.

Програми в реальному часі вимагають спеціальної синхронізації викликів функцій, тому потрібно перевірити чи виконується програмний код моделі перед тим, як відбудеться наступний виклик функції

Наступна діаграма ілюструє синхронізацію переривання (рис.1.3).

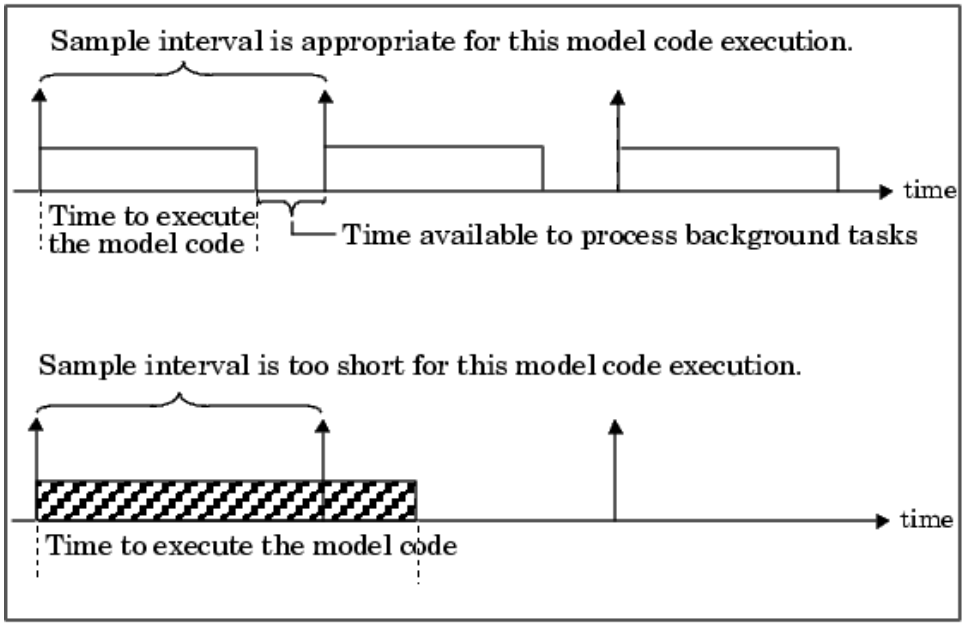

Рисунок 1.3 – Синхронізація переривання

Інтервал зразка має бути досить довгим, щоб допускати кодове виконання моделі між функціями викликів.

На рис.1.3 показано, що час між двома суміжними вертикальними стрілками і є інтервал зразка. Порожні коробки показують приклад програми, яка може завершитись за один крок у межах цього інтервалу. Сірий тон вказує, що трапляється, якщо інтервал зразка занадто короткий. Наступний виклик функції відбувається перед тим, як попередня функція буде завершена. Така синхронізація закінчується помилкою виконання.

Якщо програма в реальному часі призначена на тривалий запуск, то команда вимкнення ніколи не виконається.

Програма в реальному часі не може вимагати 100% виконання CPU. Це дозволяє запускати деякі функції протягом вільного часу.

Функції включення операцій подібно до запису даних у буфер або файл, що дає доступ, щоб програмувати дані незалежними даними. Це перевіряють інструментальні засоби, використовуючи зовнішній режим Simulink, щоб коригувати програмні параметри.

Важливо, щоб програма була здатна захопити функції у відповідний час, щоб гарантувати виконання коду моделі в реальному часі. Функції програми залежать від можливостей середовища в якому вона діє.

Зовнішній спосіб допускає зв'язки між блоковою діаграмою Simulink та окремою програмою, які побудовані із згенерованого коду програми. У цьому режимі програма в реальному часі функціонує як міжпроцесорний сервер зв'язку, що реагує на запити з Simulink. Дані, що реєструються під час виконання Single and Multitasking моделі.

Для того, щоб підсумовувати відмінності у зареєстрованих даних між singletasking та multitasking, можна відмітити наступні моменти:

- будь-який кореневий блок має певний час, який буде меншим, ніж найбільший приблизний час виконання завдання;

- будь-який блок у різних станах має час, який менший, ніж найшвидший час виконання завдань;

- будь-який блок у розблокованій підсистемі де є сигнал, що управляє включеним портом, буде повільнішим, ніж сума блоків у розблокованій підсистемі

Для перших двох випадків, навіть якщо зареєстровані величини інші між singletasking і multitasking, результати обчислень будуть схожими. Різниця між ними – точка часу, де зроблено реєстрація. Третій випадок (розблокована підсистема) закінчується затримкою, яка може бути помітна в режимі реального часу. Програмний каркас швидкого макетування забезпечує загальний програмний інтерфейс, який змінюється між визначеннями моделі.

Real-Time Workshop Embedded Coder забезпечує інший каркас, який називається як вкладений програмний каркас. Вкладений програмний каркас забезпечує оптимізований API, пристосований до вашої моделі. Коли ви використовуєте вкладений стиль згенерованого коду, ви повинні промоделювати ваш код так, щоб він виконувався у вашій вкладеній системі.

Зміст вищезгаданих функцій безпосередньо стосується блоків, що містяться в нашій моделі. Блок Simulink може бути узагальнено наступну структуру рівнянь.

Вихід у залежить від безперервного стану x_c, дискретного стану x_d і введення u. Кожен блок пише своє специфічне рівняння у відповідну секцію MdlOutput.

Дискретні значення x_d залежать від поточного значення та введення. Кожен блок має дискретне значення, яке може коригувати свій стан у MdlUpdate.

Похідні x залежать від поточного введення. Кожен блок, який має безперервні значення, забезпечує свої похідні дані на пристрій рішення в MdlDerivatives. Похідні використовуються вирішальним пристроєм для того, щоб впроваджувати безперервне рішення для виконання наступної величини.

Вихід у зазвичай записаний в блокову структуру введення / виведення. Кореневий рівень Outport виконує блокування запису на зовнішній структурі. Безперервний та дискретний стан збережено в структурі станів. Введення, u, може з'явитися з іншого виходу блоку, який розташований у блоковій структурі вводу/виводу або зовнішнього введення. Ці структури визначені у файлі model.h, що генерується Real-Time Workshop.

Блокова структура вводу / виводу – ця структура складається з усіх вихідних блочних сигналів (рис.1.4). Кількість вихідних блокових сигналів є сумою ширини вихідних портів даних віртуальних блоків в моделі. Якщо Ви активувати блокову оптимізацію введення / виводу, Simulink та Real-Time Workshop структура rtB зменшиться у розмірах:

— багаторазове використання даних у структурі rtB;

— отримання інших локальних змінних даних.

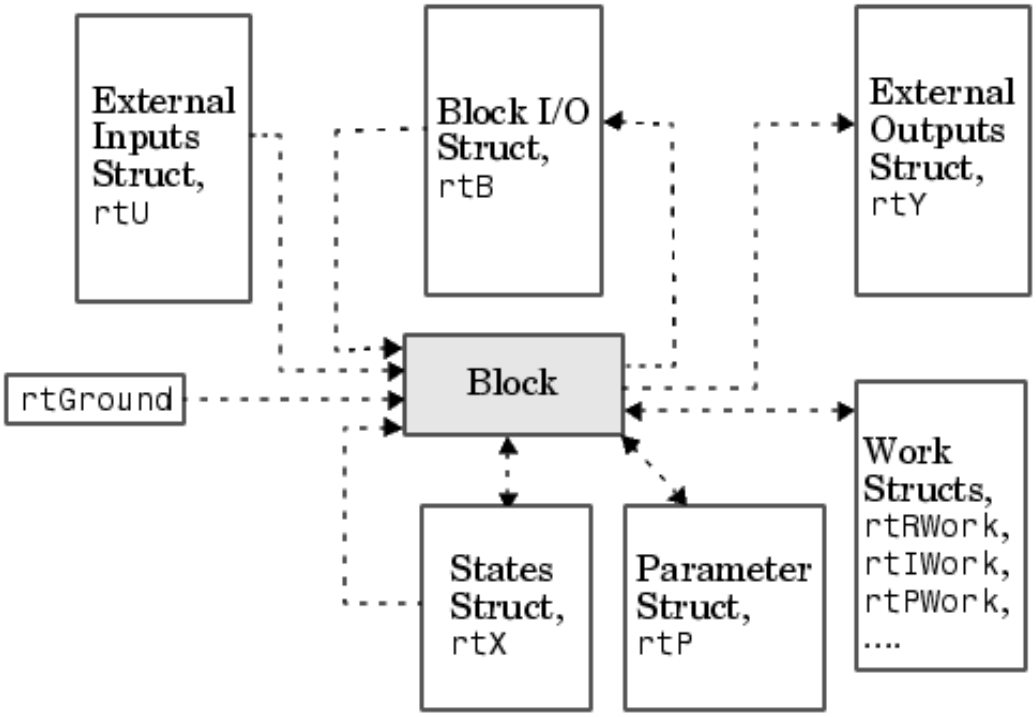

Рисунок 1.4 – Блокова структура ввід-вивід

Структурні імена області даних визначені блоковим вихідним сигнальним ім'ям або блоковим ім'ям та номером порту:

— Блокова Структура Стан (rtX) – структура станів, що містить безперервну та дискретну інформацію станів для будь-яких блоків у моделі. Структура станів має дві секції: початок – для безперервних станів, друга секція – для дискретних станів.

— Блокова Структура Параметрів (rtP) - структура параметрів містить усі блокові параметри, які можна змінити протягом виконання завдання (напр, параметр блоку Gain).

— Зовнішня структура вводів (rtU) – зовнішня структура вводів складається з усіх кореневих блокових сигналів рівня Inport. Поля назв визначені на виході, а вихідний сигнал залишений непоміченим.

— Зовнішня Структура Висновків (rtY) – зовнішня структура висновків складається з усіх кореневих блоків рівня Outport. Поля імен визначені коренево-блочними іменами рівня Outport у моделі.

1.2.2 Використання вкладених програмних функцій для побудови математичної моделі. Real-Time Workshop Embedded Coder and Ada Coder генерують такі функції:

- model_intialize – перед виконанням заданої моделі ви повинні її проініціалізувати.

Якщо відбулося одне виведення/зміна функції коду програми, тоді ви побачите: model_step(int_T tid) -- Містить висновок та змінюється код для всіх блоків у нашій моделі.

У іншому випадку ми побачимо:

model_output(int_T tid) – містить вихідний код для всіх блоків у вашій моделі.

model_update(int_T tid) – містить код корекції для всіх блоків у вашій моделі.

Якщо потрібно, завершити необхідну функцію, тоді: model_terminate – містить весь код вимкнення моделі і має бути викликано як частину системного вимкнення.

