TG Motion

verze 4

Modul Virtuální PLC

návod k obsluze

TG Drives, s.r.o.
Olomoucká 1290/79
CZ - 627 00 Brno

Tel.: +420 548 141 811
Fax: +420 548 141 890
E-mail: info@tgdrives.cz

Historie revizí

datum	verze	popis
10. 1. 2017	1.0	
31. 7. 2017	1.1	upravena typografie

Obsah

1. Virtuální PLC 4
1.1 Popis Virtuálního PLC 4
1.2 Tvorba PLC 4
1.3 Vyžadovaný algoritmus PLC 4
1.4 Komunikace s PLC 5
2. Funkce PLC 6
2.1 Obecný popis funkcí 6
2.2 Struktura PLC_DATA 6
2.3 Funkce podle priority 7
2.4 Časová souslednost volání funkcí 11
3. Nástroje pro ladění PLC 17
3.1 Control Observer 17
3.2 Výpisy 17
3.3 Oscilloscope 17
3.4 Aplikace Windows 18

1. Virtuální PLC

1.1 Popis Virtuálního PLC

Modul Virtuální PLC vykonává uživatelem napsaný PLC program. Ten provádí výpočty, zabezpečuje ovládání servopohonů, načítání a nastavování hodnot vstupů a výstupů a stará se o komunikaci s dalšími periferiemi pouze prostřednictvím sdílené paměti.

Virtuální PLC je z **TG Motion** periodicky volán v intervalu nastaveném v **Cycle_Time**. Velikost **Cycle_Time** je definována v souboru **TGMotion4xx.ini**.

Výhodou Virtuálního PLC je jeho rychlost, protože běží přímo ve strojovém kódu CPU.

Skupiny Servo a I/O unifikují ovládací rozhraní pro různé typy servopohonů a I/O jednotek. Stejný PLC kód lze aplikovat na různé servopohony nebo I/O jednotky. Operativně lze také servopohony nebo I/O jednotky měnit, aniž by se musel PLC kód přepisovat.

1.2 Tvorba PLC

Program PLC je možné vytvářet v obecném vývojovém prostředí, např. Visual Studio, Delphi. Programovací jazyk může být C, C++ a Pascal.

Ve vývojovém prostředí je třeba vytvořit soubor *.tgm, který musí zveřejnit právě šest funkcí s názvy: **Program_01**, **Program_02**, **Program_03**, **Program_04**, **Program_05** a **Program_Ini**. Uvedené názvy jsou povinné včetně velikosti písmen.

Programový modul *.tgm nesmí být propojen s jakýmikoli externími moduly či knihovnami DLL, veškerá programová funkčnost (funkce, podprogramy, …) musí být vytvořena programátorem přímo v modulu. Nesmějí se volat jakékoli funkce API Windows, je zakázáno dynamicky alokovat paměť (funkce **malloc** apod.).

Všechny dále uvedené příklady zdrojových kódů jsou uváděny pro programovací jazyk C/C++.

1.3 Vyžadovaný algoritmus PLC

Obecný algoritmus PLC vyžaduje následující postup vykonávání, a to přesně v uvedeném pořadí:

Načtení vstupů

V první fázi je nutné načíst hodnoty registrů všech požadovaných vstupů. A to jak polohy a stavy servopohonů, tak i hodnoty digitálních a analogových vstupů (viz strukturu Servo a strukturu I/O).

Zpracování hodnot a výpočty

Dalším krokem je zpracování načtených hodnot a výpočty hodnot nových (žádané polohy servopohonů a hodnoty digitálních a analogových výstupů).

Nastavení výstupů

Posledním krokem je zaslání žádaných poloh servopohonům a nastavení hodnot digitálních a analogových výstupů prostřednictvím zapsání hodnot do příslušných registrů.

1.4 Komunikace s PLC

Ostatní komponenty systému **TG Motion** a uživatelské aplikace běžící pod Windows mohou komunikovat s PLC pomocí sdílené paměti **TGM_Data**. **TG Motion** do ní nijak nezasahuje a její obsah nijak nepřepisuje. Struktura této paměti a její využití je plně v režii programátora PLC.

