```
MACHINE
    Battery+Hardware_M2        //    目標到達と計画コストの導入
REFINES
    Battery+Hardware_M1
SEES
    Battery+Hardware_C1
VARIABLES
    batteryLevel
    recharge
    solar_panels_opened
    is_charging
    currentPosition        //    ~ここまでBattery_M0とM1で定義
    atGoal        //    目標地点に到達したか[HI3]
    plan2D_cost        //    目標地点(Destination)までの推定消費電力
    plan2C_cost        //    目標地点から最寄りの充電場所までの推定消費電力
INVARIANTS
    inv1    :    atGoal ∈ BOOL        //    ゴール到達フラグ
    inv2    :    plan2D_cost ∈ N1        //    目標地点までの移動は必ずバッテリ消費あり
    inv3    :    plan2C_cost ∈ N        //    充電地点までの移動は目標地点で充電できる場合があるため目標地点＝充電地点でバッテリ消費がない場合もある
EVENTS
    INITIALISATION    ≙
      extended
    STATUS
      ordinary
    BEGIN
      act_init_battLevel    :    batteryLevel ≔ MAX_BATTERY        //    バッテリレベルの初期状態は満充電状態
      act_init_recharge    :    recharge ≔ FALSE
      act_init_solar_panels    :    solar_panels_opened ≔ FALSE
      act_init_is_charging    :    is_charging ≔ FALSE
      act_init_currentPosition    :    currentPosition ≔ START_POSITION
      act_atGoal    :    atGoal ≔ FALSE
      act_p2D    :    plan2D_cost :∈ PLAN2D_COST_SET
      act_p2C    :    plan2C_cost :∈ PLAN2C_COST_SET
    END

    Signal_Recharge_Refined    ≙        //    HI1の具体化:目標に到達し、さらに充電場所へ行くのに電力が不足するかを判断
      extended
    STATUS
      ordinary
    REFINES
      Signal_Recharge
    WHEN
      grd_arrive_charger    :    currentPosition ∈ chargers        //    ローバーが充電位置にいる場合
      grd_rechargeOff    :    recharge = FALSE        //    リチャージフラグが立っていない
      detect_BattLow    :    batteryLevel < plan2D_cost + plan2C_cost        //    目標地点＋充電地点の移動に必要なバッテリレベルが不足
    THEN
      act1    :    recharge ≔ TRUE
    END

    Consume_Battery_Refined    ≙
    STATUS
      ordinary
    REFINES
      Consume_Battery_Refined
    WHEN
      grd_batt_min    :    batteryLevel > MIN_BATTERY        //    バッテリレベルが最小より大きい場合だけ消費可能
      grd_rechargeOff    :    recharge = FALSE        //    充電が必要なフラグが立っていない場合のみ消費(移動)できる
      grd_batt_check_plan    :    batteryLevel ≥ plan2D_cost + plan2C_cost        //    バッテリレベルが十分かどうか
    THEN
      act1    :    batteryLevel :| batteryLevel' < batteryLevel ∧ batteryLevel' ≥ MIN_BATTERY
    END

    Start_Charging    ≙
      extended
    STATUS
      ordinary
    REFINES
```

```
  Start_Charging
WHEN
  grd1   :   recharge = TRUE        //   リチャージフラグがONの場合
THEN
  act1   :   is_charging ≔ TRUE
  act2   :   solar_panels_opened ≔ TRUE        //   ソーラーパネルを開く(HI6)
END

Recharge_Battery   ≙
  extended
STATUS
  ordinary
REFINES
  Recharge_Battery
WHEN
  grd1   :   is_charging = TRUE        //   充電中であること
  grd3   :   batteryLevel + step_charge < MAX_BATTERY        //   バッテリレベルがMAX未満
THEN
  act1   :   batteryLevel ≔ batteryLevel + step_charge
END

Complete_Charging   ≙
  extended
STATUS
  ordinary
REFINES
  Complete_Charging
WHEN
  grd1   :   is_charging = TRUE
  grd2   :   batteryLevel + step_charge ≥ MAX_BATTERY        //   バッテリレベルがMAXになったら
THEN
  act1   :   is_charging ≔ FALSE        //   ローバの充電終了
  act2   :   solar_panels_opened ≔ FALSE        //   ソーラパネルも閉じる
  act3   :   recharge ≔ FALSE        //   リチャージフラグもFALSE
  act_update_p2D   :   plan2D_cost :∈ PLAN2D_COST_SET        //   次の移動プランのために、目標地点までのコスト情報を集合から再取得する
  act_update_p2C   :   plan2C_cost :∈ PLAN2C_COST_SET        //   次の移動プランのために、充電地点地点までのコスト情報を集合から再取得する
END

Reach_Goal   ≙          //   HI3: 目標に到達した時のみセット
STATUS
  ordinary
WHEN
  grd1   :   currentPosition ∈ goals
  grd2   :   atGoal = FALSE
THEN
  act1   :   atGoal ≔ TRUE        //   HI3: 目標に到達した時のみセット
END

Reset_atGoal_Flag   ≙          //   atGoalフラグがTRUEだったらFALSEにする
STATUS
  ordinary
WHEN
  grd1   :   atGoal = TRUE
THEN
  act1   :   atGoal ≔ FALSE
END

END
```