1.3 Покращений процес проектування з використанням інструментарію Real-Time Workshop

Real-Time Workshop забезпечує середовище розробки реального часу, яке характеризується прямим шляхом від системного проектування до апаратного впровадження. Тут можна скоротити цикли розробки та зменшити витрати на неї за допомогою Real-Time Workshop. Воно підтримує функціонування модулів динамічної системи і добре підходить для швидкого моделювання, додатків реального часу, рішень «під ключ» і прискорює імітацію. За допомогою Real-Time

Workshop можна швидко створити C-код для дискретних, безперервних і гібридних систем.

Формати включають швидке моделювання, розробку ресурсних систем або створення компонентів для повторного використання в Simulink-імітаціях. Браузер полегшує вибір форматів кодів та шаблонів робочих файлів.

Створений код оптимізований за швидкістю та використанням пам'яті і максимально спрощений. Цей формат коду вибирається з метою запуску згенерованого коду. Розподіл статичної пам'яті використовується у вбудованому форматі C-коду. Simulink SimStruct заміщається набагато меншим об'єктом реального часу, який містить необхідну інформацію про модель у часі. Відповідно до значної багаторівневої оптимізації S-функції були підключені з використанням відповідних файлів цільового мовного компілятора – Target Language Compiler (TLC). Системи безперервного часу не підтримуються вбудованим форматом C-коду.

Ідеально підходящий для швидкого моделювання код реального часу генерує код, який досить швидкий та гнучкий для дозволу перетворення на площині параметрів моделі. Формат коду підтримує дискретні та безперервні в часі моделі, використовуючи Simulink C MEX S-функції. З форматом коду реального часу розподіл пам'яті визначається статично під час компіляції.

Цей формат коду аналогічний формату коду реального часу. Початкова різниця в тому, що формат коду реального часу malloc визначає пам'ять динамічно, щоб дозволити багаторазову інсталяцію тієї ж моделі, тому щоразу вводиться унікальний набір даних. Багато моделей можуть поєднуватися в одну виконувану без потенційного конфлікту імен.

При розробці вбудованих систем керування або проектуванні DSP пристроїв, Real-Time Workshop і Stateflow Coder можуть бути використані для генерації багатоцільових вбудованих кодів. Якщо аналіз та імітація проекту закінчено, програми можуть бути адаптовані до будь-якого вбудованого обладнання

реального часу. Це дає можливість розробити більшу частину додатків, що вбудовуються.

Багато цільових пристроїв підтримуються Real-Time Workshop. Вони включають замовні пристрої, що вбудовуються, які підтримують VME, CPCI, ISA і PC-104 архітектури. Вони також підтримують вбудовані DSP, PC та системи на основі мікропроцесорів.

Замовні вбудовані, VME, CPCI – Tornado / Vx Works від Wind River Systems забезпечує надійне вбудовуване середовище, що оперує в реальному часі для популярних мікропроцесорних архітектур. Tornado ідеально підходить для керуючих та комунікаційних додатків реального часу.

Вбудовані PC – Real-Time Windows Target компанії MathWorks – це цільове обладнання реального часу чудово підходить для швидкого настільного моделювання. Він також ідеальний для використання в університетських лабораторіях для підтримки недорогих експериментів реального часу. Це цільове обладнання підтримує багато відомих інтерфейсних плат. Згенерований С-код незалежний від цільового пристрою і може бути використаний з практично будь-яким комерційним обладнанням реального часу, а також комерційно доступними операційними системами.

Швидке моделювання для цифрової обробки та керування – dSPACE Control Development System компанії dSPACE GmbH. Середовище розробки для швидкого моделювання управління та імітації в автомобільному, аерокосмічному та комунікаційному проектуванні. ADI Real-Time Station компанії Applied Dynamics International: Цільове обладнання реального часу для автомобільних та аерокосмічних застосувань. Pi AutoSim компанії Pi Technology Ltd.: середовище інтегрованого проектування та швидкого моделювання, у тому числі для автомобільних систем керування.

Освітні цільові пристрої. Quanser Consulting забезпечує керуючі експерименти світового класу, які можуть безпосередньо адаптовані за допомогою Real-Time

Workshop. TeQuipment також забезпечує керуючі експерименти, які змінюються від 1 до 3 DOF. Educational Control Products і Feedback Instruments також забезпечують керуючі та DSP експерименти стосовно проектування керуючих систем та тренінгу на них.

1.4 Функції та завдання програмного забезпечення

Метою випускної роботи було написання програмного модуля, що реалізовує роботу віртуальної системи управління (СУ). У системі управління всі реалізовані функції об'єктом передають і приймають відгуки від “віртуального” об'єкта. "Віртуальний" об'єкт реалізований у вигляді окремого програмного модуля. У ньому прошито математичну модель об'єкта, яка реалізована в пакеті Simulink і згодом експортована в машинний код.

Однією з основних переваг цієї роботи є відкритий інтерфейс підключення додаткових модулів, у яких можуть бути реалізовані різні види віртуальних об'єктів. Недоліком є те, що процес ручного кодування неминучий, але програмні блоки стандартні для більшості моделей, і, в принципі, процес написання додаткових модулів доступний навіть неспеціалісту, не знайомому з мовою С, оскільки блоки коду, що додаються у вихідний код, згенерований Real Time Workshop стандартні, так як і стандартні місця для їх додавання.

У практиці написання систем управління реальними об'єктами існує одне слабке місце, а саме написання програмного коду, що реалізує функціональну програмну модель, адекватної реальної фізичної моделі.

Існують вже розроблені програмні функції, що реалізують алгоритми функціонування математичних блоків. Але проблема в тому, що досить складна модель, наприклад, з зворотними зв'язками, що перехрещуються, практично не підлягає переписуванню в програмний код. Існують уже готові реалізації подібних

завдань, наприклад, у тому ж Simulink, але ці розробки є результатом роботи програмістів.

У пакеті Simulink реалізовано можливість автоматичної генерації програмного коду, що реалізує практично будь-яку математичну модель, причому це зроблено так, що програміст має справу тільки з високорівневими функціями.

До функцій програмного забезпечення можна віднести:

— можливість запуску виконання програмних модулів, які реалізують різноманітні математичні моделі об'єктів управління;
— передача даних між інтерфейсом користувача та програмним модулем;
— візуалізація отриманих даних у доступному для оператора вигляді;
— можливість задавати різні вхідні значення для моделей;
— можливість збереження отриманих результатів у файл;
— можливість подальшого відкриття заздалегідь збережених результатів без запуску відповідної моделі;
— можливість підключення додаткових програмних модулів без перекомпілювання інтерфейсу користувача;

До завдань програмного забезпечення можна віднести:

— реалізація можливості моделювати роботу системи керування без підключення до реального об'єкта
— можливість навчання студентів за принципами написання систем управління з використанням моделей у пакеті Simulink.

2 КОНСТРУКТОРСЬКА ЧАСТИНА

2.1 Функціональна схема пристрою введення-виведення інформації

У функції, які виконуються блоками входять: виведення цифрового сигналу по 8 каналам і трьох канальний виведення аналогового сигналу зі зміною його як за часом так і по амплітуді з підвищенням його здатності навантаження, а також можливості зчитування цифрового сигналу з периферійного пристрою (Pic - контролеру) з подальшою передачею його на PC.

Блок складається з: двох шинних формувачів, які реалізовані на мікросхемі КР580ВА86. Дана мікросхема підвищує здатність навантаження шини управління і шини даних, а також логічної схеми управління записом / читання зібраної на мікросхемах логіки К555ЛЕ4 та К555ЛА3. Дана схема керує та синхронізує процеси читання / запису.

Реєстр адреси реалізований на буферному регістрі КР580 ІР82. До її складу входить регістр-заскочка зі статичним синхровходом STB (strobe) і тристабільний буфер. Дана мікросхема передає код пристрою, який необхідно вибрати дешифратору адреси.

Для вибору необхідних пристроїв використовують дешифратор адреси, якийи реалізований на мікросхемі К155ІД7. Вхід дешифратора адреси складається із трьох ліній даних молодших розрядів D0-D2.

Його призначення є видача:

1. L – сигналу на лінію вибору пристрою регістра коду аналогового сигналу 1, коли дані молодших розрядів міститься 0002.

2. L – сигналу на лінію вибору пристрою регістра коду аналогового сигналу 2, коду даних молодших розрядів міститься 0012.

3. Вихідного L – сигналу на лінію вибору регістра коду аналогового сигналу 3, коду даних молодших розрядів міститься 0202.

4. L – сигналу на лінію вибору пристрою регістра виведення цифрового сигналу, коду даних молодших розрядів міститься 0112.

5. L – сигналу на лінію вибору пристрою регістра цифрового введення, коду даних молодших розрядів міститься 1002.

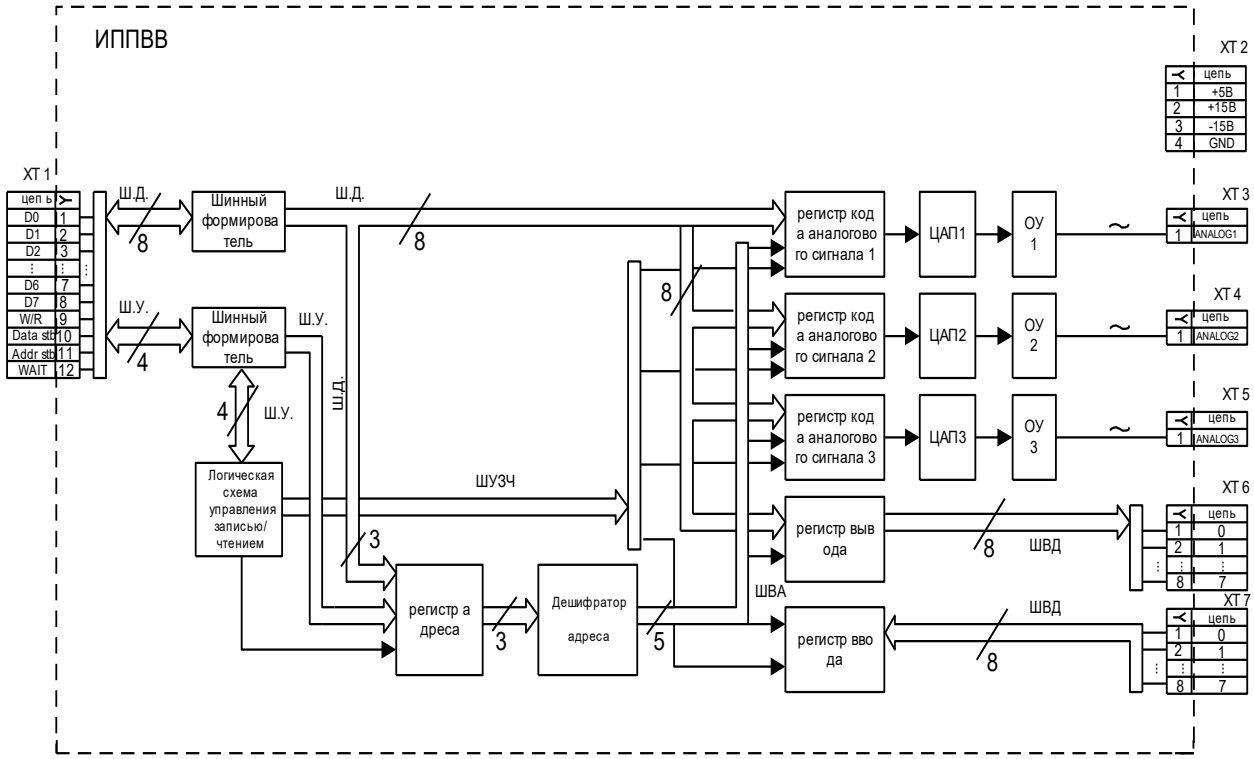

Рисунок 2.1 – Функціональна схема пристрою введення-виведення інформації

Реєстр коду аналогового сигналу реалізуються на мікросхемах КР580ІР82. До її складу входить регістр-заскочка зі статичним синхровходом STB (strobe) і тристабільний буфер. Даний буферний регістр по синхросигналу STB=1 приймає код аналогового сигналу, якщо керуючий сигнал активний, дані регістру передаються на вихід мікросхеми.

