UM0541
User manual

DCP monitor

Introduction

CPU commander is a user-friendly tool to perform data logging or to control and to monitor software applications running on MCU targets such as EVA boards, customer modules or application boards. Using CPU commander and its collection of visual instruments makes it possible to quickly build control panels for a large number of application fields.

The user can connect CPU commander to a target device using different communication systems. If the user connects CPU commander to the target device via RS232, CPU commander uses a specific protocol called DCP that is described in *Section 1.1.1: DCP protocol*. In this case, a software called DCP monitor must be present on the target side to establish the communication between CPU commander and the target.

This document provides a complete guideline to porting and to configure the DCP monitor in a target device project. *Chapter 1: Architecture* gives a brief description of the architecture and the DCP protocol. *Chapter 2: Using DCP monitor* describes how to use the DCP monitor and provides details on its configuration and initialization. *Chapter 3: DCP-Application Programming Interface* describes the API of the DCP monitor.

Requirements

CPU commander (PCs) has been designed to work with Windows 2000 and Windows XP platforms and with no particular hardware requirements. However, in order to connect your PC to the target, either a UART port or a USB port and an ST COMBox kit (driver and board) must be installed in your PC.

Contents

1 Architecture

1.1 DCP monitor

DCP monitor consists of the following files:

- DCP.c - contains all functions of the DCP monitor
- DCP.h - header file used to configure the DCP monitor
- DCPPort_xxx.c - contains all functions specific for the target
- DCPPort_xxx.h - header file for DCPPort_xxx.c

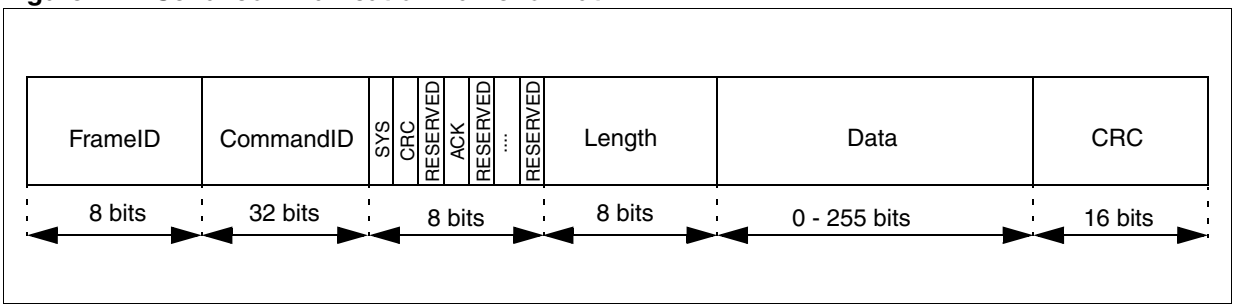
All these files have to be included in the target side project to implement the DCP monitor.

1.1.1 DCP protocol

DCP protocol is used by CPU commander during a UART communication.

Figure 1 describes the format of a DCP frame.

Figure 1. Serial communication frame format



FrameID (Frame Identifier): is used only if acknowledge management is set. It is a counter 8 bits long and identifies the frame.

Command ID (Command Identifier): is the identifier of the frame and is 32 bits long.

SYS (System) flag: when it is set, the frame is a system frame.

CRC (Cyclic Redundancy Check) flag: when it is set the CRC is present in the frame otherwise not.

ACK (Acknowledge) flag: when it is set, the receiver must send an acknowledge frame.

Length: this field is 8 bits long and indicates the length of the data in bytes (from 0 to 255 bytes) present in the current frame.

Data: it contains from 0 up to 255 bytes of data.

CRC: if CRC flag is set the last filed contains the CRC value. It is 16 bits long.

1.2 Communication mechanism

1.2.1 Introduction

This section describes the transmission mechanism and the reception mechanism between CPU commander and a target device.

1.2.2 Transmission

To send data, the sender must build a DCP frame and insert information in each field.

The sender has to perform these steps:

1. Insert the CommandID.
2. Set the correct flags for the functionality requested (SYS, ACK, CRC).
3. Insert the Length of the bytes of the data.
4. Insert the data.
5. Calculate the CRC if the CRC flag is set and insert the value at the end of the frame.
6. Call the function SendFrame(frame).

1.2.3 Reception

The reception is interrupt implemented.

Using UART communication, the receiver receives one byte at a time, so the number of the byte received depends from the properties of each frame.

