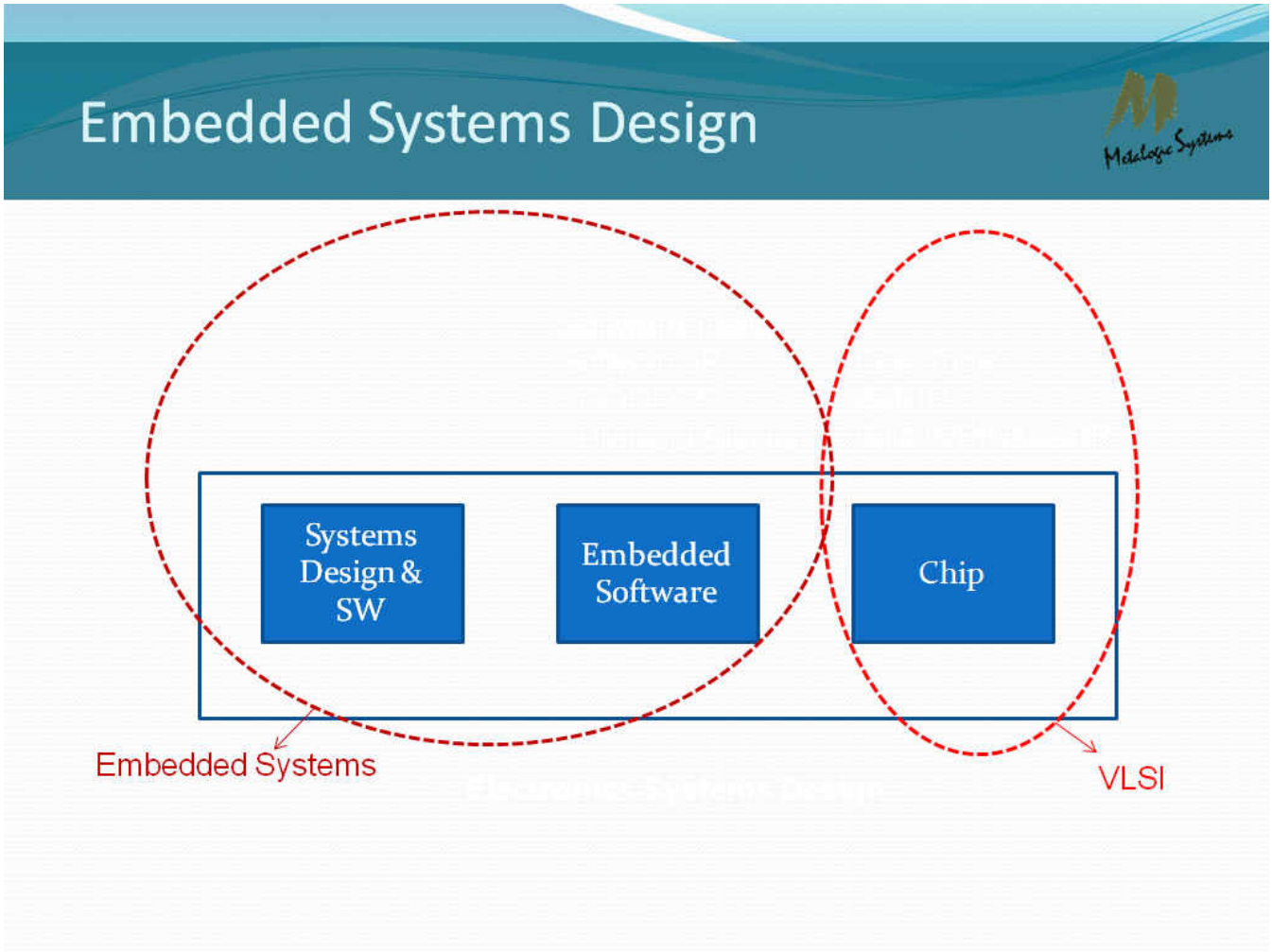

Embedded Systems Design
Metalogic Systems
Systems Design & SW
Embedded Software
Chip
Embedded Systems
VLSI

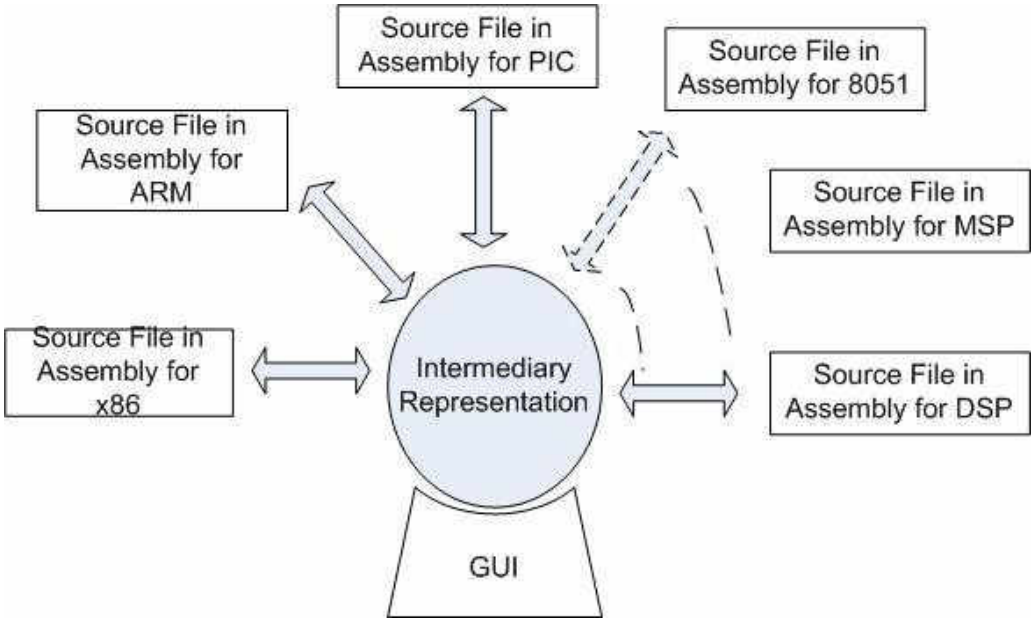
Micro-Controller Porting Tool

- Overall Specifications
- Specifications for First Release
- Stages of Development
- Work Allocation
- Milestones and Time Lines
- Appendix