Регістра виведення цифрового сигналу та регістра цифрового введення також реалізованих на мікросхемах КР580ІР82.

Цифро-аналогових перетворювачів реалізуються на мікросхемах КР5872ПА1. Мікросхема являє собою набір універсальних функціональних елементів для

побудови 10 розрядних ЦАП. Здійснюють перетворення вхідного двійкового паралельного цифрового коду у вихідний струм, пропорційний значенню коду та опорної напруги. Мають можливість реалізації повного дво- і чотириразового множення сигналів, малою споживаною потужністю. Виконані на компліментарних транзисторах із полікремневими затворами та полікремневих прецизійних резисторах. Операційний підсилювач виконує функцію перетворювача вихідного струму на напругу ЦАП.

2.2 Розробка принципової схеми пристрою

Основою центральної процесорної плати є мікроконтролер серії PIC16FXX (рис.2.2).

Для побудови процесора на цій БІС потрібен блок синхронізації та блок скидання мікроконтролера.

Блок синхронізації реалізований на кварцовому резонаторі РВ – 20/4А 20 МГц та двох ємностях по 30 pf. Цей блок формує синхроімпульси, що надходять на входи мікроконтролера OSC1 та OSC2.

Блок скидання реалізований на RC-ланцюжку та дійснює тимчасову затримку низького рівня на вході MCLR.

Кожен порт PIC-контролера виведений окремий роз'єм, до вхідних роз'ємів підведено живлення.

При включенні тумблера SA10 (рис.2.2) запалюється світлодіодний індикатор VD12,VD1 (відповідно), що сигналізує про наявність живлення напруги на платі центрального модуля Напруга надходить на виведення мікроконтролера Vdd+5V та Vss-5V. І одночасно надходить на вихідні роз'єми для живлення підключених модулів.

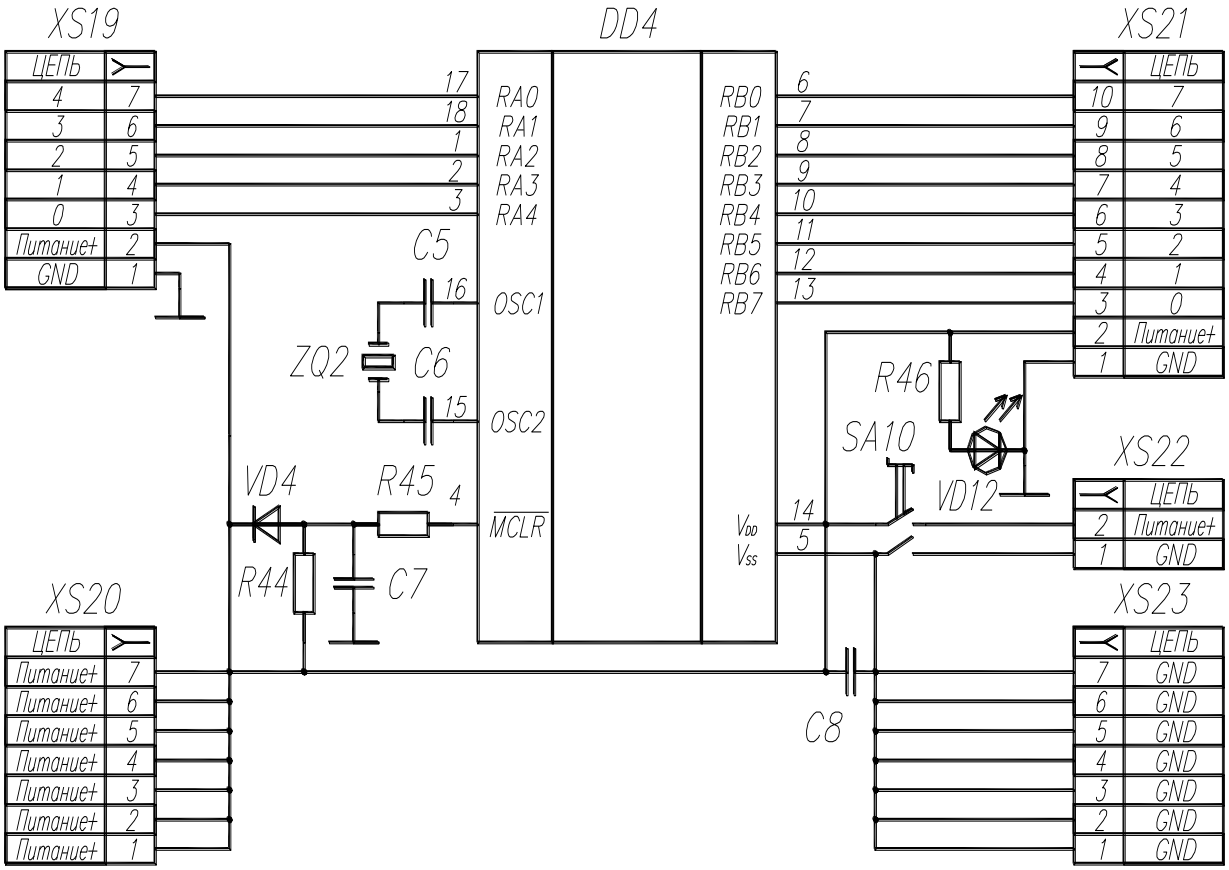

Рисунок 2.2 – Принципова схема для центрального процесорного модуля для мікроконтролерів серії PIC16FXX

Встановлюємо на шину даних код пристрою, який буде активізовано (три молодші розряди 0002). Сигнал читання / запис встановлюється у положення «0». На лінії Wait встановлюється низький рівень та дозволяємо початок циклу передачі адреси. За сигналом Add str:0 і =0 на вході STB, регістра адреси, з'являється високий рівень. Дані записуються в регістр, керуючий сигнал OE активний, дані регістра надходять на введення дешифратора адреси, де код дешифрується, і в залежності від заданого коду (0002) на заданому з виходів дешифратора (CS0) встановлюється низький рівень.

Цикл передачі адреси триває до тих пір, поки сигнал Wait не дорівнюватиме «1».

Поміщаємо на шину даних D0-D7 інформацію, що передається. Сигнал читання / запису встановлюється у «0». На лінії Wait встановлюється низький рівень і дозволяємо початок циклу передачі. За сигналом Data str=0, =0 і CS1=0 на вході STB, регістра коду аналогового сигналу 1КР580ІР82 з'являється високий рівень. Дані записуються в регістр, якщо керуючий сигнал активний, дані регістру передаються на вихід мікросхеми і надходять на вхід цифро-аналогового перетворювача.

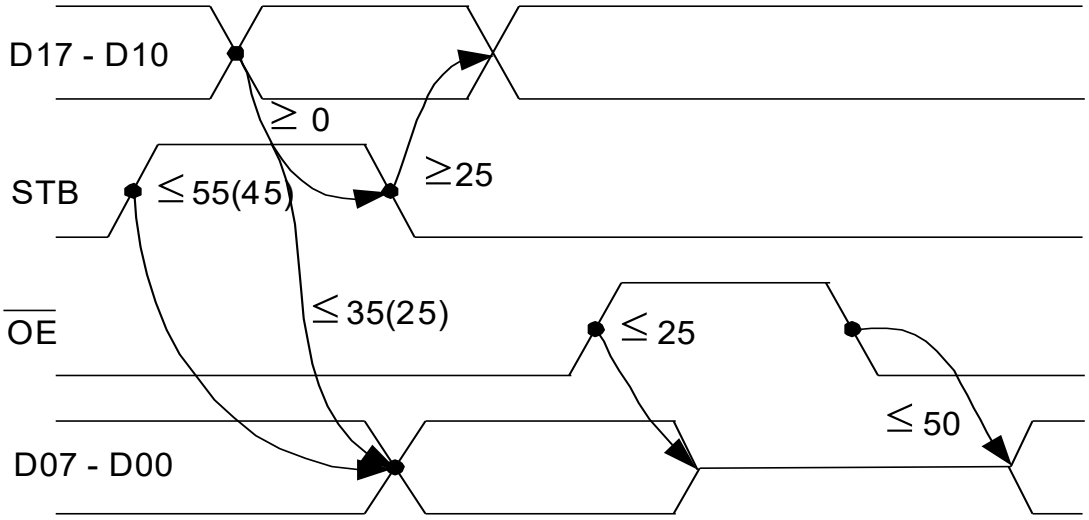

Рисунок 2.3 – Часові діаграми роботи регістру КР580ІР82

Виведення даних триває до тих пір, поки сигнал Wait не встановиться у високий рівень.

Виведення інформації через інші пристрої аналогічне, за винятком коду пристрою, який буде активізовано. Для цифро-аналогового перетворювача 2 код 001, цифро-аналогового перетворювача 3 код 010 регістру виведення цифрового сигналу 0112.

Встановлюємо на шину даних код пристрою, який буде активізовано (0002). Сигнал читання / запис встановлюється у «0». На лінії WAIT встановлюється низький рівень та дозволяється початок циклу передачі адреси. За сигналом Add str:0 і =0 на вході STB регістра адреси з'являється високий рівень. Дані записуються

в регістр, керуючий сигнал активний, дані регістра надходять на введення дешифратора адреси, де код дешифрується.

Цикл передачі адреси продовжується до тих пір, поки сигнал WAIT не буде дорівнює "1".

Сигнал читання / запис встановлюється на високий рівень. На лінії квитування WAIT встановлюється низький рівень і дозволяється початок циклу передачі. За сигналом Data str=0, =1 і CS4=0 на вході STB, регістру введення КР580ІР82 з'являється високий рівень.