If the acknowledge management is activated, the first seven bytes (FrameID, CommandID, Flags, Length) are mandatory. If the acknowledge management is not activated the first six bytes are always present (CommandID, Flags, Length). After the reception of the Length, it is possible to know how many bytes of data the receiver has to receive. After the receiver has received all data, it waits for CRC flag if the CRC flag is set, otherwise the reception is complete.

When the frame is complete, the receiver can pass the frame to the DCP protocol for the parsing calling DCP_ProcessFrame() function.

If the frame is a system (SYS) frame, DCP protocol executes the task programmed for the specific SYS. If the frame is an acknowledge frame (ACK), the protocol checks if the FrameID of the frame received corresponds to the expected FrameID. If the FrameID does not match, an error occurs.

If the CommandID of the frame is a user CommandID, the management of this frame is passed to the CallBack function managed by the user. In fact, the user has to insert the managed frame into the Callback function with a CommandID which is specific to the application.

1.2.4 ACK management

If the application uses acknowledge management, each frame will contain a FrameID field that identifies the associated frame. The FrameID is a counter that is increased for each frame sent. When the frame is sent, the sender waits for the reception of the corresponding acknowledge frame and it remains blocked in the DCP_WaitACK function.

When the receiver receives a data frame, it checks for the CRC (only if activated), builds and sends the acknowledge frame by calling the DCP_SendACK function.

Following the task executed by DCP_SendACK function:

- The function inserts the FrameID of the frame received into the FrameID of the acknowledge frame. Next, it inserts the value DCC_FACK into the CommandID field if the CRC calculation is correct otherwise it inserts the value DCC_CRC_ERR. At the end of the frame it inserts the value zero into flags and length fields.
- The last step is to call the function DCP_WriteFrame to send the frame.

At this time the sender receives the acknowledge frame and it checks if the FrameID received is the same as the FrameID attended. If it is true, the receiver checking for the CommandID of the ACK. The CommandID can be DCC_FACK and in this case the ACK is correct, otherwise it can be DCC_CRC_ERR and an error occurs.

If the ACK is lost, a timeout occurs and an error is generated. This timeout can be set during configuration phase (it can be different depending on the type of communication system and the power of the target's processor).

For each error generated, the receiver sends an error frame to the sender.

1.2.5 Resend management

This feature is enabled if the acknowledge mechanism is present. It allows to resend a frame if the sender does not receive any acknowledge frame.

If the DCP_WaitACK ends because a timeout has occurred, the sender can resend the frame up to the SendFrameTries maximum number of times.

1.2.6 CRC management

If the CRC management is enabled, either the sender or the receiver has to execute a specific task to manage the CRC.

When the sender has the frame ready to send, calculates the CRC using the function ComputeCRC().

Following the algorithm used to calculate the CRC:

- If the frame has an odd number of bytes, the sender adds a byte equal to zero until the frame has an even number of bytes.
- The sender executes an XOR operation between all 8-bit long words of the frame.
- The sender adds the 16-bit long value inside the CRC field. At this time the frame is ready to be sent.

When the receiver receives the frame, it calculates the CRC using the same function. Only when this value is calculated does the receiver compare the CRC received with the CRC calculated. If the values are the same it means that the receiver has received the frame correctly otherwise an error occurs. If the acknowledge mechanism is activated, the ACK frame contains the CRC error.

2 Using DCP monitor

2.1 Defining behavior

DCP monitor can work in two ways:

- Polling
- Autonomously

This means that the user can query information from an external tool (usually CPU commander) or the application can send the information by itself.

2.1.1 Autonomously

In order to communicate with this mode, the user can call whenever needed within the code the DCP_SendFrame function.

For parsing, the user should customize and register the DCP_Callback function.

Within this function all the necessary processing should be done. Parsing of the received user command.

2.1.2 Polling

In order to work with this mode, no specifications are needed.

The external tool instructs the target to read or write data using the DCC_READ and DCC_WRITE commands.

In this case there is no need of coding for the user. The only operation required is to initialize the DCP by calling the DCP_Init function.

Note: About endianness: the protocol works with big endianness. This means that all information sent must be converted to big endian and all the information received must be converted to the endianness of the receiving platform.

2.2 Configuring DCP

2.2.1 Communication API configuration

COM_API defines which communication API is used.

2.2.2 ACK management

ACK management can be enabled with ACK_MANAGEMENT_SEND.

ACK management on Rx can be enabled with ACK_MANAGEMENT_WAIT, specifying the timeout for waiting in ms with ACK_MANAGEMENT_TIMEOUT. If the sender is waiting for an ACK and it receives a frame, it simply processes the frame and still remains waiting for the ACK.

