```
MACHINE
    Battery+Hardware_M4        //    現在の目標と充電地点を保持する変数を追加して充電完了時にこれらを一括更新
REFINES
    Battery+Hardware_M3
SEES
    Battery+Hardware_C3
VARIABLES
    batteryLevel
    actualBatteryLevel
    recharge
    solar_panels_opened
    is_charging
    currentPosition
    atGoal
    plan2D_cost
    plan2C_cost
    lastCommand        //    〜ここまでM0-M3で定義
    currentGoal        //    現在目指している目標地点
    currentCharger        //    現在の目標に対応する最寄りの充電地点
INVARIANTS
    inv1    :    currentGoal ∈ goals
    inv2    :    currentCharger ∈ chargers
    inv3    :    currentCharger = targetCharger(currentGoal)        //    常に目標と充電地点が対応していることを保証
EVENTS
    INITIALISATION    ≙
    STATUS
      ordinary
    BEGIN
      act1    :    batteryLevel ≔ MAX_BATTERY        //    バッテリレベルの初期状態は満充電状態
      act2    :    recharge ≔ FALSE
      act3    :    solar_panels_opened ≔ FALSE
      act4    :    is_charging ≔ FALSE
      act5    :    currentPosition ≔ START_POSITION
      act_atGoal    :    atGoal ≔ FALSE
      act6    :    lastCommand ≔ 0        //    コマンドなし
      act7    :    actualBatteryLevel ≔ MAX_BATTERY_PHYSICAL        //    満充電状態
      act_init_logic    :    currentGoal, currentCharger, plan2D_cost, plan2C_cost :|
                             currentGoal' ∈ goals ∧
                             currentCharger' = targetCharger(currentGoal') ∧
                             plan2D_cost' = goalCost(currentGoal') ∧
                             plan2C_cost' = chargingCost(currentGoal' ↦ targetCharger
                             (currentGoal'))        //    依存関係のある変数を一括で初期化する
    END

    Send_Movement_Command    ≙
      extended
    STATUS
      ordinary
    REFINES
      Send_Movement_Command
    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
      grd_need_charge    :    recharge = FALSE
      grd_charging    :    is_charging = FALSE
      grd_under_act_batt_min    :    actualBatteryLevel > MIN_BATTERY_PHYSICAL
      grd_safe_consume    :    (actualBatteryLevel − (plan2D_cost + plan2C_cost)) − MARGIN_RATE > MIN_BATTERY
    THEN
      act_set_command    :    lastCommand ≔ 1        //    HI4: 移動コマンドの送信(値は仮)
      act_consume_act_batt    :    actualBatteryLevel ≔ actualBatteryLevel − (plan2D_cost + plan2C_cost)        //    HI5: 送信されたコマンドは実行され、バッテリレベルが減る
      act_update_batt    :    batteryLevel ≔ (actualBatteryLevel − (plan2D_cost + plan2C_cost)) − MARGIN_RATE        //    BM1/BM2: バッテリー監視と報告値の更新
    END

    Signal_Recharge_Refined    ≙
      extended
    STATUS
      ordinary
    REFINES
      Signal_Recharge_Refined
    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
```

```
  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_Refined   ≙
    extended
  STATUS
    ordinary
  REFINES
    Recharge_Battery_Refined
  WHEN
    grd1   :   is_charging = TRUE       //   充電中であること
    grd2   :   batteryLevel < MAX_BATTERY       //   バッテリ電圧がMAX未満であること
    grd_over_act_batt_max   :   actualBatteryLevel + step_charge + MARGIN_RATE ≤ MAX_BATTERY_PHYSICAL
  THEN
    act_charge_act_batt   :   actualBatteryLevel ≔ actualBatteryLevel + step_charge       //   実際のバッテリレベルの更新
    act_update_batt   :   batteryLevel ≔ (actualBatteryLevel + step_charge) − MARGIN_RATE       //   BM1/BM2: バッテリー監視と報告値の更新
  END

  Complete_Charging   ≙
  STATUS
    ordinary
  REFINES
    Complete_Charging
  ANY
    g       //   目標地点
  WHERE
    grd1   :   is_charging = TRUE
    grd2   :   batteryLevel + step_charge ≥ MAX_BATTERY       //   バッテリレベルがMAXになったら
    grd_g   :   g ∈ goals
  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 ≔ goalCost(g)       //   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 ≔ chargingCost(g ↦ targetCharger(g))       //   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
    act_update_goal   :   currentGoal ≔ g
    act_update_charger   :   currentCharger ≔ targetCharger(g)
  END

  Reach_Goal   ≙
    extended
  STATUS
    ordinary
  REFINES
    Reach_Goal
  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   ≙
    extended
  STATUS
    ordinary
  REFINES
    Reset_atGoal_Flag
  WHEN
    grd1   :   atGoal = TRUE
  THEN
    act1   :   atGoal ≔ FALSE
  END

END
```