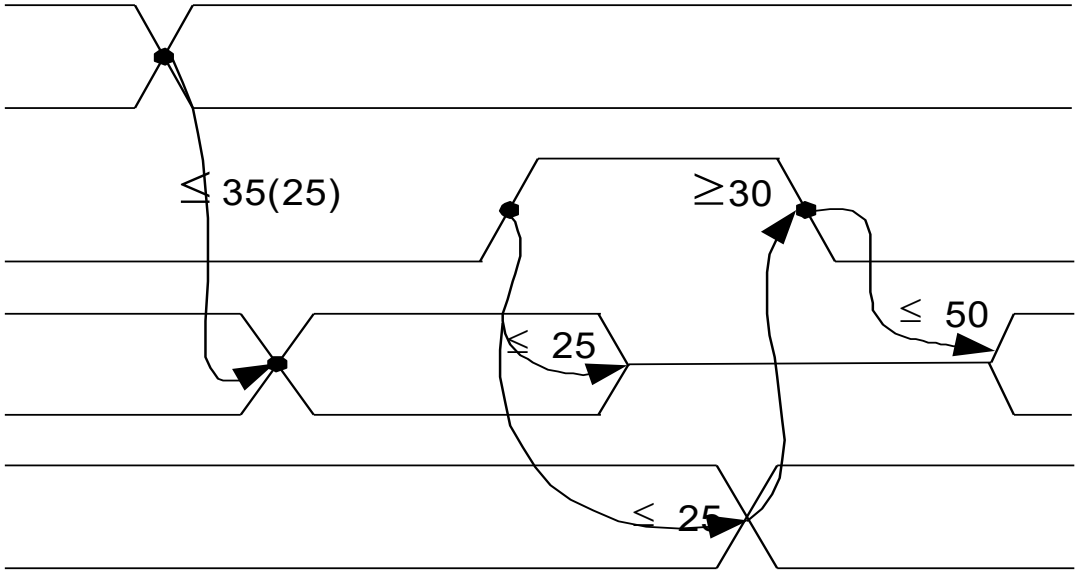

Рисунок 2.4 – Часові діаграми роботи шинних формувань КР580ВА86

2.3 Тимчасові діаграми інформаційного обміну даними через EPP-порт

Паралельні інтерфейси характеризуються тим, що для передачі біт використовуються окремі сигнальні лінії та біти передаються одночасно. Паралельні інтерфейси використовують логічні рівні ТТЛ (транзисторно-транзисторної логіки), що обмежує довжину кабелю через невисоку схильність до перешкод ТТЛ – інтерфейсу. Гальванічна розв'язка відсутня. Передача даних може бути як однонаправленою (Centronics), так і двонаправленою (Bitronics). Іноді паралельний інтерфейс використовують для зв'язку між комп'ютерами.

Поняття Centronics стосується як набору сигналів, так і протоколу взаємодії, зокрема 36-контактного роз'єму на принтерах.

Традиційний порт SPP (Standart Parallel Port) є односпрямованим портом, через який програмно реалізується протокол обміну Centronics. Порт виробляє апаратне переривання імпульсу на вході Ack#. Сигнали порту виводяться на роз'єм DB-255, який встановлений безпосередньо на платі адаптера або плоский шлейф, що з'єднується з нею.

Адаптер паралельного інтерфейсу – це набір регістрів, розташованих у просторі вводу / виводу. Регістри порту адресуються щодо базової адреси порту, стандартними значеннями якого є 3BCh, 378h та 278h. Порт може використовувати лінію запиту апаратного переривання, зазвичай це IRQ7 чи IRQ6. Порт має зовнішню 8-бітну шину даних, 5-бітну шину сигналів стану та 4-бітну шину керуючих сигналів.

BIOS підтримує до чотирьох LPT-портів (LPT1 – LPT4) своїм сервісом – перериванням INT 17h, що забезпечує через них зв'язок із принтером за інтерфейсом Centronics. Цим сервісом BIOS здійснює виведення символу.

Стандартний порт має три 8-бітові регістри, розташовані за сусідніми адресами в просторі вводу / виводу, починаючи з базової адреси порту (BASE).

Data Register (DR) – регістр даних. Дані, які записані у порт, виводяться на вихідні лінії інтерфейсу. Дані, які з цього регістру, залежно від схемотехніки адаптера відповідають або раніше записаним даним, або сигналам тих самих лініях. Якщо в порт записати байт з одиницями у всіх розрядах, а на вихідні лінії інтерфейсу через мікросхеми з виходом типу «відкритий колектор» подати будь-який код, то цей код може бути зчитаний того ж регістру даних.

Таким чином, на багатьох старих моделях адаптерів можна реалізувати порт введення дискретних сигналів, однак вихідним ланцюгам передавача інформації доведеться «боротися» з вихідним струмом логічної одиниці вихідних буферів адаптера.

Схемотехніка ТТЛ такі рішення не забороняє, але якщо зовнішній пристрій виконано на мікросхемах КМОП, їх потужності може не вистачити для відповідної роботи у цьому шинному конфлікті. Однак сучасні адаптери часто мають у вихідному ланцюгу узгоджуючий регістр із опором до 50 Ом. Вихідний струм короткого замикання виходу землю звичайно вбирається у 30мА.

IEEE 1284 визначає п'ять режимів обміну, один з яких повністю відповідає стандартному висновку протоколу Centronics. Стандарт визначає спосіб, яким ПЗ може визначити режим, доступний і хосту (PC), і ПУ (або приєднаного другого комп'ютера).

При описі режимів обміну фігурують такі поняття:

— хост – комп'ютер, що має паралельний порт;

— ПУ – периферійний пристрій, що підключається до порту;

— Ptr – у позначеннях сигналів позначає передавальне ПУ;

— прямий канал – канал виведення даних від хосту до ПУ;

— зворотний канал – канал даних у хост із ПУ.

Порти мають 5 ліній введення стану, використовуючи які ПУ може посилати в хосту байт зошитами (nibble - напівбайт, 4 біти) за два прийоми. Сигнал ASK#, що викликає переривання, яке може використовуватися в даному режимі, відповідає біту регістру 6 стану, що ускладнює програмні маніпуляції з бітами при складанні байту. Тимчасові діаграми наведено на рис. 2.5.

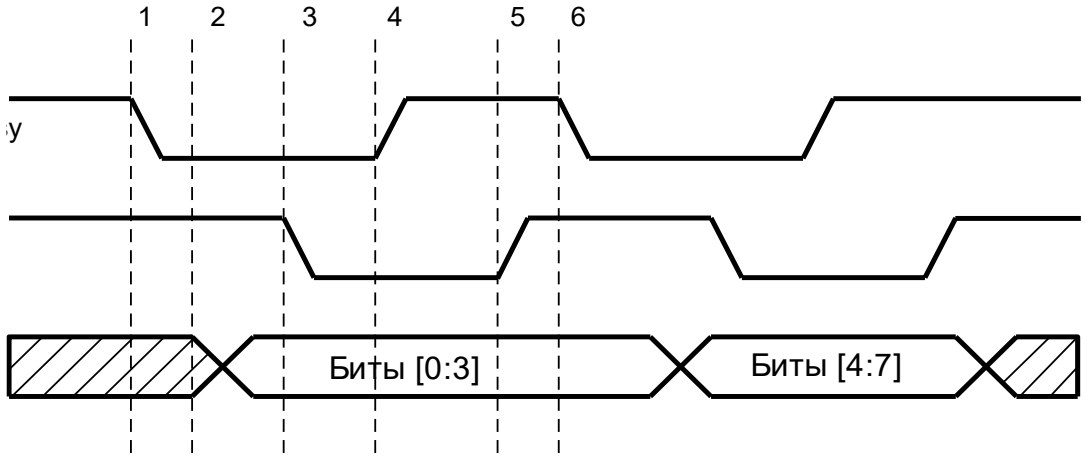

Рисунок 2.5 – Прийом даних у напівбайтному режимі

Прийом байту даних у напівбайтному режимі складається з наступних фаз:

1. Хост сигналізує про готовність прийому даних установкою низького рівня лінії HostBusy.

2. ПУ у відповідь поміщає його на вхідні лінії стану.

3. ПУ сигналізує про готовність установки низького рівня лінії Ptrclk.

4. Хост встановлює високий рівень на лінії HostBusy,

5. ПУ відповідає установкою високого рівня лінії Ptrclk.

6. Кроки 1-5 повторюються наступним.

Напівбайтний режим сильно навантажує процесор, і підняти швидкість обміну вище 50 Кбайт/с не вдається. Перевага в тому, що він працює на всіх портах. Його застосовують у тих випадках, коли потік даних невеликий. Однак при зв'язку з адаптерами локальних мереж, зовнішніми дисковими накопичувачами та CD-ROM прийом великих обсягів даних вимагає терпіння з боку користувача.

У цьому режимі дані приймаються з використанням двонаправленого порту, у якого вихідний буфер даних може відключатися установкою CR.5=1. Як і попередні, режим є програмно-керованим – всі сигнали квітування аналізуються та встановлюються драйвером (рис.2.6).

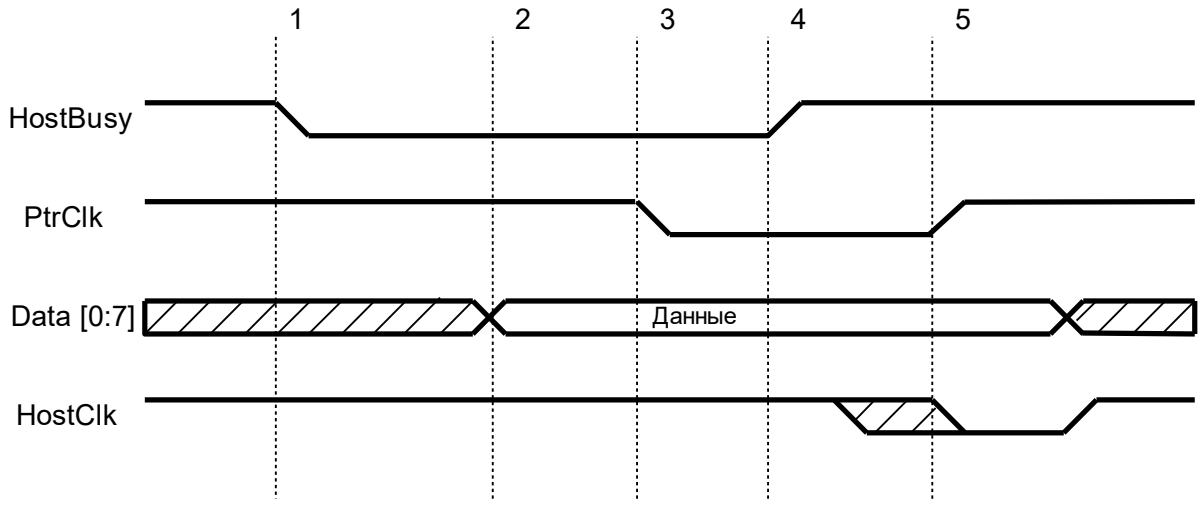

Рисунок 2.6 – Прийом даних у байтному режимі

Протокол EPP (Enhanced Parallel Port – покращений паралельний порт) був розроблений компаніями Intel, Xircom та Zenith data Systems. Він призначений для підвищення продуктивності обміну по паралельному порту EPP та був реалізований у чіпсеті Intel 386SL. Версії протоколу, реалізовані до прийняття IEEE 1284, відрізняються від цього стандарту.