Overall Specifications

1. Port the assembly program written for source processor to an assembly program for target processor

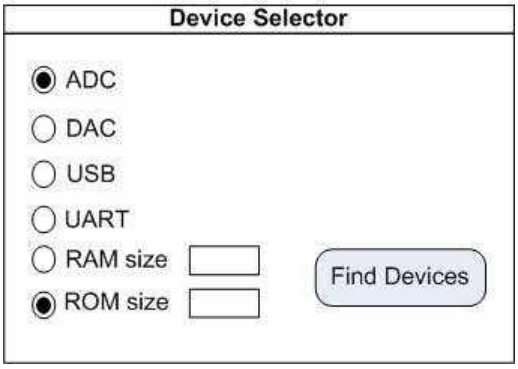
Metalogic Systems Pvt. Ltd.



2. Supports **all the available architectures** of the processors (DSP, microcontroller..)available in the market. The user selects the source and target processor from a list.

3. Device Profiler – for an assembly program, the porting tool should provide an analysisreport for device selection based on

   (i) Least execution time
   (ii) Available peripherals on the chip
   (iii) The user can get a performance profiler based on
      (a) Processors, controllers
      (b) Devices that have a specific set of peripherals, speed, I/O ports



4. The control flow (logic) of the program can be expressed as a flowchart. The IR will serve as the base for creating the flow chart.

5. Optimization – for
    (i) Reducing the execution time
    (ii) Power reduction
    (iii) Compact program – LUT pattern analyzer.

6. **Peripheral Configuration Code Generator (PCCG)** – for a configuration specified by the user the tool should be capable of generating the corresponding code. Only the initialization code will be generated. This has no connection with the porting tool, in a sense that the database created for the porting tool will be used by PCCG. The output will be an assembly language program that can be used in the program. This will help user the user to configure the peripherals without having the knowledge of the associated SFRs. See Fig.

7. **Peripheral Configuration Porting (PCP)** – With the available peripheral configuration for a specific device, an equivalent code for other device can be generated. This provides the user a quick code without a need to learn the bit pattern in SFRs. The initialization instructions for one device can be ported to other device(s).

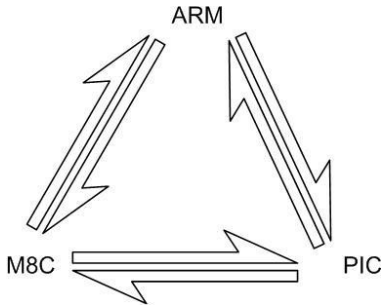
    For instance the user knows the configuration of the UART of the source processor. For the same configuration but for a different device the settings can be ported. The SFR bit patterns are suitably modified to reflect the source processor settings. The input can be selected from the drop down menu or alternatively the initialization section can be copy and pasted from the source assembly file.. See fig below.

Specifications for the first Release

Stage 1 : Basic

1. **Porting:** Port the assembly program written for source processor to an assembly program for target processor provided it falls in the selected architecture category.

2. **Architectures:** Supports the following architectures
    (i) ARM
    (ii) M8C
    (iii) PIC



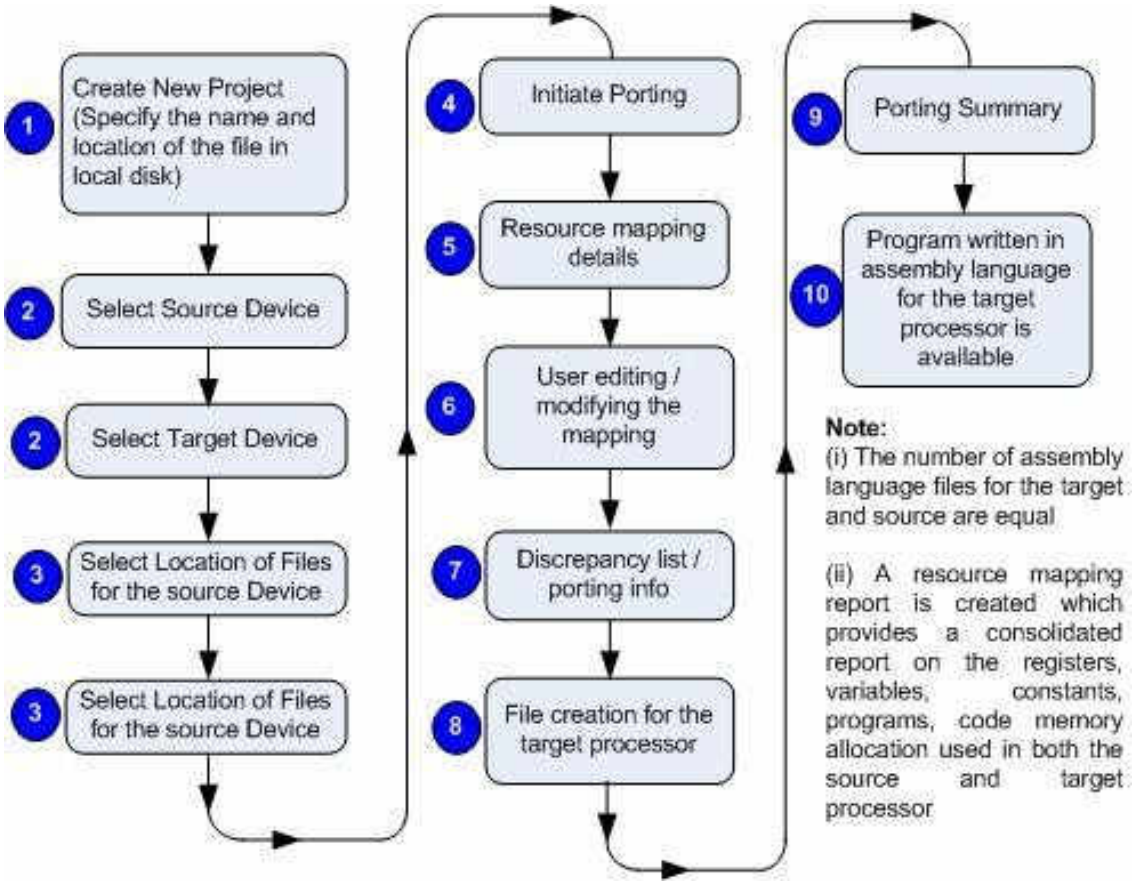
3. **IDE:** The IDE has the following options