2.2.3 DCP version

The version of the protocol is specified by DCP_VERSION.

2.2.4 Communication error management

Resend of not acknowledge frames can be activated by RESEND_ON_ERROR. In this case, MAX_SEND_TRIES specifies the number of tries and MAX_WAIT_LOOPS the time for acknowledgement waiting.

2.3 Initializing the monitor

To initialize the DCP monitor, the user has to call the DCP_Init function that initialize the used target communication interface. The user also has to call the following functions:

- DCP_Register (DCCallback): DCCallback is a pointer to callback that processes received frames.
- DCP_RegisterErrorHandler (DCErrorHandler): DCErrorHandler is a pointer to function that processes errors.

2.4 Writing the callback

This function must be implemented by the user in application code. It contains the management of the frame with specifics CommandID for the application. In fact, the user has to associate some specific task to a specific CommandID.

2.4.1 Sample

```
UINT32 DCCallback(UINT32 pFrame)
{
  FRAME frame = {0};

  // Handle DCP messages
  switch(((FRAME*)pFrame)->CommandID)
 {
    case IDC_LED
    {
      // Switch on/off the LED
      SetPort(LED_PORT, frame.Param32)
      break;
    }
    case IDC_GET_FIRMWARE_VERSION:
    {
     // Initializes reply frame
        framex.CommandID = IDC_FIRMWARE_VERSION;
        frames.length = 1;
        framex.Data[0] = FIRMWARE_VERSION;
       DCP_SendFrame(&frame);
       break;
    }
    default:
      break;
  }
  return DCN_SUCCESS;
}
```

3 DCP-Application Programming Interface

3.1 Introduction

This section describes the API (Application Programming Interface) for the DCP protocol. Following a list of the functions implemented to realize the communication using DCP protocol:

- DCP_Register
- DCP_RegisterErrorHandler
- DCP_SendFrame
- DCCallback
- DCErrorManagement

3.2 API reference

3.2.1 DCP_register

Definition

UINT32 DCP_register(PFN_DCB2 pfnDCB).

Arguments

pfnDCB [in] a pointer to callback that processes received frames.

Returns

This function returns DCN_SUCCESS on successful completion, DCN_BAD_PARAMETER if pfnDCB is NULL.

Description

The function DCP_register is used by the user application to register the callback function defined by the user.

3.2.2 DCP_RegisterErrorHandler

Definition

UINT32 DCP_RegisterErrorHandler(PFN_DEH pfnDEH).

Arguments

pfnDEH [in] a pointer to function that processes errors.

Returns

This function returns DCN_SUCCESS on successful completion, DCN_BAD_PARAMETER if pfnDEH is NULL.

Description

The function DCP_RegisterErrorHandler is used by the user application to register the error handler function defined by the user.

3.2.3 DCP_SendFrame

Definition

UINT32 DCP_SendFrame(UINT8 u8Channel, LPFRAME pFrame).

Arguments

u8Channel[in] number of the channel;

pFrame [in] pointer to the frame to send.

Returns

This function returns DCN_SUCCESS on success, DCN_BAD_PARAMETER on bad parameter, DCN_ERROR otherwise

Description

This function is used to send a frame and to wait for the acknowledge frame if the acknowledgement mechanism is enabled. If the timeout occurred, the function sends the frame again up to a maximum number of times defined by user.

3.2.4 DCP_Callback

Definition

UINT32 DCCallback(LPFRAME pFrame).

Arguments

pFrame[in], a pointer to the received frame.

Returns

This function returns DCN_SUCCESS if the frame has been handled, DCN_NOT_IMPLEMENTED if has not been handled, DCN_ERROR on error.

Description

This function is implemented by the user and contains the management of the used defined frame.

3.2.5 DCErrorManagement

Definition

UINT32 DCErrorManagement(UINT32 ErrCode).

Arguments

ErrCode[in], a code that describes the type of error.

Returns

This function returns DCN_SUCCESS if the error has been handled, DCN_NOT_IMPLEMENTED if has not been handled, DCN_ERROR on error.

Description

This function is implemented by the user and contains the management of the error occurred.

3.2.6 DCP_Init()

Definition

UINT32 DCP_Init(void).

Arguments

Void.

Returns

This function returns DCN_SUCCESS on success and DCN_MUTEX_ERR on error in mutex's creation.

Description

Initializes all peripherals for DCP.

Create semaphores for synchronization.