Протокол EPP забезпечує чотири типи циклів обміну:

— запис даних;

— читання даних;

— запис адреси;

— читання адреси.

Цикл запису даних складається з наступних фаз (рис.2.7):

1. Програма виконує цикл виведення (IOWR#) у порт 4 (EPP Data Port).

2. Адаптер встановлює сигнал Write# (низький рівень) і дані розміщуються на вихідну шину LPT-порту.

3. За низького рівня Wait# встановлюється обробка даних.

4. Порт чекає на підтвердження від ПУ (переведення Wait# у високий рівень).

5. Знімається спроб даних – зовнішній EPP цикл завершується.

6. Завершується процесорний цикл виведення.

7. ПУ встановлює низький рівень Wait#, вказуючи можливість початку наступного циклу.

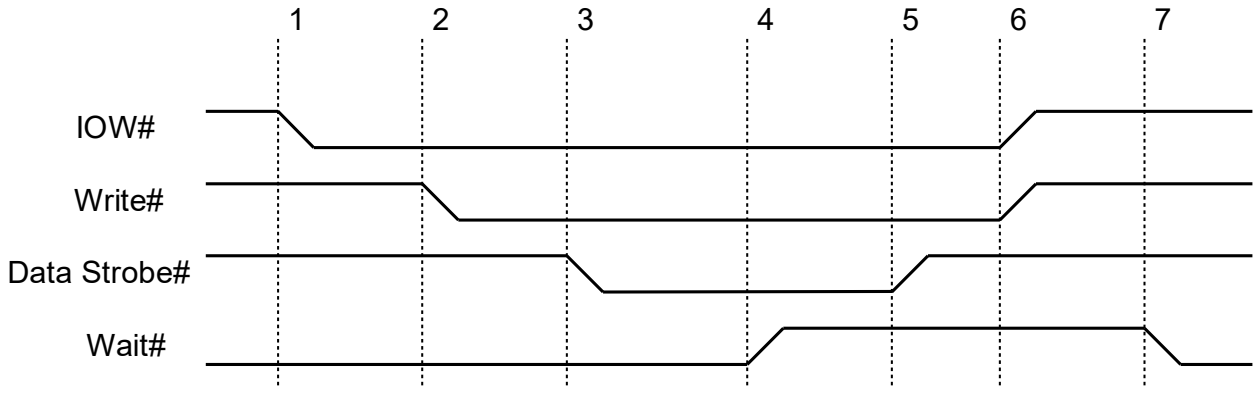

Рисунок 2.7 – Цикли запису даних у EPP

Приклад адресного циклу читання наведено рис.2.8. Цикл читання даних відрізняється лише застосуванням іншого пробного стибкового сигналу.

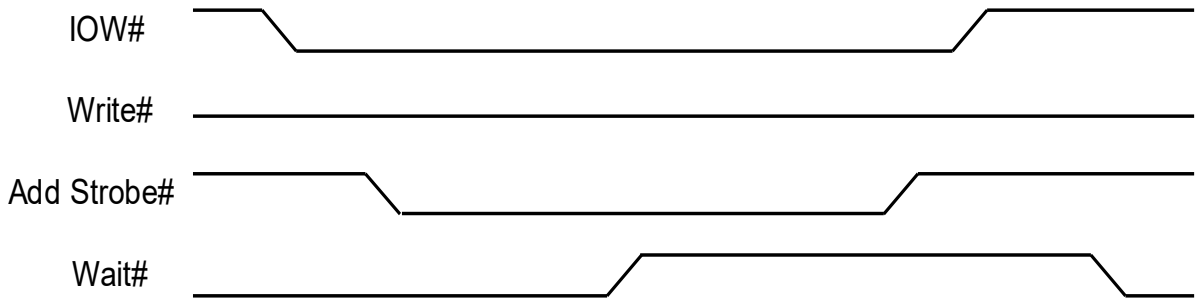

Рисунок 2.8 – Адресний цикл читання EPP

Головною відмінністю EPP є виконання зовнішньої передачі під час одного процесорного циклу введення / виводу. Це дозволяє досягати високих швидкостей обміну даними (0,5...2 Мбайт/с). ПУ, який підключений до паралельного порту EPP, може працювати зі швидкістю пристрою, що підключається через слот ISA. Протокол (interlocked handshakes) дозволяє автоматично налаштовуватися на швидкість обміну, доступну і хосту. ПУ може регулювати тривалість всіх фаз обміну за допомогою лише одного сигналу Wait#. Протокол автоматично підлаштовується під довжину кабелю – затримки, що вносяться, призведуть тільки до подовження циклу. Оскільки кабелі, відповідні IEEE 1284, мають однакові хвильові властивості для різних ліній, порушення передачі, пов'язаного із «змаганнями» сигналів, не повинно відбуватися. При підключенні мережних адаптерів або зовнішніх дисків до порту EPP можна спостерігати незвичне явище: зниження продуктивності в міру подовження інтерфейсного кабелю.

3 АНАЛІЗ СИСТЕМ РЕАЛЬНОГО ЧАСУ

Поняття "реальний час", "робота в реальному часі", "операційні системи реального часу" відомі всім, але пояснюються вони часто по-різному. Кількість ілюзій та міфів у світі реального часу велика. Наприклад, часто плутають такі поняття, як "реальний час" та "швидкість". Іноді вважають, що застосування операційної системи реального часу автоматично вирішить проблеми створення надійної передбачуваної системи. Іноді, навпаки, вважають, що системи реального часу – заняття для теоретиків, а будь-яке завдання реального часу можна вирішити, використовуючи популярні операційні системи загального призначення.

ОС загального призначення особливо розраховані на багато користувачів та орієнтовані на оптимальне розподіл ресурсів комп'ютера між користувачами і завданнями. В ОСРВ подібне завдання відходить на другий план - все відступає перед головним завданням - встигнути зреагувати на події, що відбуваються на об'єкті .

Інша відмінність – це застосування операційної системи реального часу, які завжди пов'язані з апаратурою, з об'єктом, з подіями, що відбуваються об'єкті. Система реального часу, така як апаратно-програмний комплекс, включає датчики, що реєструють події на об'єкті, модулі вводу-виводу, потім вони перетворюють показання датчиків в цифровий вигляд, які придатні для обробки цих показань на комп'ютері, і, власне, комп'ютер з програмою, що реагує на події, що відбуваються на об'єкті.

З іншого боку, застосування ОСРВ завжди конкретно. Якщо ОС загального призначення зазвичай сприймається користувачами як готовий набір додатків, то ОСРВ служить лише інструментом створення конкретного апаратно-програмного комплексу реального часу. І тому найбільш широкий клас користувачів ОСРВ – розробники комплексів реального часу, тобто це люди, які проектують системи управління та збору даних. Проектуючи та розробляючи конкретну систему

реального часу, програміст завжди знає точно, які події можуть відбутися на об'єкті, знає критичні терміни обслуговування кожної з цих подій.

Система має відреагувати на подію, що сталася на об'єкті, своєчасно, тобто протягом часу, критичного для цієї події (meet deadline). Величина критичного часу для кожної події визначається об'єктом і самою подією, і тому може бути різною, але час реакції системи має бути передбачено (обчислено) під час створення системи. Відсутність реакції у передбачений час вважається помилкою для систем реального часу. Система повинна встигати реагувати на події, що відбуваються одночасно. Навіть якщо дві чи більше зовнішніх подій відбуваються одночасно, система повинна встигнути зреагувати на кожну з них протягом тимчасових інтервалів, критичних для цих подій.

Розрізняють системи реального часу двох типів – системи жорсткого реального часу та системи м'якого реального часу.

Системи жорсткого реального часу не допускають жодних затримок реакції системи за жодних умов, оскільки:

— результати можуть виявитися марними у разі запізнення;

— може статися збій у разі затримки реакції;

— вартість запізнення може бути нескінченно велика.

Приклади систем жорсткого реального часу – бортові системи керування, системи аварійного захисту, реєстратори аварійних подій.

Системи м'якого реального часу характеризуються тим, що затримка реакції допустима, хоча може призвести до збільшення вартості результатів і зниження продуктивності системи загалом.

Приклад – робота комп'ютерної мережі. Якщо система не встигла обробити черговий прийнятий пакет, це призводить до тайм-ауту на стороні. Дані при цьому не втрачаються, але продуктивність мережі знижується.

Основна відмінність між системами жорсткого та м'якого реального часу можна виразити так: система жорсткого реального часу ніколи не запізнюється з

реакцією на подію, система м'якого реального часу – не повинна запізнюватися з реакцією на подію.

Назвемо операційною системою реального часу таку систему, яка можна використовуватиме побудови систем жорсткого реального часу. Це визначення виражає ставлення до ОСРВ як до об'єкта, що містить необхідні інструменти, але також означає, що ці інструменти ще потрібно використовувати.

Кількість операційних систем реального часу, незважаючи на їхню специфіку, дуже велика. В оогляді "Real-Time Magazine" було згадано близько шістдесяти систем. Напевно цих систем ще більше, якщо мати на увазі некомерційні ОСРВ. Проте сама специфіка застосування ОСРВ вимагає гарантій надійності, зокрема й юридичних - цим, певне, можна пояснити те що, що серед некомерційних систем реального часу немає скільки-небудь популярних.

Серед комерційних систем реального часу можна виділити групу провідних систем – за обсягами продажу та за популярністю. Це системи: VxWorks, OS, PSOS, LynxOS, QNX, VRTX.

Ознаки систем цього типу - різні платформи для систем розробки та виконання. Додаток реального часу розробляється на host-комп'ютері (комп'ютер системи розробки), потім компонується з ядром і завантажується в цільову систему для виконання. Як правило, додаток реального часу – це одне завдання, і паралелізм тут досягається за допомогою ниток (threads).

Системи цього типу мають низку переваг, серед яких головне - швидкість і реактивність системи. Головна причина високої реактивності систем цього типу - наявність лише ниток (потоків) і, отже, короткий час перемикання контексту з-поміж них (на відміну процесів). З цією головною гідністю пов'язаний і ряд недоліків: "зависання" всієї системи при "зависанні" нитки, проблеми з динамічним підвантаженням нових додатків.