S.No	Description
1	Create New Project
2	Open Existing Project
3	Create New File
4	Open New File
5	Select Source device
6	Select Target Device The user will be provided an option to select the source and target processor. Exact device number will be selected by the user from the drop down list.
7	Search Devices – enter the number
8	Parallel scrolling of the current source and target files
9	If user selects specific segment of the code in the source / target file, the corresponding converted segment will be highlighted in the target / source file
10	Same number of files will be created for the target file after porting
11	The user can see the peripherals enabled in the assembly source. The list of peripherals that are available but not enabled in the source assembly file is shown.
12	During the process of porting the user will be prompted with a window notifying the mapping of the resources. The resource mapped will be (i) Registers (ii) Constants (iii) Variables in code memory location

S.No	Description
	(iv) Variables in Data memory location (v) Initialization section
13	During the process of porting the number of instructions and execution speed in each of (i) Routine (subroutines, ISR) (ii) File will be provided by the tool

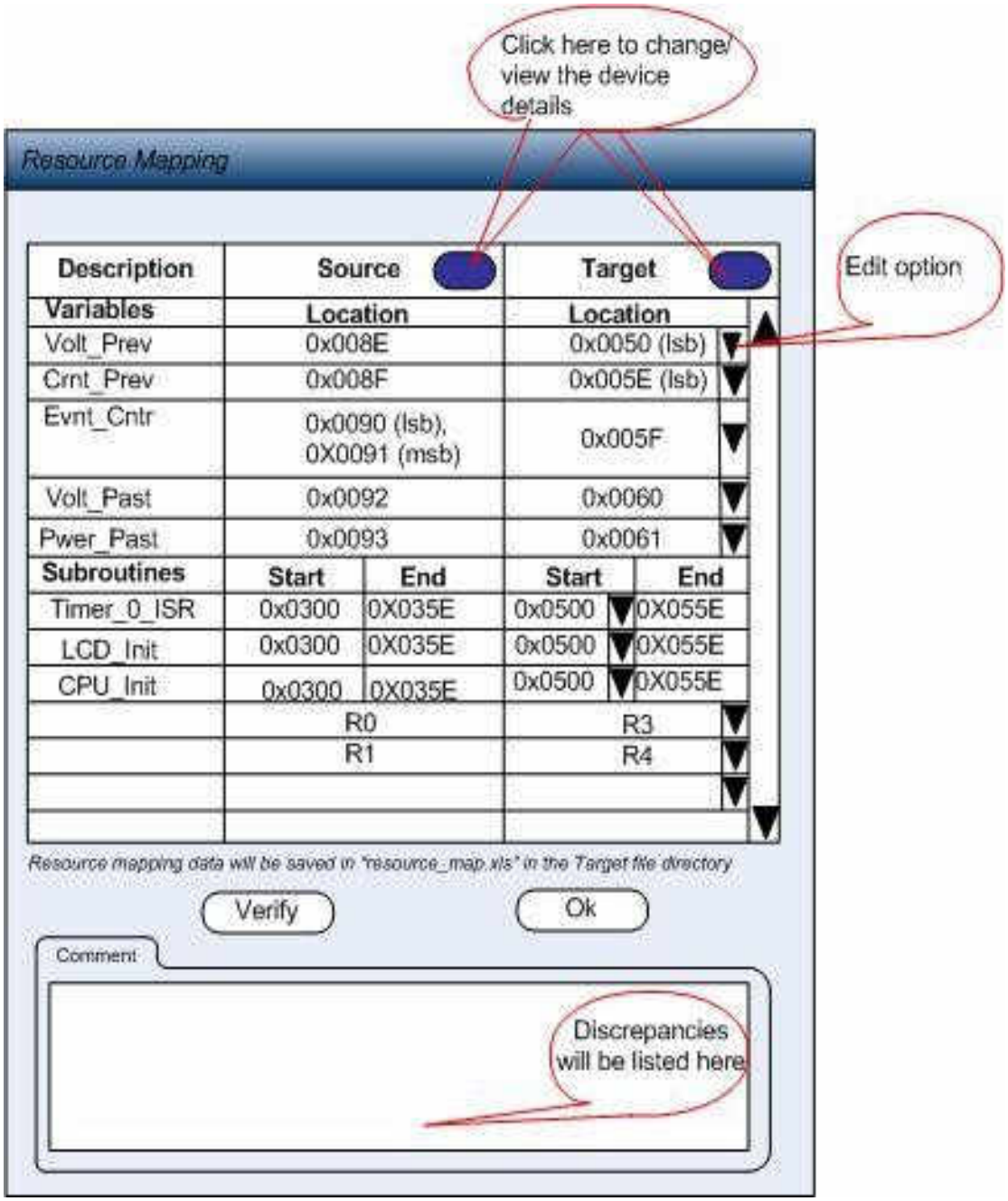
3.1 Porting Process

The following figure provides the specific steps involved in the porting operation.