Velikost sdílené paměti **TGM_Data** je obvykle 524 288 byte. Skutečnou velikost paměti obsahuje registr **TGM_System.HEADER.Mem_Size_Data**.

Paměť může obsahovat uživatelské registry, data vaček, atd. Nejčastěji je využívána aplikacemi pro vizualizaci běžícími pod Windows.

2. Funkce PLC

2.1 Obecný popis funkcí

Programovací modul Virtuálního PLC má k dispozici právě 6 funkcí. Ty se liší prioritou provádění a požadavkem na své úplné provedení během jednoho **Cycle_Time**. Pro správný a bezchybný chod Virtuálního PLC je nutné, aby požadovaný kód byl umístěn v příslušné funkci (viz dále):

Program_Ini( ) – funkce volaná pouze jednou při spuštění PLC.
Program_01( ) – nejnižší priorita, volitelná perioda volání.
Program_02( ) – nižší priorita, volitelná perioda volání.
Program_03( ) – vyšší priorita, volitelná perioda volání.
Program_04( ) – nejvyšší priorita, volá se každý Cycle_Time.
Program_05( ) – nejvyšší priorita, volá se synchronně s výpočtem poloh interpolátoru (vícekrát během jednoho Cycle_Time).

Všechny funkce musejí být implementovány, některé nemusejí obsahovat výkonný kód. Vždy je nutné zabezpečit návratovou hodnotu funkce. Viz příklad kódu funkce Program_02.

2.2 Struktura PLC_DATA

Struktura slouží pro komunikaci mezi **TG Motion** a PLC. **TG Motion** vytvoří 6 instancí struktury **PLC_DATA**. Každá z 6 funkcí (Program_01 – Program_05, Program_Ini) má právě 1 parametr, ukazatel na strukturu **PLC_DATA**. Při volání funkce jí **TG Motion** předá ukazatel na jí náležející instanci struktury.

Struktura **PLC_DATA** obsahuje ukazatele na sdílené paměti a ukazatele na vnitřní diagnostické funkce **TG Motion**.

Definice diagnostických funkcí

```
#ifdef __cplusplus
extern "C" {
#endif
  typedef int _cdecl RTWPRINTF_STRING(LPCWSTR strText); // výpis řetězce do konzole
  typedef int _cdecl RTWPRINTF_LONG(LPCWSTR strFormat, long lVal);
                                                        // výpis hodnoty proměnné lVal do konzole
  typedef int __cdecl SWPRINTF(wchar_t *buffer, size_t sizeOfBuffer, const wchar_t *strFormat, ...);
                                                        // zápis formátovaného textu do řetězce buffer
  typedef void __cdecl SLEEPFT(PLARGE_INTEGER Pause);   // čekání
  typedef BOOL __cdecl CAN_TRANSMIT(ULONG Number, ULONG Id, ULONG Dlc, PUCHAR Tx_Data );         // rezervováno
  typedef BOOL __cdecl CAN_TRANSMITREMOTE(ULONG Number, ULONG Id, ULONG Dlc, PUCHAR Tx_Data);    // rezervováno
  typedef int __cdecl RTWPRINTF_EX(int severity, const wchar_t *strFormat, ...);
                                                        // výpis formátovaného textu do konzole
#ifdef __cplusplus
};
#endif
```

Ukazatele na diagnostické funkce

```
typedef struct _PLC_IMPORT_FUNCTIONS
{
  RTWPRINTF_STRING      *pRtWprintf_String;
  RTWPRINTF_LONG        *pRtWprintf_Long;
  SWPRINTF              *pswprintf;
  SLEEPFT               *pSleepFt;
  CAN_TRANSMIT          *pCAN_Transmit;
  CAN_TRANSMITREMOTE    *pCAN_TransmitRemote;
  RTWPRINTF_EX          *pRtWprintf_Ex;
} PLC_IMPORT_FUNCTIONS;
```