3.3 Data structures and constants

Trace

To show debug messages

```
#define TRACE OutputDebugString
```

Trace1

To show debug messages

```
#define TRACE1(a, b)
{
   char sz[256]="";
   wsprintf(sz, (a), (b));
   OutputDebugString(sz);
}
```

Trace2

To show debug messages

```
#define TRACE2(a, b, c)
{
   char sz[256]="";
   wsprintf(sz, (a), (b), (c));
   OutputDebugString(sz);
}
```

3.3.1 Type definition

Tag_frame

Structure that contains the frame section

```
Typedef struct tagFRAME
   {
      UITN8        FrameID;
      UINT32       CommandID;
      BITFIELD     SYS : 1;
      BITFIELD     CRC : 1;
      BITFIELD     rsvd: 1;
      BITFIELD     ACK: 1;
      BITFIELD     rsvd: 4;
      UITN8        lENGTH;
      UITN8        Data[];
   } FRAME;
```

LPframe

Pointer to a frame structure

```
typedef FRAME* LPFRAME;
```

PFN_DCB2

Function pointer to the ProcessFrame function

```
typedef UINT32 (*PFN_DCB2)(UINT32 LPFRAME pFrame);
```

PFN_DEH

Function pointer to the error handling function

```
typedef UINT32 (*PFN_DEH)(UINT32 u32Error);
```

3.3.2 Configuration constants

The following #define in DCP.h can be used to configure protocol behavior.

Table 1. Constants in DCP.h

Name	Description
ACK_MANAGEMENT	Transmit ACK frames
ACK_MANAGEMENT_TIMEOUT	ACK waiting delay in ms
RESEND_ON_ERROR	Uncomment to activate resend of frames on error
MAX_SEND_TRIES	Max number of send retries
CHECK_PARAMETERS	Enable check arguments
USE_ERROR_HANDLER	Enable error handler
NO_TRACE	Disable debug messages
DCP_VERSION	Version of DCP Protocol
DCP_NOTIFICATIONS	To avoid of redefine DCN_ constants
DCP_COMMANDS	To avoid of redefine DCC_ commands
PFN_DCB2_DEFINED	To avoid of redefine PFN_DCB
PFN_DEH_DEFINED	To avoid of redefine PFN_DCB

The following parameters, set by the compiler, are used to conditionally compile code for a specific platform.

The following #define in DCPPort_x.h can be used to configure protocol behavior.

Table 2. Constants in DCPPort_x.h

Name	Description
ACK_MANAGEMENT_TIMEOUT	Time for acknowledgement waiting
DCP_ENDIANNESS	Set the endianness
DCP_COM_VIA_UART	Set UART as communication channel
DCP_COM_VIA_CAN	Set CAN as communication channel

The following parameters, set by the compiler, are used to conditionally compile code for a specific platform.

Table 3. Preprocessor definitions

Name	Description
WIN32	Includes this block of code if compiling for Windows.
__CC_ARM	Includes this block of code if compiling for ARM.
__ppc	Includes this block of code if compiling for PPC.

3.3.3 Command reference

Table 4. Command reference

CommandID	Value	Description
DCC_SYS0	0x80	System message
DCC_SYS1	0x81	System message 1
DCC_SYS2	0x82	System message 2
DCC_SYS3	0x83	System message 3
DCC_SYSNC_ACK	0x84	Synchronization acknowledgement
DCC_FACK	0x85	Frame successfully received
DCC_CRC_ERR	0x86	CRC Error
DCC_READ	0x87	Read memory location
DCC_WRITE	0x88	Write memory location
DCC_DATA	0x89	Data message
DCC_INFO	0xC2	User Frame 2
DCC_USR3	0xC3	User Frame 3
DCC_USR4	0xC4	User Frame 4
DCC_USR5	0xC5	User Frame 5
DCC_USR6	0xC6	User Frame 6
DCC_USR7	0xC7	User Frame 7
DCC_USR8	0xC8	User Frame 8
DCC_USR9	0xC9	User Frame 9

3.3.4 Return codes

The following codes are returned by DCP functions.