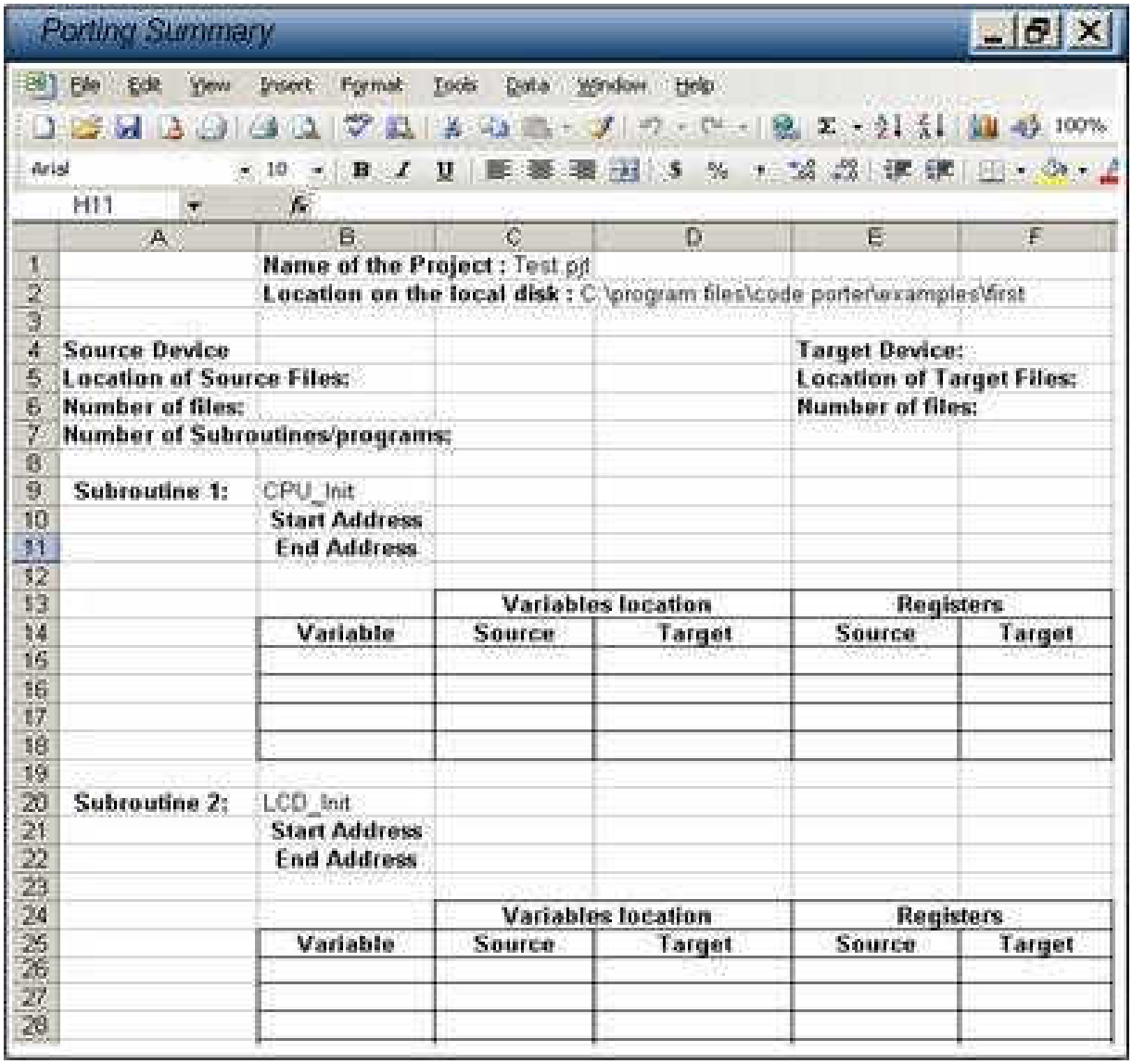
3.2 Resource Mapping

The resource mapping during the porting is a pseudo- tool in a sense that mapping will be done by the porting tool itself. Further the user is provided the flexibility to choose the address/memory location for the variables, constants. The following provides the snapshot of the resource mapping by the porting tool.