Незважаючи на всю неоднозначність відношення традиційних користувачів систем реального часу до всього, що пов'язано з Microsoft, необхідно констатувати

той факт, що з'явився новий клас ОСРВ, а саме розширення реального часу для Windows. Результати незалежних тестувань цих продуктів показують, що їх можна використовувати для побудови систем жорсткого реального часу. Область застосування розширень реального часу – великі системи реального часу, де потрібна візуалізація, робота з базами даних, доступ в Інтернет тощо.

Один із видимих процесів – зближення ОСРВ різних класів. Так багато ОСРВ класу "ядра реального часу" і UNIX реального часу з'явилися останнім часом кросові системи розробки високої якості, що було характерно для ОСРВ класу "виконавчі системи реального часу ". Так, наприклад, потужні кросові системи розробки з'явилися в таких ОСРВ, як OS9 ("ядра реального часу") і LynxOS (UNIX реального часу).

Що стосується систем виконання ОСРВ, тут спостерігається така ж картина: у системах класу "UNIX РВ" та "ядра РВ" з'являються нові компактні варіанти систем виконання з коротким часом перемикання контексту (OS9, QNX, LynxOS).

Ще одна тенденція, яку не можна не помітити – це поява у багатьох ОСРВ таких продуктів як Java реального часу (Real-Time JAVA) та вбудований Java (Embedded JAVA).

4 ПРОГРАМНА ЧАСТИНА

4.1 Інтерфейс з пакетом Simulink Real-Time Workshop

Після генерації коду, яка призначена для роботи з моделлю, ми отримуємо набір файлів з вихідним кодом згенерованого програмного представлення моделі. Якщо не змінювати вихідних кодів, то результатом роботи скомпілованого додатка буде mat-файл з матрицею виходів моделі. Цей принцип малозастосовний для реального використання.

Компіляція програмного пакета здійснюється за допомогою компілятора LCC, що поставляється з пакетом MathLab. Для автоматичної генерації exe-файлу достатньо запустити на виконання виконуваний файл, ім'я якого збігається з ім'ям моделі. Слід відзначити, що компіляція пройде лише у випадку, якщо з останньої компіляції був змінений хоча б один із файлів з програмним кодом (рис.4.1).

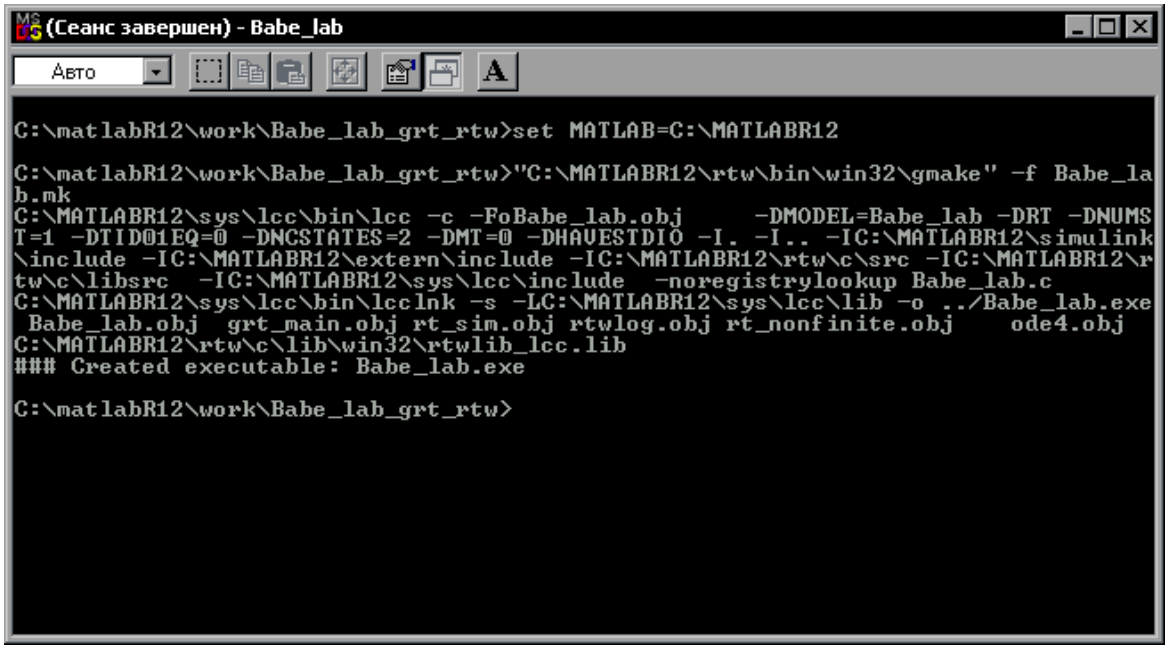

Рисунок 4.1 – Вид виконуваного файлу

Результатом успішної компіляції програми буде рядок повідомлення "Created executable" (рис.4.2).

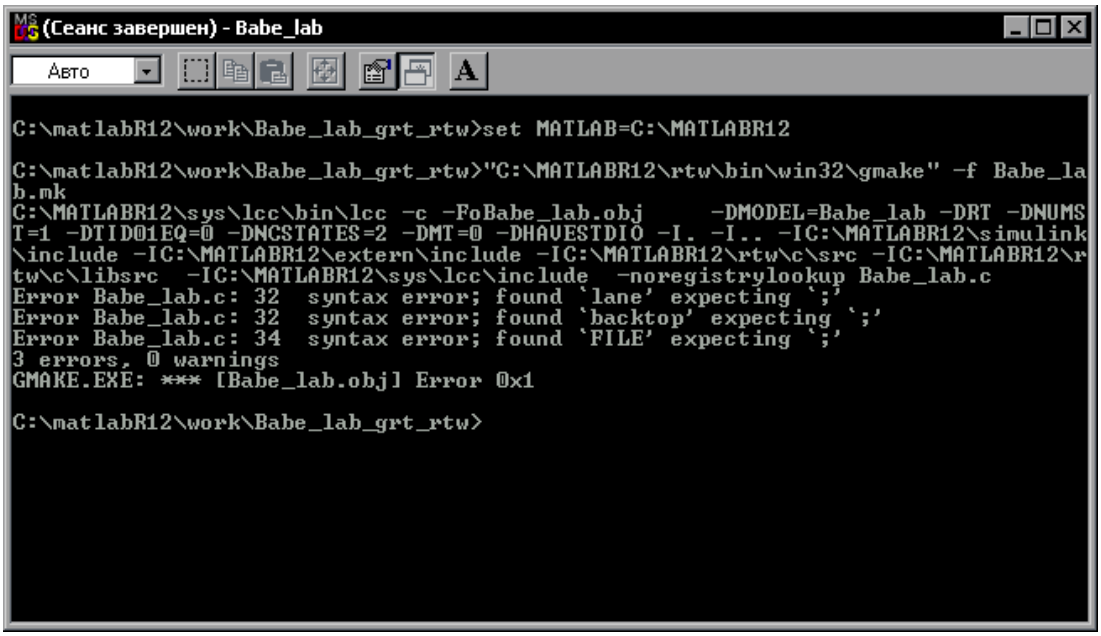

Рисунок 4.2 - Вид виконуваного файлу при виявленні помилки

При цьому компілятор зазначає імена файлів із зазначенням номерів рядків, де, на його думку, є помилки. Після усунення цих помилок слід перекомпілювати програму.

Редагування вихідних кодів програми може проводитися в текстовому редакторі або в спеціалізованій оболонці середовища розробки. Компіляція під іншим компілятором можлива, але досить проблематична. RTW може генерувати код, придатний до компіляції як під LCC, так і під Wacom C++, C++ Builder, MS Visual C++. Код, що генерується RTW, може виконуватися як в одному потоці (singletasking), так і в багатопотоковому режимі (multitasking). В останньому випадку код трохи змінюється, і у всі параметри, що передаються функціям, передається так званий ідентифікатор завдання (task identifier - tid). Це може бути корисним, коли з однією моделлю працюють кілька зовнішніх програмних модулів. У такому випадку, робота виконуваного ядра нагадує механізм роботи бібліотеки, що динамічно підключається, якою вона, по суті, і є.

Для того, щоб мати можливість згенерувати програмний код за допомогою Real-Time Workshop необхідно виконати таке:

- зібрати необхідну модель (рис.4.3)

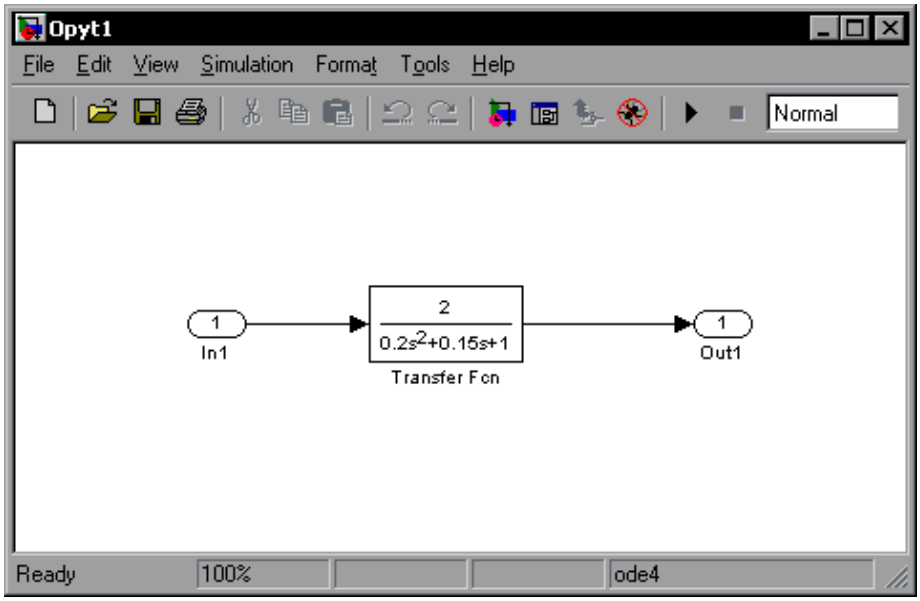

Рисунок 4.3 – Головна математична модель проекту

У меню Tools > Real Time Workshop виберіть пункт Options та заповнити всі необхідні поля в діалозі, що з'явився (рис.4.4).