Vlastní struktura PLC_DATA

```
typedef struct _PLC_DATA
{
  size_t structSize;                          // velikost struktury v bytech
  void *PSystem_Memory;                       // ukazatel na sdílenou paměť TGM_System
  void *PData_Memory;                         // ukazatel na sdílenou paměť TGM_Data
  void *POsc_Memory;                          // ukazatel na sdílenou paměť TGM_Oscilloscope
  void *PCam_Memory;                          // ukazatel na sdílenou paměť TGM_Cam_Profile
  void *PServo_Memory;                        // ukazatel na sdílenou paměť TGM_Servo
  void *PDio_Memory;                          // ukazatel na sdílenou paměť TGM_Dio
  void *PInterpolator_Memory;                 // ukazatel na sdílenou paměť TGM_Interpolator
  void *Pointer_interpolator_params;          // ukazatel na sdílenou paměť TGM_InterpolatorWriteMemory
  void *Pointer_interpolator_get_position;    // ukazatel na sdílenou paměť TGM_InterpolatorReadMemory
  void *PCNCEx;                               // ukazatel na sdílenou paměť TGM_CNCEX
  void *PGCode;                               // ukazatel na sdílenou paměť TGM_GCODE
  void *PReserve3_Memory;                     // rezervováno
  void *PReserve4_Memory;                     // rezervováno
  void *PReserve5_Memory;                     // rezervováno
  void *PReciveDataCan1;                      // rezervováno
  void *PReciveDataCan2;                      // rezervováno

  PLC_IMPORT_FUNCTIONS functions;             // struktura s ukazateli na diagnostické funkce
} PLC_DATA, *PPLC_DATA;
```

2.3 Funkce podle priority

V této kapitole jsou funkce řazeny od nejnižší priority po prioritu nejvyšší. U funkce **Program_Ini** se o prioritu v pravém slova smyslu nejedná. Tato funkce je volána pouze jednou při spuštění Virtuálního PLC.

Program_Ini

Deklarace: **`long Program_Ini(PLC_DATA *pData)`**

Tato funkce je volaná jen jednou, a to při spuštění Virtuálního PLC. Slouží zejména k inicializaci proměnných Virtuálního PLC. Výkonný kód funkce **Program_Ini** nesmí být prázdný. Délka vykonávání funkce není omezena.

Při startu Virtuálního PLC (v těle funkce Program_Ini) je vhodné zkontrolovat verze PLC a TG Motion.

Návratové hodnoty funkce

0 – chyba (spouštění PLC se zastaví; uživatel musí chybu vyřešit a znovu spustit PLC).
1 – funkce proběhla v pořádku.

*Při nahrání Virtuálního PLC se neinicializují hodnoty globálních proměnných, nevolají se konstruktory globálních objektů. Inicializaci je nutné provést ve funkci **Program_Ini**.*

Ukázka zdrojového kódu

```
long Program_Ini(PLC_DATA *pData)
{
  if (pData == NULL || pData->structSize != sizeof(PLC_DATA))    return 0;
  if (pData->functions.pRtWprintf_Long == NULL)                  return 0;
  if (pData->functions.pRtWprintf_String == NULL)                return 0;
  if (pData->functions.pswprintf == NULL)                        return 0;
  if (pData->functions.pSleepFt == NULL)                         return 0;
  if (pHeader->Compatibility_Id != ID_COMPATIBILITY)
  {
        pData->functions.pswprintf(info_ini, SIZE_INFO, L"Error start of PLC PLC_COMPABILITY_ID = %d
TGM_COMPABILITY_ID = %d \n", ID_COMPATIBILITY, pHeader->Compatibility_Id);
        pData->functions.pRtWprintf_String(info_ini);
        return 0;
  }

  //************************************** Update PLC Version **********************************

  Verze_PLC = Get_Version(PLC_VERSION);

  //********************************************************************************************

  return 1;
}
```

Program_01

Deklarace: **`long Program_01(PLC_DATA *pData)`**

Funkce je z **TG Motion** volána s periodou danou v konfiguračním souboru **TGMotion4xx.ini**. Perioda je definována položkou **Cycle_Time_Program_01**, její hodnota se pohybuje v rozmezí 100–10 000 µs (horní hranice není omezena). Délka vykonávání funkce by neměla přesáhnout 20 % **Cycle_Time_Program_01**, aby zbyl čas na vykonání funkcí **Program_02** a **Program_03**.