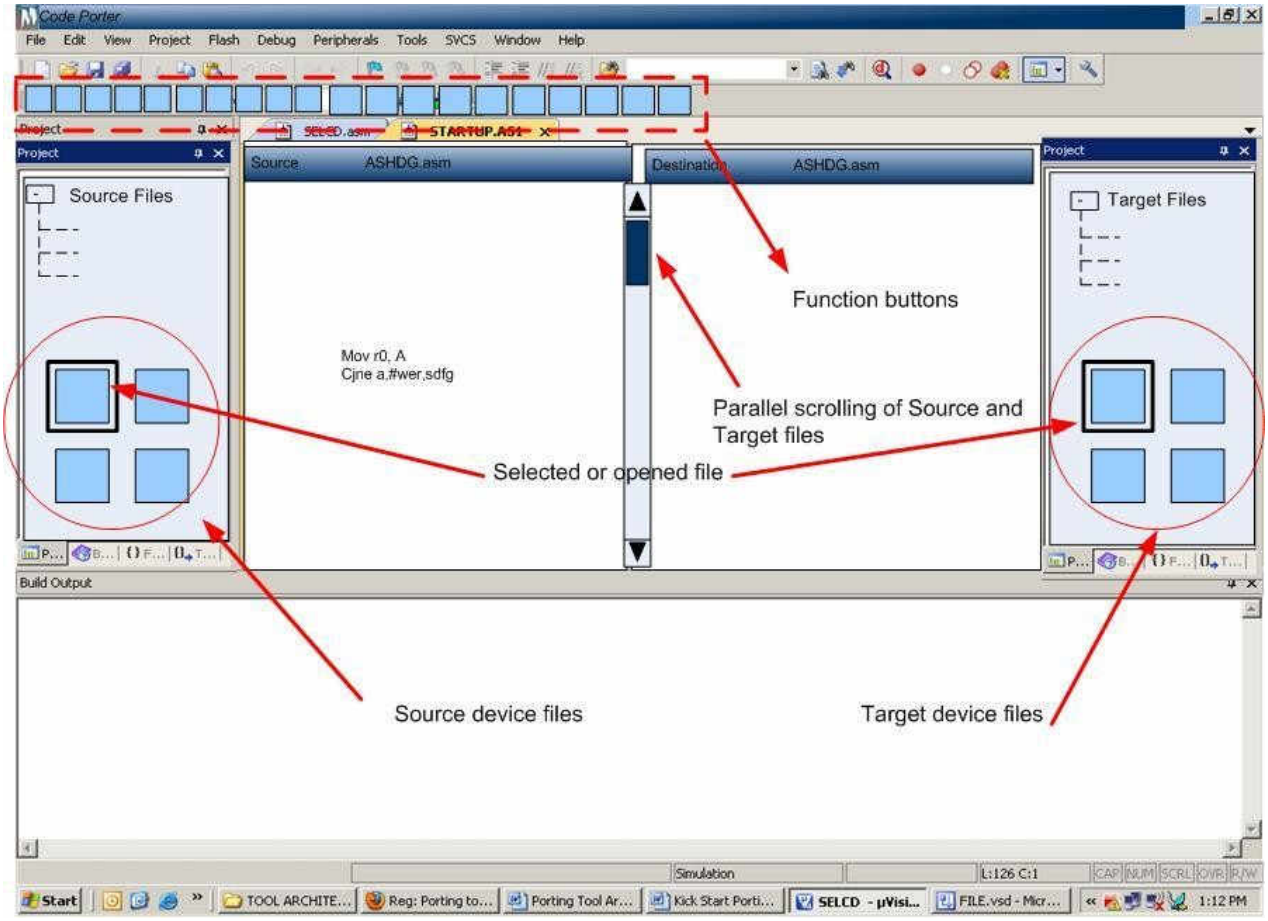
3.3 Port summary

The porting summary of the project created for porting process will be as shown below.