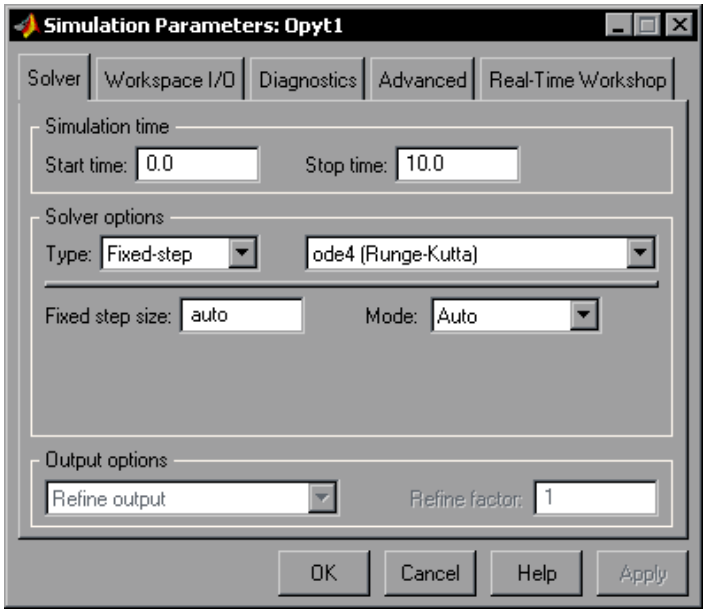

Рисунок 4.4 – Програмне меню діалогового вікна

У списку Solver options необхідно вибрати Fixed-step, інакше генерація буде неможлива. Можливий вибір платформи компіляції (рис. 4.5). Платформа є у списку використовуваних функцій. Вона використовує функції файлів з розширенням .tlc, які представляють бібліотеки даних.

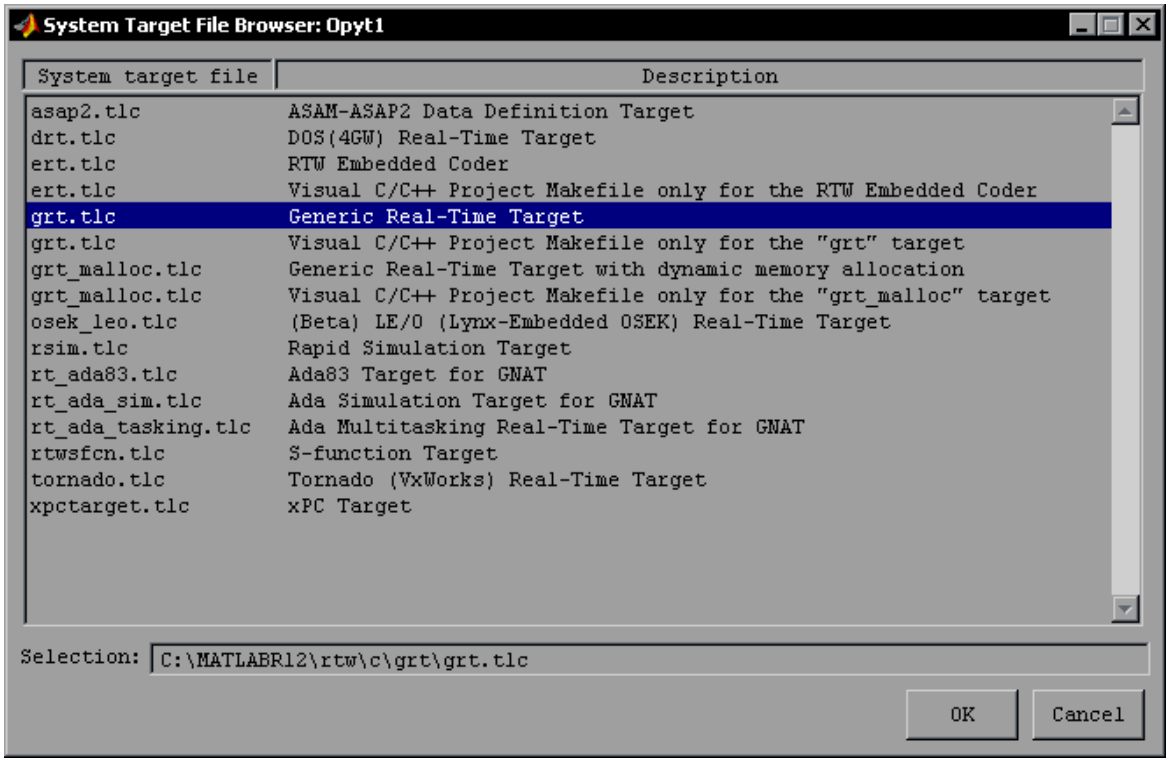

Рисунок 4.5 – Вид платформи компіляції файлу

У даному випадку погодимося із запропонованим варіантом за промовчанням – Generic Real-Time Target. Після заповнення всіх потрібних параметрів потрібно натиснути кнопку Build. Помилок на цьому етапі не повинно бути. Якщо вони все ж таки є, то необхідно переглянути правильність введення параметрів.

4.2 Користування програмним продуктом

Розроблений програмний продукт служить для моделювання управління віртуальним об'єктом.

Програма не потребує інсталяції як такої, тому для функціонування програми достатньо скопіювати необхідний набір файлів.

Окрема розмова про бібліотеки, що підключаються. Їх може бути необмежена кількість самими користувачами. Для цього файли бібліотек повинні відповідати масці plugin*.exe, наприклад plugin_асинх_двигатель.exe

Інтерфейс програми представлений на рис. 4.6.

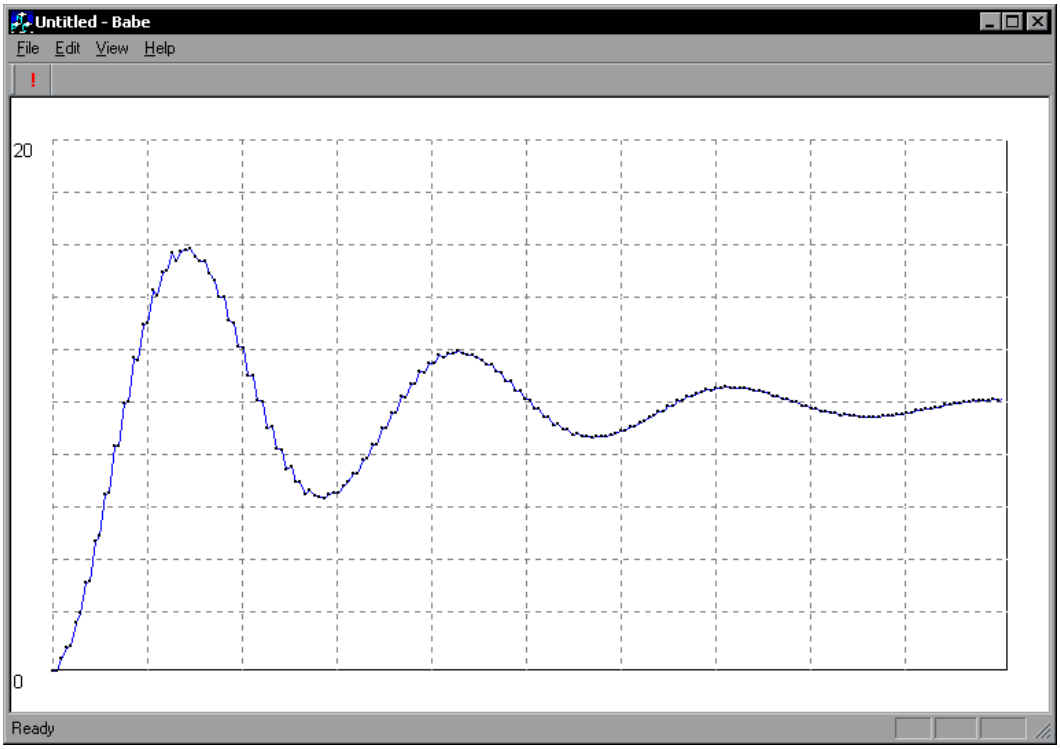

Рисунок 4.6 – Вид інтерфейсу програми

Основний вид екрану займає область графіка з візуалізацією відгуку модуля, що підключається. Графік автоматично масштабується відповідно до того, щоб отриманий графік максимально повно розміщувався в області вікна.

Для того, щоб промоделювати новий запуск, необхідно натиснути кнопку із відповідним зображенням в панелі інструментів. Результатом цього стане поява вікна діалогу (рис. 4.7) завдання параметрів.

Рисунок 4.7 – Вікно задання параметрів

Перше поле введення служить для завдання вхідного значення для моделі, другий список – служить для вибору самої моделі. Після натискання кнопки OK отримаємо графік, що відображає відгуки моделі.

Існує можливість збереження відгуків моделі в окремому файлі. Для цього слід натиснути комбінацію клавіш Ctrl-S і в вікні задати ім'я файлу для збереження. Згодом відкрити цей файл можна шляхом натискання комбінації Ctrl-O. У вікні діалогу вибрати ім'я файлу і натиснути OK.

4.3 Розробка процедури обміну даними та підключення модулів

Для того, щоб програмний модуль, який згенерований і відкомпільований Real Time Workshop можна перетворити на модуль, що підключається, та слід внести певні зміни в код на C++. Ці програмні блоки стандартні і визначають лише звані канали обміну інформації з основним додатком-оболонкою.

Перед тим, як генерувати код за допомогою Real Time Workshop, необхідно змінити модель. Якщо вхідне значення задається одним із блоків із групи Sources (Constant, Step …) то його слід замінити блоком In із групи Signal&Systems. Таким же чином, на виході має бути присутнім блок Out із групи Signals&Systems.

Якщо, наприклад, модель має ім'я babe_lab, RTW генерує такі файли:

— Babe_lab.h, babe_lab.c

— Babe_lab_common.h

— Babe_lab.export.h

— Babe_lab_prm.h

— Babe_lab_reg.h

— Debug.txt

Основним файлом для роботи є babe_lab. так як код генерується мовою С, то будемо дотримуватися правил і використовувати лише С-функції.

У ній визначено кілька функцій - так званих callback (функцій-відгуків, що викликаються при настанні певної події).

MdlInitialize(void) – початкова ініціалізація моделі.

void MdlStart(void) – старт початку розрахунку відгуків моделі.

void MdlOutputs(int_T tid) – функція, що викликається при розрахунку відгуку моделі при досягненні певного часу.

void MdlUpdate(int_T tid) - викликається при "відновленні" моделі, тобто після MdlOutputs.

void MdlDerivatives(void) – функція розрахунку.

void MdlTerminate(void) – викликається при завершенні моделі.

Для цього додамо деякі свої змінні:

— FILE *stream - дескриптор файлу, куди ми записуватимемо відгуки моделі.

— FILE *in_stream; - дескриптор файлу, звідки ми читатимемо вхідні значення.

— float in_data; - вхідне значення для моделі, зчитане з in_stream.

— int index; - номер минулої дискрети часу.

— float data; - масив значень відгуків моделі.

Логіка роботи доопрацьованої програми буде такою: виділити пам'ять під змінні в MdlStart() ->Зчитування вхідних параметрів з файлу -> При виклику MdlDerivates необхідно підставити це значення у вхід In -> При виклику MdlOutputs збільшити на одиницю номер і записати значення відгуку масив вихідних значень -> При виклику MdlTerminate( ) записати весь масив у файл.