Tato funkce se nejčastěji používá pro základní obsluhu zařízení, která nemusejí být obsluhována pravidelně každý **Cycle_Time**.

Funkce **Program_01** má nejnižší prioritu a kdykoli může být přerušena funkcemi **Program_02**, **Program_03**, **Program_04** a **Program_05**.

Program_02

Deklarace: **`long Program_02(PLC_DATA *pData)`**

Funkce je z **TG Motion** volána s periodou danou v konfiguračním souboru **TGMotion4xx.ini**. Perioda je definována položkou **Cycle_Time_Program_02**, její hodnota se pohybuje v rozmezí 100–10 000 µs (horní hranice není omezena). Délka vykonávání funkce by neměla přesáhnout 20 % **Cycle_Time_Program_02**, aby zbyl čas na vykonání funkcí **Program_01** a **Program_03**.

Tato funkce se většinou implementuje jako prázdná funkce.

Funkce **Program_02** má nízkou prioritu a může být kdykoli přerušena funkcemi **Program_03**, **Program_04** a **Program_05**.

Ukázka zdrojového kódu

```
long Program_02(PLC_DATA *pData)
{
  return 1;
}
```

Program_03

Deklarace: **`long Program_03(PLC_DATA *pData)`**

Funkce je z **TG Motion** volána s periodou danou v konfiguračním souboru **TGMotion4xx.ini**. Perioda je definována položkou **Cycle_Time_Program_03**, její hodnota se obvykle pohybuje v rozmezí 100–10 000 µs (horní hranice není omezena). Délka vykonávání funkce by neměla přesáhnout 20 % **Cycle_Time_Program_03**, aby zbyl čas na vykonání funkcí **Program_01** a **Program_02**.
Tato funkce se většinou implementuje jako prázdná funkce.

Funkce **Program_03** má střední prioritu a může být kdykoli přerušena pouze funkcemi **Program_04** a **Program_05**.

Ukázka zdrojového kódu

```
long Program_03(PLC_DATA *pData)
{
  return 1;
}
```

Program_04

Deklarace: **`long Program_04(PLC_DATA *pData)`**

Funkce je z **TG Motion** volána synchronně s komunikací v rámci **Cycle_Time**, tedy jednou během každého **Cycle_Time**. Ten je definován v souboru **TGMotion4xx.ini** položkou **Cycle_Time** (250 µs, 500 µs, 1000 µs).

Funkce **Program_04** se nejčastěji používá pro modifikaci žádané polohy servopohonů a obsluhu I/O jednotek. Má nejvyšší prioritu (stejně jako funkce **Program_05**) a vždy se vykoná celá bez přerušení.

Délka vykonávání funkce **Program_04** nesmí přesáhnout 10 % **Cycle_Time**, aby byla zabezpečena časová přesnost komunikace se servopohony a I/O jednotkami.

*Ve funkci **Program_04** nesmí být volána funkce **SleepFt**.*

Program_05

Deklarace: **`long Program_05(PLC_DATA *pData)`**

Funkce je z **TG Motion** volána synchronně s interpolátorem, tedy několikrát během každého **Cycle_Time**.