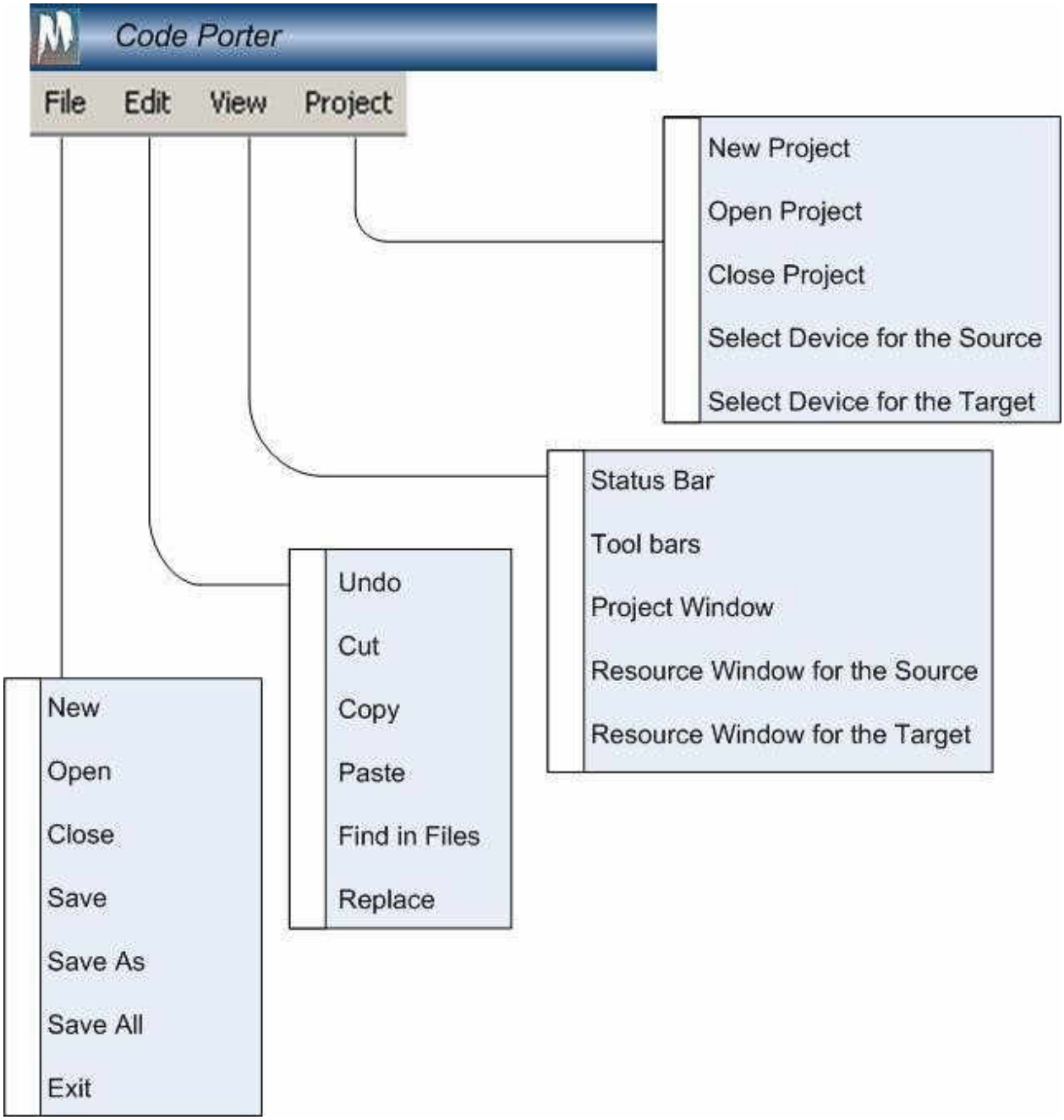


3.4 Layout of IDE

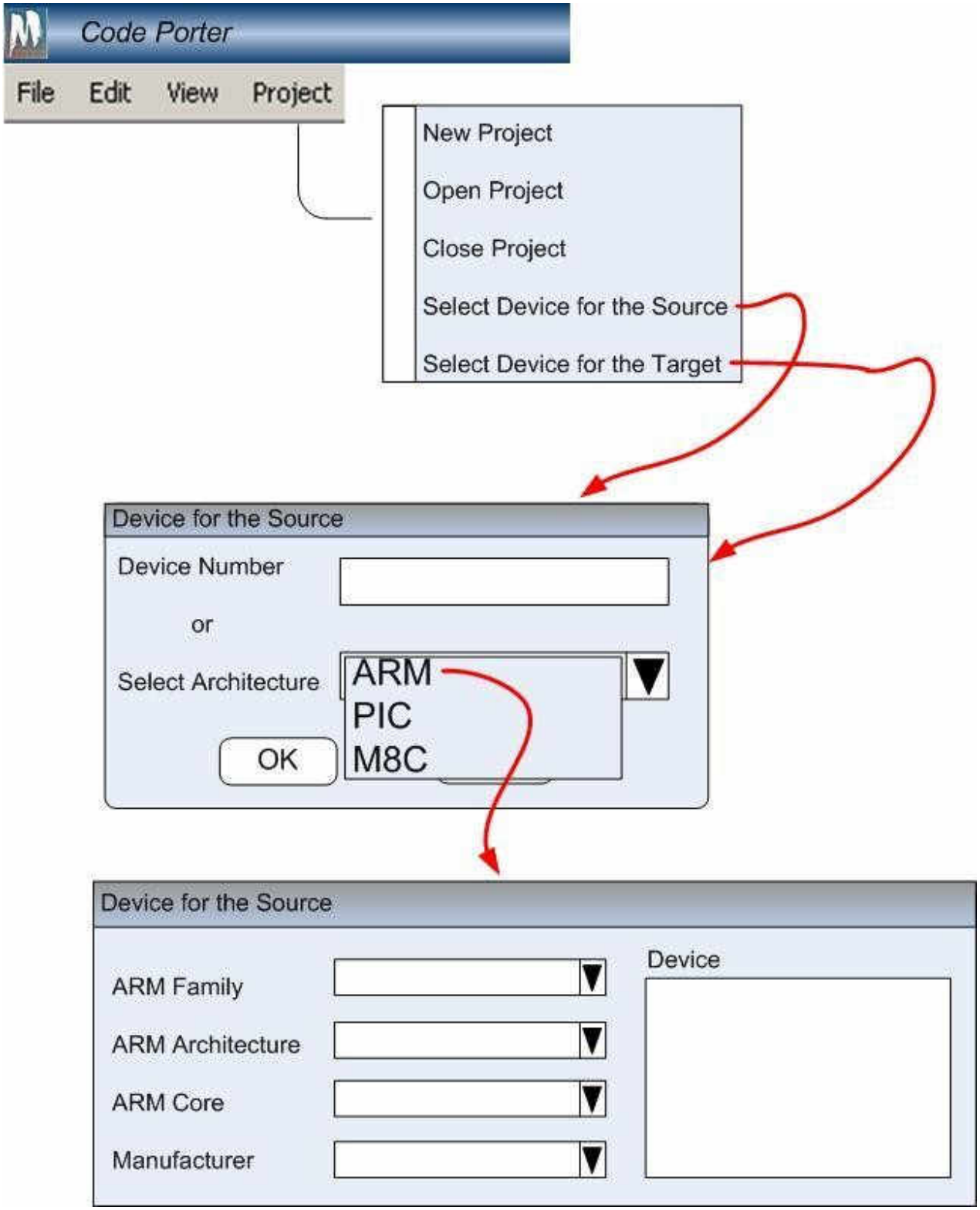
The layout of the IDE environment for the 'porting tool' is shown below.



A generic tool bar with modifications required for the porting tool is shown in the following fig.



3.5 Selection of Source and Target device



4. **Config-info:** Meaningful information regarding the configuration settings (included in the program) of the on-chip peripherals can be extracted. Those not included can be represented

as peripheral not used. The configuration is recovered by reading the bit pattern in the SFRs. The source language is assembly.

Example 1
For Instance, an UART configuration can be represented as follows.

a) Baud Rate - 9600bps
b) Stop bit - 2
c) Start bit - 1
d) Parity - Odd
e) Buffer Size - 1 byte
f) Direction - Transmit Only - in subroutine 1
- Transmit and Receive - in other subroutines and main

Example 2
Reading an on-chip ADC configuration will provide the following details

a) Sampling Rate - 1k samples/sec
b) No of Bits - 8 bit

5. **Valid Data Config:** The tool has the data base of the valid configurations of the SFR for the supporting processors. The tool should allow configuration editing for the same.

6. **Peripheral –info:** The on-chip peripherals used by the processor is got from the assembly program.

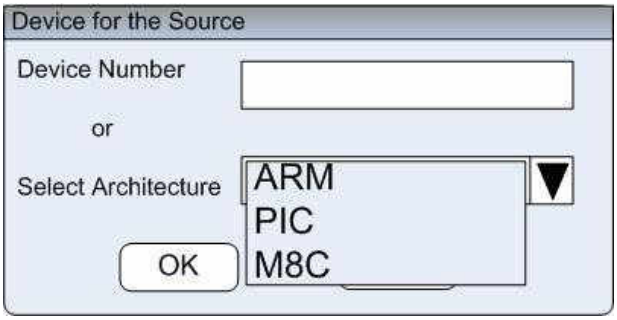
7. **Flowchart:** The logic of the program can be extracted and be represented as a flowchart which provides better readability.