Table 5. Return codes

Name	Description
DCN_SUCCESS	Operation completed successfully
DCN_ERROR	Generic error occurred
DCN_BAD_COMMAND	Bad command received
DCN_BAD_PARAMETER	Bad parameter received
DCN_NOT_IMPLEMENTED	Function not implemented
DCN_NO_POWER	Not used by DCP
DCN_NO_DATA	Not used by DCP
DCN_NOT_RUNNING	Not used by DCP
DCN_READ_ONLY	Not used by DCP
DCN_WRITE_ONLY	Not used by DCP
DCN_TIMEOUT	Timeout occurred during the operation
DCN_WRONG_SYSFRAME	An unexpected SYS frame has been received
DCN_WRONG_NACK	An incorrect ACK has been received
DCN_WRONG_FID	A frame with a wrong FrameID has been received
DCN_WRONG_CRC	A frame with a wrong CRC has been received
DCN_NOT_ENOUGH_MEMORY	Not enough memory to complete the operation
DCN_MUTEX_ERROR	On error occurred in mutex's creation
DCN_INFO	A frame with DCP configuration information

Appendix A Software end-user agreement

End-user license agreement

You (“you”) should carefully read the following terms and conditions before using this software (the “software”) which is licensed by STMicroelectronics nv (“ST”) to its customers for their use only as set forth below.

Acceptance

If You agree with and accept the terms and conditions of this agreement it shall become a legally binding agreement between You and ST and You may proceed to install, copy and use the software in accordance with the terms and conditions of the agreement.

Rejection and right to a refund

If You do not agree with or do not wish to be bound by the terms and conditions of this agreement You may NOT install, copy or use the software.

License

ST hereby grants to You, subject to the terms and conditions of this agreement, a non-exclusive, non-transferable, worldwide license, without the right to sub-license, to use the software to develop software applications for ST microcontroller product only. You are not permitted to lease, rent, distribute or sublicense the software or to use the software in a timesharing arrangement or in any unauthorized manner.

Restrictions on use of the software

The software is delivered free of charge by electronically means. You are allowed to copy the software in its initial form to be installed on as many computers as needed.

You shall only use the software on a single computer connected to a single monitor at any one time.

Limited warranty

ST warrants to You that the software will perform substantially in accordance with the accompanying documentation.

ST's total liability and your exclusive remedy for breach of the limited warranty given above shall be limited to ST, it is ST's sole option, using reasonable efforts to correct material, documents, reproduce defects in the software.

Except as provided above ST expressly disclaims all other representations, warranties, conditions or other terms, express or implied, including without limitation the implied warranties of non infringement, satisfactory quality, and fitness for a particular purpose.

Limitation of liability

To the maximum extent permitted by applicable law, in no event shall ST be liable for any indirect, special, incidental or consequential damages (including loss of profits) arising out of the use or inability to use the software whether based on a claim under contract, tort or other legal theory, even if ST was advised of the possibility of such damages.

ST does not seek to limit or exclude liability for death or personal injury arising from ST's negligence and because some jurisdictions do not permit the exclusion or limitation of liability for consequential or incidental damages the above limitation relating to liability for consequential damages may not apply to You.

Third party rights

Software provided under this agreement may contain or be derived from portions of materials provided by a third party under license to ST. The third party disclaims all warranties express or implied with respect to the use of such materials in connection with the software, including (without limitation) any warranties of satisfactory quality or fitness for a particular purpose.

Software provided under this agreement may contain or be derived from portions of materials provided by a third party under license to ST. The third party may enforce any of the provisions of this agreement to the extent such third party materials are effected. Additionally, any limitation of liabilities described in this agreement also applies to any third-party supplier of materials supplied to You. ST and its third party supplier limitations of liabilities are not cumulative. Such third party is an intended beneficiary of this section.

Term and termination

This agreement shall remain in force until terminated by You or by ST.

Without prejudice to any of its other rights if You are in breach of any of the terms and conditions of this agreement then this agreement will terminate immediately. Upon such termination You agree to destroy the software and documentation together with all copies in any form.

You may terminate this agreement at any time by destroying the software and documentation together with all copies in any form.

General

This agreement is governed by the law of France.

This is the only agreement between You and ST relating to the software and it may only be modified by written agreement between You and ST.

This agreement may not be modified by purchase orders, advertising or other representation by any person.

If any clause in this agreement is held by a court of law to be illegal or unenforceable the remaining provisions of the agreement shall not be affected thereby.

The failure by ST to enforce any of the provisions of this agreement, unless waived in writing, shall not constitute a waiver of ST's rights to enforce such provision or any other provision of the agreement in the future.

Use, copying or disclosure by the US Government is subject to the restrictions set out in subparagraph (c)(1)(ii) of the Rights in technical data and computer software clause at DFARS 252.227-7013 or subparagraphs (c)(1) and (2) of the Commercial Computer software - Restricted Rights at 48 C.F.R. 52.227-19, as applicable.

You agree that You will not export or re-export the software to any country, person or entity or end user subject to U.S.A. export restrictions, or other countries' export restrictions.

4 Revision history

Table 6. Document revision history

Date	Revision	Changes
22-May-2008	1	Initial release

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER’S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com