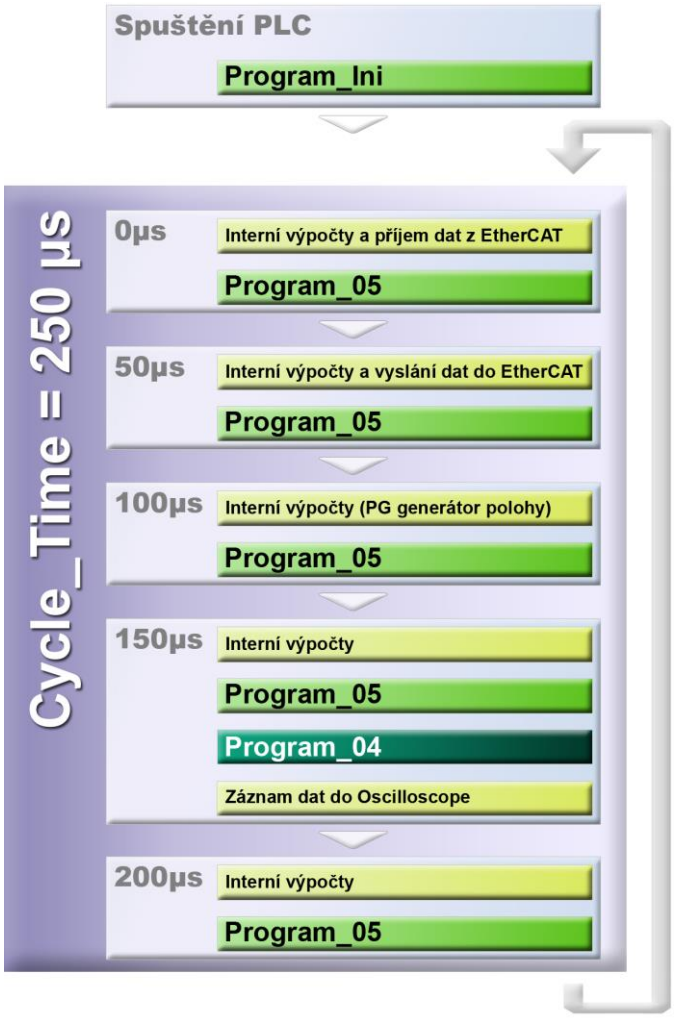
Funkce **Program_05** se nejčastěji používá pro modifikaci polohy vypočtené modulem interpolátoru. Má nejvyšší prioritu (stejně jako funkce **Program_04**) a vždy se vykoná celá bez přerušení.

Délka vykonávání funkce **Program_05** nesmí přesáhnout 10 µs, aby byla zabezpečena časová přesnost komunikace se servopohony a I/O jednotkami.

*Ve funkci **Program_05** nesmí být volána funkce **SleepFt**.*

2.4 Časová souslednost volání funkcí

Cycle_Time = 250 µs



Obr. Algoritmus vykonávání PLC pro Cycle_Time = 250 µs

Po spuštění PLC a úspěšném vykonání funkce **Program_Ini** se spustí cyklické volání smyčky trvající 250 µs. Ta je rovnoměrně rozdělena na 5 stejných časových úseků volaných pravidelně každých 50 µs.

Čas 0 µs

Provedou se potřebné interní výpočty a přijmou se data z EtherCAT. Poté se zavolá funkce **Program_05**, která se vykoná celá bez přerušení.

Čas 50 µs

Provedou se potřebné interní výpočty a pošlou se data do EtherCAT. Poté se zavolá funkce **Program_05**, která se vykoná celá bez přerušení.

Čas 100 µs

Provedou se potřebné interní výpočty a vypočtou se data žádané polohy pomocí PG generátorů. Poté se zavolá funkce **Program_05**, která se vykoná celá bez přerušení.