Stage 2 : Extended

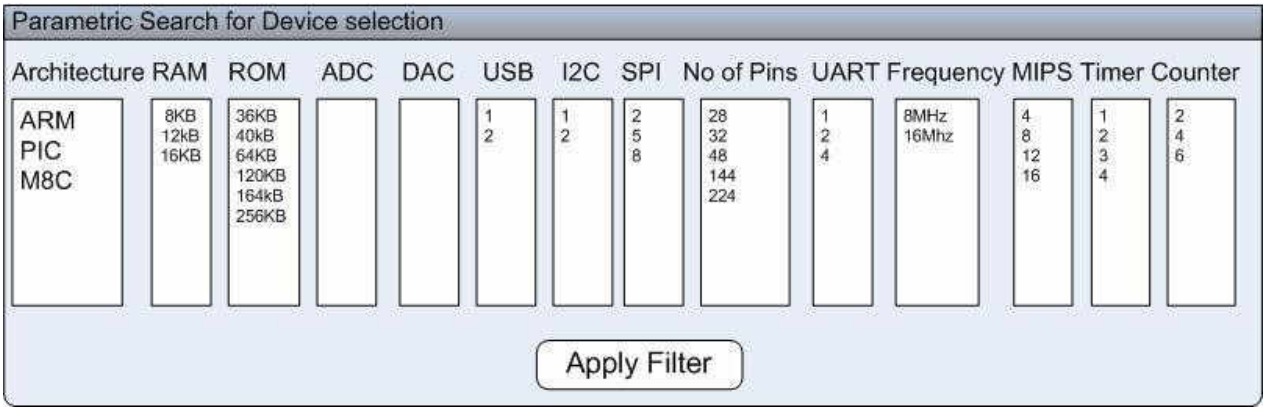
1. **Device Selection:** The tool should allow selection of devices for source and target devices based on

   (i) **Architecture** : (ARM, PIC or M8C). See fig below

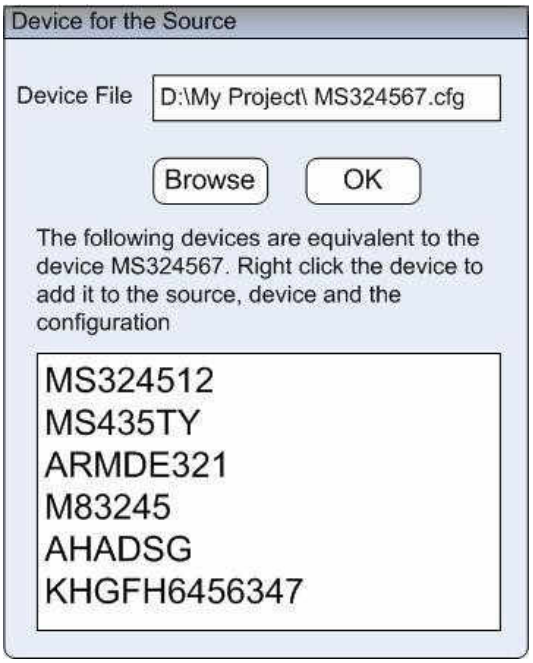
   (ii) **Device number**: See fig below



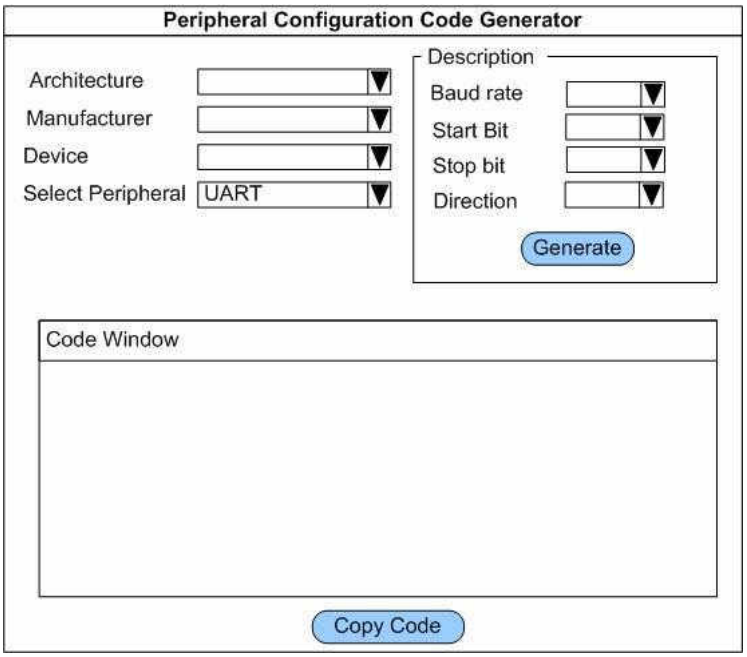
   (iii) **Parametric search :** Specify the parameters like the following. See Fig Below.



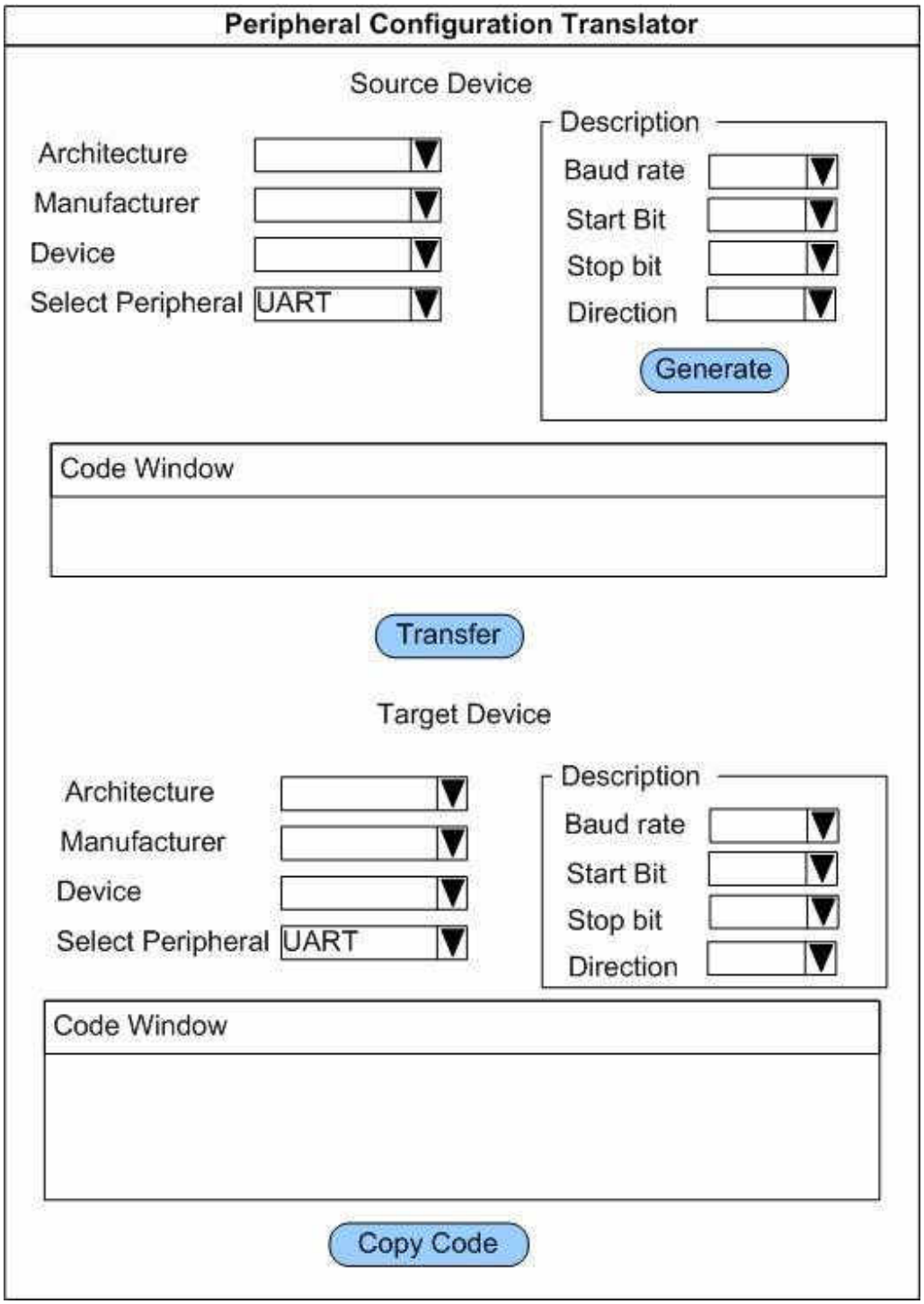
   (iv) **Like Devices:** Specify a device (by pointing it a device file) and find an equivalent device in terms of the on-chip resources. Resources greater than or equal to will be listed. See fig below.



