1 Subroutine setup_options()

Convert several options from m_input_vars to internal variables defined in m_options.
TODO/FIXME:There might be name collision if there are the same variable defined in both modules. Need to think a better scheme for this.

```
SUBROUTINE setup_options()
  USE m_input_vars
  USE m_options
  IMPLICIT NONE
```

- Option for the choice of method to solve Poisson equation

```
  IF( assume_isolated == 'sinc' ) THEN
    SELECT CASE( poisson_solver)
    CASE( 'ISF', 'isf' )
      I_POISSON_SOLVE = 1
    CASE( 'DAGE', 'dage' )
      I_POISSON_SOLVE = 2
    CASE DEFAULT
      I_POISSON_SOLVE = 1
    END SELECT
  ELSE
    ! we are calculating periodic system, use the default Poisson solver
    I_POISSON_SOLVE = 0
  ENDIF
```

- Option for the choice of method to solve Kohn-Sham equation

```
  SELECT CASE( KS_Solve )
  CASE( 'Emin_PCG', 'Emin_pcg', 'Emin-PCG', 'Emin-pcg', 'Emin_cg' )
    I_KS_SOLVE = 1
  CASE( 'SCF', 'scf' )
    I_KS_SOLVE = 2
  CASE DEFAULT
    WRITE(*,*) 'Using default value for I_KS_SOLVE = ', I_KS_SOLVE
  END SELECT
```

- How to calculate parameter β in conjugate gradient method.

```
  SELECT CASE( cg_beta )
  CASE( 'Fletcher-Reeves', 'FR', 'F-R' )
    I_CG_BETA = 1
  CASE( 'Polak-Ribiere', 'PR', 'P-R' )
    I_CG_BETA = 2
  CASE( 'Hestenes-Stiefel', 'HS', 'H-S' )
    I_CG_BETA = 3
  CASE( 'Dai-Yuan', 'DY', 'D-Y' )
    I_CG_BETA = 4
  CASE DEFAULT
    WRITE(*,*) 'Using default values for I_CG_BETA = ', I_CG_BETA
  END SELECT
```

- Iterative diagonalization methods

```
  SELECT CASE( diagonalization )
  CASE( 'davidson-qe' )
    I_ALG_DIAG = 1
  CASE( 'davidson' )
    I_ALG_DIAG = 2
  CASE( 'LOBPCG', 'lobpcg' )
    I_ALG_DIAG = 3
  CASE DEFAULT
```

```
    WRITE(*,*) 'Using default values for I_ALG_DIAG = ', I_ALG_DIAG
  END SELECT
```

- Number of electronic steps (for direct minimization and SCF cycle)

```
  IF( electron_maxstep /= -1 ) THEN
    Emin_NiterMax = electron_maxstep
    SCF_NiterMax = electron_maxstep
  ENDIF
```

- Mixing parameter.

```
  IF( mixing_beta > 0.d0 ) THEN
    SCF_betamix = mixing_beta
  ENDIF
```

- Charge-density mixing in SCF

```
  SELECT CASE( mixing_mode )
  CASE( 'linear' )
    MIXTYPE = 0
  CASE( 'linear-adaptive' )
    MIXTYPE = 1
  CASE( 'broyden-elk' )
    MIXTYPE = 3
  CASE DEFAULT
    MIXTYPE = 1
  END SELECT
```

- Total energy convergence criteria

```
  IF( conv_thr > 0.d0 ) THEN
    Emin_ETOT_CONV_THR = conv_thr
    SCF_ETOT_CONV_THR = conv_thr
  ENDIF
END SUBROUTINE
```