Čas 150 µs

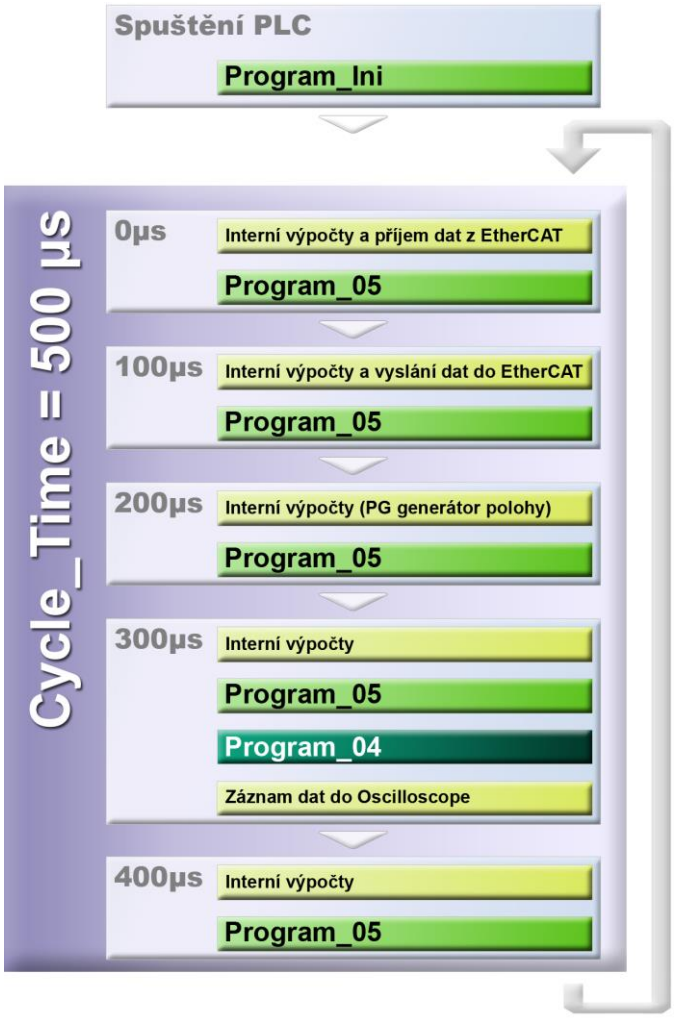
Provedou se potřebné interní výpočty a zavolá se funkce **Program_05**, která se vykoná celá bez přerušení. Poté se zavolá funkce **Program_04**, která se také vykoná celá bez přerušení. Nakonec se zaznamenají všechna data potřebná pro Oscilloscope.

Čas 200 µs

Provedou se potřebné interní výpočty a zavolá se funkce **Program_05**, která se vykoná celá bez přerušení.

Pokud je během kteréhokoli cyklu volná výpočetní kapacita, jsou v případě potřeby obsluhovány funkce **Program_01**, **Program_02** a **Program_03**.

Cycle_Time = 500 µs



Obr. Algoritmus vykonávání PLC pro Cycle_Time = 500 µs

Po spuštění PLC a úspěšném vykonání funkce **Program_Ini** se spustí cyklické volání smyčky trvající 500 µs. Ta je rovnoměrně rozdělena na 5 stejných časových úseků volaných pravidelně každých 100 µs.

Čas 0 µs

Provedou se potřebné interní výpočty a přijmou se data z EtherCAT. Poté se zavolá funkce **Program_05**, která se vykoná celá bez přerušení.

Čas 100 µs

Provedou se potřebné interní výpočty a pošlou se data do EtherCAT. Poté se zavolá funkce **Program_05**, která se vykoná celá bez přerušení.

Čas 200 µs

Provedou se potřebné interní výpočty a vypočtou se data žádané polohy pomocí PG generátorů. Poté se zavolá funkce **Program_05**, která se vykoná celá bez přerušení.