2. **Peripheral Configuration Code Generator (PCCG)** – for a configuration specified by the user the tool should be capable of generating the corresponding code. Only the initialization code will be generated. This has no connection with the porting tool, in a sense that the database created for the porting tool will be used by PCCG. The output will be an assembly language program that can be used in the program. This will help user the user to configure the peripherals without having the knowledge of the associated SFRs. Also see Overall specifications number 6.



3. **Peripheral Configuration Porting (PCP)** – With the available peripheral configuration for a specific device, an equivalent code for other device can be generated. This provides the user a quick code without a need to learn the bit pattern in SFRs. The initialization instructions for one device can be ported to other device(s). For instance the user knows the configuration of the UART of the source processor.



For the same configuration but for a different device the settings can be ported. The SFR bit patterns are suitably modified to reflect the source processor settings. The input can be selected from the drop down menu or alternatively the initialization section can be copy and pasted from the source assembly file. See screen shot of the front end application in the 'overall specifications' number 7.

4. **Other languages:** The porting tool should also support the porting from binary(.hex) file (source processor) to the binary(.hex) file (target processor).

Stages of Development

1. Front-end development – GUI – to be developed in eclipse.
2. The features available will be ported to eclipse. The pre-written code will be reused.
3. IDE development – features, design layout, visual representation of features of the porting Tool.
4. Collecting the device details that belonged to each of ARM, M8C and PIC architectures.
5. Grouping the devices based on the architecture, on-chip peripherals, RAM, ROM.
6. Identifying valid SFR configurations in each of the devices and creating a data base.
7. Back-end- Tool development
8. Creating a database for reading assembly language from the binary/opcodes.
9. Creating a Target Specific IR (TSIR) for each of the architecture/devices.
10. Linking the TSIR with the generic IR
11. Optimization development for performance –
    a) Reducing the execution time
    b) Power reduction
    c) Compact program – LUT pattern analyzer.
12. Testing the back-end tool for all the features that it claims. An example from a real time application is preferred

APPENDIX

The following information is extracted through the first parsing of the program

1. Starting and ending address of the program.
2. No of subroutines used in the program.
3. SFRs used in the program (specific subroutines).
4. Registers used
5. Single bit ports
6. multiple bit Ports used
7. Labels
8. Timers
9. Counters
10. No of pins used

    Types of I/Os. – UART, ADC, DAC, USB, RS232, RS485, RS422, SPI, CAN, I2C, USART

11. No of Digital Inputs
12. No of Digital Outputs
13. No of Digital I/Os
14. No of Analog Inputs
15. No of Analog Outputs
16. No of ISR
17. Data width of the processor used
18. Vector Address of the ISR
19. Size of the subroutines
20. No of user defined software flag
21. Maximum execution time of the Programs/Subroutines
22. Maximum execution time of the ISRs
23. LUT location – RAM wise / ROM wise
24. No of Time sliced Routines
25. Execution time of each time-sliced routine – this may be included in the exec time of the subroutine

Porting Sequence

1. Each program /routine is a black box
2. Find the logic of execution between the I and Os
3. Identify the AI,DI, AO,DO, constants and Variables in RAM &ROM
4. Determine the logic flow from input to output
5. Create the IR sequence from the source assembly file
6. Identify the mnemonic equivalent of the target processor

Porting Type

1. Only chip replacement (Pull out old IC and Plug in the new IC)
2. Xtend peripheral – some peripherals off chip of the old/source have to be replaced by onchip new/target peripherals
3. Data width/clock speed change in requirement

User Specifies the following

1. Pin replacement
2. Register replacement
3. Variables from RAM and ROM
4. AI, AO, DI, DO replacement
5. PROM space availability

Areas of Intelligence required by the tool

1. Functions of valid SFR configurations
2. Read the mnemonics of the porting processor (source)
3. Capability to represent the mnemonics in IR
4. Capability to represent IR in mnemonics / instructions of the target processor
5. Reads the binary/hex file
6. Creates the binary/hex file to be downloaded in the target
7. Creates the flowchart of the main/subroutines