Тут механізм передачі був для простоти і наочності реалізований з допомогою файлового обміну. Для практичного застосування необхідно переписати цей процес з використанням каналів (pipes) або за допомогою іншого способу передачі даних між процесами (Inter Process Communication). Можливий варіант із поділом області пам'яті для двох процесів.

Після виклику в додатку діалогу завдання початкових параметрів моделі ми отримуємо змінну типу float.

Цю змінну ми записуємо у файл data.in

```
void CParamDlg::OnOk()
{
  UpdateData();
  if(link!=NULL)

  *link=this->m_input;
  if(exefile!=NULL)
  *exefile=this->m_Selection;
  EndDialog(IDOK);
}
```

Всі модулі, що підключаються, мають ім'я виду plugin*.exe.

При запуску діалогу налаштування параметрів каркас програми вибирає всі файли, що відповідають масці, і пропонує їх користувачеві (рис. 4.8) у ComboBox.

Рисунок 4.8 – Діалогове вікна проекту

```
void CParamDlg::OnShowWindow(BOOL bShow, UINT nStatus)
{
	CDialog::OnShowWindow(bShow, nStatus);
	WIN32_FIND_DATA data;
	HANDLE find;
	find=FindFirstFile("plugin*.exe",&data);
	if(find==INVALID_HANDLE_VALUE){
		return;
	}
	this->m_plugin.AddString(data.cFileName);
	while(FindNextFile(find,&data)){
		this->m_plugin.AddString(data.cFileName);
}

	FindClose(find);
	try {
		this->m_plugin.SetCurSel(0);
	}
	catch(...){
	}
}
```

Після того, як ядро відпрацює вхідний параметр і віддасть керування, необхідно рахувати дані та відобразити їх. Дані прочитуються з файлу data.out і зберігаються в масиві CbabeDoc::data_in. Алгоритм малювання простий та ефективний. Спочатку визначаються межі зміни вихідного параметра від мінімуму до максимуму, і збільшується на 30%. Аналогічно розраховуються межі зміни значення часу. Потім визначаються межі розмірів клієнтської області вікна та

проводиться переведення фізичних координат у логічні. Малювання проводиться засобами WinApi.

При отриманні повідомлення про те, що область вікна зіпсувалася (наприклад, його перекрило вікно іншої програми), воно автоматично перемальовується. При зміні користувачем розміру вікна графік перемальовується (рис. 4.9).

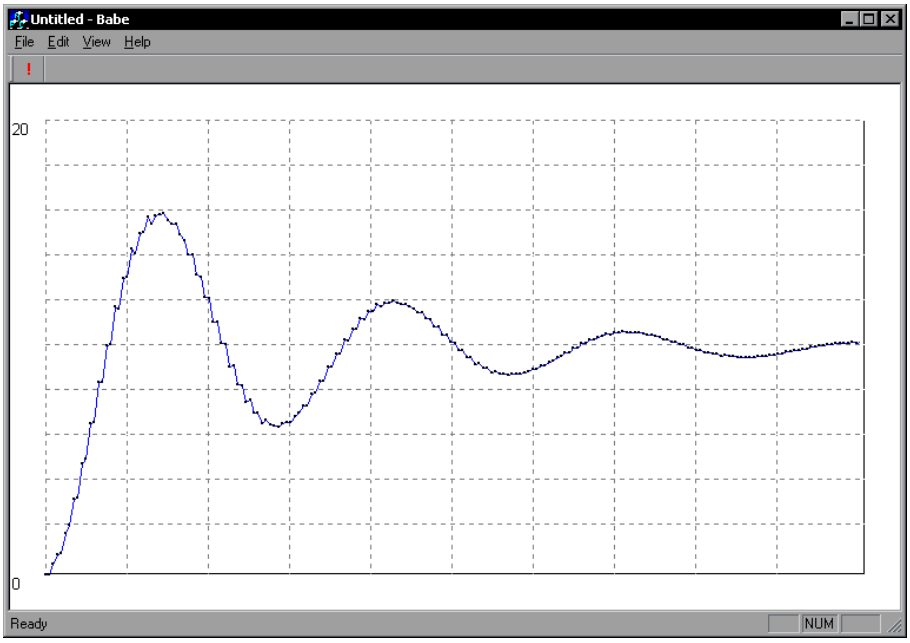

Рисунок 4.9 – Вид інтерфейсу розробленої програми

Одне із завдань, які розв'язує програміст, це розробки додатків, які можуть створювати й редагувати документи різних типів, у тому, щоб надати користувачеві можливість записати внутрішнє подання документа файл і відновити його (цей процес називається серіалізацією даних).

Такі програми, підготовлені за допомогою AppWizard, використовують цей механізм за допомогою методів класу CDocument. Програмістові пропонується лише перевизначити метод Serialize цього класу для роботи з конкретними даними програми.

Програміст може визначити свій клас для роботи з даними і скористатися механізмом запису і відновлення об'єктів, що розглядається нижче.

Бібліотека класів MFC визначає механізм запису та відновлення об'єктів (serialization), причому підтримка цього механізму здійснюється засобами класу CObject. Класи, успадковані від CObject, також можуть забезпечувати роботу механізму запису та відновлення об'єктів. Для цього при оголошенні класу треба вказати макрокоманду DECLARE_SERIAL, а при визначенні – макрокоманду IMPLEMENT_SERIAL.

ВИСНОВКИ

У цій кваліфікаційній роботі було вирішено кілька важливих завдань, які відкривають нові перспективні можливості розробників систем управління.

Найважливішим моментом була розробка прикладного програмного забезпечення, що забезпечує інтерфейс обміну даними з ядром, який скомпілюваний програмною надбудовою Real Time Workshop.

Використання коду, що реалізує поведінку моделі, замість ручного написання дозволило, скоротити час розробки системи управління, дозволило уникнути помилок програміста під час виконання ним роботи, призначеної для системотехніка. Таким чином, розробка систем управління з використанням нового підходу, ділиться на два потоки, коли в першому системотехнік готує та налагоджує в пакеті MathLab модель, а програміст у цей час готує інтерфейс користувача.

Окрім цього, розроблений механізм підключення програмних модулів дозволив зробити цю систему максимально відкритою та доступною для розширення. Спосіб написання додаткових програмних модулів є простим, що дуже важливо для використання системи програмістами не дуже високої кваліфікації.

Загалом це дозволило створити так звану систему управління віртуальним об'єктом, коли всі відгуки об'єкта емулюються електронною платою. У такому разі, для налагодження програм систем управління зовсім необов'язкове підключення до реального об'єкта. Це дозволило скоротити час та вартість розробки програм систем управління.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кваліфікаційна робота: методичні вказівки до оформлення кваліфікаційних робіт для здобувачів першого (бакалаврського) та другого (магістерського) рівнів вищої освіти всіх освітніх програм денної та заочної форм навчання / уклад. Н.В. Ковальчук, Ю.Г. Фесіна, І.Л. Заблоцька Луцьк : ЛНТУ, 2023. 46 с.

2. Інноваційні підходи в підготовці магістрів з прикладної механіки : навч. посіб. / Т. Є. Божко, Б. П. Валецький, Л. М. Самчук, Т. І. Четвержук . – Луцьк : Вежа-Друк, 2024. – 324 с.

3. Методичні рекомендації до написання кваліфікаційної роботи за ступенем вищої освіти «магістр» : метод. рекомендації для здобувачів другого (магістерського) рівня вищої освіти освітньої програми «Прикладна механіка», спец. 131 Прикладна механіка, галузі знань 13 Механічна інженерія денної та заоч. форм навч. / уклад.: Т. І. Четвержук, Р. М. Полінкевич. – Луцьк: ЛНТУ, 2024. 48 с.

4. Диференціальні моделі. Стійкість / А. М. Самойленко, С. Д. Борисенко, Дж. Матараццо та ін. – К., 2000.

5. Борозенець Г.М., Павлов В.М., Семак І. В. Деталі машин: Навчальний посібник. – К.: Видавничий дім «Кондор», 2021. – 220 с.

6. Головня В.Д. САПР технологічних процесів : конспект лекцій / В.Д. Головня. – Житомир : Житомирська політехніка, 2019. – 200 с.

7. А. Шпиняк Дослідження універсальної системи моделювання складних динамічних об’єктів // Тези IV студентської науково-технічної конференції факультету транспорту та механічної інженерії “Інноваційні технології в транспорті та механічній інженерії ” (16 листопада 2024 року). – Луцьк: ЛНТУ – 2024 – С.9-11.

8. Жерновий Ю. В. Імітаційне моделювання систем масового обслуговування : практикум / Ю. В. Жерновий. – Львів : Видавничий центр ЛНУ імені Івана Франка, 2007. – 307 с.

9. Кучеренко Є. І. Сіткові моделі в задачах аналізу складних систем: навч. посібник для вузів / Є. І. Кучеренко. – Х. : ХТУРЕ, 1999. – 99 с.

10. Патент США №2002157388A1 Pump-integrated flexible actuator /SETO TAKESHI, TAKAGI KUNIHIKO/ Опубліковано 2002-10-31. 9.

11. Патент США №4815782A Grappling device /CRAIG PRESTON S, FISHER JEFFREY/ Опубліковано 1989-03-28.

12. Павленко І.І., Мажара В.А. Роботизовані технічні комплекси / Под ред. Павленко 1.1. - Навчальний посібник 2012 - 393с.

13. Khalifa H. Harib, Kamal A.F. Moustafa, A.M.M. Sharif Ullah and Salah Zenieh: Parallel, Serial and Hybrid Machine Tools and Robotics Structures: Comparative Study on Optimum Kinematic Designs - 110-124p.

14. Технологічні машини. Розрахунок і конструювання: Навчальний посібник / Ю.В. Кодра, З.А. Стоцько; За ред. З.А. Стоцька. – Львів: Бескид БІТ, 2004. – 466с.

15. Павленко І.І., Мажара В.А. Продуктивність функціонування двозахватних промислових роботів на позиціях допоміжних пристроїв // Прогресивні технології і системи машинобудування. Міжнародний збірник наукових праць. – Вип. 30 – Донецьк: ДонНТУ, 2005. – С. 170 – 175.

16. Павленко І.І., Мажара В.А. Конструктивно-кінематична структура двозахватних пристроїв промислових роботів // Надійність інструменту та оптимізація технологічних систем. Збірник наукових праць. – Вип. 19 – Краматорськ – Київ, 2006. – С. 104 – 109.

17. Антонюк В.С., Клименко С.Ан., Клименко С.А. Теплові явища при обробці різанням: Навч. посібник. – К.: НТУ України «КПІ», 2014. – 156 с.

18. Методи, моделі та інформаційні технології оцінювання станів складних об’єктів: монографія / Є. І. Кучеренко, В. Є. Кучеренко, І. С. Глушенкова, І. С. Творошенко; ХНАМГ, ХНУРЕ. – Х. : ХНАМГ; ХНУРЕ, 2012. – 278 с.