Čas 300 µs

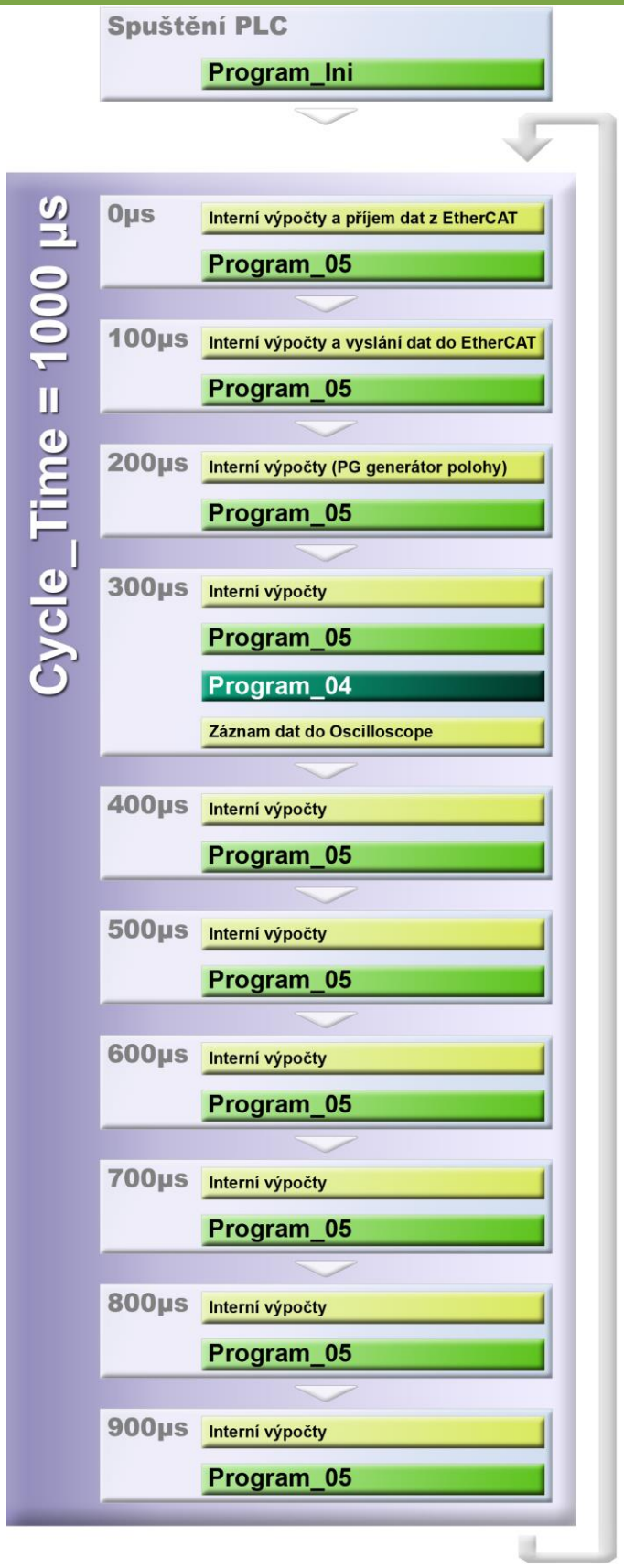
Provedou se potřebné interní výpočty a zavolá se funkce **Program_05**, která se vykoná celá bez přerušení. Poté se zavolá funkce **Program_04**, která se také vykoná celá bez přerušení. Nakonec se zaznamenají všechna data potřebná pro Oscilloscope.

Čas 400 µs

Provedou se potřebné interní výpočty a zavolá se funkce **Program_05**, která se vykoná celá bez přerušení.

Pokud je během kteréhokoli cyklu volná výpočetní kapacita, jsou v případě potřeby obsluhovány funkce **Program_01**, **Program_02** a **Program_03**.

Cycle_Time = 1000 µs



Obr. Algoritmus vykonávání PLC pro Cycle_Time = 1000 µs

Po spuštění PLC a úspěšném vykonání funkce **Program_Ini** se spustí cyklické volání smyčky trvající 1000 µs. Ta je rovnoměrně rozdělena na 10 stejných časových úseků volaných pravidelně každých 100 µs.

Čas 0 µs

Provedou se potřebné interní výpočty a přijmou se data z EtherCAT. Poté se zavolá funkce **Program_05**, která se vykoná celá bez přerušení.

Čas 100 µs

Provedou se potřebné interní výpočty a pošlou se data do EtherCAT. Poté se zavolá funkce **Program_05**, která se vykoná celá bez přerušení.

Čas 200 µs

Provedou se potřebné interní výpočty a vypočtou se data žádané polohy pomocí PG generátorů. Poté se zavolá funkce **Program_05**, která se vykoná celá bez přerušení.

Čas 300 µs

Provedou se potřebné interní výpočty a zavolá se funkce **Program_05**, která se vykoná celá bez přerušení. Poté se zavolá funkce **Program_04**, která se také vykoná celá bez přerušení. Nakonec se zaznamenají všechna data potřebná pro Oscilloscope.

Čas 400 µs, 500 µs, 600 µs, 700 µs, 800 µs, 900 µs

Ve všech těchto časových úsecích se provedou potřebné interní výpočty a zavolá se funkce **Program_05**, která se vykoná celá bez přerušení.

Pokud je během kteréhokoli cyklu volná výpočetní kapacita, jsou v případě potřeby obsluhovány funkce **Program_01**, **Program_02** a **Program_03**.

3. Nástroje pro ladění PLC

3.1 Control Observer

Hlavním nástrojem pro ladění Virtuálního PLC z prostředí Windows je Control Observer. Je dodáván se systémem **TG Motion**.

Jedná se o soubor utilit vyvinutý pro diagnostiku systému **TG Motion**, odlaďování PLC a obslužných Windows aplikací. Control Observer obsahuje nástroje pro přímé testování a ovládání servopohonů, načtení a spuštění PLC kódu a zobrazení parametrů systémového časovače. Další skupina utilit slouží k zobrazování, sledování, či změně zvolených registrů sdílené paměti.

Control Observer je samostatně spustitelný program **Control_Observer_II.exe** bez nutnosti instalace, který se dodává s 3 knihovnami:

TGM_Comm_Int_2.dll – zabezpečuje komunikaci s TG Motion běžícím na stejném počítači.
TGM_Mini.dll – obsluhuje **TG Motion** běžící v TGMmini.
TGM_Remote.dll – umožňuje spojení pomocí sítě LAN s TG Motion běžícím na jiném počítači.

Pro přístup k datům sdílené paměti TGM_Data slouží v utilitě Select Registers záložka Free Registers, typ paměti DAT.

Součásti Control Observeru

Servo Tester – utilita k testování a ovládání servopohonů.
PLC Loader – slouží k načtení PLC a jeho spuštění.
System Timer – zobrazuje aktuální vytížení CPU jednotlivými procesy TG Motion.
Oscilloscope – slouží ke grafickému zobrazování hodnot vybraných registrů v závislosti na čase.
Graphic Viewer – používá se ke grafickému zobrazení kontinuální řady vybraných registrů.

Pro podrobnější popis viz kapitolu Control Observer.

3.2 Výpisy

Pro výpisy do konzole RTX Server se používají diagnostické funkce ze struktury **PLC_DATA**. Jedná se o 3 typy diagnostických funkcí:

– *výpis řetězce do konzole*
– *výpis hodnoty proměnné* `lVal` *(hodnota typu long) do konzole*
– *výpis formátovaného textu do konzole*

3.3 Oscilloscope

Oscilloscope je samostatná utilita běžící v **TG Motion**. Slouží k zachycení hodnot požadovaných registrů v přesném časovém intervalu **Cycle_Time** a jejich ukládání do sdílené paměti **TGM_Oscilloscope**. To se děje ihned po vykonání funkce **Program_04**.

O zobrazení zachycených dat a nastavení parametrů ovlivňujících zaznamenávání hodnot se stará utilita Oscilloscope v Control Observeru.

Pro podrobnější popis viz kapitolu Oscilloscope.

3.4 Aplikace Windows

Přístup do sdílených pamětí mají také aplikace běžící pod operačním systémem Windows. Jejich prostřednictvím lze číst hodnoty registrů, případně je i měnit.

***TG Motion** běží v real-time prostředí, tedy s vyšší prioritou, než mají procesy běžící pod systémem Windows. Z Windows aplikací tedy nelze zajistit bezeztrátové zachycení všech potřebných hodnot, případně operativní reakci na nastalé situace.*